Sergio España Cubillo

Methodological integration of Communication Analysis into a model-driven software development framework

ADVISORS
Arturo González del Río Rams
Óscar Pastor López

December 2011

METHODOLOGICAL INTEGRATION OF COMMUNICATION ANALYSIS INTO A MODEL-DRIVEN SOFTWARE DEVELOPMENT FRAMEWORK

Cover design: Cristina Maestre, Sergio España
Construction of ○□: Cristina Maestre

External reviewers

Reviewers: João Falcão e Cunha (Universidade do Porto, Portugal)
Giancarlo Guizzardi (Universidade Federal do Espírito Santo, Brazil)
Roel Wieringa (Twente Universiteit, The Netherlands)
Substitute reviewers: Paolo Giorgini (Università degli Studi di Trento, Italy)
Xavier Franch (Universitat Politècnica de Catalunya, Spain)
Antoni Olivé (Universitat Politècnica de Catalunya, Spain)

Thesis defence committee

President: Antoni Olivé (Universitat Politècnica de Catalunya, Spain)
Members: Giancarlo Guizzardi (Universidade Federal do Espírito Santo, Brazil)
John Mylopoulos (Università degli Studi di Trento, Italy)
Roel Wieringa (Twente Universiteit, The Netherlands)
Secretary: Vicente Pelechano (Universitat Politècnica de València, Spain)
Substitute members: Nelly Condori (Twente Universiteit, The Netherlands)
Juan Sánchez (Universitat Politècnica de València, Spain)

Author: Sergio España
Address: Research Centre in Software Production Methods (ProS)
Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
Camino de Vera SN, Edificio 1F
46009 Valencia, Spain
Email: sergio.espana@pros.upv.es
Telephone: 963877007 – 83534
Fax: 963877359

METHODOLOGICAL INTEGRATION OF COMMUNICATION ANALYSIS INTO A MODEL-DRIVEN SOFTWARE DEVELOPMENT FRAMEWORK

Sergio España Cubillo

A thesis submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy in Computer Science

Advisors
Prof. Dr. Óscar Pastor López
Dr. Arturo González del Río Rams

The 1st of December 2011

To my father,
for I guess it is not a coincidence that I have become a researcher like him

Preface

"Love, work and knowledge are the wellsprings of our life.
They should also govern it."

Wilhelm Reich

I have passion for my job. I feel fortunate to research in a topic I like, and in a great environment such as the ProS Research Center and the Departamento de Sistemas Informáticos y Computación.

It is indeed true that doing a PhD thesis is a difficult endeavour; I have had to renounce to many things, to work overtime and on the weekends. But I would do it again, no doubt. If I had to start all over again, there are things I would do differently (e.g. I would make this book thiner and simpler); many issues were not worth the worry and some lessons I learnt them the hard way. In any case, the overall experience has been very rewarding and I feel that I have grown both professionally and personally. Professionally, I have discovered the transdisciplinarity of information systems research. In information systems, people and technology meet, so it is a field where very distinct disciplines need to cooperate (e.g. computer science, social sciences, psychology). I have read a lot and I have learnt to define theories, design methods, carry out empirical evaluations, communicate my work to others, etc. There will always be new things to learn, this never ends and I am happy for that. Personally, the thesis has been a bit of an ego-trip for me, with its highs and lows. At the end I like where I have landed: I am very proud of my work, but I also acknowledge that it has been a teamwork and I have learnt to admire my colleagues and lots of great researchers all around the world. I am looking forward to keep on researching and collaborating with them.

I have truly enjoyed carrying out the work and reporting it in this book. It is far from perfect, but I have done my best. Each and every figure has been designed with great care, for I believe that good diagrams improve the understandability of complex theories and models. In fact, they are key tools in business process modelling and requirements engineering. When people ask me what I do for a living I sometimes answer "I draw".

Finally, I would not have been able to finish this thesis without the support of many people for whom I feel respect and gratitude. Let me mention some of them.

Acknowledgements

Arturo, for his lectures were so enthralling that I recovered my lost interest in Computer Science and I decided to be an analyst, for while working for him in a project I enjoyed so much designing and evaluating methods that I decided to be a researcher, for sharing his knowledge with me, for his patience and trust.

Óscar, for offering me the chance to join his research group (now grown up to be a research centre), for infecting me his passion for *model-driven everything*, for teaching me how to move in academic circles, for sharing his vision and expertise with me, for his optimism and his encouragement, for his patience and trust.

My L104 lab mates at DSIC, for creating such a great athmosfere. David Anes, David Melo, Diane, Emanuel, Ignacio, Inés, Kevin, Luis, Marce, Mariam, Nathalie, Nelly, Paco, Raúl, Urko and Yeshica. I am especially grateful to Ignacio, for he is always ready to lend a helping hand, and to Nelly, for helping me see the importance of empirical validations and sharing her experience with me.

The ProS Research Center is a nice place to work. Jorge Belenguer, for the coffee-chat advices when they were most needed. Bea and Giovanni, for their amusing initiatives. Ana Cidad, for greatly pushing the centre further. And all the members of ProS, for their technical strength and their human touch.

Marce, for without her collaboration this thesis would not be the same, for she came in the right time with contagious enthusiasm, for her sense of responsibility and her commitment that make me feel proud of her, for the long way ahead.

My lunch mates, M. Carmen, José, Carlos, César, Jesús, German, Miriam, Jorge... For we have been able to unwind in the middle of our hard-working days.

Ana Gadea, for all her care and the paperwork she has done for me, from the first day to the last! And to all the secretariat at DSIC, for keeping us all in motion.

Those with whom I have studied, discussed or collaborated along these years; I hope we keep in touch. The gang in the EUI stairs, for the great moments while studying. Colleagues in Equipo Drac , COPUT (CIT), EMT, and Anecoop, for all I learnt with you. Xavier Franch et al., Antoni Olivé et al., Jaelson Castro et al., Natalia Juristo et al., Vincent Pijpers, and many more. Frederik Gailly and Geert Poels, for the nice collaboration.

Jean Vanderdonckt, for all the good advices and help on human-computer interaction and usability evaluation, for his Belgian chocolates.

José, David, M. Victoria and Javier, for the fun of working together towards an intelligence test that (according to some journalists) can be applied to... flowers? :)

The good reviewers I have come across in my publications, for their constructive comments. Especially, the reviewers and the defence committee of this thesis, for taking their time to go through this thick book and provide helpful remarks.

The Spanish Ministry of Science and Innovation, Valencian Government, European Regional Development Fund, Teccon Ingenieros Consultores and CARE Technologies, for their financial support so I could eat three times a day.

CARE Technologies, especially Juan Carlos Molina, Jose Miguel Barberá and Emilio Iborra, for their support in using Integranova (formerly known as *OLIVA**NOVA***); especially for letting us use their model compiler, and for their involvement in the experiment in Twente. Thanks, Elena Tejadillos, for travelling there and training the students in Integranova.

With regards to my stay in Twente, I am grateful to many other people as well. Ignacio for carrying out the live code-generation demo. Cristina for her graphic designs and her moral support. Klaas Sikkel and Maya Daneva, for their interest and logistic support, for their patience. Suse Engbers, for being on top of everything, for her sandwiches, for her kindness. My lab mates Silja, Lianne, Hassan, and the rest of the nice members of the IS group, for making my stay there very pleasant. Renske, Frank and Henry, for participating actively in the course. Roel Wieringa, for hosting me in his group, for the discussions on empirical validations, for showing me how to structure and report my research.

Jerónimo Bellido, for teaching me how to breathe, for helping me reach here and go yonder. The groups, for sharing their deepest emotions and knowledge.

The people from Campanar, those from La Cañada, the ones who have emigrated, the Saber Beber club... All my friends, for reminding me that there are more places to be besides my lab and my home, for they keep calling me despite so many "I can't, I've got loads of work."

Everyone I forgot to include above. You are probably in my mind at the moment of writing these lines, but my mind has been elsewhere lately; please accept my apologies. Thank you, _____ (write your name here).

Always in my heart. My family, for creating a safe an happy environment for me to grow and explore the world. My parents, Pilar and Paco, for nurturing me with love and affection, for the world-wide travels, for transmitting to me their thirst for knowledge. Cris, for sharing joys and woes, for loving me and letting me love her, for her support, for designing a fabulous cover, for the home warmth. To them all I dedicate this thesis.

Valencia, December 2011 Sergio España

Agradecimientos

Arturo, porque sus clases eran tan interesantes que recuperé la vocación por la Informática y decidí ser analista, porque trabajando con él en un proyecto disfruté tanto diseñando y evaluando métodos que decidí ser investigador, por compartir su conocimiento conmigo, por su paciencia y su confianza.

Óscar, por ofrecerme la oportunidad unirme a su grupo de investigación (ahora centro), por contagiarme su pasión por el *todo dirigido por modelos*, por enseñarme a moverme por el mundo académico, por compartir su visión y su experiencia conmigo, por su optimismo y sus ánimos, por su paciencia y su confianza.

Mis compañeros del laboratorio L104 del DSIC, por crear tan buen ambiente. David Anes, David Melo, Diane, Emanuel, Ignacio, Inés, Kevin, Luis, Marce, Mariam, Nathalie, Nelly, Paco, Raúl, Urko y Yeshica. Especialmente agradecido a Ignacio, por estar siempre dispuesto a echar una mano. Nelly, por ayudarme a ver la importancia de las validaciones empíricas y compartir su experiencia.

El Centro de Investigación ProS es un buen lugar para trabajar. Jorge Belenguer, por los consejos tomando café cuando más falta hacían. Bea and Giovanni, por sus entretenidas iniciativas. Ana Cidad, por su buen trabajo potenciando el centro. Y todos los miembros del ProS, por su fortaleza técnica y calidad humana.

Marce, porque sin su colaboración esta tesis no sería lo mismo, porque llegó en el momento apropiado y con un entusiasmo contagioso, porque su sentido de la responsabilidad y su compromiso me hacen sentir orgulloso, por el camino por delante.

Mis compañeros de la comida, M. Carmen, José, Carlos, César, Jesús, German, Miriam, Jorge... Porque hemos podido desconectar a mitad de la dura jornada.

Ana Gadea, por todas sus atenciones y los papeleos que ha hecho por mí, ¡desde el primer días hasta el último! Y a toda la secretaría del DSIC, porque mantienernos a todos en marcha.

Aquellos con quienes he estudiado, discutido o colaborado durante estos años; espero que mantengamos el contacto. La peña de las escaleras de la EUI, por los grandes momentos durante la carrera. Colegas en Equipo Drac , COPUT (CIT), EMT, y Anecoop, por todo lo que aprendí con vosotros. Xavier Franch et al., Antoni Olivé et al., Jaelson Castro et al., Natalia Juristo et al., Vincent Pijpers y muchos más. Frederik Gailly y Geert Poels, por la colaboración que realizamos.

Jean Vanderdonckt, por los buenos consejos y su ayuda en interacción persona-ordenador y evaluación de usabilidad, por sus chocolates belgas.

José, David, M. Victoria y Javier, por la diversión al trabajar juntos para conseguir un test de inteligencia que (según algún periodista) podemos aplicar a ¿flores? :)

Los buenos revisores que he encontrado en mis publicaciones, por sus críticas constructivas. Especialmente, los revisores y tribunal de defensa de esta tesis, por encontrar tiempo para leer este libro tan grueso y ofrecer comentarios útiles.

El Ministerio de Ciencia e Innovación, la Generalitat Valenciana, el Fondo Europeo de Desarrollo Regional, Teccon Ingenieros Consultores y CARE Technologies, por su financiación, por la cual he podido comer tres veces al día.

CARE Technologies, especialmente Juan Carlos, Jose Miguel Barberá y Emilio Iborra por su apoyo en el uso de Integranova (anteriormente llamado *OLIVA**NOVA***), especialmente por permitirnos usar su compilador de modelos, y por su participación en el experimento en Twente. Gracias, Elena, por viajar allá y entrenar a los estudiantes en Integranova.

Con respecto a mi estancia en Twente, me siento agradecido a muchas personas. Ignacio por llevar a cabo la demo de generación de código en vivo. Cristina Maestre por sus diseños gráficos y su apoyo moral. Klaas Sikkel y Maya Daneva, por su interés y apoyo logístico, por su paciencia. Suse Engbers, por haberse ocupado de todo, por sus sándwiches, por su amabilidad. Mis compañeros de laboratorio, Silja, Lianne, Hassan, y el resto de miembros del grupo IS, por hacer que mi estancia fuera tan agradable. Renske, Frank and Henry, por participar activamente en el curso. Roel Wieringa, por acogerme en su grupo, por las discusiones sobre validaciones empíricas, por enseñarme a estructurar y comunicar mi investigación.

Jerónimo Bellido, por enseñarme a respirar, por ayudarme a llegar hasta aquí y a ir más allá. Los grupos, por compartir sus conocimientos y emociones más profundas.

Los de Campanar, los de La Cañada, los que han emigrado, el club del Saber Beber... Todos mis amigos, por recordarme que hay otros lugares aparte de mi laboratorio y mi casa, por seguir llamándome pese a tantos "no puedo, es que tengo mucho curro".

Todos a quienes he olvidado incluir arriba. Probablemente te tengo en mente en el momento de escribir estas líneas, pero últimamente tengo la mente en otra parte; por favor, acepta mis disculpas. Gracias, _____ (pon tu nombre aquí).

Siempre en mi corazón. Mi familia, por crear un entorno seguro y feliz en el que crecer y descubrir el mundo. Mis padres, Pilar y Paco, por criarme con amor y afecto, por los viajes por el mundo, por contagiarme el hambre de conocimientos. Cris por compartir alegrías y penas, por amarme y dejarse amar, por su apoyo, por diseñarme una portada fabulosa, por el calor de hogar. A tod@s ell@s dedico esta tesis.

Valencia, diciembre de 2011 Sergio España

Abstract

It is widely recognised that information and communication technologies development is a risky activity. Despite the advances in software engineering, many software development projects fail to satisfy the clients' needs, to deliver on time or to stay within budget. Among the various factors that are considered to cause failure, an inadequate requirements practice stands out. Model-driven development is a relatively recent paradigm with the potential to solve some of the dragging problems of software development. Models play a paramount role in model-driven development: several modelling layers allow defining views of the system under construction at different abstraction levels, and model transformations facilitate the transition from one layer to the other. However, how to effectively integrate requirements engineering within model-driven development is still an open research challenge. This thesis integrates Communication Analysis, a communication-oriented business process modelling and requirements engineering method for information systems development, and the OO-Method, an object-oriented model-driven software development method provides automatic software generation from conceptual models. We first provide a detailed specification of Communication Analysis intended to facilitate the integration; among other improvements to the method, we build an ontology-based set of concept definitions in which to ground the method, we provide precise methodological guidelines, we create a metamodel for the modelling languages included in the method, and we provide tools to support the creation of Communication Analysis requirements models. Then we perform the integration by providing a technique to systematically derive OO-Method conceptual models from Communication Analysis requirements models. The derivation technique is offered in two flavours: a set of rules to be manually applied by a human analyst, and an ATL model transformation that automates this task. The proposals described this thesis have been validated by several means, ranging from theoretical evaluations such as ontological analysis to empirical evaluations such as controlled experiments. The results demonstrate the benefits of Communication Analysis and the success of its integration with the OO-Method. Obviously, there is much space for improvement that we will address in future works, but the result has fulfilled the initial research and engineering goals.

Resumen

Es ampliamente reconocido que el desarrollo de tecnologías de la información y la comunicación conlleva numerosos riesgos. A pesar de los avances en la ingeniería del software, gran parte de los proyectos de desarrollo de software fracasan en satisfacer las necesidades de los clientes, se exceden en tiempo o sobrepasan el presupuesto. La práctica inadecuada de ingeniería de requisitos destaca entre los factores que influyen en este fracaso. El desarrollo dirigido por modelos es un paradigma relativamente reciente con el potencial de aliviar algunos de los problemas recurrentes del desarrollo de software. Proporciona varias capas de modelado que permiten definir vistas del sistema en construcción con diferentes niveles de abstracción y se apoya en transformaciones de modelos que facilitan la transición de una capa a la otra. En cualquier caso, sigue siendo un reto de investigación abierto cómo integrar de manera efectiva la ingeniería de requisitos en el desarrollo dirigido por modelos. Esta tesis integra el Análisis de Comunicaciones, un método de modelado de procesos de negocio e ingeniería de requisitos orientado a comunicaciones, y OO-Method, un método de desarrollo de software dirigido por modelos que provee generación automática de código a partir de modelos conceptuales. En primer lugar, proporcionamos una especificación detallada del Análisis de Comunicaciones con al intención de facilitar la posterior integración; entre otras mejoras al método, fundamentamos el método en un conjunto de definiciones de conceptos basadas en una ontología de referencia, creamos un metamodelo para los lenguajes de modelado incluidos en el método y proveemos de herramientas para soportar la creación de los modelos de requisitos. En segundo lugar llevamos a cabo la integración proporcionando una técnica para derivar sistemáticamente modelos conceptuales de OO-Method a partir de modelos de requisitos de Análisis de Comunicaciones. La técnica de derivación se ofrece en dos formatos: un conjunto de reglas de derivación para ser aplicadas manualmente por un analista humano y una transformación de modelos implementada en ATL que automatiza esta tarea. Las propuestas descritas en esta tesis se han validado de diversas maneras, abarcando desde evaluaciones teóricas como el análisis ontológico hasta evaluaciones empíricas como experimentos controlados. Los resultados demuestran los beneficios del Análisis de Comunicaciones y el éxito de su integración con OO-Method. Evidentemente, existen numerosos aspectos mejorables que abordaremos en trabajos futuros, pero el resultado ha cumplido los objetivos de investigación e ingeniería iniciales.

Resum

És àmpliament reconegut que el desenvolupament de tecnologies de la informació i la comunicació comporta nombrosos riscos. A pesar dels avanços en l'enginyeria del programari, gran part dels projectes de desenvolupament de programari fracassen a satisfer les necessitats dels clients, s'excedeixen en temps o sobrepassen el pressupost. La pràctica inadequada de l'enginyeria de requisits destaca entre els factors que influeixen en aquest fracàs. D'una altra banda, el desenvolupament dirigit per models és un paradigma relativament recent amb el potencial d'alleujar alguns dels problemes recurrents del desenvolupament de programari. Proporciona diverses capes de modelatge que permeten definir vistes del sistema en construcció amb diferents nivells d'abstracció i es recolza en transformacions de models que faciliten la transició d'una capa a l'altra. En qualsevol cas, segueix sent un repte d'investigació obert com integrar de manera efectiva l'enginyeria de requisits en el desenvolupament dirigit per models. Aquesta tesi integra l'Anàlisi de Comunicacions, un mètode de modelatge de processos de negoci i enginyeria de requisits orientat a comunicacions, i OO-Method, un mètode de desenvolupament de programari dirigit per models que proveeix generació automàtica de codi a partir de models conceptuals. En primer lloc, proporcionem una especificació detallada de l'Anàlisi de Comunicacions amb a la intenció de facilitar la posterior integració; entre altres millores al mètode, fonamentem el mètode en un conjunt de definicions de conceptes basades en una ontologia de referència, creem un metamodel per als llenguatges de modelatge inclosos en el mètode i proveïm d'eines per a suportar la creació dels models de requisits. En segon lloc, duem a terme la integració proporcionant una tècnica per a derivar sistemàticament models conceptuals de OO-Method a partir de models de requisits d'Anàlisi de Comunicacions. La tècnica de derivació s'ofereix en dos formats: un conjunt de regles de derivació per a ser aplicades manualment per un analista humà i una transformació de models implementada en ATL que automatitza aquesta tasca. Les propostes descrites en aquesta tesi s'han validat de diverses maneres, abastant des d'avaluacions teòriques com l'anàlisi ontològica fins a avaluacions empíriques com els experiments controlats. Els resultats demostren els beneficis de l'Anàlisi de Comunicacions i l'èxit de la seua integració amb OO-Method. Evidentment, existeixen nombrosos aspectes millorables que abordarem en treballs futurs, però el resultat ha complit els objectius d'investigació i enginyeria inicials.

Table of contents

List of figures

List of tables

PART I.
PROBLEM INVESTIGATION

Chapter 1
Motivation

"Communication is the lifeblood of any organization..."

Clinton O. Longenecker, Jack L. Simonetti, Thomas W. Sharkey

Two million years ago, hominids used wooden and stone tools to increase the efficiency of their tasks. As hominids evolved, the complexity of the tasks they undertook grew, but so did their technology. As time elapsed, they used fish hooks, bone needles, pottery, wheels, astrolabes, cranes, steam engines... The advent of computational machines was a giant leap for human technology. At some point in time, computational machines started having their behaviour specified by sets of rules, now known as *software*. The complexity of the tasks that were intended to be solved by means of automatic computations grew so much, that a series of phenomena coined as *software crisis* became evident [Naur and Randell 1969]: software-related projects were difficult to manage and often exceeded the estimated time and budget, software had low quality and did not satisfy the users' requirements, the software code was difficult to maintain and degraded over time, etc. In short, there was a widening gap between the expectations and the software that was actually produced.

As a reaction to the so-called software crisis, an effort was made to manufacture software by following theoretical foundations and adopt practical disciplines that were traditional in the established branches of engineering: *software engineering* was born[1]. The main objective of software engineering was to increase quality; i.e. to be able to deliver software products that meet the real needs of the clients and final users (product quality), and to do so on time and within budget (process quality). For this purpose, the software engineering community has proposed numerous software life-cycle paradigms (e.g. cascade development, spiral, iterative), analysis and design paradigms (e.g. structured

[1] The term "software engineering" was initially coined with the intention of being provocative, since many considered software production an artistic activity.

analysis and design, object orientation, goal orientation) and implementation paradigms (ranging from monolithic applications to multi-tier architectures, and even cloud deployment); but the problem persists.

There is no agreement in information systems development project failure rates. The widely cited 1994 Chaos Report suggests that 84% of projects are unsuccessful [The Standish Group 1995][2]. A 2006 Chaos Report highlights "a major up-tick" (sic) in success rates: the rate of unsuccessful projects drops to 65% [The Standish Group 2008]. The evolution of success rates can be seen in Figure 1. In any case, it is widely recognised that information and communication technologies development is a risky activity. Various factors are considered to cause failure. For instance, project complexity increases risk significantly [Daniels and LaMarsh 2007], and an inadequate requirements practice is also considered a major factor [IT Cortex]. This thesis deals with software requirements and it is specifically focused on a type of human tool named *information systems*.

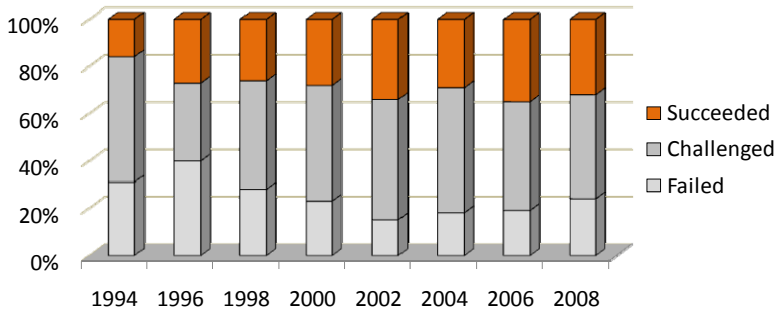


Figure 1. Evolution of software development project success rates[3]

1.1 *Enterprise information systems*

Information systems as a support for organisational communication

Among all the possible types of software that can be developed, this thesis focuses on information systems[4]. An *information system* (IS) is a socio-technical

[2] The report uses an ambiguous concept of failure and, therefore, its validity has been questioned [Glass 2006] [Eveleens and Verhoef 2010]

[3] The chart data comes from several Chaos Reports, the latest being issued in 2010 [The Standish Group 2010].

system that aims to support communication between the an organisation and its environment and also within the organisation itself [Langefors 1977]. An information system can be seen as a collection of agents of different nature that cooperate in order to acquire, process, store, retrieve and distribute information with the purpose of observing, controlling or influencing a portion of the world [Lockemann and Mayr 1986].

Notice that, according to the definition above, an information system does not necessarily include any software. It is important to make the intellectual effort to distinguish between information systems and computerised information sub-systems. For instance, at the time of the Inca (c. 1200-1475 AD) there was no computer technology; this wealthy pre-Columbian civilisation did not even possess writing, and yet they deployed an successful information system with which they administered the far reaches of their empire [Beynon-Davies 2007].

Information systems are created within an organisation for the purpose of improving the effectiveness of that organisation in achieving its goals, and it is computerised with the purpose of increasing the efficiency of the work practice. Whether these purposes are achieved or not depends on the capabilities of the information system, but also on the characteristics of the organisation [Silver, Markus et al. 1995]. Thus, it is necessary that the information technology is aligned with the organisational goals. Ensuring that the information system appropriately supports the organisation business processes facilitates this alignment. However, ascertaining organisational goals and business processes is not a simple task.

The importance of engineering requirements

Information systems development and computerisation is a wicked problem[5]. To a large extent, this is due to their socio-technical nature [Lockemann and Mayr 1986] and to the intervention of multiple stakeholders with often conflicting goals, needs and world views. To overcome conflicts and to reach agreement, stakeholders' perceptions have to be clarified and shared with information system developers. Requirements engineering (RE) facilitates this process by offering techniques for the discovery and description of stakeholders' goals and needs. The discipline is still young and many research questions remain open,

[4] This means that the methods discussed and proposed on this thesis, are mainly intended to be used for information system development (although they may apply as well to other types of software).

[5] Among other characteristics, wicked problems do not have a unique solution and their statement is not clear until they are solved (in part due to stakeholder discrepancies) [Rittel and Webber 1973].

such as a clear distinction between what a requirement is and what it is not (e.g. what vs. how [Davis 1990], why [Rolland 2007]).

There are also different orientations towards requirements. Attending to academic literature and industrial practice, various conceptions of information systems are found. Some authors consider information systems as a mere representation of reality [Wand and Weber 1995]. Under this perspective, information systems can be described following an ontological approach; that is, focusing on the objects perceived in the universe of discourse (e.g. OO-Method [Pastor and Molina 2007]). Other perspectives focus on organisational intentions (e.g. Maps [Rolland 2007]), value object exchanges (e.g. e3-value [Gordijn and Wieringa 2003]), speech acts (e.g. Language-Action Perspective [Winograd 1986]), or business processes (e.g. ARIS [IDS Scheer 2005]).

The sizeable scientific production in the area of requirements engineering indicates that we are dealing with an open problem. Many challenges issued by Bubenko in 1995 [Bubenko 1995] are still valid today. As some studies show [Juristo, Moreno et al. 2002] [Kaindl, Brinkkemper et al. 2002], new requirements elicitation techniques keep on being proposed but they do not live up to industry expectations. Of the many factors hinder industry adoption of requirements engineering practices, one of most decisive is the inherent complexity of dealing with human beings.

Despite the open issues, it is widely accepted the fact that engineering requirements is a good practice that has a strong impact in the software development process; an invalidity or an ambiguity in early information system specifications leads to costly errors in later design and implementation stages [Sommerville 2006].

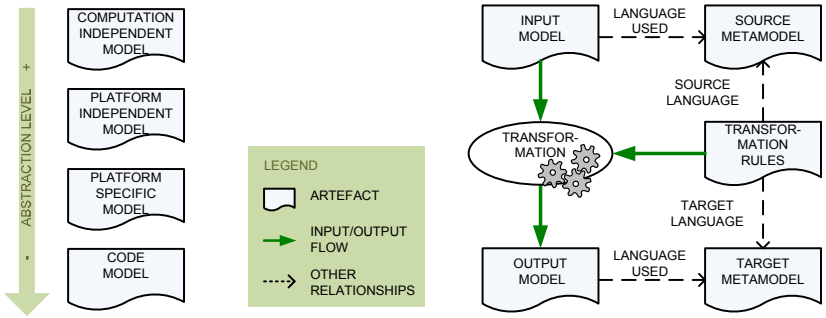
Model-driven development: a promising paradigm

"The entire history of software engineering is that of the rise in levels of abstraction" said Grady Booch in his talk "The promise, the limits, the beauty of software". In fact, if software conception and specification is analysed, a series of increases in abstraction can be distinguished; punched cards, assembler programming languages, third generation programming languages, visual programming languages... In recent times, a new paradigm is gaining recognition: model-driven development. The aim of *model-driven development* (MDD) is to raise the abstraction level of specifications in order to bring them closer to the problem space. This is achieved by means of creating (generally graphical) models that represent the problem at hand (i.e. the task that needs to be supported) and using model transformation technologies to derive a software product (i.e. the tool that will support the task).

The acceptance of a modelling notation standard such as the UML [OMG 2009] and the establishment of Model Driven Architecture [OMG 2003] as a reference

modelling framework are steps towards the wide adoption of model-driven development as a software development paradigm. *Model Driven Architecture* (MDA) proposes a software development lifecycle in which higher-abstraction models are gradually converted into lower-abstraction models (see Figure 2.a). Model Driven Architecture distinguishes four modelling layers: the *computation independent model* (CIM) describes a system disregarding whether it will be computerised of not, the *platform independent model* (PIM) describes the system disregarding the specific technology that will support it, the *platform specific model* (PSM) takes into account the target platform, and the *code model* (CM) actually corresponds to the final software source code.

A basic principle of Model Driven Architecture is that, when shifting from one modelling layer to the next, models should be operated by means of (preferably automatic) model transformations. For each modelling primitive, a correspondence with a software pattern is defined, in a way that it is possible to generate the software code. Since the grammars of modelling languages in the field of software engineering are often expressed by means of metamodels (informally, a metamodel is a model of a model), model transformations can be expressed in terms of rules that convert metamodels elements from the source metamodel into metamodel elements of the target metamodel (see Figure 2.b).



a) MDA modelling layers **b)** Generic metamodel-based transformation

Figure 2. Model-Driven Architecture principles

In many recent proposals, programming gives way to modelling. For Extreme Non-Programming [Morgan 2002], the key artefact of the software production process is a conceptual model that describes the business domain; the software application is generated by means of a model transformation. Conceptual Schema-Centric Development [Olivé 2005] also claims that defining a conceptual schema is necessary and sufficient for developing a computerised information sub-system. As discussed below, the OO-Method is an MDD method that fully realises the intended code generation.

1.2 Problem statement

Model-driven development + requirements engineering

Although the term is not well established in the literature, we can refer as *model-driven requirements engineering* to the practice of engineering requirements by placing the emphasis on modelling and model transformations. This way, initial versions of the later models can be derived automatically from the requirements model.

There is currently no approach that successfully integrates a requirements engineering method into a model-driven software production method. Menzies [2003] observes that there is no complete approach to model-driven requirements engineering. This observation is confirmed by the systematic literature review by Loniewski, Insfran et al. [2010]; the authors conclude that, despite many MDD methods include some requirements engineering activities, there does not exist a complete tool-supported solution to manage requirements models and make further use of them in an automated way. Moreover, the traceability among requirements, elements of later models and pieces of the generated software needs much improvement.

This thesis is motivated by the need to define a full-fledged model-driven software production method that covers all the stages of software development. This challenge can be faced in two ways:

- Providing new methods and tools that are conceived within the MDD paradigm (e.g. [Kabanda and Adigun 2006]).
- Integrating existing model-based methods and tools (e.g. [Alférez, Kulesza et al. 2008]).

This thesis takes the second path by integrating **Communication Analysis** [España, González et al. 2009], a communication-oriented requirements engineering method for information systems development, and the **OO-Method**, an object-oriented software production method that is MDA-compliant and provides automatic software generation from conceptual models. The objective is

to offer a software development method that starts with requirements engineering and allows seamlessly deriving and completing the conceptual model as well as generating the final software application by means of a model compilation.

Why the OO-Method

This thesis takes OO-Method as the reference MDD method [Pastor, Gómez et al. 2001]. The OO-Method methodological core is the *object-oriented conceptual modelling* stage, the result of which is the *Conceptual Model*, an object-oriented model that describes the computerised information sub-system disregarding the implementation platform. This way, the OO-Method Conceptual Model corresponds to the PIM layer of the Model Driven Architecture (see Figure 3). This model is composed of four complementary submodels: the Object Model offers the static view of the computerised-information sub-system, the Dynamic Model and the Functional Model offer the behavioural view, and the Presentation Model offers the interaction view. Section 2.4 further describes the state of the OO-Method at the start of this thesis.

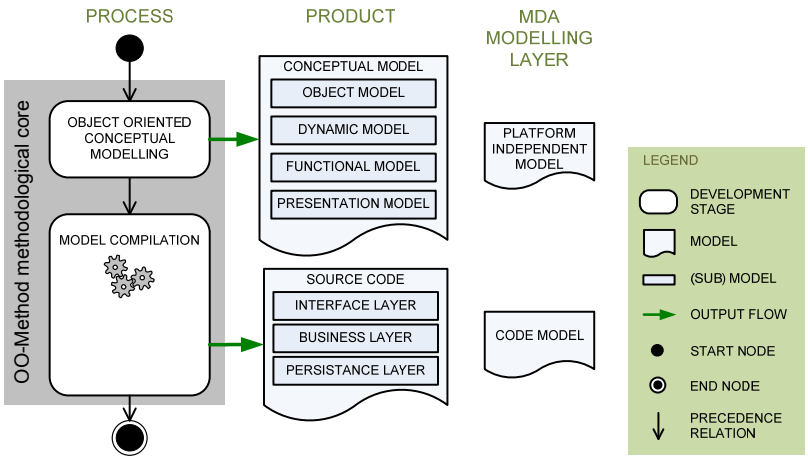


Figure 3. Methodological core of the OO-Method

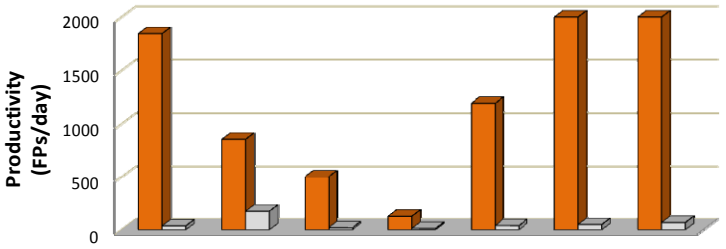
After the conceptual modelling stage, a model compiler[6] takes as input the conceptual model and a set of compilation parameters (e.g. the selected database management system and programming language) and it generates the source code of a software application that is functionally equivalent to the conceptual model (i.e. the application fulfils the specifications determined by the model). The

[6] A model compiler is a software program that automates model transformations in order to generate software source code.

automatically generated software application is fully functional and it is organised in a three-layer architecture: interface, business logic and persistence.

The methodological core of the OO-Method is mature and it is industrially supported by CARE Technologies [CARE Technologies]. This company has developed Integranova Model Execution system, a tool suite that supports the OO-Method methodological core. The suite of tools includes Integranova Modeler, a computer-aided software engineering (CASE) tool that allows specifying the Conceptual Model, and a model compiler. Both the OO-Method and the Integranova technology are used in practice to develop enterprise information systems.

Several reports by Gartner have demonstrated the advantages of using the Integranova technology. According to a 2003 study [Molina 2003], time to market is nearly 6 times better and performance ranges from 12 to 47 times higher than similar projects using other development tools. Moreover, the defects ratio[7] is over 15 times better than in the peer group (see Figure 4).



	PROJECT MANAGEMENT	QUALITY	SPECIFICATIONS	PROGRAM-MING	ACCEPTANCE TESTS	INITIAL & POST-START-UP	OTHER ACTIVITIES
CARE TECHNOLOGIES	1845	856	502	127	1193	15860	13703
PEER GROUP (AVG.)	37	174	18	13	40	50	69

Figure 4. Productivity ratios according to the Gartner report [Molina 2003]

Therefore, the OO-Method is a mature MDD method with industrial support. Also, the research group in which this thesis is developed has expertise in the OO-Method, and there exists a collaboration environment with the CASE tool developer, as well as with OO-Method practitioners.

With regards to the current requirements engineering practice in the software development company that uses the OO-Method, is the following. The analysts (conceptual modellers) interview the users in order to find out their needs. After each interview, the analyst creates the proceedings that account for what has been discussed and sends them to the client. This documentation is mainly used to

[7] The defects ratio indicates to the amount of defects per 1000 function points.

specify the scope of the development project, in order to solve possible contractual disagreements. Requirements specification is perceived as a burden, mainly because there is no implemented strategy for automatically transforming requirements models into conceptual models[8].

Why Communication Analysis

Since information systems are a support to organisational communication, a communicational approach to information systems analysis is necessary; that is, we[9] claim that information systems requirements engineering should take into account users' communicational and informational needs. Some authors have even demonstrated that a communicational orientation for business process models is convenient because it is associated with increased business process model quality, which in turn affect positively the success of business process redesign efforts [Kock, Verville et al. 2009]. Communication Analysis is a requirements engineering method that analyses the communicative interactions between the information system and its environment [España, González et al. 2009]. Therefore, the method focuses on external information system functions: information acquisition and distribution[10].

The methodological core of Communication Analysis is the *information system analysis* stage, the result of which is an *analysis specification*, a communication-oriented documentation that describes the information system disregarding its possible computerisation (see Figure 5). This way, the analysis specification produced by Communication Analysis corresponds to the CIM layer of the Model Driven Architecture. This specification includes a *system decomposition* that intends to reduce complexity, a set of *communicative event diagrams* that describe business processes from a communicational perspective, a set of *event specification templates* that describe each business activity in detail, and the *object requirements* that specify business objects from a user perspective. Section 2.1 further describes the state of Communication Analysis at the start of this thesis.

---

[8] In a way, from the point of view of Integranova analysts, this thesis tackles a *technology-push problem*. Most analysts applying the OO-Method and using the Integranova technology do not care that much about model-driven requirements engineering, but the researchers see an improvement oportunity and expect its adoption.

[9] This is the PhD thesis of an individual named Sergio España. However, throughout the document, the first person of the plural is often used. This is both a stylistic resource and an acknowledgement to my advisors and colleagues. Without them, this thesis would not exist.

[10] We advocate that these two functions should the focus of information systems analysis. The description of information storing, retrieval and processing corresponds to information systems design.

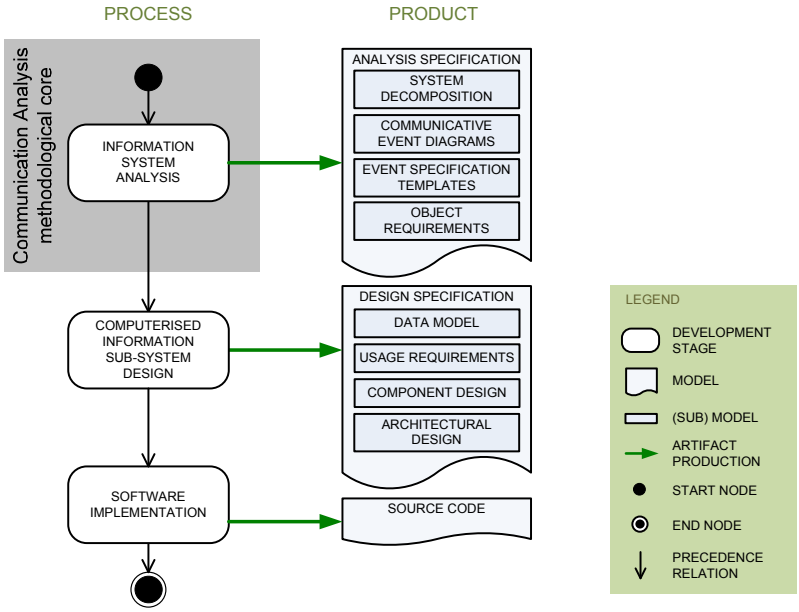


Figure 5. Methodological core of Communication Analysis within a wider development method

After the information system analysis stage, several situations are possible. For instance, the analysts can create the *design specification*. Then the implementation can be carried out within the organisation or it can be outsourced. In either case, upon the start of this thesis, the implementation has always been carried out manually (i.e. not using model compilers).

Communication Analysis has been successfully put into practice in medium and big software development projects in enterprises and governmental organisations. A few prototype tools exist, but practitioners mainly use general-purpose diagramming tools (e.g. Visio) and text processors (e.g. Microsoft Word) for the purpose of specifying requirements. Later stages, such as design, implementation and testing, although driven by the requirements specification, do not make use of model transformation technologies.

Research questions

As discussed above, recent systematic reviews of the literature in model-driven requirements engineering show that integrating a requirements engineering method in an MDD framework is still an open research problem. Evidences indicate that there remain several challenges yet to be solved. First, it is not clear which analytic perspective suits model-driven development of information systems best (e.g. object orientation, use case orientation, goal orientation). In any

case, one perspective needs to be chosen in order to provide an operable method[11]. This thesis chooses to focus on business processes from a communicational perspective. Then, it needs to be investigated whether the available documentation of a specific requirements engineering method serves the purpose of integrating the requirements engineering method into an MDD framework, since some changes may need to be made to the documentation in order to facilitate the integration. Also, it is not a straightforward task to provide the appropriate transformation guidelines to derive conceptual models from requirements models. Furthermore, the traceability of requirements across the different model layers and software components needs to be provided. And of, course, apart from providing solutions to the aforementioned challenges, the validity of the solutions needs to be empirically proved.

It is an initial assumption in this thesis that the OO-Method practitioners can benefit from a well integrated requirements engineering method, and that Communication Analysis practitioners can benefit from using model transformation technologies. The following research goal is aimed at.

> *Main research goal*. To successfully integrate Communication Analysis and the OO-Method, in order to obtain a full-fledged software development method that covers from requirements engineering to automatic code derivation.

To address this goal, we define several research questions that guide the research described in this thesis. Some research questions are further decomposed into other research questions. We distinguish between knowledge problems (KP) and design problems (DP)[12] as defined by Wieringa and Heerkens [2006].

- ***RQ1*** (KP). What elements are common to all enterprise information systems, disregarding whether it is computerised or not? The answer to this question should clarify the terminology and facilitate reasoning about the relation between requirements engineering and conceptual modelling in the information systems domain.
  - ***RQ1.1*** (KP). What conceptual frameworks about information systems exist today?
  - ***RQ1.2*** (KP). Do they provide enough level of detail to facilitate the above-mentioned reasoning?

[11] Some methods propose a combination of several orientations, but often one orientation provides the backbone of the modelling framework.

[12] A *knowledge problem* is a lack of knowledge about the world (as long as someone desires to fill this gap), whereas a *design problem* (a.k.a. practical problem) is the difference between the way the world is and the way someone thinks it should be.

  - ***RQ1.3*** (DP). If not, how can an information systems conceptual framework be extended with the purpose of aiding the research goal of this thesis?
- **RQ2** (KP). Which RE methods exist that can be effectively used in an MDD framework?
  - ***RQ2.1*** (KP). What kind of model-driven requirements engineering approaches exist?
  - ***RQ2.2*** (KP). How suitable these requirements engineering approaches are for developing information systems the MDD way?
  - ***RQ2.3*** (KP). What characteristics needs to have a requirements engineering method specification in order to integrate it into an existing MDD framework?
  - ***RQ2.4*** (KP). Can Communication Analysis be integrated into the OO-Method, given the characteristics of its current specification?
  - ***RQ2.5*** (DP). If not, what changes in the method specification are needed?
- ***RQ3*** (DP). How can Communication Analysis be integrated into the OO-Method framework? With regards to the derivation of the conceptual model, two situations are taken into account: (i) a situation in which the derivation is performed by the information system analyst, and (ii) a situation in which the derivation is performed automatically by a transformation engine.
  - ***RQ3.1*** (DP). What methodological guidelines are needed in order to facilitate a developer the manual derivation of a conceptual model from a requirements model?
  - ***RQ3.2*** (DP). What model transformations are needed in order to automatically derive a conceptual model from a requirements model?
- ***RQ4*** (KP). How can Communication Analysis be validated as a requirements engineering method?
  - ***RQ4.1*** (KP). What quantitative and qualitative metrics are useful for validating a requirements engineering method?
  - ***RQ4.2*** (KP). How can the plausibility of the conclusions about Communication Analysis be ensured and improved?
- ***RQ5*** (KP). How can the integration of Communication Analysis into the OO-Method framework be validated?
  - ***RQ5.1*** (KP). What quantitative and qualitative metrics are useful for validating the integration of an requirements engineering method into an MDD framework?
  - ***RQ5.2*** (KP). How can the plausibility of the conclusions about the integration of Communication Analysis into the OO-Method framework be ensured and improved?

1.3 Main contributions

The contributions of the thesis are as follows:

- ***C1***. We analyse the fundamental concepts of information systems and the relationships among them, building upon previous theoretical studies.
- ***C2***. We present a detailed specification of Communication Analysis. Taking as input the available method documentation, we create an improved documentation with several goals in mind: (i) it serves as a departure point for creating a handbook for practitioners (although it is far too academic for this purpose), (ii) it is suitable for the integration of the method in an MDD framework (it includes the corresponding metamodels and tool support).
- ***C3***. We integrate Communication Analysis and the OO-Method taking into account two software development scenarios: (i) a human-centred process in which the creation of the conceptual model is performed by the practitioner by applying a set of derivation guidelines, and (ii) a tool-supported process in which the initial creation of the conceptual model is performed automatically by executing a metamodel-based transformation.
- ***C4***. We illustrate how the detailed specification of Communication Analysis can be validated. More precisely, we describe the results of an ontological evaluation, several lab demos and a controlled experiment:
  - ***C4.1***. An ontological evaluation that assesses the alignment of the concepts of Communication Analysis with the concepts of a reference ontology.
  - ***C4.2***. An experiment that compares the performance of two RE methods; namely Use Cases and Communication Analysis.
  - ***C4.3***. Moreover, the thesis contributes an extension to current empirical software engineering frameworks. It does so by proposing new metrics that can be used to validate requirements modelling methods (e.g. C4.2).
- ***C5***. We illustrate how the integration of Communication Analysis and the OO-Method can be validated. More precisely, we describe the results of an ontological evaluation, several lab demos and two controlled experiments:
  - ***C5.1***. An ontological evaluation that assesses the alignment of OO-Method concepts with a reference ontology. Also, a theoretical evaluation of the integration of Communication Analysis and the OO-Method.
  - ***C5.2***. An experiment that assesses the effect that an early stage of communication-oriented business process modelling has in the outcome of a later stage of object-oriented conceptual modelling.
  - ***C5.3***. An experiment that compares two conceptual model derivation approaches with regards to the conceptual model quality: (i) a set of heuristics applied to textual requirements specifications, and (ii) a set of derivation guidelines applied to Communication Analysis models.

1.4 Research methodology

This thesis presents the results of a 5-year research. Of course, it builds upon previous works; especially upon the results of the research lines leaded by Óscar Pastor (e.g. [Pastor, Gómez et al. 2001]) and Arturo Gonzalez (e.g. [González 2004]). The research follows the general guidelines of the scientific method [Bunge 1998]. The type of research that this thesis undertakes corresponds to the design science framework since it aims to design a new artefact (an integration of two methods), by means of acting and deciding on the basis of a systematic body of evidence [Hevner, March et al. 2004] [Peffers, Tuunanen et al. 2008]. The research methodology can be explained by means of a rational problem decomposition, as proposed by Wieringa [2009]. Figure 6 shows the regulative cycles that have been enacted[13] in order to answer the research questions.

The integration of two methods is a practical problem[12] and, therefore, a matter of design. Engineering cycle EC1 is a rational problem decomposition for the practical problem of integrating Communication Analysis and the OO-Method. EC1 starts with a problem investigation; related tasks are the definition of thesis motivation and goals (T1.1, already discussed above) and the definition of a theoretical framework on information system development and MDD (T1.2). The solution design requires reviewing the state of the art in RE approaches for the OO-Method (T2.1); this review reveals that there exist previous RE methods that have been integrated into the OO-Method. Despite this, it is decided to integrate Communication Analysis into the OO-Method framework.

An initial attempt to perform the integration (T2.2a) shows that the available specification of Communication Analysis is insufficient for this purpose. This situation leads to another engineering cycle, EC2, aimed at providing a detailed specification of Communication Analysis that facilitates the integration. Since many RE methods already exist by the time EC2 starts and, at this point, Communication Analysis has not yet been presented to the academic community, problem investigation involves motivating the need for another RE method (T3.1). Also, a conceptual framework of information systems concepts is built, so as to clarify terminology and settle some foundations for the rest of the research (T3.2). A review of the literature on RE methods is performed (T4.1). Then Communication Analysis is specified in detail, including the requirements structure prescribed by the method (T4.2) and the metamodels that enable an integration into an MDD framework (T4.3).

[13] Although the daily research practice that was actually carried out included some false starts and deviations, it is convenient to explain the research method and to report the results according to the standard regulative cycles, as argued in [Wieringa 2009].

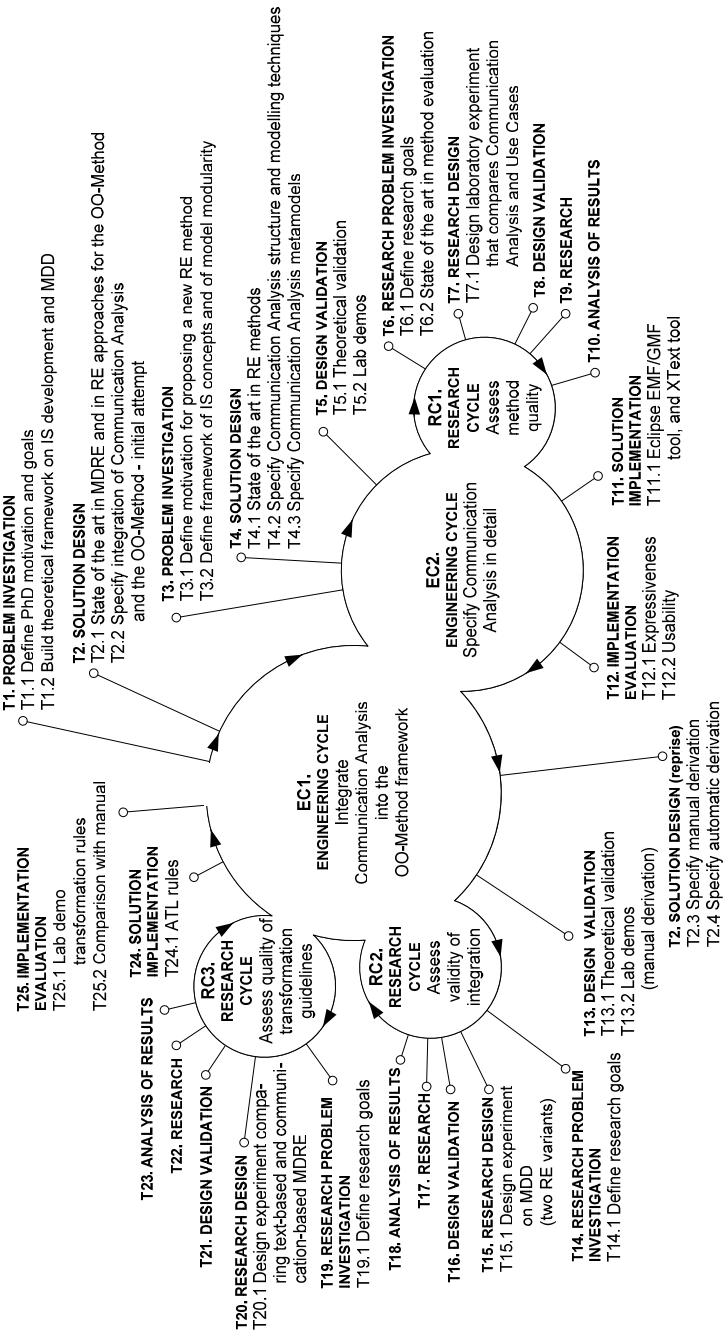


Figure 6. Research structure in terms of regulative cycles

Communication Analysis is then validated in several ways: a theoretical argumentation of the validity of the approach (T5.1), several lab demos that illustrate the use of the method (T5.2), and a laboratory experiment that compares Communication Analysis and Use Cases with regards to the quality of the resulting requirements models (this requires a research cycle of its own; i.e., RC1).

Each research cycle is decomposed into the research problem investigation (e.g. the goals and hypotheses of the research are stated, and a review of similar empirical studies is typically carried out), the research design (in the case of an experiment it corresponds to the experimental design, the planning, and the construction of all the necessary experimentation instruments), the design validation (e.g. the analysis of the threats to the validity of the experiment, and changing the experimental design/planning to mitigate the threats), the research itself (e.g. the execution of the experiment) and the analysis of results (including the preliminary data preparation). We skip the details of the research cycle (i.e. tasks T6 to T10).

After validating the detailed specification of Communication Analysis, a tool that supports the method is implemented (T11.1). The Eclipse Modeling Framework is chosen as the technological environment since it is itself an MDD framework and it facilitates the integration with other existing MDD frameworks (including Integranova). The implementation is evaluated with regards to expressiveness and usability criteria (T12.1).

Once EC2 ends, it is possible to fully specify the integration of Communication Analysis and the OO-Method. The intention is both to give support to a human developer (e.g. an analyst) that manually derives the conceptual model from the requirements model (T2.3), and to support automatic model-to-model transformations (T2.4). The integration of both methods is validated in several ways: a theoretical argumentation of the validity of the integration (T13.1), a laboratory case study that illustrates the manual use of the integration (T13.2), a laboratory experiment that assesses whether the application of Communication Analysis requirements engineering before undertaking OO-Method conceptual modelling is worth the effort (this requires a research cycle of its own; that is, RC3), and a laboratory experiment compares two approaches to conceptual model derivation –one based on textual specifications and one based on Communication Analysis specifications– with regards to the quality of the resulting conceptual models (this also requires a research cycle of its own; that is, RC4). The details of these research cycles are omitted (i.e. tasks T14 to T23).

The integration is implemented in Eclipse as an ATL transformation (T24). Lastly the transformation is evaluated by applying it to the laboratory case study (T25.1) and then comparing the results of manual and the automatic derivations (T25.2).

1.5 Outline of the thesis

The present document is divided in five parts that group related chapters. Table 1 displays the structure of the thesis and relates chapters with research tasks and research questions[14].

- *Part I* describes the problem investigation. *Chapter 1* motivates the research of this thesis. *Chapter 2* discusses related work. *Chapter 3* presents three conceptual frameworks of information systems, model modularity and model-driven development that serve as a foundation for the rest of the research.
- *Part II* designs a solution to the main research goal of the thesis (see page 13). *Chapter 4* introduces a detailed specification of Communication Analysis that is also extended with the artefacts that are needed to integrate the method into an MDD framework (e.g. metamodels). *Chapter 5* introduces the integration of Communication Analysis and the OO-Method, considering both a manual derivation of the conceptual model and an automatic model-to-model transformation.
- *Part III* describes the research activities that have been carried out to validate the solutions proposed in Part II. *Chapter 6* illustrates how an RE method can be validated; specifically, the chapter presents (i) a theoretical validation of Communication Analysis, (ii) several lab demos that illustrate the use of the method, (iii) a controlled experiment that compares Communication Analysis and Use Cases with regards to the quality of the resulting requirements models. *Chapter 7* validates the integration of an RE method into an MDD framework; specifically, the chapter presents (i) a theoretical validation of the integration of Communication Analysis and the OO-Method, (ii) several lab demos that illustrate the manual derivation of a conceptual model, (iii) a controlled experiment that compares two approaches to conceptual model derivation (namely, a set of derivation guidelines that take a textual requirements specification as input, and a set of guidelines that take a Communication Analysis requirements specification as input) with regards to the quality of the resulting conceptual model, and (iii) a controlled experiment that attempts that to assess the effect that an early stage of business process modelling has in the outcome of a later stage of object-oriented conceptual modelling.

[14] The relations mean the following. Chapters report the results of research tasks. Chapters address research questions.

- *Part IV* describes the available tool support for the solutions proposed in Part II. *Chapter 8* describes the tools that support Communication Analysis requirements modelling, as well as the model ATL transformation that provides the automatic derivation of OO-Method conceptual models.
- *Part V* concludes the thesis. *Chapter 9* discusses the lessons learned and the conclusions drawn from the research. *Chapter 10* outlooks future work in the same research line.

Table 1. Structure of the thesis

	Regulative cycles and tasks					**Research questions**				
	EC1	EC2	RC1	RC3	RC4	RQ1	RQ2	RQ3	RQ4	RQ5
Part I. Problem investigation										
Chapter 1. Motivation	T1.1 T1.2									
Chapter 2. Related work	T2.1	T3.3				RQ1.1	RQ2.1 RQ2.2 RQ2.3			
Chapter 3. Theoretical framework		T3.2				RQ1.2 RQ1.3				
Part II. Solution design										
Chapter 4. Detailed specification of Communication Analysis		T3.1 T4.2 T4.3					RQ2.4 RQ2.5			
Chapter 5. Integration of Communication Analysis and the OO-Method	T2.2 T2.3 T2.4							RQ3.1 RQ3.2		
Part III. Design validation										
Chapter 6. Validation of Communication Analysis		T5.1 T5.2	T6.1 T6.2 T7.1 T8 T9 T10						RQ4.1 RQ4.2	
Chapter 7. Validation of the conceptual model derivation technique	T13.1 T13.2			T14.1 T14.2 T15 T16 T17 T18	T19.1 T19.2 T20 T21 T22 T23					RQ5.1 RQ5.2
Part IV. Solution implementation										
Chapter 8. Technological support	T24.1 T25.1 T25.2	T11.1 T12.1 T12.2						RQ3.1 RQ3.2	RQ4.1 RQ4.2	
Part V. Conclusion										
Chapter 9. Final discussion										
Chapter 10. Outlook										

1.6 Philosophical stance

Philosophy is, in general terms, a methodical, systematic and rational reflection that aims to comprehend and express the nature of reality and existence, the use and limits of knowledge and the principles that govern and influence moral judgment. Philosophy is the reaction of human beings to the infinite curiosity that arouses the wonders of the world that surrounds us, the cosmos. This curiosity leads us to ask ourselves questions that we try to answer using reason. Philosophy and science have in common the use of reason, what distinguishes them from religion, which is ultimately based on faith. Philosophy differs from science in that the latter focuses on concrete aspects of the being and it has to respect the limits imposed by the scientific method. Philosophy can critique and reason about science itself and the scientific method. The *philosophy of science* is the investigation of the general nature of scientific practice. It studies assumptions, foundations, objectives and implications of science.

There exists a philosophical issue that is debated since the very origins of written philosophy [Plato 2000, (written ca. 380 BC)], which is coined the *problem of universals* [Klima 2008]. In brief, a *universal* is an abstract entity; i.e. a type of concrete things (e.g. humanity), a property (strong) or a relation (father of). On the other hand, a *particular* is a concrete entity that only has one instance (e.g. Socrates).

The debate was resumed with great interest during the Middle Ages and it confronted nominalists and conceptualists. In the face of the problem of universals, a philosophical and epistemological perspective named constructivism grew in the middle of the 21st century. *Constructivism* offers a perspective of knowledge based on mediation: knowledge is constructed by humans (e.g. scientists) and not discovered from the world [Piaget 1967]. Constructivists claim that universals (and the concepts of science, in general) are mental constructs proposed and agreed in order to explain our sensory experience. Therefore, reality is a human construction[15].

Constructivism does not accept the pre-existence of universals. Constructivists accept the existence of an outside world of perceptible things, but they also consider that the outside world is only apprehended through the senses and it is only comprehensible by means of modelling (see Figure 7). By means of abstracting and applying patterns to perceptions, each person creates a mental model which acts as a subjective reality that conforms to those perceptions.

[15] The world outside our minds indeed exists and it contains perceptible things (particulars), but we cannot reason about them but through the concepts that conform the constructed reality.

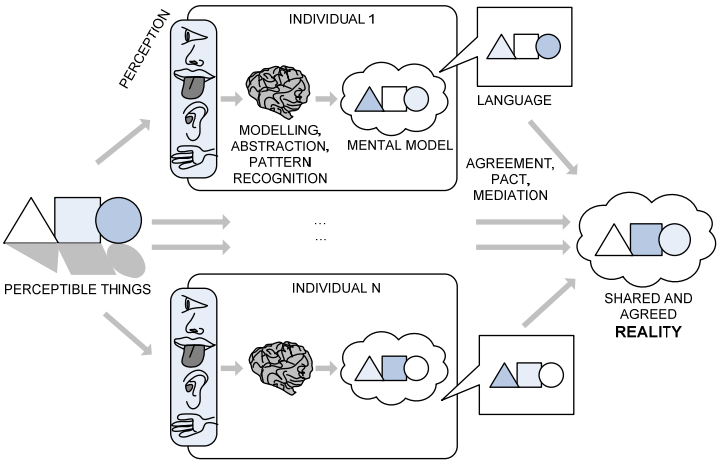


Figure 7. Reality construction according to constructivism

Language allows us to share our mental models, to compare and discuss them with others, therefore participating in agreeing a shared reality. In other words, individual experiences construct a subjective reality, by sharing those experiences humans find out commonalities and construct an inter-subjective reality, finally reaching agreement on the categories of things and the frontiers of individuals. This agreement gives rise to a shared model that constructivist consider to be reality.

One of the issues that have generated strong debate in the philosophy of science is determining which the objective of science is and how scientific results ought to be interpreted. In the face of this issue, scientific realism and instrumentalism hold contrasting positions. For scientific realism, the view of the world described by science (its concepts and theories) "is" the real world; that is, one ought to regard scientific theories as true or approximately true [Leplin 1984, p. 1]. For instrumentalism, concepts and theories are simply useful instruments whose value does not rely of its truth but on its effectiveness in explaining and predicting phenomena; it is in terms of usefulness rather than correctness that the theories should be judged [Duhem 1954]. I consider instrumentalism to be compatible with constructivism, since the concepts (and the theories in which concepts are used) gain agreement as they are more widely accepted by the scientific community (usually because they succeed in explaining and predicting phenomena observed by other scientists).

From a philosophical and epistemological point of view, **I declare myself constructivist** and close to the view of scientific theories held by instrumentalists. The consequence of this stance is that definitions made in this work (i.e. the concepts and theories) intend to be the result of a sensible and shared reasoning, and intend to be useful in explaining and predicting the world. This way, far from considering them as an ontological dogma, I am always open to debate about definitions. The aim is to offer useful definitions to reason about information systems understanding and development (our universe of discourse). Similarly, the proposals made in this work (i.e. techniques, notations and guidelines) intend to serve a purpose: facilitating the process of producing computerised information sub-systems of good quality. Therefore, I must take into account aspects such as conformance to theory (e.g. the modelling primitives in modelling techniques must represent well defined and agreed concepts), and adequacy to human factors (e.g. modelling primitives must be easy to manipulate by developers, both cognitively and instrumentally, resulting models must be easy to communicate to and understandable by users).

Chapter 2
Related work

"Those who cannot remember the past are condemned to repeat it."

George Santayana

2.1 Motivation

According to the design-science guidelines in [Wieringa 2009], the second step of the engineering regulative cycle is to design a solution to a practical problem. However, it may occur that the researcher[16] does not have to invent something new but make an inventory of solutions available in the market or in the company, or described in engineering literature (that is a knowledge problem). Alternatively, the researcher can assemble a solution from known elements (that is a knowledge question -which elements are available- and a creative problem -how to assemble them-). This is why state-or-the-art studies are mandatory: researchers must ascertain that there exist no previous solutions to the problem at hand before proposing a new one.

During the research described in this thesis, the latter has been done. We first reviewed the literature to find out whether there was an existing model-driven requirements engineering method available for use. Section 2.3 describes the study of the state of the art in this topic. Since no solution was available, we opted to integrate Communication Analysis and the OO-Method. Sections 2.4 and 2.1 describe the state of the OO-Method and Communication Analysis at the time this thesis started, respectively. However, before starting research in a new field, certain preliminary knowledge is needed. With regard to this thesis, this preliminary knowledge is described in Section 2.2.

[16] In design science, researchers usually have to solve both knowledge and practical problems, so they actually play the two roles of researcher and engineer.

2.2 *Propaedeutic*

Propaedeutic is the body of preliminary knowledge needed to undertake a specific scientific activity, such as carrying out the research described in this thesis. Propaedeutic is even considered a stage that is previous to defining the scientific method to be applied in a research. In the case of proposing a method that facilitates the development of computerised information sub-systems, a multidisciplinary knowledge is convenient. In the following we provide a (non-detailed and non-exhaustive) list of the areas of knowledge that is necessary to study (with more or less depth) in order to carry out the research described in this thesis. Some terms are underlined in order to highlight the relationships among the areas of knowledge.

Semiotics

The languages to which we refer in Figure 7 are composed of signs (spoken or written words, graphical symbols, modelling primitives, etc.). *Signs* are the means that humans employ to share meanings (concepts, ideas, theories, emotions, models, etc.). Semiotics is the field of science that studies signs. Semiotics can even be considered the basic science that addresses how thought works, since it tries to answer the question of how humans know about the surrounding world, how we interpret it, how we create knowledge and how we share it. It is common to use geometrical figures to reason about the distinct aspects with which a sign is related (different authors have different conceptions of these aspects and their relations: Odgen, Saussure, Hjelmslev, Peirce…) [Braun, Hesse et al. 2000].

The FRISCO report [Falkenberg, Hesse et al. 1998] proposes a semiotic tetrahedron that merges the elements of Odgen's representational triangle and Pierce's semiotic triangle (see Figure 8). Representation is the physical form that the sign has (e.g. a sticky figure doodled in a paper). The *domain* or referent is the object to which the sign refers (e.g. a person); the representation substitutes the domain object in a model (e.g. in the drawing of a child). The *actor* of the conceptualisation process plays the role of systemic observer. The actor can indistinctly adopt the role of representer or interpreter. As a representer, the actor is capable of creating a representation from a domain object (the child observes his father and represents him with a sticky figure). As an interpreter, the actor is capable of interpreting the representation and evoking the domain object. The *conception* (a.k.a. concept) is the thought that takes shape in the mind of the actor during the conceptualisation process (e.g. the mental figure of the father in the mind of the child); the conception is the mental surrogate of the domain object and it is evoked by perceiving either the domain object or the representation.

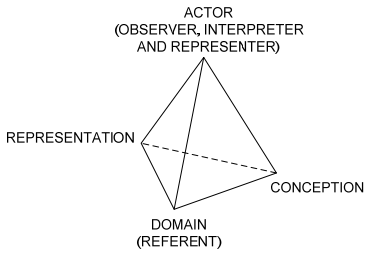


Figure 8. The semiotic tetrahedron according to FRISCO

Systems Theory

General Systems Theory is widely accepted as a development paradigm for theories related to complex phenomena. It is a transdisciplinary approach to systems research, aiming to find common properties of different kinds of systems that have been traditionally addressed by different scientific disciplines (e.g. ant colonies, electronic devices, galaxies, societies). In the following, we briefly describe several concepts from systems theory that are relevant to our research. An *observer* who is interested in conceiving of a series of phenomena as a system for a given purpose (e.g. increasing the understanding of the phenomena) marks out a portion of the world (encapsulates the system). Then the observer chooses a set of *modelling languages* to describe the system; that is, systems are reasoned and described by using signs. Systems are more than the sum of their components. Some properties of a system cannot be understood (or explained) by simply studying system components; these properties, which are called *emergent properties*, appear when the components of a system interact. The appropriate way of analyzing a system is by studying its *interaction* with its environment, by focusing on the system frontier. Only this way it is possible to apprehend the behaviour of a system.

Communication Theory

Communication Theory is the scientific field that studies the nature of communication and how it takes place. It is a transdisciplinary discipline whose theories often rely in Semiotics, Linguistics, Systems Theory, etc. Three theories are particularly relevant for our research: those of Shannon, Jakobson and Bunt.

Shannon and Weaver's Model of Communication

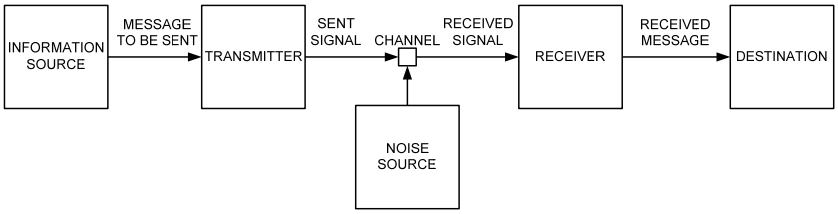


Figure 9. Schematic diagram of a generic communication system [Shannon 1948]

Shannon [1948] defines a model of communication that emphasises the role of coders (transmitters) and decoders (receivers) in *message* conveyance. Its mathematical theory implies a *shared code* between transmitter and receiver. Shannon defines a communication system as depicted in Figure 9. It consists of essentially five parts:

- An *information source* which produces a message or sequence of messages to be communicated to the receiving terminal. E.g. the message may be a single function of time `f(t)` as in radio or telephony.
- A *transmitter* which operates on the message in some way to produce a signal suitable for transmission over the channel. E.g. in telephony this operation consists merely of changing sound pressure into a proportional electrical current.
- The *channel* is merely the medium used to transmit the signal from transmitter to receiver. E.g. it may be a pair of wires, a coaxial cable, a band of radio frequencies, a beam of light.
- The *receiver* ordinarily performs the inverse operation of that done by the transmitter, reconstructing the message from the signal.
- The *destination* is the person (or thing) for whom the message is intended.

Additionally, he considers the possible existence of a *noise source* that perturbs the signal during transmission or at one or the other of the terminals. This means that the received signal is not necessarily the same as that sent out by the transmitter.

Jakobson's model of the communicative act

Jakobson [1990] proposes a model of the communicative act that takes into account several elements that were disregarded in Shannon's model (see Figure 10, elements appear in upper-case letters).

- *Addresser*. This is the one who sends the message. It is also referred to as sender.
- *Addressee*. This is the one who is expected to receive the message. It can also be absent or implicit, during the communicative act. It is also referred to as receiver.
- *Context*. It is the referent or subject matter of the discourse, what it refers to. The context includes the shared knowledge between the addresser and the addressee, which is crucial for reaching a common understanding.
- *Code*. It contains information about the lexical code with which the message is built. It is fully or partly known to addresser and addressee (see shared code in Shannon's model of communication). Codes need to be learnt.
- *Channel*. It refers to the channel or connection (physical or psychological) between the two parties. It is also known as contact.
- *Message*. It is the thing being conveyed, information being communicated.

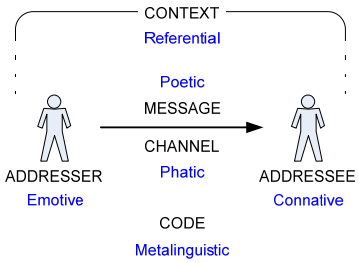


Figure 10. Adaptation of Jakobson's model of the communicative act

Jakobson defines six possible functions of language that are related to the six elements of the communicative act (in Figure 10, functions appear in lower-case).

- *Emotive* function. It is the direct expression of the speaker's attitude toward what s/he is speaking about. It is related to the addresser. The purpose of this function is to convey emotion. Also referred to as expressive function. An example: -What a beautiful voice!-
- *Connative* function. It is related to the addressee. The purpose of this function is to convey commands; that is, the aim is to (attempt to) transform reality or people, to affect the course of events or behaviour of individuals. Therefore, it is not liable to a truth test (it is neither true nor false; it is an order or a suggestion). Also referred to as vocative or imperative. -Hear her sing!-

- *Referential* function. It is related to the context. The purpose of this function is to convey information. In any case, human language is never completely neutral, so it is usually combined with other functions. Also referred to as informative function or denotative function. An example: -Mary Fahl is singing.-
- *Metalinguistic* function. It is related to the code. The purpose of this function is to convey code analysis. Language is capable of self-interpretation. Also referred to as metalingual function. An example: -This is the concert programme.-
- *Phatic* function. It is related to the contact. The purpose of this function concerns contact, both at the physical and psychological levels. It aims to establish, maintain contact, and ensure operation of the channel of communication. Also referred to as contact function. An example: -Hello. Nice voice, huh?-
- *Poetic* function. It is related to the message. The purpose of this function is to convey play or pleasure. It aims to play with form (e.g. rhyme, repetition, alliteration) or meaning (e.g. artful exploitation of synonyms, metaphors). An example: -She sings like an angel.-

A communicative act does not necessarily/usually have only one function. Nevertheless, it can be considered that, in each, communicative act, one function predominates.

Dynamic Interpretation Theory

Among the several communication theories that raise the abstraction to the level of discourse, we choose Dynamic Interpretation Theory because it defines concepts that are useful to build the information systems conceptual framework presented in Chapter 3. Also, this theory is compatible with an ISO standard for dialog annotation currently under development [ISO/DIS 2009].

A *dialog* can be considered a communicative interaction between agents that, by means of making communicative acts by turns, aim to achieve a (generally) non-communicative goal. Communicative goals are (generally) instrumental to the underlying non-communicative goal. The task of achieving the non-communicative goal that motivates a dialogue is referred to as the *underlying task* of the dialogue. The participants in a dialogue may each have their own underlying goal, which may overlap, coincide, conflict or be complementary [Bunt 1994]. Bunt focuses on *information dialogues*, which are dialogs that serve the purpose of exchanging certain specific information concerning some domain of discourse[17] [Bunt 1995]. For the sake of keeping research viable, it is convenient to simplify the goals involved in such dialogues: only the information-seeking

[17] We agree with Blunt in that this type of dialogue is the one that applies best to information systems research.

participant is considered to have an overall communicative goal, the information-providing participant is therefore characterised by a general cooperative attitude. Participating in a dialogue typically involves two tasks at the same time: the underlying task of achieving the non-communicative goal, and the task of communicating successfully in order to achieve the associated communicative goal.

Every dialogue occurs in a context. Bunt refers as *context* of a dialog to the factors that are relevant to understanding communicative behaviour, and groups these factors into five categories [Bunt 1995]: linguistic context, semantic context, cognitive context, physical and perceptual context, and social context. The notion of semantic context mostly coincides with Jakobson's notion of context, and it also includes shared knowledge.

The meaning of a communicative act is defined in terms of how the act changes the context. Bunt defines *dialogue act* as a communicative act that is used in a dialogue; more precisely, it is a functional unit used by the speaker to change the context (see [Bunt 1994] for an early taxonomy of functions[18]). A single utterance can include many dialogue acts, since each part of an utterance can have a different function. For instance, the utterance "Excuse me, sir. I am looking for my dad, John Smith… Do you know where his office is?" includes several dialog acts; "Excuse me, sir." has a greeting function, "I am looking for my dad, John Smith…" has an information-providing function, and "Do you know where his office is?" has an information-seeking function. Bunt distinguishes between two types of dialogue acts (based on two types of functions):

- *Dialogue control acts* are dialogue acts that are motivated by the communicative task and controlling aspects that require attention in communication more generally. For instance, monitoring attention, ensuring the correctness of understanding, taking turns, repairing communicative failures, following certain social rules (e.g. greetings, apologies, acknowledgements, farewells), etc. Note that the functions that relate to dialogue control include the Jakobson's phatic function. Dialogue control acts do not change the semantic context.
- *Task-oriented acts* are dialogue acts that are directly motivated by the underlying task and contribute to its accomplishment. For instance, questions, answers, checks and confirmations. Task-oriented acts do change the semantic context because they provide or request facts that are closely related to the underlying task.

[18] At the time of writing this thesis, there is an updated taxonomy that includes function definitions available at http://dit.uvt.nl

Information Systems Theory

Information systems are socio-technical systems that aim to support organizational communications [Langefors 1977; Lockemann and Mayr 1986]. Being a type of systems, a sensible approach to information system analysis is to study its interaction with the environment (see Systems Theory above).

Many theories and conceptual frameworks have been developed on the area of information systems. For instance, the Bunge-Wand-Weber ontological model considers the information system as a mere representation of a domain, ignoring any social aspects [Wand and Weber 1990]. It provides a formal approach upon which many works have been built. However, we are interested in interdisciplinary theories that take into account social and communicational aspects.

ISO technical report 9007:1987

This report [ISO 1987] presents a conceptual framework for information systems, defining a basic interaction architecture (see Figure 11). According to this report, every information system reacts to messages depending on different objectives. The objective of the information system is to memorise reported facts[19]. The main objective of the organisation is to react to events. The ISO report makes the following definitions that are relevant to our research:

- The *universe of discourse* is the collection of all objects (entities) that ever have been, are, or ever will be in a selected portion of a real world or postulated world of interest that is being described.
- The *environment* is a part of the real world which may be disjoint from, or overlap with, the universe of discourse. The environment originates messages.
    - A *message* is a collection of sentences and/or command statements to be used as an information exchange between the environment and the information system.

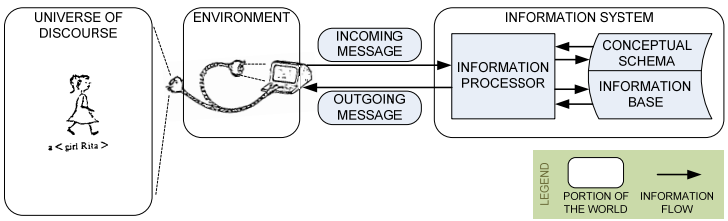


Figure 11. Information system model, adapted from [ISO 1987]

[19] The authors of [ISO 1987] acknowledge that they adopt a naive realist approach to information systems.

- An *information system* consists of a conceptual schema, an information base, and an information processor.
  - An *information processor* is a mechanism that in response to a command executes an action on (i.e. produces a change in or retrieves information from) the conceptual schema and the information base.
- The *conceptual schema* is the description of the possible states of affairs of the universe of discourse including the classifications, rules, laws, etc., of the universe of discourse. The conceptual schema is obtained by applying perceiving the universe of discourse and conceptualising the perceptions (e.g. applying classification, abstraction, etc.).
- The *information base* is the description of the specific objects (entities) that in a specific instant or period in time are perceived to exist in the universe of discourse and their actual states of affairs that are of interest. The information base is obtained by recording facts and happenings about the universe of discourse.

The report considers the information system to be formal and fully predictable, and excludes its users from belonging to it (they are said to belong to the environment). From a socio-technical perspective, however, information systems also include people.

The FRISCO report

The FRISCO report [Falkenberg, Hesse et al. 1998] contributed an essential a foundation stone on the subject of information systems. In the midst of terminological fuzziness, FRISCO adopted a constructivist stance and proposed a set of definitions that addresses the underlying concepts behind any information systems from various semiotic angles. It proposes a rigorous framework of information system concepts.

The FRISCO report proposes distinguishing the following systems:

- An *organisational system* is a special kind of system, being normally dynamic, active and open, and comprising the conception of how an organisation is composed (i.e. of specific actors and actands) and how it operates (i.e. performing specific actions in pursuit of organisational goals, guided by organisational rules and informed by internal and external communication), where its systemic properties are that it responds to (certain kinds of) changes caused by the system environment and, itself, causes (certain kinds of) changes in the system environment.

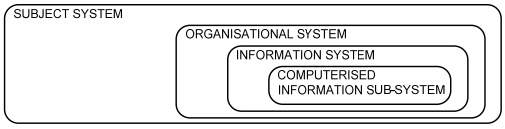


Figure 12. The systems involved in the information systems discipline

- An *information system* is a sub-system of an organisational system, comprising the conception of how the communication- and information–oriented aspects of an organisation are composed (e.g. of specific communicating, information-providing and/or information-seeking actors, and of specific information-oriented actands) and how these operate, thus describing the (explicit and/or implicit) communication-oriented and information–providing actions and arrangements existing within that organisation.
- A *computerised information sub-system* is a sub-system of an information system, whereby all actions within that sub-system are performed by one or several computer(s).

We add the following definition (we will further elaborate on it in Chapter 3):

- A *subject system* is a portion of the world in which an organisational system is interested; that is, to achieve the organisational goals, the subject system needs to be observed, influenced or controlled.

These systems can be complexly imbricated; Figure 12 simplifies the imbrications as a subsumption.

FRISCO modelled the dialog occurring in an interaction. In the FRISCO model of the information system (see Figure 13), *users* are actors that specify *input messages* to the processor and receive *output messages* from it. *Processors* are actors that are responsible for checking input messages, keeping the *domain model denotation* (i.e. the information base as defined in [ISO 1987]) logically consistent with the *language representation* (i.e. the conceptual schema as defined in [ISO 1987]), and retrieving information to produce output messages; in short, a processor is responsible for the information system reaction to external stimuli.

Although this model is said to refer to computerised information sub-systems, the same concepts can apply to an information system with no computerised actors (e.g. a human and paper-based information system).

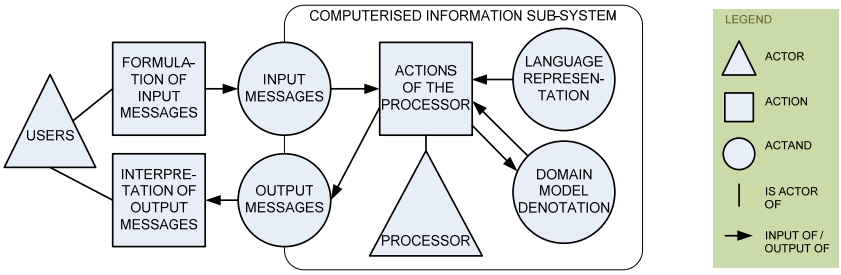


Figure 13. Model of a computerised information sub-system, according to the FRISCO report

Wicked Problems

Rittel and Webber [1973] enunciated the characteristics of a type of very complex problems that they coined **wicked problems**. Wicked problems are often not fully understood until a solution has been found, since every wicked problem is essentially unique and novel. Solutions to these problems are not right or wrong, simply "better," "worse," "good enough," or "not good enough". Indeed, it is the social complexity of these problems, not their technical complexity, which overwhelms most current problem-solving approaches. To appropriately deal with wicked problems, it is common to carry out *opportunity driven problem solving*. At any given time the developers are seeking the best opportunity to progress toward a solution, regardless of whether they are going up or down in the abstraction ladder. Rittel and Webber identified a type of problematic situation that can only be solved if representatives of all the stakeholders participate in a joint effort. Since the participants must be totally committed to solving the problem, much communication, willingness and synergy are needed.

Wicked problems need contingent approaches; that is, the problem solver needs to tackle the problem from distinct angles, using different problem solving techniques. The application of the contingency principle to the organisational area was introduced by Fiedler [1964]. In a time when many scholars assumed that there existed only one good leadership style, Fiedler proposed a model of contingency that defined leadership effectiveness depending on the interaction among several factors. This gave rise to a field of studies that tried to define the adequacy of a strategy to a situation.

Software Engineering and model-driven development

Software Engineering is the discipline devoted to formulating theories and proposing methods related to the software lifecycle (e.g. software development methods) [Naur and Randell 1969]. The main objective of Software Engineering is to obtain good quality products. In this context, quality is understood as the ability of the final software product to meet the users and clients needs and expectations.

Information systems development and computerisation is a wicked problem. To a large extent, this is due to their socio-technical nature [Lockemann and Mayr 1986] and to the intervention of multiple stakeholders with often conflicting needs and world views. To overcome conflicts and to reach agreement, stakeholders' perceptions have to be pooled together (e.g. in a model) and shared with information system developers. *Requirements Engineering* (RE) facilitates this process by offering techniques for the discovery and description of stakeholders' needs. Modelling allows raising the abstraction level of information system descriptions and, therefore, it is part of the current practice of information system analysts and requirements engineers.

Moreover, software development has particularities depending on the application context. This way, the development of a computerised information sub-system for enterprise management cannot be tackled the same way as the development of an industrial control computerised sub-system; the applied techniques need to be appropriate to the problem at hand (as the contingency principle states).

Method Engineering

During the 90s, the scientific community realised that organisations did not enact methods just as they are defined in method descriptions (e.g. books, manuals, methodological guidelines). Methods were used more laxly, they were adapted [Russo, Wynekoop et al. 1995]. This recognition started a trend to describe methods by means of fragments that can be assembled to fit a specific project or organisation. This was called method engineering, a discipline whose concepts and metamodel are specified by the ISO/IEC 24744 report [ISO/IEC 2007b].

Method engineering offers a structured framework in which methods for a given purpose can be designed, constructed, and adapted (in this thesis we are interested in software development methods).Methods are created by assembling several individually identifiable parts commonly referred as *method fragments*, method chunks or method components [Henderson-Sellers, González-Perez et al. 2007; Henderson-Sellers, Gonzalez-Perez et al. 2008]. Method fragments are defined as the instantiation of a metamodel and they are stored in a repository [ISO/IEC 2007b]. A method engineer (a.k.a. method designer) assembles several method fragments with the purpose of creating a method[20]. Then, project managers can instantiate the method in order to enact the development process.

From this perspective, the work presented in this thesis is related to method engineering because it integrates Communication Analysis and the OO-Method.

The consequence of the tangle of cognitive, social and abstraction-level issues that are involved in information systems development is that no single ingenuous solution is adequate. Methodologists need to offer software-development methods that facilitate opportunity-driven problem solving. An evident corollary is that it is important to deal with contingency in software development. Methods must be flexible in the sense that the techniques to be applied in each moment are determined by various factors: the characteristics of the system to be computerised (i.e., automatic teller machines are not dealt with the same way as information systems), the nature of a specific problem presented at a given

[20] Some authors, as well as the ISO/IECC 24744 report, use method and methodology as synonyms. However, in this work, we use the term method. Methodology is used to refer the study of methods or to qualify something as being related to methods (e.g. methodological guideline).

moment (e.g. to design a user interface, to specify business objects, to design strategic-level reports), the expertise of the development team (which techniques they know best), the maturity of the organisational system, the users' organisational and technological knowledge, the predisposition of the stakeholders to be involved in development, etc.

The application of the contingency principle to method engineering has lead to *situational method engineering*, a discipline that proposes to configure methods so as to adapt them to specific projects [Kumar and Welke 1992; Brinkkemper 1996; Ralyté, Brinkkemper et al. 2007]. This topic is especially relevant and useful in areas such as requirements engineering where various situation-specific factors, e.g. project objective, application domain, features of the product to be developed, stakeholders involved, and technological conditions and constraints, exert a significant influence [Ågerfalk and Ralyté 2006].

Ågerfalk and Ralyté have identified an initial and traditional assembly-based approach in method engineering [Ågerfalk and Ralyté 2006]. In their work the procedure has been used to describe methods and processes in metamodels to be used as a basis for computer supported instantiation of situational methods, typically through the assembly of a number of method fragments from different methods stored in a method base [Brinkkemper, Saeki et al. 1999; Lyytinen and Welke 1999]. Recently, however, method engineering research and practice have extended beyond the traditional assembly-based approach to address a variety of issues including method requirements specification [Gupta and Prakash 2001], method configuration [Karlsson and Ågerfalk 2004] and roadmap-driven approaches [Mirbel and Ralyté 2005]. Recent works also pay more attention to method rationale (i.e. the reasons behind and arguments for the method) and the tension between method-in-concept (as described in method handbooks) and method-in-action (as enacted in actual engineering practice) [Lundell and Lings 2004; Ågerfalk and Fitzgerald 2006; Karlsson and Wistrand 2006].

Contingency theory underlies Situational Method Engineering. For instance, Mirbel and Ralyté propose to characterise the project situation and the project engineer's profile using several reuse frames. A "reuse frame contains the knowledge about the reuse context of […] method chunks and provides criteria for project and project engineer situation characterization" [Mirbel and Ralyté 2005].

2.3 *Model-driven requirements engineering*

Model-driven requirements engineering is a relatively novel approach that has started once model-driven conceptual modelling has been proved successful.

The systematic review by Loniewski, Insfrán and Abrahão [2010] answers the following research question: what requirements engineering techniques have been employed in model-driven development approaches and what is their actual level of automation. The authors follow the guidelines proposed in [Kitchenham 2004] for performing systematic reviews. A total of 72 papers are selected for data extraction and synthesis.

2.3.1 Descriptive analysis of results

The results are summarised in Table 2 and discussed in the following. Criteria are highlighted in bold, their possible values or levels appear in italics.

2.3.1.1 Point of view of the modelling language

With regards to the **requrirements type** (i.e. the systemic level), 60% of the papers focus on *software requirements*; i.e. for the computerised information sub-system. Only 40% of the papers include so-called *business requirements*; i.e. statements that are not only related to the functionality of the intended software but also address aspects of the subject, the organisational or the information systems (e.g. business context, structure of the enterprise, business processes). Many works that take into account business requirements are based on the i* Framework [Yu and Mylopoulos 1994] (e.g. [Mazón, Trujillo et al. 2007]). Others generate UML models from business requirements models (e.g. [Raj, Prabhakar et al. 2008]).

With regards to the **requirements structure** (i.e. modelling language), 64% of the papers propose or describe the use of graphical languages, half of which use *standard models* (i.e. UML, being the most frequent Class Diagram, Use Case Diagram, Activity Diagram and Sequence Diagram) and the other half use *non-standard models* (i.e. non-UML, such as goal-oriented or aspect-oriented). The rest of the papers use non-graphical modelling languages: 22 % of the papers use *structured natural language*, 10% use unstructured *natural language*, 1% use *templates*. The remaining 3% use other types of modelling languages. No paper is reported to combine graphical and non-graphical modelling languages.

With regards to the **type of models**, of the papers using graphical modelling languages, 69% of them use so-called *behavioural* modelling languages (e.g. Use Case Diagram, i*), 15% use *structural* modelling languages (e.g. Class Diagram),

10% use *functional* modelling languages (e.g. Activity Diagram), and 8% use *other* types of modelling languages (e.g. Requirements-Design Coupling).

Table 2. Results from the systematic review in [Loniewski, Insfran et al. 2010]

Selection criteria		**Total**	**%**
Requirements type	Software	43	60
	Business	29	40
Requirements structure	Standard model	25	32
	Non-standard model	25	32
	Template	1	1
	Structured natural language	17	22
	Natural language	8	10
	Other	2	3
Type of models	Structural	9	15
	Behavioural	39	69
	Functional	6	10
	Other	5	8
Transformations provided	Yes	58	81
	No	14	19
Transformations level	Endogenous	9	15
	Exogenous	52	85
Standard transformations	Yes	7	13
	No	46	87
Transformations automation	Automatic	27	45
	Interactive	11	18
	Manual	22	37
Traceability requirements	To analysis	20	24
	To design	12	14
	To implementation	12	14
	None	41	48
Traceability automation	Automatic	17	59
	Manual	12	41
Tool support	Traceability only	9	13
	Transformation only	23	32
	Traceability and transformations	2	3
	None	37	52
Type of validation	Survey	1	1
	Case study	32	45
	Experiment	2	3
	None	37	51
Approach type	Academic	53	73
	Industry	20	27

2.3.1.2 Point of view of model transformation

With regards to model **transformations provided**, 81% of the papers reviewed describe different types of transformations from requirements specifications. The authors distinguish several types of transformations (this is discussed below): mappings, pattern- and template-based transformations, transformations implemented in model transformation languages (e.g. QVT, ATL), linguistic operations on textual requirements, graph transformations, etc. 19% of the papers do not provide model transformations and focus on other MDD aspects (e.g. traceability).

With regards to model **transformation level**, most of the papers that address model transformations (85%) describe *exogenous* model transformations; i.e. transformations in which both the source and the target models are represented using the same modelling language and they correspond to the same abstraction level (according to [Mens, Czarnecki et al. 2005]). The systematic review indicates that many of the papers that describe exogenous model transformations start with business requirements specification (e.g. [Debnath, Leonardi et al. 2008], [Kherraf, Lefebvre et al. 2008]). The rest of the papers (15%) describe *endogenous* model transformations; i.e. transformations in which either the source and target models are expressed in different modelling languages, or the models have different abstraction levels (e.g. [Laguna and Gonzalez-Baixauli 2005]).

With regards to the use of **standard transformation** languages, only 13% of the papers represent model transformations with ATL and QVT (e.g. [Koch, Zhang et al. 2006]), all the rest (87%) use non-standard model transformation languages in the form of ad-hoc mappings, pattern-based transformations, etc. (e.g. [Raj, Prabhakar et al. 2008]).

With regards to the **transformations automation**, 45% of the papers describe an *automatic* model transformation (i.e. one that can be performed by a transformation engine without human intervention, e.g. [Zhang and Jiang 2008]), 19% of the papers describe an *interactive* or semi-automatic transformation (e.g. [Machado, Fernandes et al. 2005]), and the remaining 38% describe *manual* model transformations (e.g. [Laguna and Gonzalez-Baixauli 2005]).

With regards to the degree of automation, only 35% of the papers claim to provide tool support for model transformations; that is 32% provide **tool support** for *transformation only* and 3% provide tool support for both *traceability and transformations*.

2.3.1.3 Point of view of traceability

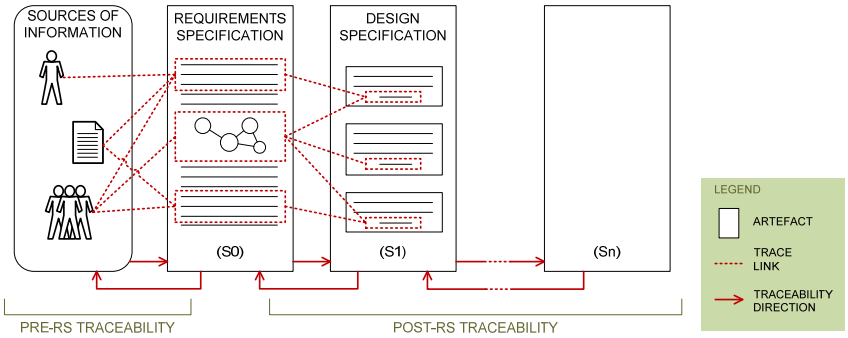


Figure 14. Pre- and post-RS traceability (adapted [Gotel and Finkelstein 1994])

[Gotel and Finkelstein 1994] proposes distinguishing two types of requirements traceability (see Figure 14):

- *Pre-RS traceability* is concerned with those aspects of a requirement's life prior to its inclusion in the requirements specification (RS).
- *Post-RS traceability* is concerned with those aspects of a requirement's life that result from its inclusion in the RS.

Furthermore, from a different perspective, the following types of traceability can be distinguished:

- *Forward traceability* establishes maps from an element of a model to an element of another model that appears later on in the development process.
- *Backward traceability* establishes maps that have the opposite direction, from an element of a model to an element of another model that appeared earlier.

The systematic review focuses on post-RS traceability, since it deals with tracing requirements to elements of later analysis and design models. The majority of the reviewed papers that describe model transformations assume that forward traceability exists, but not all approaches provide explicit mechanisms to support traceability.

With regards to the target of trace links (**traceability requirements**), the systematic review indicates that 24% of papers provide backward traceability from requirements to so-called *analysis* models, 14% to *design* models, 14% to *implementation* artefacts, and 48% do not provide traceability at all (*none*).

With regards to the degree of automation, only 16% of the papers claim to offer **tool support** for traceability generation (13% *traceability only*, 3% both *traceability and transformations*). However, once the requirements traceability has been established (either manually or automatically), 59% of the papers provide *automatic* support for managing the traceability (**traceability automation**).

2.3.1.4 Point of view of validation and applicability

Approximately half of the papers do not present any **type of validation** (i.e. 51% *none*). Instead, they often provide an example with illustration purposes that proves the feasibility of the approach; papers without validation often come from academy. The systematic review indicates that 45% of the papers present *case studies* with various levels of detail; these papers mainly come from the industry. 3% of the papers describe *controlled experiments*. 1% of the papers describe a validation based on *surveys*.

With regards to the **approach type**, only 27% of the papers provide evidences that the approach has ever been used in industrial settings (*industry* implies that the approach has been applied in real development projects, e.g. [Escalona, Gutiérrez et al. 2009]), whereas the rest (73%) can be considered an *academic* proposal that has not yet been applied in conditions of practice.

2.3.2 Perceived threats to the validity of the results

The systematic review by Loniewski, Insfrán and Abrahão [2010] provides a good overview of the state of the art in model-driven requirements engineering and it shows that there is space for improvement in the area. However, we have noticed some issues related to the data extraction protocol with the potential to threat the validity of some results.

The main concern is related to the fact the literature of software engineering is full of fuzzy and polysemic terms. This way, the authors of the reviewed articles sometimes make claims that may be inaccurate, if not completely wrong. However the procedures for performing systematic reviews do not prescribe that the reviewers have to dispute the statements found in the articles; unless the reviewers are extremely cautious and contrast the statements with other statements or illustrative examples within the reviewed article, the results can be flawed.

For instance, with regards to the **requirements type**, the reviewers consider that [Raj, Prabhakar et al. 2008] deals with *business requirements* probably because the paper actually claims that the Semantics of Business Vocabulary and Rules (SBVR) framework corresponds to the CIM level of the MDA paradigm. However, in the article, this modelling language is not only applied to specify computation-independent requirements but also to specify the requirements of computerised technology in a way that actually corresponds to the PIM level. For instance, the statement "customer owns account" does indeed describe a computation-independent requirement, but the statements "it is necessary that each atm [automatic teller machine] card has exactly one pin [personal identification number]" and "it is obligatory that each atm display exactly one

main-screen" are actually describing a computation-dependent requirement, in a platform independent way (these statements are drawn from examples found in [Raj, Prabhakar et al. 2008]). Therefore this article deals with *software requirements* as well. Thus, the percentage of articles describing exogenous model transformations that start with specifications exclusively containing *business requirements* could actually be lower than 40%.

The guidelines related to the criterion **traceability requirements** are arguable; especially, the distinction between analysis and design models. This is not surprising since it is a thorny issue [Génova, Valiente et al. 2009]. The following argumentation reflects the view of the author of this thesis. As an example, the authors of the systematic review consider [Lapouchnian, Yu et al. 2006] to address analysis models. However, the paper describes model transformations from goal models to statecharts and feature models. The generated statecharts describe "the behavioural variability of the system-to-be, for each goal the software system is responsible for a state that represents that goal being achieved by the system is introduced" [Lapouchnian, Yu et al. 2006]. Therefore, the so-called analysis model is actually a design model. In fact, the paper claims to shift "from goal models to high-variability software designs." Consequently, the percentage of articles that provide backward traceability from analysis models to requirements could actually be lower than 24%.

With regards to the **type of validation**, the figures could also be different than reported in the review. The term "case study" appears frequently in software engineering research papers. However, the presented studies range from very ambitious and well organised field studies, to small toy examples that claim to be case studies [Runeson and Höst 2009]. We strongly believe that the actual number of case studies in model-driven requirements engineering is much lower that reported in the systematic review. As an example, Santos, Moreira et al [2008] presents a running example that is referred as "case study", there is no evidence that the proposed technique has actually been used under conditions of practice. The extended material describing the validation (cited as online material in [Santos, Moreira et al. 2008, reference [17]]) is not available anymore at the time of writing this thesis. In any case, it seems that the validation consisted of a lab demo in which the technique is used by one of the researchers in a somehow realistic example, but lacking real conditions of practice. Even if conditions of practice existed (i.e. the development project was carried out in an industrial setting), the validation would actually be "action research" and not a case study. Consequently, the percentage of articles that report an actual case study is probably much lower than 45%.

2.3.3 Conclusions

The results of the systematic review by Loniewski, Insfran et al. [Loniewski, Insfran et al. 2010] prove that model-driven requirements engineering is an active research topic with several open research challenges.

- There is a need for technological support for model transformations; that is, to provide automatic or interactive transformations and to implement the corresponding tools.
- There is a need for better technological support for requirements traceability; that is, to provide guidelines for forward and backward post-RS traceability and to implement the corresponding tools.
- There is a need for empirical validations of model-driven requirements engineering proposals.

2.4 State of the OO-Method at the beginning of this thesis

The OO-Method was first proposed in the academy [Pastor, Insfrán et al. 1997; Pastor, Gómez et al. 2001], on top of a formal object-oriented algebra named OASIS [Pastor, Hayes et al. 1992]. Soon a spin-off company named CARE Technologies was created, with the aim of creating a CASE tool to support the OO-Method. The result of this endeavour was the Integranova model-driven development framework (Figure 15 shows a snapshot of Integranova Modeler, the modelling tool offered by the framework). At the time of starting this thesis, Integranova was a full-fledged CASE tool and CARE Technologies client list included international organisations that had some of their software developed by means of automatic software code generation.

The OO-Method conceptual model is comprised of four interrelated models:

- The *Object Model* allows specifying the static aspect of the system in the form of a UML-compliant class diagram. This model allows describing the following elements of the computerised information sub-system (among others): the business objects in terms of classes of objects, structural relations among classes of objects, agents of the system (an abstraction of the users of the software system) and relations among agents and class services.
- The *Dynamic Model* specifies the possible sequences of method invocations that can occur in the life of an object; it is expressed as a State-Transition Diagram.

- The *Functional Model* specifies the effect that the method invocations have in the state of the objects; it is expressed as generic pseudo-code specifications (which is independent of any programming language).
- The *Presentation Model* offers an abstract description of the computerised information sub-system interface. This model is structured in three abstract pattern levels.

As other object-oriented modelling methods, the OO-Method follows an ontological approach to information systems analysis; the focus in put on the objects that are perceived in the subject system.

Without entering into detail, the analyst takes the following steps during the conceptual modelling stage. First, the static aspect of the system is specified in the Object Model; the analyst identifies classes of objects, attributes and relationships among the classes of objects. Then, the analyst defines the methods that correspond to each class of objects. The order in which methods can be invoked is specified in the Dynamic Model and their reaction is specified in the Functional Model. Lastly, the interface is designed using the Presentation Model.

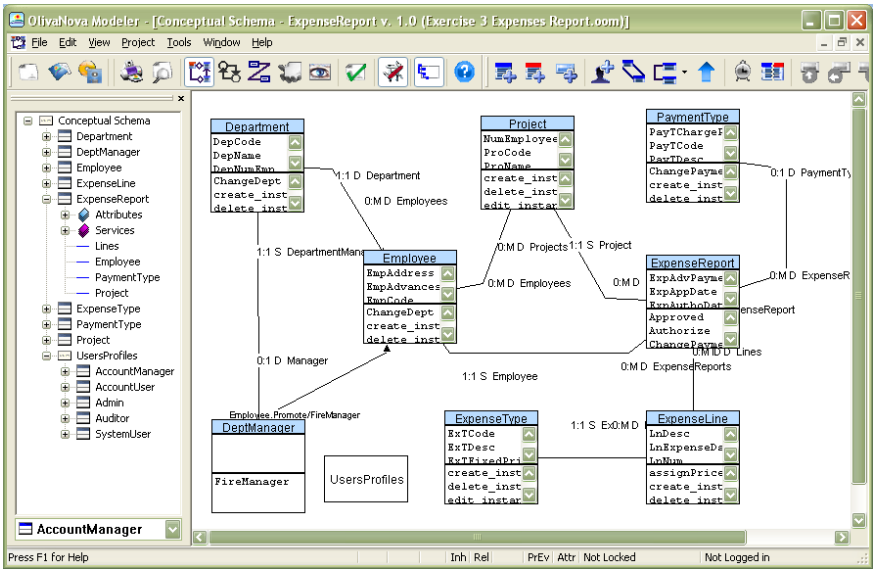


Figure 15. Snapshot of the Integranova Modeler tool for the specification of OO-Method conceptual models[21]

[21] The model is borrowed from the training material used by CARE Technologies, http://www.care-t.com, and Integranova, http://www.integranova.es.

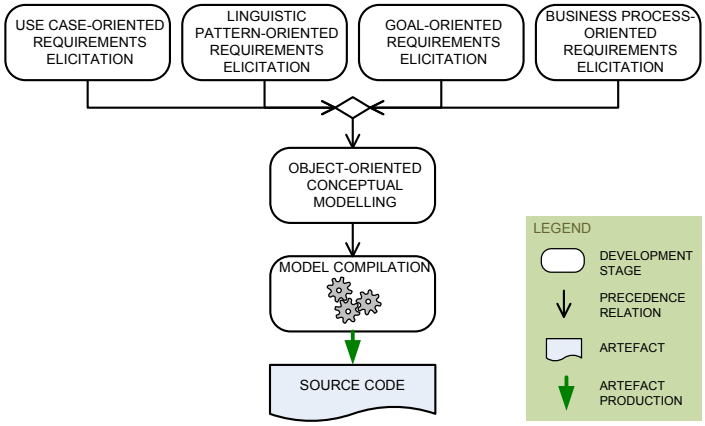


Figure 16. Several approaches to requirements engineering in the OO-Method

Several proposals build upon the methodological core of the OO-Method. Some of these proposals extend the method with modelling techniques aimed at a specific type of software system. It is the case of OOWS [Fons, Pelechano et al. 2003] for web applications, and PervML [Muñoz and Pelechano 2005] for pervasive systems. Some proposals in the area of requirements engineering offer techniques to capture requirements following different orientations. We now offer a brief description of these proposals (see Figure 16).

Use case-oriented requirements engineering

The proposal defines a requirements model that describes *what* the computerised information sub-system has to do, and a requirements analysis process that offers methodological guidance to derive a conceptual model [Insfrán, Pastor et al. 2002].

The proposal consists of the following steps:

- A *Requirements Model* is created, which (see Figure 17) is a model of knowledge shared with the users that combines three complementary techniques:
  - *Mission Statement*: it defines very concisely the purpose of the software that is under construction. It also defines the most significant responsibilities and defines the design charter (i.e. the scope of the project).
  - *Function Refinement Tree*: it decomposes information system functions until the leafs of the tree define use cases.
  - *Use Case Diagram*: it specifies the relation with the stakeholders. Each use case is described in detail by means of a template that specifies the composition of steps that the actor performs, as well as the software reaction.

- The *Requirements Analysis Process* is a strategy to describe concrete details such as the composition of the computerised information sub-system.
  - First the use cases are refined by means of *Sequence Diagrams* that describe the reaction of the software to external stimuli within a use case.
  - The identification of software components and the creation of *Function Decomposition Tables* help to derive a first version of the Object Model.

A tool named RETO supports this proposal [Insfrán 2004]. More details about this approach can be found in [Insfrán 2003].

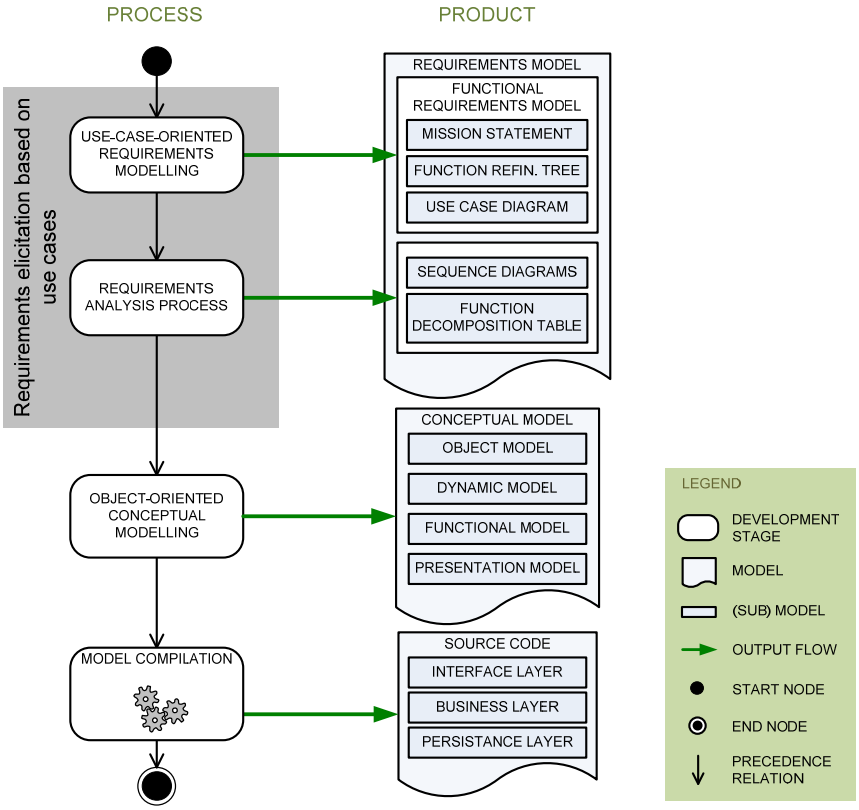


Figure 17. OO-Method extended with use case-oriented requirements engineering

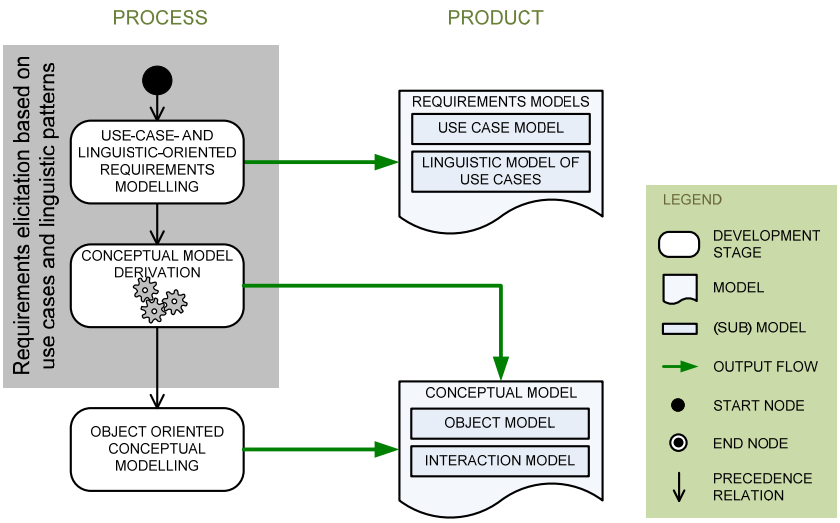


Figure 18. OO-Method extended with linguistic-pattern-based requirements engineering

A natural step after defining a requirements engineering approach based on use cases is the definition of a restricted natural language for the textual templates [Díaz, Losavio et al. 2004]. By means of the model transformation rules described in [Díaz, Sánchez et al. 2005], the use case templates can be processed to derive an initial version of the OO-Method conceptual model. The proposal consists of the following steps (see Figure 18):

- The requirements models are created.
  - First, the analyst creates a *Use Case Model*, which includes a use case diagram, as well as a set of templates that specify the use cases in detail.
  - Then, the use case templates are normalised using a set of transformation rules that are based on the CREWS style and content guidelines. The result is the *Linguistic Model of Use Cases*.
- A set of transformation patterns is applied to the Linguistic Model of Use Cases so as to derive an initial version of the conceptual model.
  - The *Object Model* is the structural description of the computerised information sub-system, as explained above.
  - The *Interaction Model* uses the UML Sequence Diagram modelling language to specify how objects interact to provide the intended functionality.

Note that this methodological extension proposes a different set of views for the OO-Method Conceptual Model. Thus, it is not possible to use the existing industrial tools to generate the software code automatically. In any case, it is feasible to complement the object model with the other three views so as to apply model compilation.

Goal-oriented requirements engineering

[Estrada, Martínez et al. 2003] proposes creating business models based on the i* framework. The business models are later used to derive a software requirements specification based on use cases and scenarios (see Figure 19).

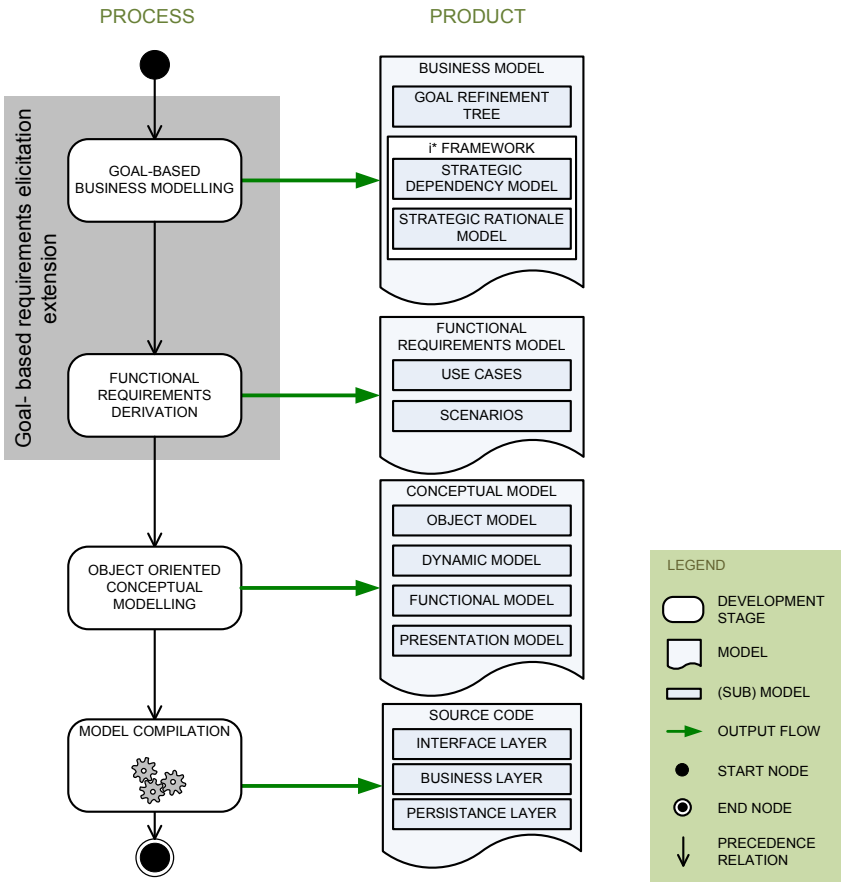


Figure 19. OO-Method extended with goal-oriented business modelling

The proposal consists of the following steps:

- A *Business Model* is created.
  - A goal elicitation technique is used to create a *Goal-Refinement Tree*, which is a hierarchical structure that describes the organisational context from the point of view of organisational goals.
  - The Goal-Refinement Tree is used to create the two complementary i* models. The Strategic Dependency Model shows the dependencies that exist among actors of both the organisational system and the subject system in order to fulfil their goals, to carry out their tasks, and to provide or consume resources. This model is represented by means of a graph in which the vertices represent actors and the edges represent dependencies.
  - The *Strategic Rationale Model* allows a deeper reasoning on the reasons behind each dependency relationship. From the elements of the Strategic Dependency Model, the tasks needed to fulfil a goal are decomposed, and the different options to carry out a task are explicitly described.
- The strategic models are used to derive a functional specification.
  - The functional requirements model is based on *Use Cases*. First, the analyst identifies those tasks in the Strategic Rationale Model that are going to be computerised. Then those tasks are assigned to an actor that represents the computerised information sub-system. From this model, a set of use cases is derived, along with their correspondent actors.
  - Use cases are complemented with *Scenarios*, which are a detailed specification of the steps in which a use case is decomposed.

The authors of this proposal have empirically evaluated its usage in an industrial environment (real projects), detecting the strengths and weaknesses of the approach, and they enumerate some improvement issues [Estrada, Martínez et al. 2006].

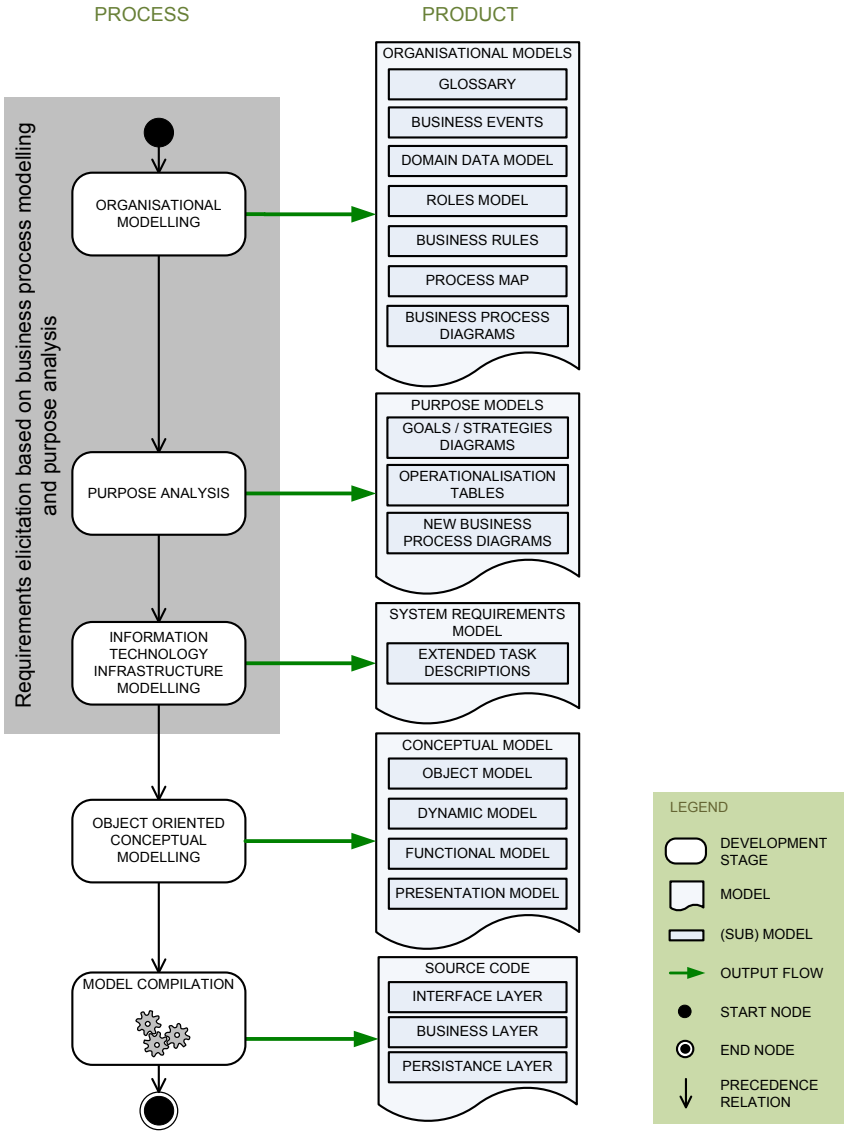


Figure 20. OO-Method extended with business process-oriented and goal-oriented requirements engineering

This extension proposes several complementary models aimed at describing the organisation and its work practice (see Figure 20). Business process models are described using the BPMN notation [OMG 2006a]. Using the Map technique [Rolland 2007] it is possible to carry out a business process reengineering and reasoning about developing or modifying a computerised information sub-system. The final result of the requirements engineering stage is the specification of a series of task that will be supported by the software system.

The proposal consists of the following steps:

- The first step is to perform *organisational modelling*, in order to understand the organisation and its current work practice (a.k.a. as-is model):
  - The *Glossary* defines the most important concepts of the subject system and the organisational system. These concepts later appear in process definitions.
  - *Business events*: the authors refer as business events to recurrent and significant things that happen while the organisation activity goes by and to which the organisation has to respond.
  - *Domain data model:* it is a simplified class diagram that represents the business objects at a high abstraction level.
  - *Roles model:* it specifies the roles that are played by organisational actors; each role carries out a series of activities that are part of business processes.
  - *Business rules:* they constrain or define the organizational behaviour and data.
  - *Process map:* it classifies the processes in three categories: managerial scope, operational and support processes.
  - *Business process diagrams:* BPMN is used to describe processes. In this methodological proposal, a business process is considered to be a complete and dynamically coordinated set of collaborative and transactional activities that deliver value to customers [Smith and Fingar 2003].
- Then the problems and needs of the organisational system are analysed. This way, the analyst can take advantage of the creation or modification of a computerised information sub-system to perform a process reengineering. By means of *purpose analysis*, a new definition of the business processes is obtained (i.e. to-be model).
  - *Goals/strategies Diagrams*: the analysts create a Map, along with the users, following methodological guidelines that are adapted from the *Map construction process* [Rolland 2007]. For each of the problems detected a map is created; the goals that the users formulate in order to solve a problem are modelled as intentions, the characteristics of the

computerised information sub-system that will help to fulfil the goals are modelled as strategies.

- *Operationalisation tables*: the tables enumerate the strategies within the maps and assign them elements of the Business Process Diagrams as well as participant actors. The elements can be part of the as-is model, they can be newly defined elements, or it can be decided to discard the element.
- *New business process diagrams*: from the diagrams of the as-is model, new diagrams are designed, adding or discarding elements depending on the operationalisation of the map strategies. The elements are labelled according to how the computerised information system will support them.

- Then the analyst performs the specification of system requirements, which is part of the information technology infrastructure. In order to ensure the alignment of the business strategy and the technology infrastructure, the requirements models are derived from the result of the purpose analysis.
  - Functional requirements are specified by means of Extended Task Descriptions, which are templates that are based on essential use cases [Constantine and Lockwood 1999] and task & support descriptions [Lauesen 2003].
- This proposal also offers a strategy to derive a class diagram and a set of state transition diagrams, which are part of the OO-Method conceptual model.

For more information about this proposal, see [de la Vara and Sánchez 2007] and [de la Vara, Sánchez et al. 2008]. It should be noted that this methodological extension did not exist at the time this thesis started.

Communications: the lacking approach in the OO-Method framework

In view of the requirements engineering extensions that have been proposed for the OO-Method, it can be noted that none places the focus on organisational communication; that is, in the communicative interactions between actors of the organisational and subject systems, and the information system. No approach tackles with the essence of information systems: input and output messages.

2.1 State of Communication Analysis at the beginning of this thesis

At the time this thesis started, there was a sound foundation for Communication Analysis [González 2004]. There also existed several method specifications [González 2002] [González 2005] [España 2005] although there were some inconsistencies between them. The main artefacts of the requirements model are the following.

- The *Communicative Event Diagram* specifies the organizational work practice by modelling business processes from a communicational perspective. In order to guide business process modularity, a set of unity criteria are used. This diagram graphically specifies the communicative events, the precedence relationships among the events (i.e. their temporal order) and the primary actors that are involved in each event.
- Each identified communicative event is further specified by means of an *Event Specification Template*. These templates include contact requirements (e.g. what communication channels can be used), message requirements (e.g. the information that is conveyed in the event), and reaction requirements (e.g. how the message is treated).

The method had been taught for several years to Computer Science students, as part of an optional course[22]. Some of the students had later been hired by companies as analysts and/or programmers and could apply their knowledge of the method to development projects.

There was some experience with the method in action. By means of technology transfer projects, the method had been adopted by several companies[23]: (a) the Valencia Port Authority, (b) the Infrastructure and Transport Ministry of the Valencian Regional Government, (c) and Anecoop S. Coop. Anecoop is a second degree cooperative and the major commercialisation agent in the Spanish fruit and vegetables sector. Anecoop intermediates between its associated cooperatives (circa 100) and its worldwide clients. Therefore, its information system is highly complex, since it has to deal with disparate information needs (e.g. invoices conforming to different tax laws). Communication Analysis has changed the way Anecoop tackles information systems development. Anecoop has attained a successful software development process. A 2-6 people team elicits requirements. All the implementation effort is outsourced. They have implemented a software

[22] A brief description of the course can be found here:
http://www.inf.upv.es/webei/webETSIA/english/descargas/STUDY_GUIDE0809.pdf

[23] Spanish names of the companies: (a) Autoridad Portuaria de Valencia; (b) Consellería de Infraestructuras y Transporte.

framework to support their internal business processes and they have also deployed a web service-based B2B solution to work with their associates. At the moment, Anecoop leads a big project aimed to unify the internal processes of their associates. Obviously, this success can not be attributed only to the requirements practice; other practices as project management and risk management have also been undertaken.

Requirements models were mainly specified using word processors and general-purpose diagramming tools (e.g. Microsoft Visio). A prototype tool named Diccio (see Figure 21) gave additional support to the specification.

No attempts to integrate Communication Analysis in a model-driven development framework had been made. Furthermore, the method lacked a metamodel specification.

2.2 *Summary*

In Chapter 1, the motivation of this thesis has been defined; we intend to propose a model-driven development method for information system development. Information systems research is acknowledged to be a transdisciplinary discipline because of the socio-technical nature of the object of study. Thus, the pool of knowledge needed before undertaking the study of information systems draws from different disciplines, ranging from semiotics to method engineering (see Section 2.2). Furthermore, research, as well as engineering, never starts from scratch. An important task when carrying out an investigation or designing a solution is reviewing previous results to the same problem. A review of the state of the art in model-driven requirements engineering has been presented (see Section 2.3). Also, since two methods are integrated to achieve the goal of this thesis (Communication Analysis and the OO-Method), we describe their state at the moment the thesis began (Section 2.4 and Section 2.1 respectively). Some other reviews of the state of the art in specific topics are scattered across the thesis.

Chapter 3 presents the theoretical framework that underpins the research carried out in this thesis, which includes several interrelated conceptual frameworks.

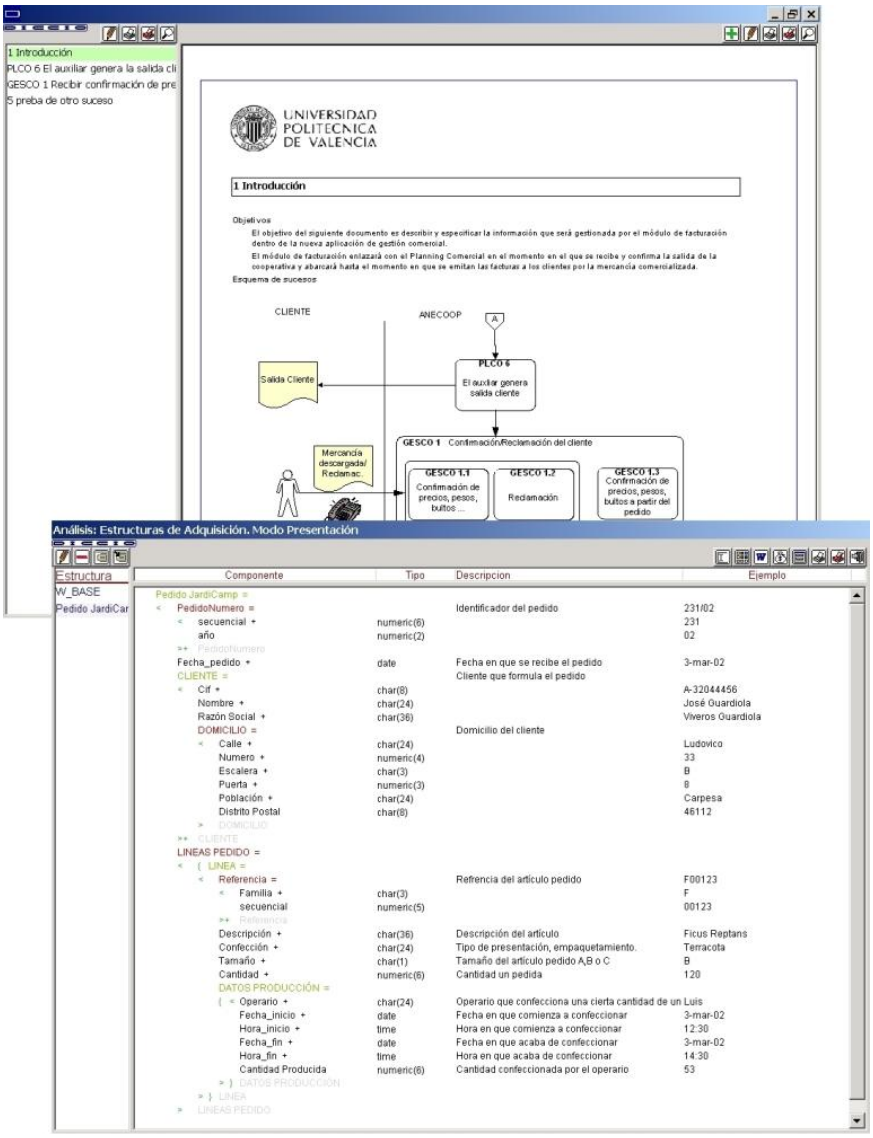


Figure 21. Snapshots of the Diccio tool for the specification of Communication Analysis requirements models

Chapter 3
Theoretical framework

"To be radical is to grasp things by the root."

Karl Marx

3.1 Motivation

In the literature of information systems, it is frequent to find different authors giving different names to the same concept. The variety of terms is not necessarily a problem as long as each author specifies the intended meaning and keeps these meanings consistent throughout his/her work. However, this is often not the case, what leads to fuzziness and ambiguity in the argumentations. Fortunately, this situation can be avoided by using reference frameworks that define concepts rigorously. This strategy is well-known in sound information systems research. For instance, a basic framework for performance engineering during information systems development is proposed in [Opdahl and Sølvberg 1992].

In this chapter, we propose a framework of well-defined and related concepts in order to reason unambiguously about information systems. Our approach is ontological yet constructivist. Instead of starting from scratch, the FRISCO report [Falkenberg, Hesse et al. 1998] is taken as a starting point. For the sake of brevity, [Falkenberg, Hesse et al. 1998] is often referred to as FRISCO throughout the thesis.

3.2 A note on ontological approaches

An *ontology*, in the philosophical sense, is a systematic explanation of existence, as it is perceived by humans. Ontology is a well established branch of philosophy

that deals with understanding elements of the world. The basis for an ontology is a fixed terminology in order to refer to entities of a particular domain. On top of this terminology, a conceptualisation can be built. According to Gruber [Gruber 1995], ontologies need the following elements in order to represent knowledge: *concepts* are the basic ideas that are being formalised and they abstract physical or abstract things of the world with the notion of class (e.g. class of objects), *relations* represent interactions and links among concepts (e.g. to build a taxonomy), *functions* are a specific type of relation in which an element of the ontology is identified by means of (operating with) other elements, *instances* are used to represent specific occurrences of concepts, *axioms* are theorems that specify relations that the elements of an ontology must satisfy so as to restrict the interpretation of meanings.

McGuinness offers a linear spectrum of ontologies[24] (see Figure 22), depending on the characteristic of the ontology and the level of detail of its specification. We argue that the conceptual framework presented in Section 3.3 is a *formal is-a ontology* since it provides an explicit hierarchy where an instance of a more specific class is always an instance of the more general class [McGuinness 2003].

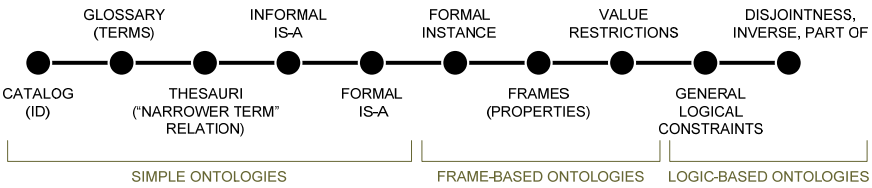


Figure 22. An ontology spectrum (adapted from [McGuinness 2003])

Depending on the philosophical stance of the authors, three categories of approaches to ontology can be distinguished [España 2008]: formal scientific realist approaches (e.g. the Bunge-Wand-Weber model [Wand and Weber 1990]), common-sense realist approaches (e.g. Chisholm's ontology, as used by Milton and Kazmierczak [Milton, Kazmierczak et al. 2000]) and constructivist approaches (e.g. the FRISCO report [Falkenberg, Hesse et al. 1998]).

Comparison of ontological approaches is inevitably biased by the epistemological stance adopted by the one who performs the comparison. My stance is biased towards the social aspects. An ontology that attempts to comprise the concepts underlying information systems comprehension and development, needs to take into account the many social issues that are present in information systems; e.g. communication, pragmatics, semiotics, etc. The following quote from Lyytinen [1987a] reflects a similar point of view on the matter:

[24] McGuinness [2003] focuses on web ontologies but her reasoning is still valid for other kind of ontologies.

> "Formalistic information models have limitations, however. They disregard the characteristics of the users and the uses of data and their impact [Stamper 1985]. Although information models attempt to capture the meaning of data, they are inadequate in this task because they are based on a too naive theory of meaning [Winograd and Flores 1987]. In effect, information models have a bias toward treating "meaning" as a stable and immutable entity originating from outside human judgment [Kent 1978]. This bias can increase bureaucratic side effects and introduce dysfunctions in the information systems' processes [Lyytinen 1987b] [Stamper 1986]." From [Lyytinen 1987a].

The conceptual framework presented in Section 3.3 builds upon the FRISCO report and therefore it follows a systemic, semiotic and constructivist approach.

3.3 *A conceptual framework for information systems: FRISCO 1.1*

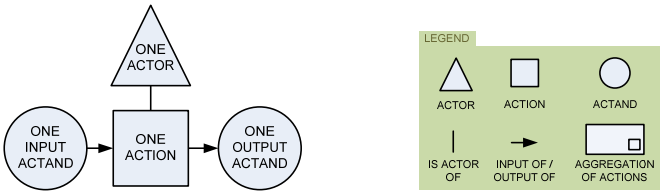
We build upon FRISCO [Falkenberg, Hesse et al. 1998] by redefining one concept and by adding new concepts to the framework. Following the FRISCO approach, new concepts are specialisations of previously defined concepts. We refer to this framework as FRISCO 1.1 to emphasise that it is an extension of its predecessor.

The following notation is intended to facilitate the comprehensibility of the conceptual framework.

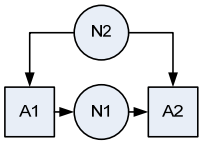
- When a concept is defined in a paragraph for the first time, it appears in bold font; for instance **transition**.
- When a concept that is used in a definition is defined elsewhere in this thesis, then it is underlined with a solid line; for instance, transition.

Each definition is identified by a code of the form Def. N, where N is a sequential number (e.g. the concept transition is defined in Def. 13). As mentioned above, since we are building upon FRISCO, several situations occur.

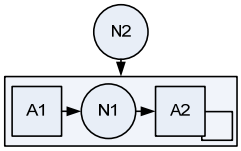
When a concept is borrowed from FRISCO as-is, without altering its definition, then the definition code that the concept was given in the FRISCO report is indicated (e.g. the concept transition is borrowed from FRISCO and, in [Falkenberg, Hesse et al. 1998], its definition had the identifier E7).



a) Example diagram and legend of extended FRISCO notation



b) Diagram with no aggregation



c) Diagram that aggregates actions

Figure 23. Notations employed for the information system models

When a concept is taken from FRISCO but it is redefined in some way, then the definition code that the concept was given in the FRISCO report is indicated, but it appears crossed out (e.g. the concept organisational system is redefined and, therefore the definition code from FRISCO is indicated as ~~E39~~)

Also, the FRISCO report proposed a concise graphical notation for the main types of elements and their relationships. This notation is shown in Figure 23.a and is intensively used throughout this work.

In addition to the symbols for actors, actions, actands and their relationships (see Section 3.3.2), we use a basic abstraction mechanism that is not addressed by FRISCO notation. Some actions are aggregated by placing them inside rectangles; these rectangles are marked with a small square in the lower-right corner that denotes that it represents an action aggregation (see the legend in Figure 23.a). This syntactical resource is used for the sake of "diagrammatic economy" (e.g. to avoid line crossings), which we believe implies greater cognitive simplicity.

This has some implications in the relationships. Note that the diagrams shown in Figure 23.b and Figure 23.c show two actions (A1 and A2) and two actands (N1 and N2). In Figure 23.b, the actand N2 is an input of both actions. In Figure 23.c, the actions A1 and A2 have been aggregated and the relationship between the actand N2 and the aggregation keeps the same semantics as in the diagram in Figure 23.b: the actand N2 is still an input for both actions.

A diagrammatic interface for FRISCO concepts that uses UML class diagram has been proposed [Wohed and Andersson 2005].. At the beginning of this thesis, we used this type of representation for our extension of the information systems conceptual framework. However, we soon chose to remain faithful to the graphical notation proposed by FRISCO because we feel that, this way, the diagrams are easier to grasp.

3.3.1 Fundamental layer

A **thing** is any part of a conception of a domain (being itself a "part" or "aspect" of the "world"). The set of all things under consideration is the conception of that domain.	E1 Def. 1

This thesis is a thing. SuperStationery, a fictional company, is also a thing.

A **predicator** is a thing, used to characterise or qualify other things, and assumed as being "atomic", "undividable" or "elementary".	E2 Def. 2

The things called "is fictional", "is a company", "is a person" are predicates.

A **predicated thing** is a thing being characterised or qualified by at least one predicator.	E2 Def. 3
A **relationship** is a special thing composed of one or several predicated thing(s), each one associated with one predicator characterising the role of that predicated thing within that relationship.	E3 Def. 4
A **set membership** is a special binary relationship between a thing (the set), characterised by the special predicator called 'has-element', and another thing, characterised by the special predicator called 'is-element-of'.	E4 Def. 5
An **adjacency relationship** is a special binary relationship between two things, both things characterised by the special predicator called 'is-adjacent-to', what (informally) means that both things exist one next to the other.	Def. 6
An **elementary thing** is a thing, not being a relationship and not being characterised by the special predicator called 'has-element'.	E4 Def. 7
A **composite thing** is a thing, not being an elementary thing.	E4 Def. 8
An **entity** is a predicated thing as well as an elementary thing.	E5 Def. 9
A **type** of things is a specific characterisation (e.g. a predicate) applying to all things of that type.	E6 Def. 10
A **population** of a type of things is a set of things, each one fulfilling the characterisation determining that type.	E6 Def. 11
An **instance** of a type of things is an element of a population of that type.	E6 Def. 12
A **transition** is a special binary relationship between a pre-state	E7

and a post-state, where both states are composite things that differ in at least one thing. | Def. 13

A **state** is a composite thing, involved as pre-state or as post-state in some transition. No element of a state may be a transition, itself. | E7 Def. 14

The **pre-state** of a transition is the state valid before that transition, and is characterised by the special predicator 'before'. | E7 Def. 15

The **post-state** of a transition is the state valid after that transition, and is characterised by the special predicator 'after'. | E7 Def. 16

State-transition structures relate transitions. Given are the transitions tx: s1 => s2 and ty: s3 => s4. The following basic state-transition structures exist in this case. | E8 Def. 17

- Sequence: sequ(tx, ty) is a sequence of transitions if s3 is a subset of s2. The resulting state-transition structure has s1 as pre-state and s4 as post-state. Longer sequences are defined: sequ(tx, ty, tz) follows from sequ(tx, ty) and sequ(ty, tz).
- Choice: choice(tx, ty) is a choice of transitions if the intersection of s1 and s3 is not empty. The result is either transition tx or ty, but not both. Wider choices are defined as follows: choice(tx, ty, tz) follows from choice(tx, choice(ty, tz)).
- Concurrency: concur(tx, ty) are concurrent transitions if the intersection of s1 and s3 is empty. The result is (s1 union s3) => (s2 union s4). Wider concurrencies are defined as follows: concur(tx, ty, tz) follows from concur(tx, choice(ty, tz)).

A **composite transition** is a state-transition structure with a unique pre-state and a unique post-state. | E9 Def. 18

A **transition occurrence** is a specific occurrence of a transition. A set of transition occurrences is subject to strict partial ordering. | E10 Def. 19

Relative time is the strict partial order (<) imposed on the set of all transition occurrences. Through the predicators before and after, relative time is introduced. | E11 Def. 20

Usually it is said that a transition occurs at a certain point in time.

Absolute time may be determined by a clock that issues (assumedly) regular pulses (transition occurrences of the clock, or clock events). | E11 Def. 21

A **rule** determines a set of permissible states and transitions in a specific context. In other terms, a rule governs a non-empty set of types of things by determining their permissible populations. | E12 Def. 22

3.3.2 The layer of actors, actions, and actands

An **actor** is a special thing conceived as being "responsible" or "responsive" and as being able to "cause" transitions.	E13 Def. 23
An **action** is a transition involving a non-empty set of actors in its pre-state, and, if not "destroyed" or "consumed" by the action, in its post-state as well, and involving a nonempty or empty set of other things (actands) as part of its pre-state, and having a nonempty or empty set of other things (actands) in its post-state.	E14 Def. 24
A **composite action** is a composite transition with the same conditions as applying for the notion of action.	E14 Def. 25
An **action occurrence** is a transition occurrence with the same conditions as applying for the notion of action.	E14 Def. 26
A **co-action** is a special action performed by at least two actors in a co-ordinated way, pursuing a common goal.	E14 Def. 27
An **actand** is a thing involved in the pre-state or post-state of an action, not considered as an actor for that action.	E15 Def. 28
An **input actand** is a part of the pre-state of an action, excluding the actors.	E15 Def. 29
An **output actand** is a part of the post-state of an action, excluding the actors.	E15 Def. 30
We refer as **resource**s to the pre-state of an action (the union of the set of actors and the set of input actands).	E15 Def. 31
The **action context** of an action is a special, optional part of the pre-state of that action, qualifying the context or situation in which that action is performed, and determining or modifying at least one of its output actands.	E16 Def. 32
The **goal** of an action is a special input actand of that action, pursued by the actors of that action and stating the desired output state intensionally.	E17 Def. 33
A **goal-pursuing actor** is an actor performing an action, who deliberately aims at a specific goal when involved in that action.	E17 Def. 34

These two definitions cover both the case of a human actor consciously attempting to achieve the intended outcome and a device which is programmed or otherwise triggered to perform a task which has been set up in a purposeful way.

3.3.3 The layer of cognitive and semiotic concepts

A **domain** comprises any "part" or "aspect" of the "world" under consideration.	E18 Def. 35
A **domain component** is any "part" or "aspect" of that domain.	E18 Def. 36

For instance, an enterprise (viewed as part of society) is composed of departments, and a department (of that enterprise) is composed of employees.

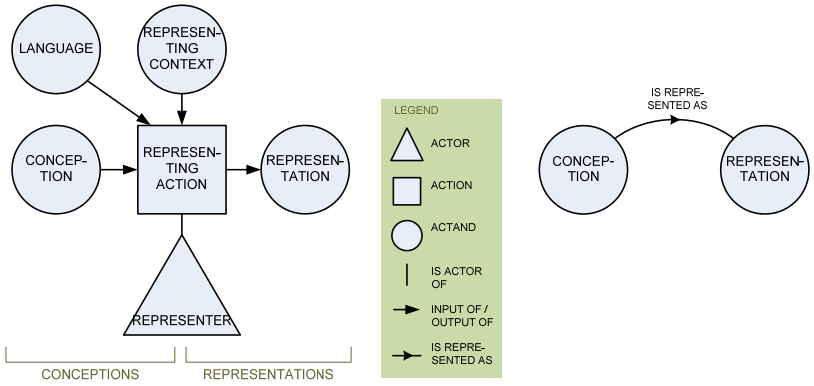
A **domain environment** is the "world" without that **domain**.	E18 Def. 37
A **human actor** is a responsible actor with the capabilities and liabilities of a normal human being, in particular capable of performing perceiving actions, conceiving actions and representing actions.	E19 Def. 38
A **non-human actor** is any actor that is not human and hence non-responsible[25].	Def. 39
A **computerised actor** is a non-human actor that is inanimate and a merely responsive device (e.g. sensors and other computerised information sub-system components such as a CPU).	D48 Def. 40
A **perception** is a special actand resulting from an action whereby a human actor observes a domain with his senses, and forms a specific (static, non-time-varying, or dynamic, time-varying) pattern of visual, auditory or other sensations of it in his mind.	E19 Def. 41
A **perceiving action** is a special action of a human actor having a domain as input actand and a perception as output actand.	E19 Def. 42
A **perceiver** is a human actor involved in a perceiving action.	E19 Def. 43
A **conception** is a special actand resulting from an action whereby a human actor aims at interpreting a perception in his mind, possibly in a specific action context.	E20 Def. 44
A **conceiving action** is a special action of a human actor having a perception and possibly some action context as input actand(s) and a conception as output actand.	E20 Def. 45
A **conceiver** is a human actor involved in a conceiving action.	E20 Def. 46

[25] This definition is based in the annotations in [Falkenberg, Hesse et al. 1998 p.30].

A **conceiving context** is the action context of a conceiving action.	E20 Def. 47
An **interpreting action** is the sequence of a perceiving action performed on a domain, resulting in a perception of that domain, followed by a conceiving action performed on that perception, resulting in a conception.	E21 Def. 48
An **interpreter** is a human actor performing an interpreting action.	E21 Def. 49
An **interpreting context** is the action context of an interpreting action.	E21 Def. 50

Conceptions are immaterial entities that only exist in the mind of human actors. If conceptions are to be documented, communicated and analyzed they must be represented in terms of a concrete artefact by using a language [Guizzardi 2006].

A **symbol** is a special entity used as an undividable element of a representation in a language.	E22 Def. 51
An **alphabet** of a language is a non-empty and finite set of symbols.	E22 Def. 52
A **symbolic construct** is a non-empty and finite "arrangement" of symbols taken from an alphabet.	E22 Def. 53
A **language** is a non-empty set of permissible symbolic constructs. The permissible symbolic constructs in a language are determined either extensionally by enumeration or intensionally by a set of rules. The rules of a language may be syntactic ("grammar") as well as semantic ("semantic rules").	E22 Def. 54
A **representation** is a special actand describing some conception(s) in a language, resulting from an action whereby a human actor aims at describing his conception(s), possibly in a specific action context.	E23 Def. 55
A **representing action** is a special action of a human actor having a conception and possibly some action context as input actand(s) and a representation as output actand.	E23 Def. 56
A **representer** is a human actor involved in a representing action.	E23 Def. 57
A **representing context** is the action context of a representing action.	E23 Def. 58



a) Diagrammatic view of the representing action

b) Abstracting the shift from conceptions to representations

Figure 24. The representing action

Figure 24.a shows a diagrammatic view of the representing action; Figure 24.b shows a simplified notation.

A **label** is a special entity being an elementary representation and used for referring to some conception in an elementary way.	E24 Def. 59
A **reference** is a special binary relationship between a conception and a representation used to refer to that conception.	E24 Def. 60
The **semiotic level** of a representation is the aspect considered in representing it. The semiotic levels are: physical, empirical, syntactical, semantical, pragmatic, and social.	E25 Def. 61
A **model** is a purposely abstracted, clear, precise and unambiguous conception.	E26 Def. 62

Models are abstractions of a domain that allow reasoning about the domain by ignoring irrelevant details while focusing on the relevant ones (adapted from [Brown, Conallen et al. 2005]). This simplification is the essence of modelling [Booch 1994]. Figure 25 shows a diagrammatic view of the modelling action.

A **model denotation** is a precise and unambiguous representation of a model, in some appropriate formal or semi-formal language.	E26 Def. 63
A **modelling action** is the sequence of a perceiving action performed on some domain, followed by a conceiving action on that perception, resulting in a model, and followed by a representing action on that model, resulting in a model denotation.	E27 Def. 64
A **modeller** is a human actor performing a modelling action.	E27 Def. 65

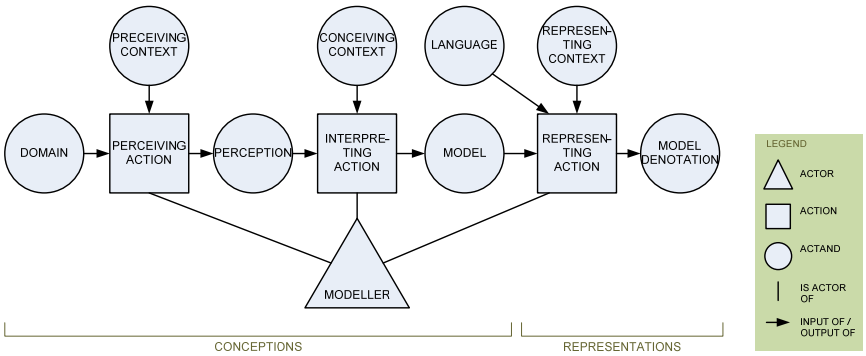


Figure 25. The modelling action

An **intensional model** is that part of a model comprising the possibilities and necessities of a domain only, i.e. the types and rules.	E28 Def. 66
An **extensional model** is that part of a model containing a specific population of the types in the corresponding intensional model, whereby this population must obey all rules determined in that intensional model.	E28 Def. 67
A **meta-model** is a model of the conceptual foundation of a language, consisting of a set of basic concepts, and a set of rules determining the set of possible models denotable in that language.	E29 Def. 68
A **meta-model denotation** is a precise and unambiguous representation of the meta-model of a language.	Def. 69
Knowledge is a relatively stable and sufficiently consistent set of conceptions possessed by single human actors.	E33 Def. 70
The term **data** denotes any set of representations of knowledge, expressed in a language. In other words, data are meaningful symbolic constructs (expressed in a language), which can be qualified as "knowledge-bearing".	E34 Def. 71
A **message** is an actand composed of data, transmitted by one actor (the sender) via a channel (a medium), and intended for a non-empty set of other actors (the receivers).	E35 Def. 72
A **message transfer** is a sequence of actions, the sending action by the sender and the receiving actions by the receivers, whereby the input actand of the sending action is the message to be sent, whereby the output actand of the sending action, being (in the simplest case) equal to the input actand of the receiving action, is	E35 Def. 73

the message on the channel, and whereby the output actand of the receiving action is the message received.	
A **sender** is an actor sending a message.	E35 Def. 74
A **receiver** is an actor receiving a message.	E35 Def. 75
Information is the knowledge increment resulting from the receiving action in a message transfer. Thus information can be defined as the difference between the conceptions interpreted from a received message and the personal knowledge before the receiving action.	E36 Def. 76
Communication is an exchange of messages, i.e. a sequence of mutual and alternating message transfers between at least two human actors, called communication partners, whereby these messages represent some knowledge and are expressed in languages understood by all communication partners, and whereby some amount of knowledge about the domain of communication and about the action context and the goal of the communication is made present in all communication partners.	E37 Def. 77
Shared knowledge is that knowledge of the individuals in a group of human actors, which they assume to be identical (or at least similar) to that of the others, as resulting from the negotiation process implicit in some communication. Thus, shared knowledge is a subset of (personal) knowledge.	E38 Def. 78
Knowledge- and data-processing actions are actions having exclusively knowledge or data as input actands or output actands.	D46 Def. 79

3.3.4 The layer of system concepts

The efficacy of the concept of system is the way it allows us to relate systems of activity (organisational systems) with systems of communication (information systems) and systems of artefacts (tools and supports such as computerised information sub-systems) [Beynon-Davies 2007].

A **system** is a special model, whereby all the things contained in that model (all the system components) are transitively coherent, i.e. all of them are directly or indirectly related to each other form a coherent whole. A system is conceived as having assigned to it, as a whole, a specific characterisation (i.e. the "systemic properties").	E30 Def. 80
A **system denotation** is a precise and unambiguous representation of a system.	E30 Def. 81
A **system component** is a non-empty set of things being contained in that system.	E30 Def. 82

To reduce complexity, it is often useful to decompose a system into sub-systems. A sub-system is a system whose system components are fully contained in another system (the super-system), but which does not contain all system components of that super-system.

Any **sub-system** S' of a larger system S is a system, itself. The set of all sub-system components of S' is a proper subset of the set of all system components of S.	E32 Def. 83
The **system environment** of a system is the set of all things not being contained in that system.	E30 Def. 84
A **system viewer** is a human actor perceiving and conceiving a domain as a system. A system viewer recognises the system, by its distinction from the system environment, by its coherence, and because of its systemic properties.	E30 Def. 85
A **system representer** is a human actor representing a system in some language.	E30 Def. 86
A **dynamic system** is conceived as capable of undergoing change, i.e. some of the system components are transitions.	E31 Def. 87
A **static system** is a system that is not dynamic.	E31 Def. 88
An **active system** is conceived as capable of doing something, i.e. some of the system components are actors performing actions on actands.	E31 Def. 89

A **passive system** is a system that is not active. E31 Def. 90

An **open system** is conceived as one which may respond to external messages or triggers, i.e. there may be transitions within the system due external causes coming from the system environment. E31 Def. 91

A **closed system** is a system whose system environment cannot cause any transitions within the system. E31 Def. 92

It is common to refer to terms such as boundary, limit or interface when we refer to system analysis and design. However, marking out system boundaries is often a difficult task. Sometimes the difficulty lies in determining a physical area (e.g. to measure the boundary of a coral reef, fractal mathematical methods are needed); other times, especially in systems not having a spatial nature, the difficulty is conceptual (e.g. an information system in the broad sense). Two fields of knowledge that place emphasis on embodying the concept of boundary are topology (a branch of mathematics) and mereotopology (a branch of metaphysics).

A **system boundary** is the set of system components that are conceived to be the frontier between the system and the system environment (even if the system does not have a spatial nature); that is, there is at least one binary adjacency relationship between each element of the system boundary and an element of the system environment. Def. 93

A **system interface** is the system boundary of an open system, where external interactions (i.e. co-actions performed by actors belonging to the system environment along with actors belonging to the system) take place. Def. 94

3.3.5 The layer of organisational and information system concepts

Organisational systems are referred by Beynon-Davies [2007] as human activity systems. He defines a human activity system as a set of activities performed by a group of persons in the fulfilment of some defined purpose; they are designed and typically specified in terms of sets of roles, procedures and rules.

~~E39~~
Def. 95

An **organisational system** is an open, dynamic and active system. It comprises the conception of how an organisation is composed (i.e. of specific actors and actands) and how it operates (i.e. performing specific actions in pursuit of organisational goals, guided by organisational rules and informed by internal and external communication), where its systemic properties are that it responds to (certain kinds of) transitions occurred in the subject system and, itself, causes (certain kinds of) transitions in the subject system.

FRISCO originally defined organisational system as follows. We highlight in bold the part of the paragraph that is affected by the redefinition.

E39
(original)

An **organisational system** is an open, dynamic and active system. It comprises the conception of how an organisation is composed (i.e. of specific actors and actands) and how it operates (i.e. performing specific actions in pursuit of organisational goals, guided by organisational rules and informed by internal and external communication), where its systemic properties are that it responds to (certain kinds of) **changes caused by the** system environment and, itself, causes (certain kinds of) **changes in the** system environment.

The rationale for the redefinition is the following. "Transition" is a more accurate concept than "change" since it has been previously defined (see Def. 13). Also, by indicating that the organisational system responds to transitions "occurred in the" (and not "caused by the") subject system we account for the fact that it may not be feasible to identify the actor that causes the change (or the organisational system is simply not interested in knowing about the actor).

For instance, an emergency service needs to know that an accident has occurred in order to send an ambulance. In this case, the change occurred in the subject system is the shift from a situation in which a vehicle is driving around without incident and its occupants have a given health condition to a situation in which the vehicle has collided with a tree and there are some people injured that require immediate medical assistance. The emergency service is not interested in knowing whether the accident has been caused by a drunken driver or by an ice

sheet (i.e. it is not interested in the actor that caused the transition) but in knowing the kind of injuries suffered by the occupants, whether the vehicle is on fire, etc. (i.e. it is interested in the result of the change, the post-state of the transition).

Actually, the how the organisational system reacts to certain types of transitions (its behaviour) and the information required to act (its communications) are usually defined by organisational norms.

Norms are socially agreed rules affecting and to a large extent directing the actions within an organisational system. E39 Def. 96

As defined above (see Def. 95), the organisational system (OS) is a group of actors collaborating to pursuit the organisational goals. According to Lockemann and Mayr [1986], the organisational goals are typically related to observing, controlling and/or influencing a portion of the world.

For instance, an organisation composed of astronomers is interested in observing part of the cosmos. A middleman company is interested in observing a market segment, as well as in influencing the actors in market segment in order to increase their demand of the company products. A car manufacturer company, aside from having similar goals as the middleman company, also wants to control its machinery.

We refer as **subject system** (SS) to the portion of the world in which the organisational system is interested (a.k.a. universe of discourse); it can be conceived as a system. Def. 97

For instance, in the case of the organisation of astronomers, the part of the cosmos in which they are interested is the subject system. In the case of the middleman company, the market segment is the subject system, as well as most of their own organisation (e.g. they want to observe the trucks they own for logistics) and part of the provider companies (e.g. at least they want to observe some of the behaviour of their providers, such as the yearly increase in their prices or the shipments that they send to the middleman company warehouses). In the case of the car manufacturer company, the subject system comprises the market segment, the providers and the factories (including their machinery).

Note that the subject system and the environment of the organisational system are not the same thing. The organisational system is not only interested in what happens outside its boundaries (i.e. in the environment) but also in what happens within the organisation. For instance, the middleman company is surely interested in the sales manager decision to stop working with a given provider. Also, the organisational system is not interested in what happens in the whole environment (observing the whole environment would be unachievable, anyway), but just in a portion of it. For instance, a star in a distant galaxy is part of the environment of a clothing shop; however, it is not of interested for the

clothing shop. Therefore, we argue that the subject system is a relevant concept that helps conceiving the organisational goals as well as the kinds of changes to which the organisational system responds.

An **information system** is a sub-system of an organisational system. It contains actors, actions and actands[26], and the relationships among these elements, whereby all actions are knowledge and data-processing actions. An information system has the systemic property that its actors are pursuing at least one specific (informational) goal.	E40 Def. 98
An **information system denotation** is a precise and unambiguous representation of an information system.	E40 Def. 99

An information system is a socio-technical system (see Def. 98), a set of agents of different nature that collaborate in order to support organisational communication; that is, communication between the organisational system and its environment, as well as within the organisational system [Langefors 1977]. It can be seen as a collection of agents of different nature that cooperate in order to acquire, process, store, retrieve and distribute information with the purpose of observing, controlling or influencing a portion of the world [Lockemann and Mayr 1986]. Olivé [2007] offers a similar definition. An information system is fundamentally concerned with communication in support of human activity using artefacts to represent and transmit data [Beynon-Davies 2007]. The information system is intended to increase the effectiveness of the organisational system in achieving its goals.

A **computerised information sub-system** is a sub-system of an information system, whereby all actions within that sub-system are performed by one or several computer(s). Since a computerised information sub-system has only computerised actors, the actands are only data.	E41 Def. 100

The computerised information sub-system is the part of the information system that is automated (see Def. 100). Information systems are not necessarily bound to computers; around 4400 BC, ancient Sumerians already used complex clay tokens with incised markings for accounting purposes [Beynon-Davies 2009]. Computers, however, are more versatile and operating with them is faster than operating with clay tokens. A computerised information sub-system helps the organisational system to increase the efficiency in achieving its goals.

[26] The information system components are related to communication and information. This way, actors in an information system are communicating, information-providing and/or information-seeking actors. Actands are information-oriented actands. Actions are communication-oriented and information-providing actions.

Communication Analysis is an information system development method; therefore, it can be applied to develop an information system that is based in paper forms. The OO-Method is a computerised information sub-system development method; therefore, its final artefact is a software application.

A **computerised information sub-system denotation** is a precise and unambiguous representation of a computerised information sub-system. [27] Def. 101

The four above-mentioned systems can be imbricated in many ways. In the case of the organisation of astronomers, the organisational system and the subject system are completely disjoint; the astronomers observe stars and other phenomena that are out of reach. In the case of the middleman company, the clothing store and the car manufacturer company, the subject system and the organisational system overlap. With regards to the information system, it is usually embedded in the organisation so it can be considered a sub-system of the organisational system. However, this is not always the case (e.g. the information system can be externalised, therefore being disjoint systems). By definition, the computerised information sub-system is a sub-system of the information system. In any case, for the sake of simplicity, the systems can be considered to be subsumed as depicted in Figure 12.

Within the transitions in which an organisation is interested and reacts, two categories can be distinguished. Some transitions are well known by the organisational system and may even have procedures defined; that is, there may exist norms that regulate the interaction with the subject system and the internal reaction when these transitions occur. These procedures may be defined in the form of textual documents, business process graphical models, etc. For instance, a middleman company will probably have procedures defined so as to regulate the organizational reaction to an order request placed by a client. We refer to these transitions as regulated transitions. Other transitions, although the organisational system is interested in them, may not be regulated by organizational procedures. For instance, a middleman company must react to a bunch of kids breaking the glasses of one of its warehouses although this transition is probably not foreseen. Also, the organisation is interested in the manager's son appearing in the tabloids because of his addiction to gambling, but this is out of the scope of our research interest.

[27] This term is not explicitly defined in the FRISCO report, but it is used in one of the figures (see [Falkenberg, Hesse et al. 1998 p. 74]).

We refer as **regulated transition** to a transition whose occurrences take place in the subject system, which is of interest for the organisational system, and for which organisational norms have been defined. Def. 102

A regulated transition, like a transition, is a category of things, their instances being regulated-transition occurrences and transition occurrences, respectively.

A **regulated-transition occurrence** is a transition occurrence with the same conditions as applying for the notion of regulated transition. Def. 103

Identifying regulated transitions is part of modelling the organisational system or the information system. It is also subject to modularity issues, since the phenomena that a modeller conceives as a single regulated transition, another modeller can conceive them as several regulated transition, depending on the abstraction level.

The organisational system in interested in knowing when a regulated transition has occurred because it affects the achievement of its organisational goals. For instance, a clothing shop aims at earning money by selling their products; if a person wants to buy a garment in the shop, then the organisational system must react to this regulated transition (e.g. sell the garment).

Since the organisational system is interested in regulated-transition occurrences, each time a regulated transition occurs, the organisation should be informed. This way, the shared knowledge of the organisation will be incremented with facts (data) about the regulated-transition occurrence. For this shared knowledge to increase, the new data needs to be reported to the organisational system. Since the information system is responsible of supporting organisational communication, the new data is actually reported to the information system, which in turn will distribute it to the organisational actors that need to be aware of the regulated-transition occurrence.

Organisational systems usually develop norms that define what information is needed to take a certain action.

The **regulated-transition information definition** is an organisational norm that defines the pieces of data that are relevant for the organisational system in order to know about the occurrence of a regulated transition. Def. 104

The regulated-transition information definition is part of an organisational system or information system model. Like regulated transitions, regulated-transition information definitions are subject to modularity issues. For instance, in the case of the clothing shop, two possible model denotations of a regulated-transition information definition are the following.

- To make a sale the clerk needs at least the order description.
- To make a sale the clerk needs at least which articles the client is interested in and, for each article, which size (if the article has different sizes) and the requested amount of items.

For the moment we tiptoe round the issue of modularity since it is dealt with in Section 3.4. The organisational system expects those pieces of data in case of a regulated-transition occurrence, so as to take informed action. Thus, when a regulated-transition occurrence takes place, then an actor should communicate to the organisational system the information in the regulated-transition information definition. This communication is supported by information system and it may involve one or several message transfers.

It should be noted that, when humans are involved, communication typically occurs in the form of dialogues. An isolated message transfer often does not convey enough information to carry on a task. Let's imagine a possible dialogue occurred in the clothing shop, transcribed in Table 3.

Table 3. A transcription of a possible dialog occurred in the clothing shop

Clerk:	Good morning madam, may I help you?
Client:	Well, yes... I want to buy a T-shirt.
Clerk:	Sure, any particular model?
Client:	I've seen a blue shirt in the shop window... The one with a yellow frog in it...
Clerk:	Oh, I love that one! Which size? I think an M will fit you.
Client:	No, it's not for me. It's a present for my husband. Let me see the XL, please.
(After checking the stock)	
Clerk:	I'm afraid we do not have that T-shirt in XL right now.
(Client remains silent, disappointed)	
Clerk:	We have one L left, but we are expecting to receive some stock tomorrow.
Client:	You know what? I'll buy it size L and, in case it's too small, I'll come back.
Clerk:	Sure, no problem. Do you want me to gift-wrap the T-shirt for you?
[...]	

The message "I've seen a blue shirt in the shop window... The one with a yellow frog in it..." does not convey enough information to sell a garment. On the contrary, the whole dialogue does indeed convey all the necessary information: "want to buy", "a T-shirt", "blue [...] with a yellow frog", "L". However, the messages transferred in the dialog also contain irrelevant data (from the point of

view of the information system). In order to address these issues, we need to use dialogue theory.

According to Dynamic Representation Theory, we should distinguish between the non-communicative goal (e.g. the client wants to order T-shirt) and the communicative goal (e.g. the client wants to make the clerk aware that she wants a T-shirt), as well as between the non-communicative underlying task (e.g. ordering a T-shirt) and the communicative task (e.g. communicating with the clerk). In the same way, the dialogue contains dialogue control acts (e.g. "please" has a social-obligations-management function) and task-oriented acts (e.g. "I'll take the L" has an information-providing function).

From the point of view of organisational systems, any of the types of dialogue functions may be important (e.g. etiquette indeed shapes business communication with Japanese organisational systems [Limaye and Victor 1991]). However, from the point of view of information systems, the dialog acts of utmost importance are task-oriented acts, which have an information-providing, information-transfer or action-discussion function.

Action-discussion dialog acts usually inform of the kind of regulated transition that has occurred and (as a side-effect) convey some data that is typically part of the regulated-transition information definition. For instance, "I want to buy a T-shirt" has a request function, and it also includes some of the information needed to take action "T-shirt". The rest of the necessary information is provided by dialog acts that have an inform function, such as "The one with a yellow frog in it..."

We can obtain the regulated-transition information by abstracting all the data that is not related to the regulated transition (e.g. dialog acts with an information-seeking or a social-obligations-management function such as "any particular model?" and "please") and by discarding overridden data (e.g. "XL" was an option but such size was unavailable, so finally the client agreed to buy an "L").

Def. 105

The **regulated-transition information** is the set of relevant pieces of data about a regulated-transition occurrence that, when provided by one or several actors as part of the messages transferred in communication with the information system, brings about an increment in the organisational knowledge. It must be as compliant as possible with its corresponding regulated-transition information description.

Figure 26 illustrates the identification of regulated-transition information in the dialog in Table 3. Pieces of regulated-transition data are highlighted in green. At the bottom of the figure, a thick arrow compiles the regulated-transition information, which is compliant with the regulated-transition information definition.

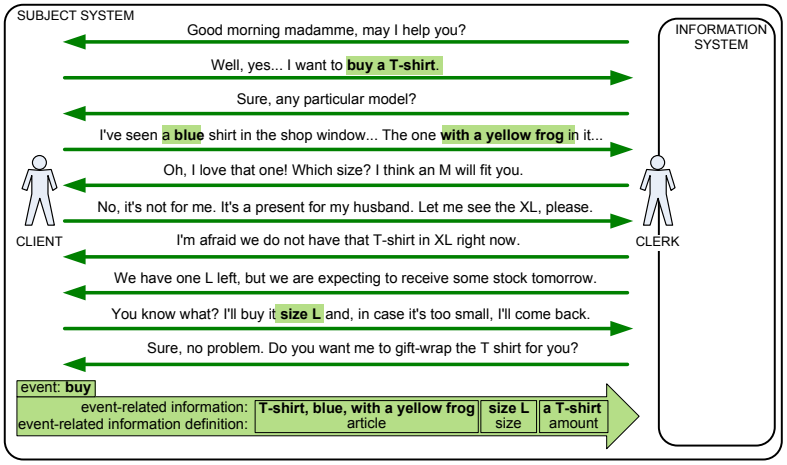


Figure 26. Identification of regulated-transition information

The organisational system may develop norms that define what actors are assigned the task of reporting the occurrence of a regulated-transition and communicating the regulated-transition information.

Def. 106

A **regulated-transition observer** is an actor to whom the norms of an organisational system have assigned the action of sensing the regulated-transition occurrences of a specific regulated transition and reporting those occurrences to the information system. For this purpose, the event observer sends the information system the regulated-transition information by means of the appropriate message transfers.

The regulated-transition observer can be a human actor. For instance, the client of the clothing shop is responsible for placing an order, the logistics department clerk of a middleman company is responsible for arranging the transportation of a shipment, etc. Despite referring to this actor as observer, its importance lies in the fact that s/he reports the facts; indeed, there may exist several regulated-transition observer but the information system is concerned about the one who reports the occurrence.

Sometimes a non-human actor is assigned the task. In such case, the non-human actor is typically designed in a way so that when it senses certain transitions that occur in subject system it sends the appropriate messages to the information system. For instance, a sensor can be assigned the task of warning the information system that the temperature in a tank has risen above a safe level.

From the point of view of the organisational system, the regulated-transition observer is the source of information. It may not be the ultimate source of information, but the organisational system does not care for that. For instance, the

husband of the clothing-shop client in the example above may have expressed to his wife his desire to own a blue T-shirt with a yellow frog in it but, as far as the clothing shop is concerned, it is the client who matters.

Although in some cases it may be obvious to identify the regulated-transition observer (e.g. the temperature sensor), in other cases many actors are involved in the chain of message transfers triggered by a regulated-transition occurrence, hindering the identification. In the above-mentioned example, it is quite straightforward to discard the husband as he does not really become involved and his existence is not even proved; however, whether the clerk is the event observer instead of the client depends on the modelling guidelines (or, in the absence of guidelines, on system modeller).

A social view of information systems

In information systems, information flows in two main directions: in and out of the information system. Figure 27 presents a view of the information system as a means to support organisational communication at the semiotic level of the "social world"; it is based on the work of Le Moigne [Le Moigne 1985; Van Gigch and Le Moigne 1989].

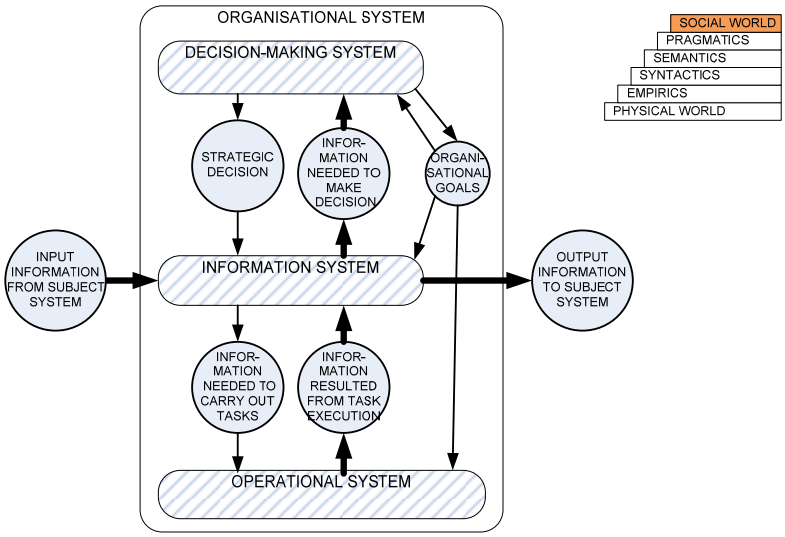


Figure 27. The information system viewed at the "social world" semiotic level

The information system provides information to both the decision-making system (which includes the organisational actors in charge of defining organisational goals[28] and the corresponding organisational norms) and the operational system (which includes the actors in charge of carrying out the tasks that fulfil the organisational goals). Information also flows from and to the environment of the organisation.

At this semiotic level the focus of analysis should be put on social issues such as how the different systems are interrelated (e.g. how the organizational system is related to other organizational systems in terms of value interactions), what social laws must the organisational system obey, what organisational norms and goals have already been established, etc.

A pragmatic view of information systems

Figure 28 presents a view of the information system as a means to support organisational communication at the semiotic level of "pragmatics". At this abstraction level, the information system is reported occurrences of transitions that occur in the subject system (ingoing message transfers). That is, the regulated-transition observer is the actor that becomes aware of a regulated transition occurrence in the subject system and reports this occurrence to the organisational system. The information system stores the information. Sooner or later, this information is distributed to those actors that should be aware of the occurrence of the transition (outgoing message transfers). That is, some actor is provided information, thus becoming a receiver of recalled facts. At this semiotic level, message transfers are an abstraction of a much complex interaction (i.e. dialogues), as discussed above.

The regulated transition observer and the recalled facts receiver can be the same person (or device) or they can be distinct. Moreover, both message transfers can occur consecutively or at different moments in time. For instance:

- In order to assign a service to a messenger, the clerk inspects the planning grids (consecutively, the recalled facts are needed a priori).
- Whenever the sensor reports a sudden increase in the tank temperature, the tank operator is informed immediately (consecutively).
- The salesman checks the orders that are pending approval when she decides to do so (at different moments in time).
- The company manager takes a look at the business indicators of the balance scorecard once every trimester (at different moments in time).

[28] The organisational system reacts to transitions according to certain organisational goals, which are usually stated by organisational actors at the strategic level. Goals affect actions and can, therefore, be considered as input actands.

- The list of damages needed to be repaired is stuck in the notice board and the team of caretakers cross them out as they repair them (at different moments in time).

Also, several outgoing message transfers can be triggered by one ingoing message transfer, and vice versa. At this semiotic level the focus of analysis should be put on pragmatic issues such as what transitions should be regulated, what information about their occurrences does the organisation need to know, which actors should be aware of these occurrences, how message transfers of both types are related, and what are the underlying intentions of the intentions of the actors involved, etc.

Note that the elements inside the information system are based on the elements from the model of a computerised information sub-system defined in FRISCO (see Figure 13) and in the ISO report (see Figure 11). However, we argue that the elements of this model can be defined disregarding any computerisation; that is, an information system that is not automated (e.g. one that is exclusively supported by paper forms) still has the elements described in Figure 28. Every information system has at least one language representation (or language denotation) that defines how the messages and the stored facts have to be represented, as well as a domain model denotation that keeps the memory of the information system (e.g. a filing cabinet full of paper forms).

Def. 107

The **information system reaction to ingoing message transfer** is a composite action having a message containing the regulated transition information and a language representation as input actands and a domain model denotation as output actand.

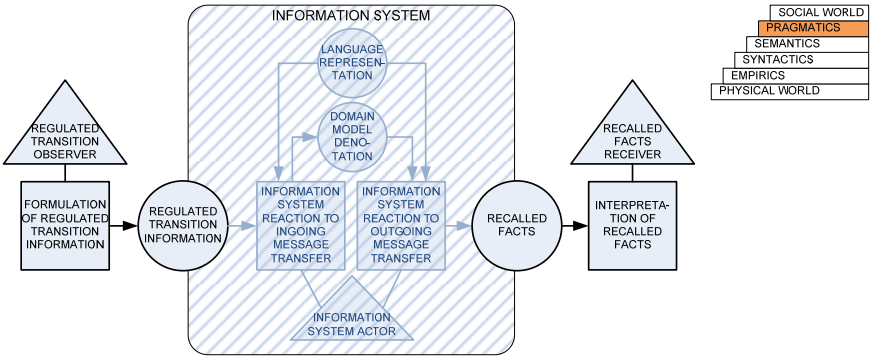


Figure 28. The information system viewed at the semiotic level of "pragmatics"

We acknowledge that this definition is a simplification that abstracts many facts about how things take place in the information system. Again, the form in which ingoing message transfers take place is actually a dialogue. In such dialogue, there also occur message transfers from the information system to the transition observer[29] (e.g. information that is needed before formulating the regulated transition information, such as the planning grids of a courier company), but the main goal concerning the dialogue is actually providing the regulated transition information. For instance, the client in the example above (see Table 3) is a transition observer who wants to communicate to the information system of the clothing shop the desire of buying a T-shirt. The abstracted regulated transition information is shown in Figure 26. A similar argument applies to outgoing message transfers, but in the opposite direction.

The language representation gathers the knowledge of the organisational system: it is the model of the information system memory and it defines the types of facts the information system can register, as well as many rules about its behaviour. Note that, for some actions of the information system, the language representation can be implicit. For instance, in message transfers between humans the language representation (e.g. a dictionary and a grammar of the English language) need not be explicit.

> The information system **language representation** is an actand of several actions of the information system that denotes the language in which messages need to be represented.
>
> [30] Def. 108

The information system reacts to new information by memorising the facts.

> A **domain model denotation** is a special actand of the information system reactions. It is built up and maintained by specific ingoing message transfers and parts of it may be retrieved by outgoing message transfers.
>
> [31] Def. 109

[29] If these outgoing message transfers were taken into account, the domain model denotation would also be an input actand of the information system reaction to ingoing message transfer.

[30] This concept is mentioned (but not rigorously defined) in FRISCO.

[31] This definition is somehow based on the explanation of domain model denotation in the FRISCO report (see [Falkenberg, Hesse et al. 1998, p. 77]). In a computerised information sub-system, the database (an extensional model) is part of the domain model denotation. However, we disagree with considering that the database schema (e.g. an intensional model containing types and rules) is part of the domain model denotation. This is inconsistent with the differentiation between language/meta model and model/model denotation (see, e.g. Figure 3.6-2 in [Falkenberg, Hesse et al. 1998, p. 57]). The intensional model is the language with which the domain model is represented. So, strictly speaking, the domain model denotation is solely the database, not the database schema.

The domain model denotation is mainly composed of information base entities (as defined by [ISO 1987]); entities and their relationships are the information system memory imprints about regulated transitions. The domain model denotation acts as the information system memory. Stored facts can be later retrieved by the actors.

The **information system reaction to outgoing message transfer** is a composite action having a language representation and a domain model denotation as input the regulated transition information and a language representation as input actands and a message containing recalled facts as output actand.	Def. 110
The **recalled facts** are pieces of data that are either part of the information system domain model denotation themselves or are facts and conclusions derived/drawn from the domain model denotation. Recalled facts are conveyed in a message transfer from the information system to a recalled facts receiver.	Def. 111

A semantic and syntactic view of information systems

Figure 29 presents a view of the information system as a means to support organisational communication at the semiotic levels of "semantics" and "syntactics". Note that this view of the interactions with the information system is different to the one at the pragmatic level. Now the model focuses on the dialogue that takes place in order to either communicate the occurrence of a transition to the information system or to recall certain facts from the information system memory. That is, for each "abstract" message transfer discussed above (see Figure 28), several interactions like the one depicted in Figure 29 take place. The loop in Figure 29 can be understood as two turns in a dialogue between the user and the information system (one turn starting with the formulation of an input message and ending with the information system reaction updating the domain model denotation and another turn starting with the retrieval of facts from the domain model denotation and ending with the interpretation of the output message.

Figure 29 is based on (and somehow refines) Figure 13 but, again, the possible computerisation of the information system is disregarded. The refinement aims to understand more in depth what takes place inside an information system. The term "user" must be taken in the general sense "someone who uses a product, machine or service" [Cambridge 2003] (the services provided by the information system) and not necessarily as a user of a software application.

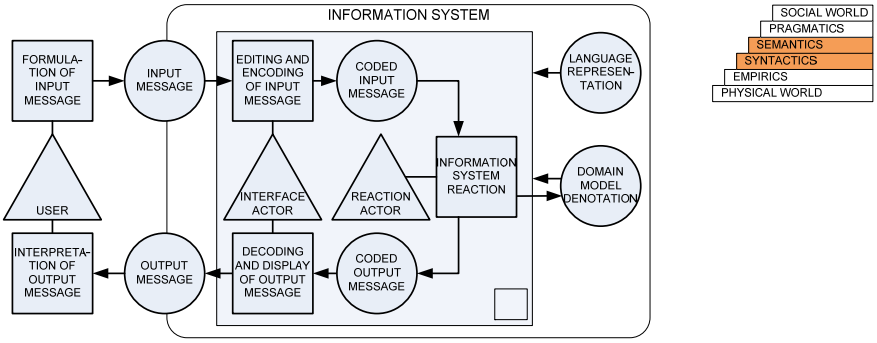


Figure 29. The information system viewed at the semiotic levels of "semantics" and "syntactics"[32]

The user can either be the transition observer or the recalled facts receiver. In order to provide or request the information, the user needs to convey one or several input messages to the information system.

A **user** is an actor that interacts with an information system by providing input messages and/or receiving output messages. — [33] Def. 112

An **input message** is a message that is conveyed to the information system by a user of the information system. It can contain regulated transition information, among other data. — [33] Def. 113

These messages may or may not be expressed in the language prescribed by organisational norms (i.e. to ensure shared knowledge an organisational system usually defines how facts should be expressed). In case the input message is not expressed in the language denoted by the information system language representation, the input message must be re-encoded. An interface actor is in charge of re-representing the input message so as to obtain a coded input message that can be treated by the information system.

An **interface actor** is an actor that is in charge of editing messages in the interface of the information system, so as to represent them following the codes and norms prescribed by the information system language representation. It can either be a human actor or a computerised actor. — Def. 114

The **edition and encoding of input message** is a composite action whereby the interface actor re-encodes the input message into a — Def. 115

[32] The language representation and the domain model denotation are input for all the actions generalised inside the box.

[33] This concept is mentioned (but not rigorously defined) in FRISCO.

coded input message so that the information system is able to react accordingly. It is actually a translation that adds no new information to the input message.

This translation is intended to re-express the message using the language representation, that acts as a shared linguistic code [Jakobson 1990] (see Figure 10). This view enforces the conception of the information system interface as a message editor. However, the interface is also a message displayer, if the opposite direction is considered.

Def. 116

The **coded input message** is a special message that contains the same knowledge as the input message but represented in the language prescribed by the information system language representation.

The primary role and interface role may be played by the same or distinct actor, who may belong to the organisation or its environment. For instance:

- A customer has a desire and phones the company so as to formulate an order (because she is providing the information, she is playing the primary role); the salesman who attended the customer in the phone enters the order in the system (because of this message editing, he is playing the interface role).
- The customer could have used the electronic form the company has in its website to make the order (this way, a person who is not member of the organisation plays the two roles,).
- In any case, later in time, the salesman gives the order the go-ahead (in this case, a member of the organisation plays the two roles).

After the input message has been expressed in a way that is tractable by the information system, the information system can process is to either update or recall facts from its memory.

Def. 117

The **information system reaction** is a composite action having a coded input message, a language representation and a domain model denotation as input actands and a domain model denotation and a coded output message as output actands.

Def. 118

A **reaction actor** is an actor performing the information system reaction. It can either be a human actor or a computerised actor (in such case it is part of the computerised information sub-system).

The reaction processor performs the following steps of the information system reaction:

1. It takes the coded input message and checks whether it corresponds to one of the expected types of messages; this knowledge is contained in the language representation (which, in this case, acts as a conceptual or database schema

that defines the types of facts that can be stored in or recalled from the memory).

2. Then, in case it needs any facts that are stored in the domain model denotation, it retrieves such facts.
3. It processes all this information according to a certain recipe of rules (i.e. an algorithm, a procedure described in the organisational norms).
4. It updates the information system memory (the domain model denotation contains the "knowledge of the organisational system").
5. It builds one or several coded output messages showing the result of its actions, and conveys them to the actors concerned.

Obviously, the information system reaction could be further refined to explicitly denote all these steps, but we chose to keep the diagrams simple.

Def. 119

The **coded output message** is a special message that contains recalled facts and/or explanatory information (e.g. instructions, errors, acknowledgements, etc.) represented in the language prescribed by the information system language representation.

The coded output message is originated by the information system reaction and it sometimes needs to be translated again to an external actor-intelligible message.

[34] Def. 120

The **output message** is a message conveyed by the information system to a user of the information system, containing the same knowledge as the output message and represented in a language that in intended to be understood by the user.

Def. 121

The **decoding and display of output message** is a composite action whereby the interface actor re-encodes the output message into a coded input message so that the user is able to interpret it. It is actually a translation that adds no new information to the output message.

[34] This concept is mentioned (but not rigorously defined) in FRISCO.

3.4 A conceptual framework for model modularity

When an observer attempts to analyse a domain and model it as a system, the complex interrelation of perceivable phenomena gives rise to a complex interrelation of conceptions. In order to actually gain knowledge, the observer needs to properly structure the conceptions, for which s/he needs to apply many cognitive abstraction processes (e.g. classification, generalisation, modularisation, refinement). Among these cognitive processes, we now focus in modularisation, which (informally) consists in defining system components in a way that they can be separated and recombined [Schilling 2000]. When attempting to create a model denotation of a system or implementing the system, the issue of modularity is unavoidable.

Modularity has been proved to pay off its price [Baldwin and Clark 1999]. However, the problem of defining proper guidelines to perform modularisation or to judge the modularity of a model remains open for some types of models. We argue that, in order to facilitate the definition of such guidelines, a framework of concepts related to modularity is needed. Since the FRISCO report did not enter this subject, this section proposes an extension[35].

Dealing with complex structures of concepts

Although the interpreting action is defined in FRISCO as resulting in a single conception (see Def. 48), innumerable interpreting actions are actually carried out sequentially or even concurrently while analysing a domain. In fact, the interpretation of the domain phenomena brings about not just one, but a set of conceptions. Figure 30 presents an informal view of the interpreting action.

[35] The conceptual framework presented herein only reasons about modularity as applied to models and model denotations. We acknowledge that modularisation can also be applied to the construction of physical things (e.g. mechanical device).

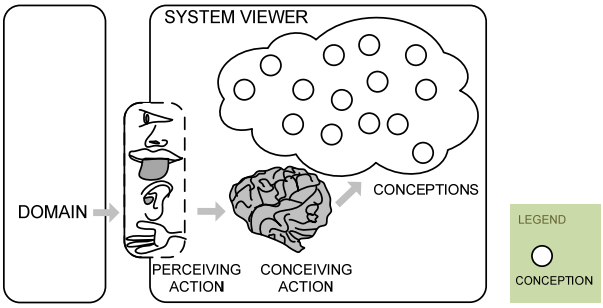


Figure 30. Diagrammatic view of the interpreting action

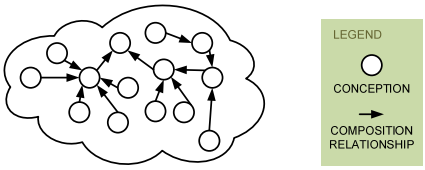


Figure 31. A set of conceptions with composition relationships among them

Conceptions do not arise as a plain set, but they are usually structured by means of composition relationships. We refer as composition to the relationship (see Def. 4) that holds between a part and the whole. They are also known as part-of, part-whole, mereological, and meronymic relationships.

> A **composition relationship** is a special binary relationship between a thing (the "whole"), characterised by the special predicator 'has-part', and another thing (the "part"), characterised by the special predicator called 'is-part-of', meaning that the latter predicated thing is considered to be part of the former predicated thing.

Def. 122

Figure 31 shows a set of conceptions in which composition relationships have been made explicit. However, revealing composition relationships in a set of conceptions is not always a trivial problem. In a set of conceptions, many times it is not clear which conceptions should be associated the predicator 'has-part' and which of them should be associated the predicator 'is-part-of'.

Two approaches can be adopted in order to reveal part-of relations among conceptions; namely, a meaning-based approach and a structure-based approach. The *meaning-based approach* focuses on the meaning of concepts, not in their structure; this approach can rely on Philosophy and/or Semiotics.

- Philosophy: There is a long-standing controversial debate in Philosophy about whether universals (pre)exist. Universals are abstract qualities that particular things show. According to Loux, the three major kinds of universals are types

(e.g. something can be predicated as being a tree), properties (two things can be regarded as being green), and relationships (e.g. one thing can be regarded as being taller than another thing) [Loux 2001]. If universals are accepted, composition relationships are given for granted; that is, it is just a matter of being able to grasp the structure underlying reality.

- Semiotics: Semiotics is the field of science that studies signs, and how they relate to conceptions. Several relationships among conceptions, signs (representations) and particular things have been proposed over time. These relationships have usually been called semiotic triangles. From a constructivist stance concepts are social constructs and, thus, the semiotic triangle includes the observer (thus becoming a tetrahedron) [Braun, Hesse et al. 2000]. In the semiotic approach, composition relationships fall out of the scope of the triangle/tetrahedron[36] (see Figure 32).

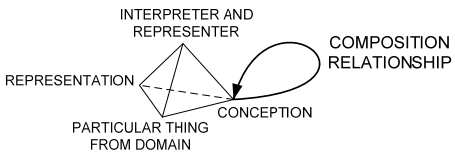


Figure 32. Part-of relations are beyond the scope of the semiotic tetrahedron

The meaning-based approach is powerful for dealing with conception definitions, but its plain usage of conceptions[37] makes this approach too simplistic when dealing with sets of conceptions structured by means of composition relationships.

The *structure-based approach* focuses on the structure underlying a set of conceptions and, therefore, it allows tackling with complex problems. This approach can rely on Mereology and/or General Systems Theory.

- Mereology: Mereology is the interdisciplinary study of composition relationships; it integrates knowledge from Philosophy, Linguistics, Cognitive Psychology and Mathematics, among other disciplines. In this approach, composition is also known as meronymy [Winston, Chaffin et al. 1987; Leśniewski 1992]. Works related with meronymic relations can be found in many fields of science; e.g. Ontology [Guarino and Welty], Biology [Pisanelli, Battaglia et al. 2002], Medicine [Gangemi, Pisanelli et al. 2000], object-oriented databases [Piattini 1994], data warehouses [Artale, Franconi et al. 1996].

[36] It can be argued that a relationship itself is a conception. This does not change the fact that (as far as we know) semiotics has not addressed the issue of composition relationships in a systematic way.

[37] We mean plain in the sense of unstructured.

- General Systems Theory: The term system denotes a way of observing and describing phenomena of a domain as a set of interrelated components; the system shows emergent properties, which are properties that cannot be attributed to any particular component [Checkland 1981]. Also, systems are analysed by focusing on external interactions[38], rather than studying its components (otherwise, emergent properties cannot be apprehended). Moreover, if the interior of the system needs to be analysed, then it is convenient to identify sub-systems. Following semiotic arguments, systems, sub-systems and components can be assimilated to their corresponding conceptions, and represented by means of signs.

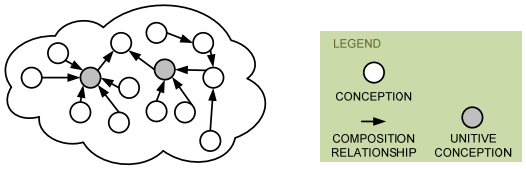


Figure 33. An illustration of set of conceptions where unitive conceptions (at a given systemic level) are identified

In information systems engineering, complex sets of conceptions are frequently confronted and represented (e.g. business process modelling, requirements modelling, conceptual modelling, database design, software architecture design). Therefore, we advocate adopting a systemic approach (other approaches can be used in a complementary way).

We refer as unitive conceptions to those composite conceptions that are the matter of interest at certain systemic level (e.g. external interactions). In systemic approaches, unitive conceptions need to be conceived or identified (see Figure 33).

Def. 123

A **unitive conception** is a special conception that is related by means of composition relationships with one or several other conceptions, the unitive conception being characterised by the predicator 'has-part', and resulting from an action whereby an interpreter aims at focusing on certain aspect of a domain which is of interest in a specific interpreting context (e.g. clarifying a certain systemic layer).

The conceptions that are "part" of a unitive conception can be cognitively treated as a single "whole".

[38] An external interaction is an interaction between the system and its environment.

It is worth noting that, when interpreting a domain and conceiving it as a system, the resulting conceptions are the system itself and its components. Therefore, in that case a unitive conception is also either the system or a system component.

Def. 124

A **conception-unifying action** is a special interpreting action having a set of (possibly interrelated) conceptions, an interpreting context as input actands and a set of interrelated conceptions as output actand, both sets addressing the same domain, where the output actand either includes a unitive concept or a composition relationship that is not included in the input actand.

When the action context of the conception-unifying action is conceiving a domain as a system, then the unitive conception is either the system or a system component.

Figure 35 shows a sequence of conception-unifying actions applied on the same set of conceptions. However, the conception-unifying action is a simplification of how things occur in practice. For instance, during the process of structuring a set of concepts, system interpreters usually act by trial and error (e.g. they change their mind about a specific unification, they split up conceptions again, they reunify, etc.).

Also, the concept-unifying action is often part of the system conceiving action. For instance, as soon as interpreters start conceiving the domain as a system, a unitive conception which is the system-as-a-whole takes shape in their mind; from that moment on, each and every conception of the system components is inevitably related by a composition relationship to the system-as-a-whole. However, many composition relationships are meditated a posteriori.

In any case, when dealing with complexly structured sets of conceptions, identifying unitive conceptions is not an easy task and guidelines are needed. We refer as *unity criteria* to the rules that aim to facilitate conceiving unitive conceptions and composition relationships. Figure 34 shows that the unity criteria are input for the conception-unifying action.

Def. 125

Unity criteria is an optional actand of the conception-unifying action consisting of a set of rules that guide the conceptualisation (conceiving a new conception) or identification (selecting an existing conception) of unitive conceptions, or the establishment of composition relationships.

When the conceiving context is system analysis then unity criteria facilitate conceiving systems and sub-systems.

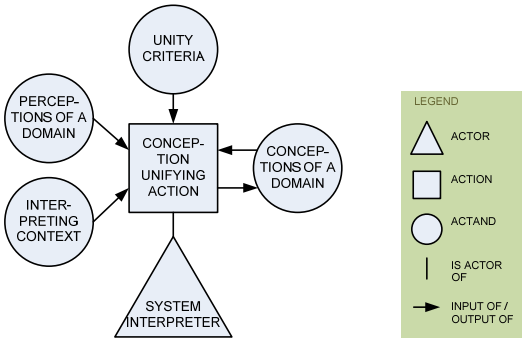


Figure 34. Conception-unifying action

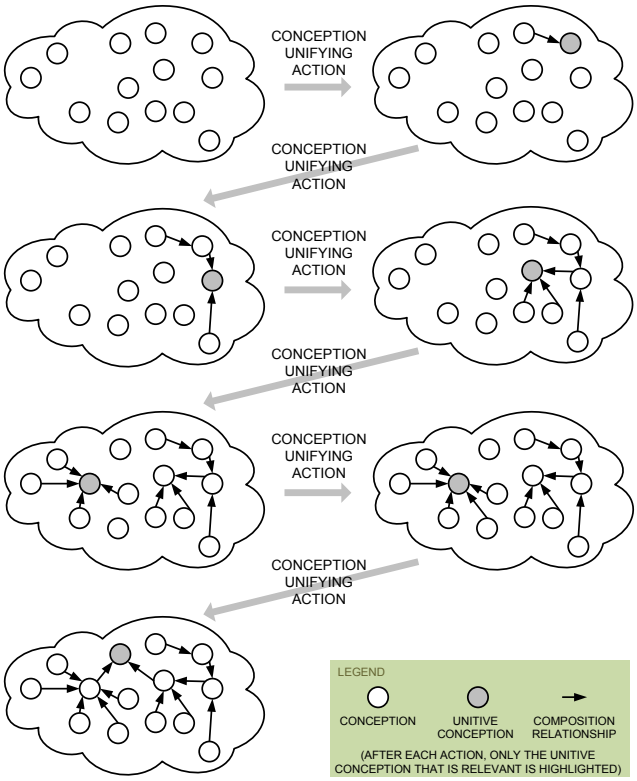


Figure 35. Illustration of a sequence of conception-unifying actions

Modularity is a systemic principle that allows determining the degree to which the components (sometimes conceived as sub-systems) of a given system may be separated and restructured [Schilling 2000]. Both a system and its sub-systems can be regarded as modules. Modularity serves the general purpose of focusing on the interface of a (sub)system in order to understand emergent properties[39] [Bertalanffy 1975].

Two mechanisms allow modularity: information hiding and encapsulation [Wirth 1971; Parnas 1972; Blair, Gallagher et al. 1991].

- *Encapsulation*. Creating encapsulations implies representing that a set of conceptions about the domain are conceived as a unit, in a way that this unit can be referred to or be manipulated by the observer.
- *Information hiding*. Hiding the representations of the conceptions underlying an encapsulation implies creating a module. Information hiding allows separating internal and external structure/behaviour of modules. In this sense, it highlights emergent properties.

We now define these mechanisms more rigorously (diagrammatic views are shown in Figure 36 and Figure 37).

Def. 126

An **encapsulation** is a representation describing several conceptions as being "part" of a single unitive conception.

When the representing context of the encapsulating action is analysing or describing a system, an encapsulation represents the system (or a sub-system, or a system component) as being "composed of" system components.

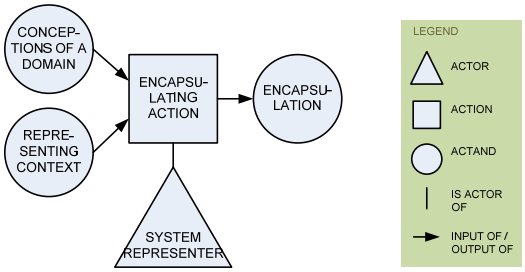


Figure 36. Encapsulating action

[39] Actually, when analysing, the purpose is to focus on the interface of the system (i.e. the boundaries off the system and the interaction with the environment); when designing, the purpose is to focus on the interface of its sub-systems and to describe the components.

Def. 127

An **encapsulating action** is a special representing action having a set of conceptions, one of them being a unitive conception, and possibly some representing context as input actand and an encapsulation as output actand.

By means of the encapsulating action, the intended interpreters of the representation are expected to conceive the underlying composition relationships, but they are allowed to treat the encapsulation, (at the representational level) and its corresponding conception (at the cognitive level) as a "whole".

If the internal details of an encapsulation that represents a sub-system are hidden and, instead, the emergent properties of the sub-system are described then a module is obtained (the sequence of both actions is shown in Figure 38).

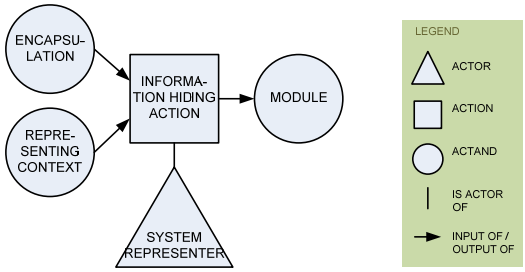


Figure 37. Information-hiding action

Def. 128

Information-hiding action is a special representing action having an encapsulation as input actand and a module as output actand. To perform the action, the system representer removes the symbolic constructs that represent the system components being "part" of the system (or sub-system) represented by the encapsulation; only the systemic properties of the module remain.

Def. 129

A **module** is a representation of a system (or a sub-system), in which instead of having the system components of the system (or the sub-system) represented, its systemic properties are described.

Modularity always serves an engineering goal; the goal depends on the engineering field to which modularity is applied. Table 4 and Table **5** present examples of how modularity is applied to several fields; namely, General Systems Theory [Bertalanffy 1975; Checkland 1981], Structured Design [Constantine and Yourdon 1975; DeMarco 1979; Yourdon 1989], Transaction design [Gray 1981] and Normalisation Theory [Codd 1970]. For each field, the goals behind modularisation, the information hiding mechanism and the encapsulation mechanism are shown.

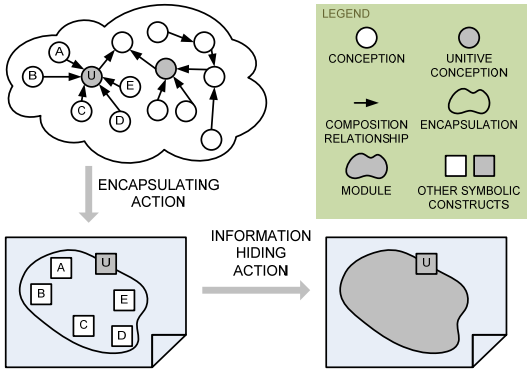


Figure 38. Sequence of encapsulating action and information-hiding action

Table 4. Modularity in general systems theory and structured design

	Gral. Systems Theory	**Structured Design**
Intention	To achieve holism; that is, to be able to not only describe composition but emergent properties (relation with the environment).	To enhance maintainability; that is, to facilitate future software design modification.
Encapsulation	Unity criteria depend on the type of analysis being done[40]. Encapsulations are often referred to as sub-systems.	The unity criterion is known as cohesion; the more cohesion, the better; this concerns grouping together functionally related pieces of reaction. Encapsulations are often referred to as modules.
Information hiding	Black box principle; by focusing on external interactions, the interior of sub-systems remains hidden.	Coupling avoidance. The less a software module relies on the internal structure of other modules, the better; this is achieved by means of module interfaces.

[40] After all, General Systems Theory provides principles that apply to systems in general.

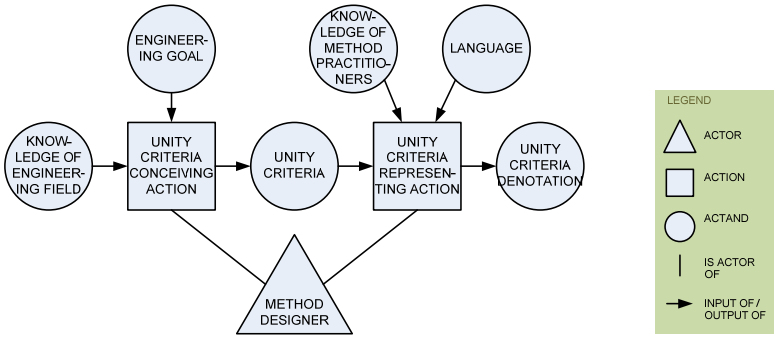


Figure 39. Actions related to unity criteria conception and representation

Table 5. Modularity in transaction design and normalisation theory

	Transaction design	**Normalisation Theory**
Intention	To ensure data integrity. To do that, transactions must have the set of properties known as ACID[41].	To prevent data anomalies by minimising redundancy.
Encapsu-lation	Unity criteria that guide transaction design are atomicity (the "all or nothing" rule) along with consistency (ensuring consistent pre- and post-states). Encapsulations are referred to as transactions or units of work.	Unity criteria are normal forms (e.g. 3NF or BCNF); the higher the normal form applicable to a table, the less vulnerable it is to inconsistencies and anomalies. Encapsulations are often referred to as tables[42].
Information hiding	Atomicity along with isolation (the restriction on intermediate states observation) prevent unexpected interactions among transactions.	Definition of logical relations[42] among data by means of identifiers (e.g. foreign keys refer to primary keys). This way the inner structure of a table can be changed without affecting external relations.

[41] ACID stands for atomicity, consistency, isolation and durability; these properties are not completely orthogonal. Therefore, they take part in information hiding and encapsulation mechanisms in a mixed fashion. Durability is a property of database management systems.

[42] In Normalisation Theory, relation is a mathematical construct that abstracts the notion of table. Here, however, the term relation refers to a connection between things.

Thus, when method designers define and propose unity criteria, they usually have some knowledge of the engineering field and an engineering goal in mind (see Figure 39). Unity criteria are conceptions that, if expected to be shared with others, need to be describer in a proper form (e.g. as a written checklist, as a handbook, etc.), using a language.

A **unity-criteria-conceiving action** is an action having certain knowledge of an engineering field and an engineering goal as input actands and unity criteria as output actand.	Def. 130
A **unity-criteria-representing action** is an action having some unity criteria, certain knowledge of method practitioners and a language as input actands and unity criteria denotation as output actand.	Def. 131
A **unity criteria denotation** is a precise and unambiguous representation of a set of unity criteria.	Def. 132

Whether a given unity criteria denotation is actually precise and unambiguous needs to be validated. Whether a given set of unity criteria actually contributes to fulfilling the engineering goal also needs to be validated.

3.5 *A conceptual framework for model-driven system development*

In the literature of model-driven development (MDD) of information systems and computerised information sub-systems many of terms that are used lack a precise definition. This section attempts to define at least some of them, starting a conceptual framework for MDD.

First of all, the notion of system development and model-driven system development are defined.

Def. 133

A **system development** is a composite action having a domain and an engineering goal as input actands and a set of physical things as output actand, where the output actand can be conceived as a system having certain systemic properties that are considered to fulfil the engineering goal.

For instance, in a computerised information sub-system development, the domain is typically an organisational system and the engineering goal is typically to give good support to the communications of the organisational system. The output actand is typically certain software (and perhaps also some hardware) that can be conceived as a computerised information sub-system, whose systemic properties are that it provides the expected support to organisational communication.

In model-driven system developments the importance of models and model denotations is stressed.

Def. 134

A **model-driven system development** is a special kind of system development where models and model transformations (as well as model denotations and model-denotation transformations) play an important role.

According to [ISO/IEC 2007b] system developments and model-driven system developments are two kinds of "endeavour"; that is, information-based domain development efforts aimed at the delivery of some product or service through the application of a method.

Many terms are used in the literature to refer to the different roles played by the participants a system development endeavour (e.g. analyst, designer). We can generalise these roles by defining a system developer as anyone involved in the development of a system (both analysts and designers are system developers); that is, a person who applies a method for some specific job, usually an endeavour [ISO/IEC 2007b]. Similarly, the notion of model-driven developer is defined.

A **system developer** is a human actor that is involved in (some of) the actions within a system development. Def. 135

A **model-driven developer** is a human actor that is involved in (some of) the actions within a model-driven system development. Def. 136

We refer as method to a systematic way of working by which one can obtain a desired result. A technique can be considered to be recipe for obtaining a certain result [Wieringa 1996]. Methods contain techniques. A method denotation is a specification of the process to follow together with the work products to be used and generated, plus the consideration of the people and tools involved, during a system development effort (adapted from [ISO/IEC 2007b]).

A **method** is a composite thing, mainly consisting of a set of rules that define how a system development should be performed. It comprises the definition of the actions that should be performed (including the order in which these actions should occur), the actors that should be involved in the actions, and the input actands and output actands of each action. Besides, the rules in a method can refer to other things (e.g. languages) that can be conceived to be part of the method as well. Def. 137

A **method denotation** is a precise and unambiguous representation of a method. Def. 138

In some organizational systems, the role of method engineer is defined for those actors who design, build, extend and maintains methods [ISO/IEC 2007b].

A **method engineer** is a human actor performing any of the actions related to conceiving and representing any part of a method. Def. 139

Methods for system development are quite complex conceptions (and method denotations are typically lengthy documents). For the purpose of understandability and reuse, methods (and method denotations) are often decomposed in smaller conceptions (parts). In some contexts, these parts are referred to as techniques. A technique is a "small-grained work unit" that focuses on how a given goal may be achieved [ISO/IEC 2007b]. However, the distinction between method and technique is quite arbitrary and mainly serves the above-mentioned purposes.

A **technique** is a part of a method which a method engineer defines so as to facilitate the communication of the method or to allow reusing the technique (e.g. to conceive other methods). Def. 140

A **technique denotation** is a precise and unambiguous representation of a technique. Def. 141

Note that, in the area of MDD, it is typical to use the term model to refer to the description of a system, instead of using the term model denotation. For instance, a diagram that describes an information system workflow using the Activity Diagram modelling language is actually a model denotation of an information system (that is, a system denotation), but it is typically said to be a model of an information system.

However, [ISO/IEC 2007b] keeps the same distinction made in FRISCO by defining a model as "an abstract representation of some subject [domain] that acts as the subject's surrogate for some well defined purpose [goal]." Also, the following annotation is made: "models are abstract constructs and therefore they are not visible or directly manageable. Documents [model denotations] are the perceivable, communicable counterparts of models."

The term modelling language is typically used in the area of software development to refer to languages that are intended for describing the system under construction.

A **modelling language** is special language, consisting of an alphabet, a set of syntactic rules (i.e. a grammar) and a set of semantic rules (i.e. definitions attempting to clarify the meaning of the symbols in the alphabet), intended to be used for representing systems under a model-driven development paradigm.

Def. 142
Syn. 54

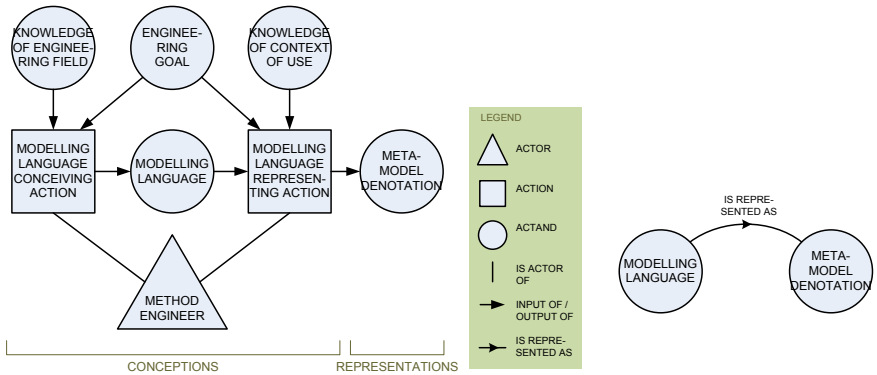
A **modelling primitive** is a symbol of the alphabet of a modelling language.

Def. 143
Syn. 51

For instance, the UML [OMG 2010b] is a widespread modelling language in the area of MDD, and it is often used for describing computerised information sub-systems. In turn, the UML is composed of several interrelated modelling languages; for instance, the Class Diagram modelling language. An example of a modelling primitive is a Class.

A **modelling-language conceiving action** is an action performed by a method engineer, having certain knowledge of an engineering field and (at least) one engineering goal as input actands and a modelling language as output actand.

Def. 144



a) Diagrammatic view of the representing action

b) Abstracting the shift from conceptions to representations

Figure 40. Method engineers conceive and represent modelling languages

In the area of information systems development, examples of engineering fields are business process reengineering, requirements engineering, conceptual modelling, database design, etc. Examples of engineering goals for the field of conceptual modelling are enhancing model comprehensibility and facilitating automatic software code generation.

If modelling languages are to be communicated then they need to be represented (see Figure 40). For instance, in the area of model-driven development, method engineers not only conceive but also represent modelling languages, model transformations, etc.

Def. 145

A **modelling-language representing action** is an action performed by a method engineer, having certain knowledge of the context of use of a modelling language (e.g. knowledge about model-driven developers who are expected to use the modelling language) and (at least) one engineering goal as input actands and a meta-model denotation as output actand.

A language can be accompanied by rules that guide how it can be used for a certain purpose. We then talk of modelling techniques (see Figure 41).

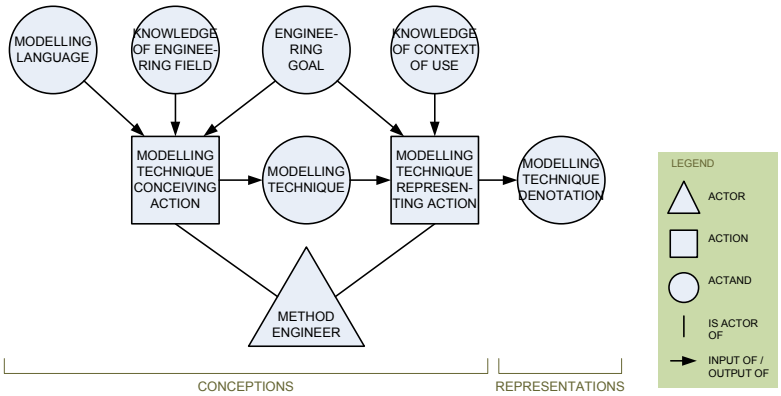


Figure 41. Method engineers conceive and represent modelling techniques

A **modelling technique** is a composite thing, consisting of a modelling language, a set of rules that guide the usage of the modelling language (rules that guide the production of models so as to ensure certain properties). Def. 146

A **modelling-technique conceiving action** is an action performed by a method engineer, having certain knowledge of an engineering field and (at least) one engineering goal as input actands and a modelling technique as output actand. Def. 147

A **modelling technique denotation** is a precise and unambiguous representation of a modelling language and a set of model denotations that serve as examples of the usage of the modelling language. Def. 148

The previous definition accounts for the importance of examples to understand conceptions; as claimed by Sager [1990], it is the examples that complete the meaning of a concept. This way, every modelling technique denotation ought to have one or several illustrative examples of its application.

A **modelling-technique representing action** is an action having certain knowledge of the context of use of a modelling technique (e.g. knowledge about model-driven developers who are expected to use the modelling technique) and (at least) one engineering goal as input actands and a meta-model denotation as output actand. Def. 149

If compared to English, a grammar and a dictionary are part of the language definition (i.e. a metamodel denotation), whereas a book on how to use English language to produce scientific documents (e.g. [Halliday and Martin 1993]) is part

of a technique that explains how to use the language for an intended purpose (i.e. a modelling technique denotation).

A modelling technique can include many kinds of rules. For instance, it can include unity criteria intended to guide the creation of modular model denotations.

The model-driven development paradigm encourages carrying out model transformations in order to derive some models from other models, as well as to generate software code from models. OMG [2003] defines model transformation as "the process of converting one model to another model of the same system." A model transformation is guided by rules that prescribe how to take the elements of the source model as input and generate the elements of the target model as output (see Figure 42).

A **model transformation rule** is a special rule determining the set of permissible states and transitions within a model transformation. Def. 150

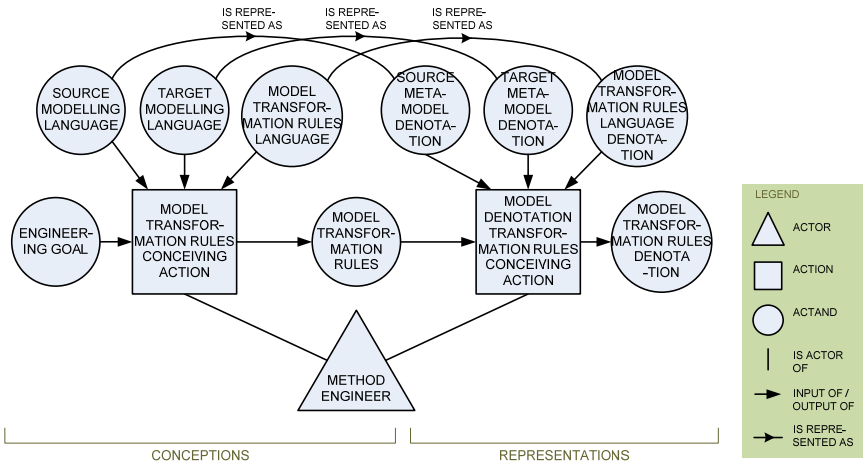


Figure 42. Method engineers conceive and represent model transformations

A **model-transformation rule denotation** is a precise and unambiguous representation of a model transformation rule. Def. 151

Two model-transformation rule denotations can correspond to the same model-transformation rule. For instance, given a set of model transformation rules intended to guide the conversion of a class diagram into a relational database schema, one model transformation rule can define that, for each class in the class

diagram, a table is created in the relational schema. This rule[43] is a conception that can be represented in many ways. Table 6 shows (part of) two possible model-transformation rule denotations, using two different model-transformation rules languages; namely the ATL transformation language and QVT.

A **model transformation** is an action having (at least) one source model (often a system), a set of model-transformation rules and possibly some model transformation parameters as input actands and (at least) one target model (often a system) as output actand, where the input and output models differ in at least one thing. Def. 152

A **model transformation parameter** is an optional actand of a model transformation whose model-transformation rules include at least one state-transition structure of type "choice". The transformation parameter is a piece of data that the model transformation actor uses to decide which branch of the "choice" to perform. Def. 153

Table 6. Two model-transformation rule denotations that correspond to the same model-transformation rule

```
rule Class2Table {
from
  c : Class!Class
to
  out :
Relational!Table (
  name <- c.name,
  ...
}
```

```
protected void ruleClass2Table(EObject pkg,
EObject schema) {
  List<EStructuralFeature> classifiers =
getMulFeature(pkg, "classifiers");
  List<EStructuralFeature> tables =
getMulFeature(pkg, "tables");
  for(EObject classifier : classifiers) {
    EObject newTable = createObject(OUT,
"Table");
    if (isKindOf(classifier, "Class"){
      setFeature(newTable, "name",
getFeature(classifier,"name"));
      tables.add(newTable);
      ...
    }
  }
}
```

Model transformations can be manual (see Figure 43.a) or automated (see Figure 43.b), depending on who the transformation actor is.

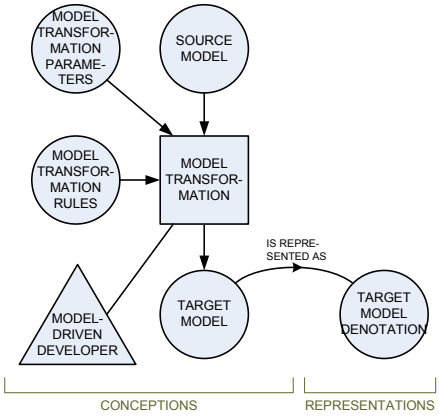
A **model-denotation transformation** is an action having a model denotation (often a system denotation), a set of model-transformation rule denotations and possibly some model- Def. 154

[43] Actually, we had to represent the rule in written English so as to communicate our argumentations to the reader. If the rule denotation was unambiguous enough, the conception that has been formed in the reader's mind is indeed the model transformation rule.

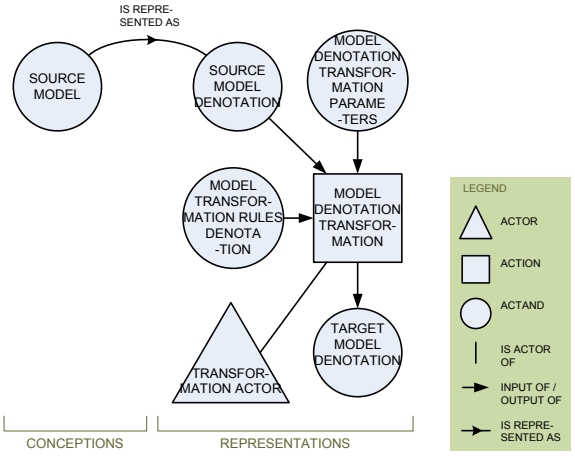
denotation transformation parameters as input actands and a model denotation (often a system denotation) as output actand, where the input and output model denotations differ in at least one thing.

A **model-denotation transformation parameter** is a precise and unambiguous representation of a model-transformation parameter. Def. 155

A **model transformation actor** is a computerised actor performing a model denotation transformation. Def. 156



a) "Manual" model denotation transformation



b) "Automated" model denotation transformation

Figure 43. "Manual" vs. "automated" model denotation transformation

To better understand the elements that are involved in an automated model denotation transformation, Figure 44 presents a view of a metamodel-based transformation[44] (this figure is a precise definition of the example in [OMG 2003, Figure 3-2, p. 3-9]). A source model denotation that conforms to its corresponding source metamodel denotation is intended to be transformed into a target model denotation that conforms to its corresponding target metamodel denotation. The model transformation rules are expressed in terms of a mapping between the metamodel denotations. This mapping can be defined as relations between metamodel elements or it can be complex enough to need declarative or imperative transformation rules.

In the following we provide another useful definition of a term that is commonly used in the literature of model-driven development.

Def. 157

An **analyst** is a system viewer, in the sense that they typically perceive a domain and conceive it as a system (i.e. a model), as well as a model representer, in the sense that they typically create system denotations (i.e. descriptions of the system, or model denotations).

However, when analysts participate in a model-driven development project, they also play the role of model-driven developers.

3.6 Summary

This chapter presents the theoretical framework upon which the rest of the thesis builds. It consists of three related conceptual frameworks that can be regarded as constructivist ontologies intended to prevent terminological confusion (see Section 3.2). Our work follows the steps of the FRISCO approach, so as to avoid a techno-centric view of the area and aiming for a broad, interdisciplinary approach. We share its intention to bridge the gap between reality and its modelling concepts basing the line of argument on *semiotics*, i.e. the theory of *signs*, their *form* (syntax), *meaning* (semantics) and *effect* (pragmatics) [Hesse and Verrijn-Stuart 2000]. The core framework deals with information system concepts (see Section 3.3).

[44] The MDA Guide version 1.0.1 refers to this type of model denotation transformation as "metamodel transformation" but we consider that this term can lead to confusion. It is not actually the transformation of a metamodel but a model transformation that is based on metamodels and, thus, a "metamodel-based transformation."

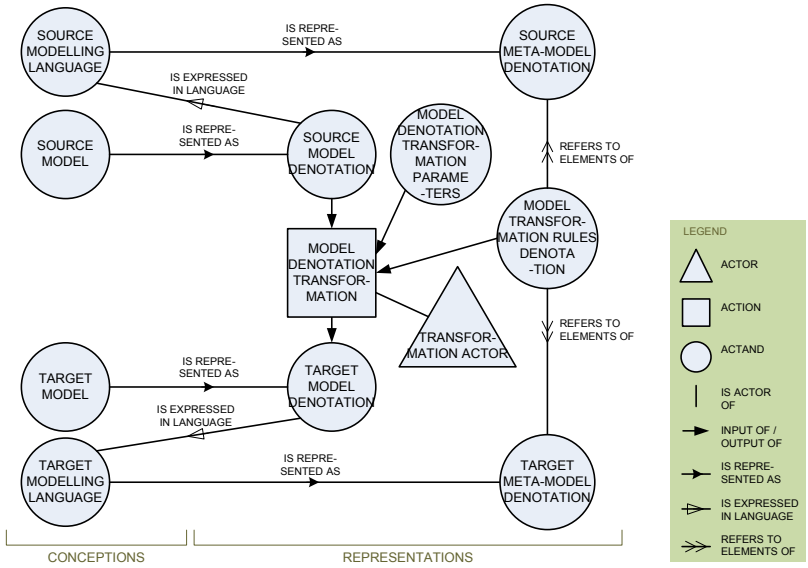


Figure 44. Metamodel-based transformation

We have also defined a conceptual framework about model modularity (see Section 3.4) and another one about model-driven systems development (see Section 3.5) because there is a string focus on both issues in the following chapters: Section 4.3.1.2 defines guidelines for business process modularity and Chapter 5 presents a model-driven approach to derive conceptual models from requirements models. These frameworks altogether serve as an ontology[45] that sets a sound conceptual foundation aiming to facilitate further investigations on model-driven information systems development.

[45] We have focused on defining the concepts rigorously because meaning is a key point in ontologies. After all, "ontologies consist of a vocabulary along with some specification of the meaning or semantics of the terminology within the vocabulary" [Fox and Gruninger 1998]. The formalisation of the concepts in terms of algebraic formulas did not serve our purposes.

PART II.
SOLUTION DESIGN

Chapter 4
Detailed specification of Communication Analysis

"It's not where you take things from - it's where you take them to."

Jean-Luc Godard

4.1 Motivation

We intend to integrate Communication Analysis into the OO-Method model-driven development (MDD) framework. As discussed in Chapter 1, Communication Analysis needs to be specified in detail so as to allow for such integration. Among other aspects that need to be worked out, rigorous definitions of the method concepts and, moreover, a method metamodel are prerequisites for a sound integration. This chapter provides a detailed specification of Communication Analysis.

> **Communication Analysis** is a method that aims to facilitate the interpretation and representation of a domain as an information system and (possibly) the development of a computerised information sub-system.

Def. 158

[España, González et al. 2011] includes a requirements model that describes the work practice of an organisation applying Communication Analysis; many of the examples in this chapter are drawn from that requirements model.

The remainder of the chapter provides a detailed specification of Communication Analysis and it is organised as follows. Section 4.2 proposes a structure for Communication Analysis requirements specifications; this structure defines how the elicited requirements ought to be arranged. Section 4.3 presents some modelling techniques that are used within Communication Analysis; aside

from describing the grammar (the modelling constructs and how they relate to each other), methodological guidelines are provided and the usage is illustrated. Section 4.4 presents some requirements elicitation and analysis techniques that are applied in Communication analysis to discover requirements. Since we intend to integrate this method into an MDD framework, a metamodel of the method needs to be provided; Section 4.5 presents a platform independent metamodel of Communication Analysis (no particular tool-implementation platform is considered yet). Section 4.6 reviews some works related to Communication Analysis and discusses the similarities and differences. Section 4.7 concludes with a summary.

4.2 *Requirements structure and concept definitions*

There is actually no consensus in the area of Requirements Engineering on the concept of requirement. Some authors consider that requirements describe *what* the system should do whereas design focuses on *how* the system should do it; but Davis [1990] acknowledges that the difference between the *what* and the *how* is not objective (one person's *how* is often another's *what*). Other authors consider that requirements should focus on the *why* (the underlying intentions) [Rolland 2007]. In any case, there seems to be some agreement on the fact that requirements should offer an external view of the system under development. For instance, many authors agree on the fact that requirements should describe the user perception of the system [IEEE 1990; Zave and Jackson 1997], and requirements are often considered as functions and characteristics of a system that can be perceived externally [Davis 1990]. Davis [2003b] also provides an analogy by which stating requirements is likened to defining the phenotype of the system we desire.

It is not enough to define a frontier among requirements and non-requirements; a further structure is convenient for a requirements engineering method to be usable. Standard definitions of requirement (e.g. a condition or capability needed by a user to solve a problem or achieve an objective [IEEE 1990]) neither specify the scope, granularity or structure of a requirement. To overcome this problem, the concept of requirement needs to be reinforced with a well-defined requirements structure, prescriptive elicitation and analysis guidelines, and instructive exemplification that facilitate method understanding and adoption.

In this section, we propose a structure for enterprise information system requirements that is founded on Systems Theory and Communication Theory, among other fields of science.

In Communication Analysis, requirements specifications are mainly organised around the set of communicative interactions that enterprise actors need to perform in order to carry out their tasks. Communicative interactions constitute *what* the enterprise needs in terms of information (the problem); therefore we consider them to be the main requirements to be discovered and described during information system analysis. The rest of requirements qualify communicative interactions by stating aspects, qualities, constraints, etc. of the communication. These solution requirements constitute *how* the system has to be implemented or has to perform. Communication and information requirements are related to *efficacy*; that is, the adequacy of the information supplied to achieve a task. Solution requirements relate to *efficiency*, to the minimisation of operation and usage costs. E.g. usability requirements and response-time constraints are intended to reduce costs. We propose a layered structure for requirements that covers both the problem and the solution spaces. This structure is the backbone that supports Communication Analysis requirements engineering (including the elicitation, the specification and the analysis of requirements).

We define a requirements structure with five levels; each requirement is ascribed to one requirements level. Figure 45 shows the requirements levels, as well as the main activities performed and the artefacts created at each level. The requirements structure offers two dimensions. One dimension is related to the static-dynamic duality (horizontal axis of Figure 45). The other dimension is related to refinement (vertical axis of Figure 45).

In organisational systems, a duality appears among dynamic and static aspects: business interactions are such things because they affect business objects and, in the opposite way, certain objects are considered business objects because organisational interactions deal with them. Communication Analysis facilitates dealing with this duality by offering techniques for interaction analysis (activities 2, 4 and 6) and for the analysis of business objects (activities 3, 5 and 7). This way, following a systemic approach, Communication Analysis allows using stepwise refinements techniques in a twofold perspective. From a dynamic perspective, analysing business processes and obtaining, from the process specification, the business objects structure. From a static perspective, discovering the business objects structure and, then, reasoning the communicative interactions that allow the users to deal with those business objects. This intertwining of both perspectives makes the method flexible and contingent.

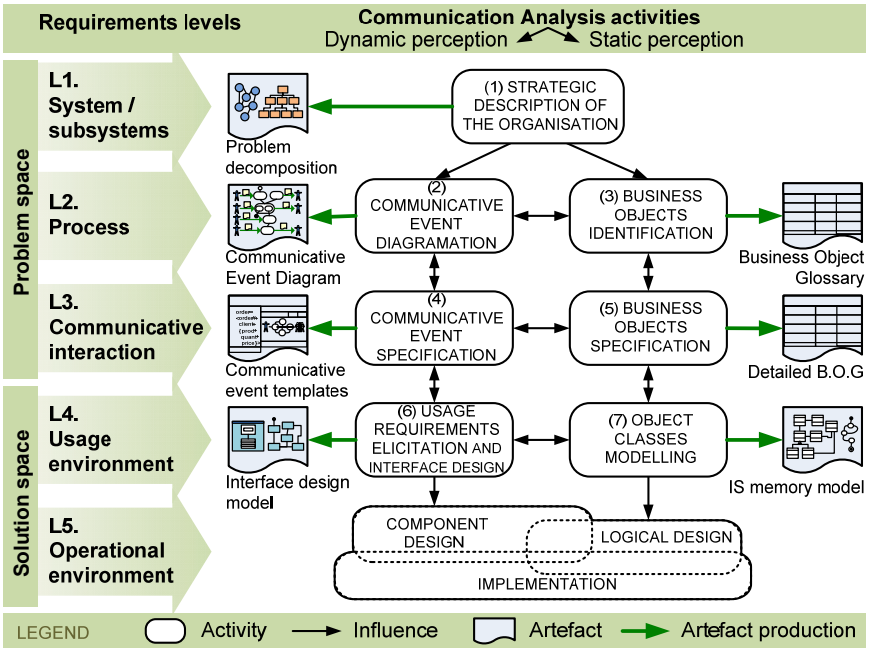


Figure 45. The requirements structure and the Communication Analysis workflow

The requirements discovered in each activity influence the other activities. Often a requirement of a higher abstraction level is refined in lower levels, but horizontal influences also exist. For instance, the bidirectional arrow between activities 2 and 3 represents that the identification of business objects may involve the identification of new communicative events and vice versa. The bidirectional arrow between activities 4 and 5 represents that new business-object requirements may arise from the specification of communicational content, and vice versa.

In Communication Analysis, the focus is put on organisational communications; we consider each discovered requirement to be of one of these kinds:

- *A communicative interaction* between the information system and its environment. E.g. the reception of an order from a customer is an ingoing communicative interaction, whereas a business indicator is an outgoing communicative interaction.
- *A refinable aggregate of interactions*; that is, a set of communicative interactions that can later be refined. A sub-system can be seen as a set of interactions; for instance, the accounting department can stand for the set of communicative

interactions for which the department is responsible. Similarly, a business process can be seen as a set of communicative interactions (e.g. sales management).

- *Part of the specification of a communicative interaction*. For instance, the description of a message, the description of a specific message field.
- *A characteristic of a (set of) requirement(s) of any the previous kinds*. E.g. statements such as "every amount shall be stored with 9 decimals", "every listing for the accounting department shall have a timestamp".

In Communication Analysis, each and every requirement is associated to a specific communicative interaction or to a set of interactions. However, this raises the issue of modularity. How can sub-systems be identified? How many processes should be defined? What is the appropriate granularity of a communicative interaction? When should the analyst stop refining communicative interactions and start designing their support? To answer these questions, Communication Analysis provides guidelines to define requirements models modularly.

In levels L1 and L2, the analyst seeks to refine a complex system into sub-systems and to obtain the repertoire of communicative interactions that the users need for their work practice. From level L2 to level L3 the analyst makes a quantum leap and starts specifying the identified communicative interactions. Another qualitative difference exists between levels L3 and L4, since the analyst (or designer) shifts the focus of the specification from a pragmatic perspective to a semantic and syntactic perspective; the requirements model is added details about how the messages are edited and displayed (i.e. interface design). Also, whereas in L3 the focus was messages (data in motion), L4 focuses on the information system memory (data at rest) intended to ensure the persistence of the communicative interactions. Lastly, the technological architecture is designed and the software application is implemented (this can again involve data in motion in the form of component interaction and network communication, but now the semiotic level is lower).

In the following, the requirements structure is explained in more detail.

Def. 159

A **requirements structure** is a composite thing that serves as a set of rules for representing a system (e.g. an information system) or a computerised information sub-system., in a way that it offers a set of predicators (e.g. the types requirements, taxa of a taxonomy) and a set of relationships among the predicators (e.g. levels of requirements and relationships among them).

4.2.1 L1. System/Subsystems Level

The first requirements level is related to the point of view of organisational structure and strategy; i.e. the information needed to manage the enterprise. When big enterprises are confronted, it is convenient to decompose the problem into sub-systems such as organisational units; this reduces problem complexity (the entropy is reduced). However, most organisations have already defined some sort of organisational structure that the analyst can gather. Also, organisational systems typically have a mission, several organisational goals, a strategy to achieve these goals and some way of measuring the success in achieving the goals. Depending on the scope of the project, the analyst may need to gather this information or even help organisational actors to define it.

Concepts

When organisations are complex, a typical way of structuring them is by defining a hierarchy of organisational units.

- An **organisational unit** is a sub-system of an organisational system, usually conceived with the goal of reducing the complexity of organisational administration and management. Def. 160

Organisational units are usually defined by the managerial staff and they can take different shapes depending on the type of unit and the unity criteria employed to define them (e.g. organisational areas, departments, centres, divisions, directorates, teams).

Organisations usually make use of available installations (e.g. buildings, warehouses, shopping centres) to develop their activities. An installation situated in a specific place is referred as organisational location (a.k.a. business location).

An **organisational location** is a specific place in space and time where a sub-system of the organisational system is situated. Def. 161

Organisational locations are usually defined by the managerial staff and many factors usually influence their situation (e.g. closeness to raw materials and market, availability of communication and power supply infrastructures, government incentives). One organisational unit can be situated in many organisational locations, and vice versa.

An **organisational goal** is a goal that ought to affect the actions of an organisational system (that is, the behaviour of its actors). Def. 162

Organisational goals often refer to organisational units and organisational locations. A top-level organisational goal that is usually defined is the mission statement of the organisation, which is a concise natural-language description of the main purpose of the organisation, its *raison d'être*. According empirical

research by Bart [1997], mission statements typically describe several of the following elements: purpose, values, distinctive competence, desired competitive position, identification of stakeholders, general corporate goals, one big goal, concern for customers, concern for employees, concern for shareholders, and vision. For many purposes, however, we suggest defining these three elements:

- *Key market*. The primary target client or customer of the organisational system.
- *Contribution*. The product or service the organisational system provides to the client.
- *Distinction*. The characteristics of delivered products or services that makes them unique or desirable, so that the client would choose that organisational system in particular.

Although the mission statement helps understanding organisational behaviour, it is not possible to precisely determine whether the organisational system is being successful in pursuing its mission. In order to be able to measure the degree of achievement of a goal, the goal needs to be expressed in terms of business indicators.

Def. 163

A **business indicator** is a metric (a formula) that allows measuring a specific aspect of a set of phenomena occurring in a subject system (or predicting future phenomena), along with the rules that allow to carry out the measurement action and to interpret the result.

A business indicator typically measures or predicts a specific aspect of a kind of actions, actands or actors at any of the systemic levels (thus, this concept often refers to concepts of other layers and ascribing it to L1 is an arbitrary decision). Some examples of business indicators are stock price, consumer confidence, time to market, average number of customer claims, mean response time to a client order, current number of shipments *en route*, computer network bandwidth consumption. Indicators need to have an associated metric; for a precise conceptual framework on metrology, see [ISO/IEC 2007a].

In organisational systems, defining goals is often part of a coarser process of planning the organisational strategy. Strategic planning consists of defining the goals, operationalising them (i.e. defining their corresponding business indicators, a current value and a target value for each indicator) and planning the actions that are expected to lead to achieving the goals.

Def. 164

An **organisational strategy** is the set of interrelated organisational goals that is defined by the managerial staff of an organisational system or sub-system, along with their corresponding business indicators, a current value for each business indicator, a target value for each business indicator, and an action plan intended for achieving the organisational goals.

Since complex organisational systems structure themselves in many organisational units, several organisational strategies are usually defined (e.g. corporate strategy, firm strategy, departmental strategy, a specific product marketing strategy). Strategies defined by different organisational units could, in principle, be conflicting. It is a duty of the managerial staff to solve those conflicts of interest, but the analyst should be aware of them.

Modelling support

Figure 46 shows a view of the Communicating Analysis platform independent metamodel containing the metaclasses that belong to level L1. We acknowledge that these metaclasses are insufficient for modelling complex organisations. It is not the intention of this thesis to provide full support to organisational modelling. The metaclasses in this level provide, at least, a starting point to allow a metamodel extension if needed.

Communication Analysis does not define a specific modelling technique for this requirements level. It neither provides new guidelines for eliciting, specifying or analysing the hierarchy of organisational units, the network of organisational locations or the business strategy. There exist several good methods for this purpose; choosing one or another is contingent upon factors such as enterprise size, maturity, etc. In any case, we provide a metamodel which, although it is simplistic, it can be useful whenever organisational modelling is not crucial for analysing the information system.

In case the hierarchy of organisational units is of a manageable complexity, an organisation chart can be used. However, the analyst should know that using organisation charts entail risks, since they can be misleading: they only depict formal structure but disregard how information and decisions actually flow through the organisation, as well as tacit power relationships [Mintzberg 1979]. Recent research on organisational structure that adopts a constructivist stance can be found in [Dissanayake and Takahashi 2006]. For more versatile metamodels dealing with organisational structure and human resource modelling, refer to the existing literature (e.g. [zur Muehlen 1999]).

In case the network of organisational locations is of a manageable complexity, a list of locations and a map where they can be spotted should suffice. For more complex models and analysis, refer to the literature of management sciences.

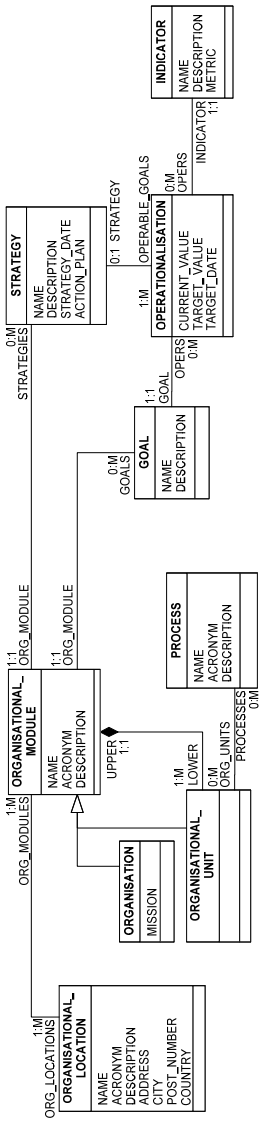


Figure 46. Platform independent metamodel view that corresponds to requirements level L1

In case the business strategy is of a manageable complexity, a simple template such as the one specified in Table 7 should suffice. For the purpose of performing a detailed modelling of organisational goals refer to existing metamodels such as the Business Motivation Model [OMG 2010a]. If an in-depth analysis of organisational goals is intended, then refer to goal-oriented methods such as Tropos [Bresciani, Perini et al. 2004] or i* [Yu and Mylopoulos 1994] (see a comparative evaluation in [Ayala, Cares et al. 2005; Grau, Cares et al. 2006]). Also, if complex business indicators are needed, the analyst can design a Balanced Scorecard [Kaplan and Norton 1996].

Table 7. Simple template for defining the business strategy

FIELD	OP	DOMAIN
STRATEGY =		
< Name +	i	text
Strategy date +	i	date
Description +	i	text
Action plan +	i	text
Organisational module +	i	Organisational module
OPERATIONALISATIONS =		
{ OPERABLE GOAL =		
< Goal +	i	Goal
Business indicator +	i	Business indicator
Current value +	i	[number | text]
Target value +	i	[number | text]
Target date +	i	date
>		
}		
>		

Example

In order to illustrate Communication Analysis, we use a case about a fictional company named SuperStationery Co. that provides stationery and office material to its clients. The company acts as an intermediary: the company has a catalogue of products that are bought from suppliers and sold to clients. The mission statement of the company is "to provide small and medium enterprises with a wide range of high-quality, moderately priced office material, serving and shipping the goods diligently within regional distance". The case is described in full detail in [España, González et al. 2011].

The SuperStationery Co. case is of manageable size. Although it is a small enterprise, the company has a director and is divided into five organisational units that correspond to departments; namely, Management Department, Sales Department, Logistics Department, Quality Assurance Department and Insurance Department. There is a single organisational location so no particular specification has been made of this aspect.

Table 8. Relationship among business processes and departments

Business process		**Departments**				
Acro-nym	**Name**	**Manage-ment**	**Logis-tics**	**Quality Assess.**	**Sales**	**Insu-rance**
CLIE	*Client management*	X			X	
PROD	*Product management*	X		X		
LOGI	*Logistics*		X			
SALE	*Sales management*		X		X	X
RISK	*Risk management*					X
ACCO	*Accounting*				X	
SUPP	*Supplier management*	X			X	

The company work practice has been decomposed into seven business processes (that can be considered sub-systems), each having a name (e.g. *Sales management*) and an acronym (e.g. *SALE*). Table 8 relates the business processes and the departments that participate in them.

Some business processes fall entirely within the scope of a single department (e.g. *Logistics* business process), while others crosscut several departments (e.g. *Sales management* business process).

Currently, the information system is supported by means of paper forms and filing cabinets. It is the first time that the company undertakes a computerisation project. They are interested in improving the efficiency of their paperwork but, for the moment, the company has displayed no interest in defining business indicators that would provide them better monitoring of their performance (this will probably be considered in future projects). In any case, possible strategic indicators ascribed at level L1 are the increase in the number of clients and the yearly revenue.

4.2.2 L2. Process Level

Communication is the essence of business process analysis in Communication Analysis. The analyst seeks to discover the set of communicative interactions of the enterprise. The aim is to discover communicative interactions between the information system and its environment, and to describe them taking into account their dynamic and static aspects; that is, creating the Communicative Event Diagram and the Business Objects Glossary, respectively.

Concepts

In the following, a series of definitions clarify the concepts upon which the modelling techniques are built.

> An **organisational actor** is a goal-pursuing actor that interacts with the organisational system in some way. Organisational actors either belong to the organisational system (e.g. they are bound to the organisational system by, e.g., a contractual relationship, and thus their actions are expected to conform to organisational norms) or they are external parties (i.e. they belong to the subject system but not to the organisational system).

Def. 165

Although non-human actors can also be organisational actors (e.g. a sensor that reports the temperature of a brewing tank), most of the times organisational actors are human. For instance, members of the enterprise (John, Mary, Frank, Joseph), customers (Phil, Nathalie) and suppliers (Phil, Christine) are organisational actors. For the purpose of analysing information systems, it is convenient to group them according to their roles within the organisational work practice.

> An **organisational role** is a type of organisational actors that are expected to perform a certain type of actions conforming to the organisational norms; they are often considered to have a similar goal in common, or to have a similar function.

Def. 166

Organisational roles are usually defined by the managerial staff and they can be determined intensionally (e.g. "a salesperson is a worker whose duty is to sell the products of the company") or extensionally ("Mary and Frank are salespersons"). Several organisational actors can play the same organisational role (e.g. both Mary and Frank are salespersons). Also, the same actor can play different organisational roles (e.g. Phil is both a customer and a supplier).

Organisational actors belonging to the organisational system can perform many different kinds of actions intended to achieve organisational goals. Organisational roles are assigned a set of rights and obligations; for instance, responsibilities (the duties that organisational actors belonging to that role ought

to perform), the type of information they are entitled to know or manage, the set of other organisational actors they are in charge of, etc. For instance, "salespersons attend clients when they place orders; to do so..." Besides defining which actions ought to be performed by each organisational role, organisational norms also define how those actions are carried out. Business process model denotations represent this aspect of the organisational norms by specifying predefined flows of actions (e.g. how a business product is produced, how it is shipped to the customer).

Def. 167

A **business process** is a sub-system of the subject system, often expressed as an organisational norm defining (among other things) a set of actions that intend to fulfil one or several organisational goals, the organisational roles that take part in those actions, the state-transition structures that relate those actions, and the expected pre-states (e.g. technology, materials, installations) and post-states (e.g. products being created, services being provided) of the actions.

From a communicational perspective we are interested in the relationship between business processes and the information system. If organisational actors carried out their actions without communicating, one actor would not know about the state of affairs (e.g. about what other actors did before) and therefore coordinated action would be hampered, if not impossible at all... Then the chances of achieving the organisational goals would be low.

Organisation necessarily entails communication; in other words, communication is absolutely essential to organisations [Simon 1976]. Communication Analysis conceives the following relationship between organisational action and organisational communication. Business processes fall within the realm of the subject system, so many business-process actions (informally, the activities that compose the business process) are often regarded as regulated transitions and, therefore, their occurrences are regulated-transition occurrences. This implies that the organisation (meaning some of the organisational actors) is probably able to define what information is relevant for the organisational system about the action occurrence; in other words, for some actions it is possible to agree a regulated-transition information definition.

As argued in Section 3.3.5 communication often takes place in the form of a dialog (a set of message transfers). In the context of Communication Analysis, we refer to dialogs as communicative interactions[46].

[46] The term interaction is used as a synonym of the term co-action (see Def. 27).

Def. 168

A **communicative interaction** is an exchange of messages, i.e. a sequence of mutual and alternating message transfers between at least two organisational actors, called interacting partners (being either human or non-human), whereby these messages contain some data about the subject system and are expressed in languages understood by all communication partners, and whereby the action context and the goal of the communicative interaction is made present in all interacting partners.

This definition is almost equivalent to that of communication (see Def. 77), except for the fact that it allows for interacting partners to be non-human. This way, a sensor or a software application can participate in a communicative interaction.

Depending on the main direction of communication, the following types of communicative interactions can be distinguished:

- *Ingoing communicative interactions* primarily feed the information system memory with new meaningful information. For instance, the placement of an order by a client, the reporting of a fault in a packer machine by an operator, the decision of the director of the Department of Risk Management to accept an investment. These interactions are often (but not always) supported by business forms.
- *Outgoing communicative interactions* primarily consult information system memory. These interactions are often materialised as business indicators, listings and printouts. For instance, the list of clients with a debt that is greater than 6000€, a cash flow chart, a receipt, a payroll list.

In the following, rigorous definitions are provided.

Def. 169

An **ingoing communicative interaction** is a communicative interaction whereby one of the interacting partners is a regulated transition observer and another interacting partner belongs to the information system. The main communicative goal of the interaction is to report to the information system a regulated transition occurrence, for which purpose the regulated transition observer conveys pieces of regulated-transition information to the information system actor.

Def. 170

An **outgoing communicative interaction** is a communicative interaction whereby one of the interacting partners belongs to the information system and another interacting partner is a recalled facts receiver. The main communicative goal of the interaction is to retrieve knowledge (i.e. recalled facts) from the information system and to convey it to the recalled facts receiver.

Industrial experience has shown us that ingoing communicative interactions entail more analytical complexity whereas the complexity of outgoing

communicative interactions is related to their design. Therefore, we advise the analyst to focus, first of all, on ingoing communicative interactions[47]. This way, from ingoing communicative interactions we will derive the memory of the information system (e.g. a data model); once the memory is defined, the outgoing communicative interactions can be designed more easily.

However, not every ingoing communicative interaction is equally important for Communication Analysis. For instance, an interaction that provides no information that is actually new for the information system can be disregarded during early analysis. Also, the analyst needs to take into account model modularity; whether some phenomena should be conceived as only one interaction or as several interactions really matters when applying Communication Analysis. For the purpose of clarifying these issues, we now present the concept of communicative event.

A *communicative event* is the organisational action that is triggered as a result of a given change in the world (i.e. in the subject system), intended to account for that change by gathering information about it. In each communicative event, someone conveys a message containing new meaningful information to the information system (this message is specified by means of a message structure in level L3). Communication Analysis offers guidelines to allow identifying communicative events (communicative event unity criteria), also enabling modularity. This way, a communicative event can be seen as an ingoing communicative interaction that fulfils the communicative event unity criteria.

Communicative event unity criteria are a set of unity criteria that serve the purpose of guiding the information systems modelling action. It consists of three rules; namely trigger unity, communication unity and reaction unity.	Def. 171
A **communicative event** is a composite transition; more specifically, a unitive conception unifying several conceptions about an ingoing communicative interaction and the corresponding reaction of the organisational system, whose modularity fulfils the communicative event unity criteria.	Def. 172
A **communicative event occurrence** is a transition occurrence with the same conditions as applying for the notion of communicative event.	Def. 173

[47] Communication Analysis also takes into account outgoing communicative interactions. However, when the organisational system needs complex indicators for performance management, advanced methods such as the Balanced Scorecard are recommended.

Note that the term even has been overloaded with different meanings by different authors.

- In the literature on real-time, embedded, and control systems, the concept of event is often used in a broad sense (disregarding whether the transition occurrence is external or internal to the system).
- In programming environments, the term event designates any stimulus that interrupts a system component activity.
- The same happens in object-oriented analysis and design techniques. For example, an individual stimulus from one object to another is an event for [Rumbaugh, Blaha et al. 1991]. Also, "in the context of state machine, an event is an occurrence of a stimulus that can trigger a state transition" [Booch, Rumbaugh et al. 1999]. Olivé and Raventós [2006] define a structural event as "an elementary change in the population of an entity or relationship type" (up to five kinds of primitive structural events are defined in [Costal, Olivé et al. 1997]); this notion of event is often within the modelling scope of most object-oriented conceptual modelling approaches.
- However, we use the term event in a way similar to [ISO 1987] and [Falkenberg, Hesse et al. 1998]. In the context of our work, an event is a stimulus that occurs in the outside world and to which the organisational system must respond [Yourdon 1989]. It is related to the notion of domain event, which is "a state change that consists of a set of one or more structural events that are perceived or considered as a valid change in the domain" [Olivé and Raventós 2006].

The communicative event unity criteria consist of three criteria that guide business process model modularity. Trigger unity implies that the event occurs as a response to an external interaction and, therefore, some actor triggers it. Communication unity implies that each and every event involves providing new meaningful information. Reaction unity implies that the event is a composition of synchronous activities. If an encapsulated part of the organisational work practice fulfils the trigger unity and the communication unity, then it can be considered a logical communicative event. If an encapsulated part of the organisational work practice fulfils the communication unity and the reaction unity, then it can be considered a physical communicative event. Logical events are far more important to Communication Analysis than physical events; unless otherwise noted, whenever communicative event is mentioned we refer to logical communicative events. Unity criteria are further discussed in Section 4.3.1.2 .

A specialised communicative event indicates that different alternatives in the path through the business process are possible[48]. Specialisation can be induced

[48] This type of specialisation is somehow related to business process variability, although further investigation is needed to clarify the similarities and the differences between both notions.

externally, either by structural or domain variants in the message structure that is associated to the event, or by a decision explicitly made by the primary actor. Specialisation can also be induced internally, as a result of previous events. We refer as *event variant* to each alternative behaviour within a specialised communicative event. We advice using an encapsulated notation for specialisation (i.e. to graphically encapsulate all the event variants within the specialised communicative event) to reinforce the notion of unity. Each event variant has a corresponding *specialisation condition*, which is a well-formed formula that can refer to one or several fields of the message structure (in case the specialisation is externally induced). However, the specialisation condition is typically defined in terms of the content of the message, so it belongs to the level L3.

Def. 174

An **event variant** is each of the alternative transitions (if any) that take part in the state-transition structure that defines the composition of a communicative event. Such event is said to be a "specialised" communicative event.

Within each communicative event, several business actors may interact, each of them playing a specific *communicative role*.

The *primary actor* of a communicative event is the actor that is responsible for communicating the new meaningful information to the information system (e.g. a client that formulates a request). It is usual that the primary actor also triggers the communicative interaction (i.e. they initiate the interaction by establishing contact with the organisational system; e.g. a client comes by and places an order), but it is not always the case (e.g. a sales representative can visit a client and convince him to place an order).

Def. 175

A **primary role** for a given communicative event is a set of organisational actors that are expected to act as regulated-transition observers, in the action context of that communicative event; that is, the set of organisational actors that are designated responsible for reporting to the information system an occurrence of a certain type of communicative event, by providing data.

Def. 176

A **primary actor** is an organisational actor playing a primary role in the action context of a communicative event.

A *receiver actor* is expected to be communicated some information as a result of a communicative event (e.g. the Sales Manager receives a copy of the order form).

A **receiver role** for a given communicative event is a set of organisational actors that are expected to act as a recalled facts receivers in the action context of a communicative event; that is, the set of organisational actors that (having the rights to do so) request the retrieval of data from the information system memory (i.e. the domain model denotation). Def. 177

A **receiver actor** is an organisational actor playing a receiver role in the action context of a communicative event. Def. 178

With regards to the behaviour of the organisation, organisational norms usually define the set of states in which the occurrence of each communicative event is admissible. An intensional definition of such set is usually referred to as the precondition of the communicative event.

The **precondition** of a communicative event is a rule determining the states in which the communicative event is allowed to occur (i.e. the possible pre-states). Def. 179

Preconditions often establish temporal relationships among communicative events. In Communication Analysis, we are interested in precedence relations among communicative events.

A *precedence relation* between two communicative events A and B indicates that event A must necessarily occur before B occurs. This can as well be expressed as "A is a direct precedent (communicative event) of B" or "B is a direct successor (communicative event) of A".

A **precedence relation** between two communicative events (say, A and B) is a rule consisting of a relationship that defines the relative time between them. It indicates that, for communicative event B to occur, A has necessarily occurred before. A is said to be a "direct precedent" (communicative event) of B; conversely, B is said to be a "direct successor" (communicative event) of A. The precedence relation is part of the precondition of the "direct successor" communicative event. Def. 180

Following the FRISCO notation for relationships: {<A, before>, <B, after>}.

Logical nodes (a.k.a. logical gates) can be used to complement the semantics of the precedence relation, in order to define more complex preconditions. They are means to express logical formulas that include mainly "or" and "and" operators (although other logical nodes are possible). Their specific use and semantics depend on the notation; the logical nodes provided by the Communicative Event Diagram are explained in Section 4.3.1.1.

The previous definitions mainly address dynamic aspects of the systems under analysis. The static aspects correspond to the concept of business object. We should distinguish between a "physical" business object (the thing with which the business works; e.g. a specific T-shirt), a class of business objects (a set of business objects of the same type; e.g. T-shirts), the information that the organisation wants to know about a class of business objects in order to work with or make decisions concerning a business object (e.g. the colour, the size and the type of textile of a T-shirt), and a business object representation (the representation of the information about a business object).

A **business object** is a composite thing of the subject system, which an organisational system is interested in acquiring knowledge about. Def. 181

A **business object class** is a type of business objects. Def. 182

At the requirements level L2, classes of business objects are simply identified; their informational structure is specified at level L3.

Modelling support

Figure 47 shows a view of the Communication Analysis platform independent metamodel containing metaclasses that correspond to level L2. Some metaclasses from level L1 are represented in grey so as to specify how they are related to this level.

For the description of the dynamic view of the information system, Communication Analysis proposes to specify the flow of communicative events by means of the *Communicative Event Diagram* (CED). It is recommended that, for each business process, a communicative event diagram is created.

Communicative Event Diagram is a modelling technique intended for the representation of the communicative events of (part of) an information system. It also depicts the precedence relations among communicative events, the primary actors of ingoing communicative interactions and (optionally) the receiver actors of the outgoing communicative interactions and the involved support actors. Def. 183

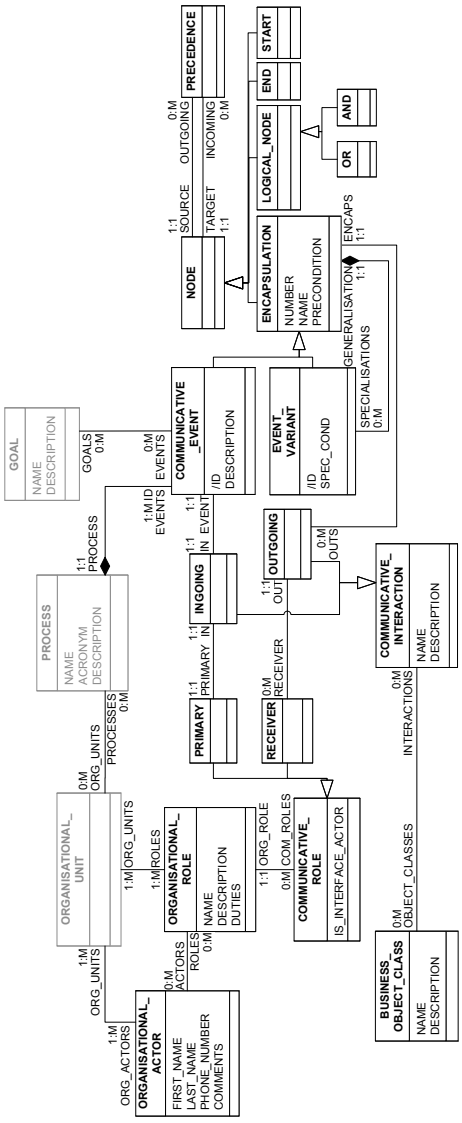


Figure 47. Platform independent metamodel view that corresponds to requirements level L2

The primitives of this modelling technique are shown in Figure 48 and are explained in detail in Section 4.3.1.

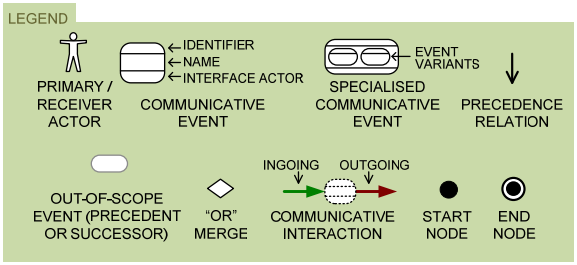


Figure 48. Modelling constructs of the Communicative Event Diagram

Apart from depicting communicative events, the Communicative Event Diagram specifies precedence relations among events and the organisational roles that play each communicative role; that is, which organizational roles can play the role of the primary actor, the receiver actor and the support actor. Note that the two types of roles (organisational and communicative) are orthogonal.

For the description of the static view of the information system, Communication Analysis proposes to specify classes of business objects by means of a *Business Object Glossary*. This glossary has two different levels of detail:

- At the requirements level L2. Process, it is enough to identify business object classes and describe them briefly with natural language, also gathering some examples of business object representations (e.g. a client order form).
- At the requirements level L3. Communicative interaction, the business object classes are described with greater detail, specifying their corresponding templates (i.e. the pieces of data about each type of business object that the organisational system wants the information system to store).

Additionally, those business indicators that measure any aspect about business processes or business objects correspond to this layer. These indicators allow an organisation to monitor their running and performance (e.g. the state of an order, number of outstanding orders waiting to be served, mean response time to an order).

Example

As said above, the company is divided into five departments; Figure 49 shows the organisation chart, which specifies the most relevant organisational roles.

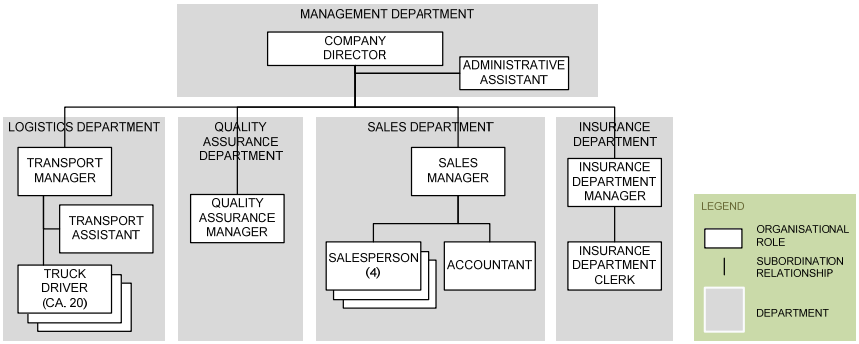


Figure 49. Organisation chart of SuperStationery Co.

The business processes are specified by means of communicative event diagrams. For instance, Figure 50 presents the communicative event diagram that corresponds to the *Sales management* business process.

At this requirements level, some business indicators that are associated to sub-systems or processes can be elicited. Although SuperStationery Co. is mature enough to consider measurements, some indicators they could be interested in the future are productivity and profitability indicators (e.g. delivery performance to customer, ratio of order acceptance of a supplier), client-related indicators to monitor customer loyalty (e.g. consumption rates), payments and takings indicators to monitor debts (e.g. client indebtedness).

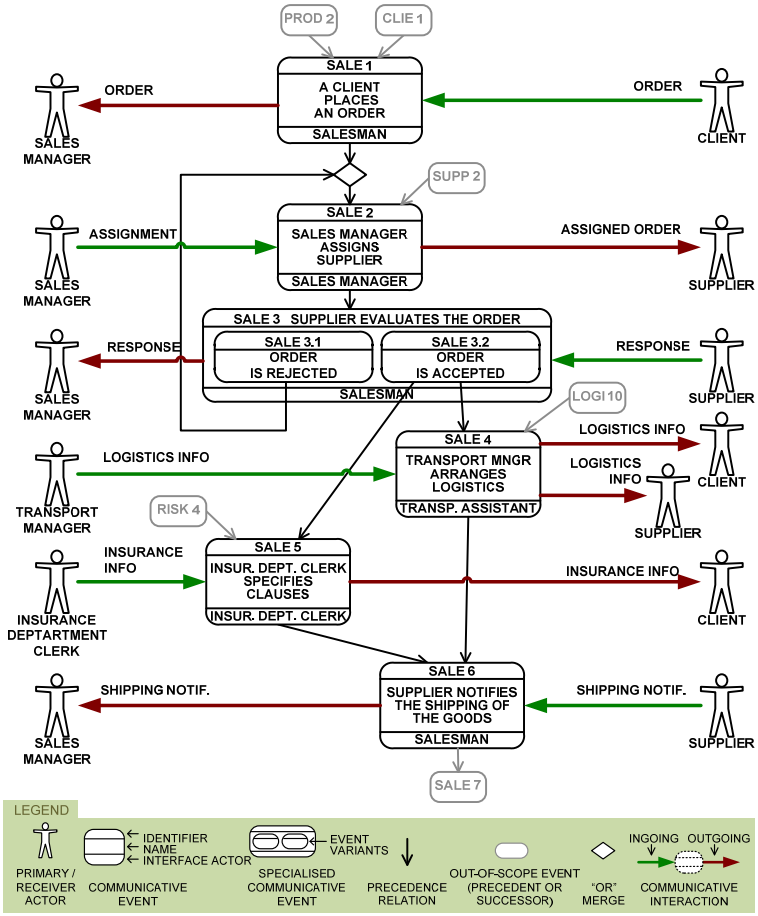


Figure 50. The SuperStationery Co. *Sales management* business process depicted as a communicative event diagram

4.2.3 L3. Communicative Interaction Level

In the previous requirements level, business process analysis is carried out with the intention to discover communicative events. The aim of this requirements level is describing them.

Concepts

Communicative events that appear in the communicative event diagram need to be specified in detail. Requirements related to communicative events are further structured in three wide categories: contact, communication and reaction requirements. Each category can contain requirements of several types; however, not all types of requirements correspond to this requirements level. We provide just some examples of types of requirements that correspond to level L3:

- *Contact requirements* are related to the triggering of the event by an actor who wishes to communicate something to the information system.
  - *Precondition* of the communicative event.
- *Message requirements* specify the contents of the message being communicated to or from the information system. This category of requirements needs to include a description of the message conveyed by the primary actor to the information system. The requirements concerning the message (its structure and constraints) conform the regulated transition information definition (see Def. 104).
  - *Structure of the message* that is conveyed to the information system.
  - *Composition* of message fields.
  - *Domains* of the message fields.
  - *Message constraints* over the structure of the message (e.g. cardinalities), the context of communication (e.g. domains of the message fields), particularisations of the messages to organisational roles depending on the information that concerns them, etc.
- *Reaction requirements* are related to the information system reaction to the conveyed message. At the level L3, these requirements are related to distributing the information to other actors so that they can act accordingly, and conditioning future behaviour to the result of this event.
  - *Linked behaviours* refer to how the occurrence of this communicative event affects future occurrences of events. For instance, the information provided in this communicative event can condition certain future behaviours of the system. This includes business rules or complex conditions (e.g. decision tables) that determine future reactions depending on the values provided in the communicative event.

- *Linked communications* specify to whom the occurrence of the communicative event must be communicated; that is, which organisational actors (or roles) need to know about this communicative event so as to take further actions (e.g. make decisions).

Message requirements define the message that corresponds to a communicative interaction; we refer to this as message structure.

A **message structure** is a model of the message that corresponds to a communicative interaction. Def. 184

Also, a communicative event can be mapped to the organisational goals that it contributes to fulfil. This is in line with Dynamic Interpretation Theory. According to this theory, each dialogue has at least one underlying non-communicative goal that motivates it; since the ingoing communicative interaction of a communicative event is a dialogue, then each communicative event has at least one non-communicative (organisational) goal.

Organisational norms usually include templates that define the types of information that the business wants to know about each business object class. The information about business objects (business object representations) is maintained by the information system in order to facilitate the actions of the organisational system concerning that business object.

A **business object class property** is an organisational norm that defines a piece of data that is relevant for the organisational system in order to know about a specific business object class. Def. 185

A **business object representation** is a representation of the knowledge about a business object. It must be as compliant as possible with the information definition of the corresponding business object class (i.e. it must include the representation of the business object class properties). Def. 186

All the business object class properties related to the same business object class constitute the definition of the relevant information about the class and this definition is part of the information system language representation, whereas business object representations are part of the domain model denotation.

Business object information definitions are usually complex aggregates of business object class properties. At this stage, business object classes are not specified from a (typical) object-oriented perspective. Object orientation splits business knowledge into homogeneously identifiable aggregates of properties whereas, at this level, the interest is in conceiving business objects as a whole. For instance, users consider an order to be a single business object, but object orientation splits an order into several object classes (e.g. order head and order lines). At this requirements level, the structure of properties that are of interest for each business object is specified but keeping the view of the object as a whole.

Modelling support

Figure 51 shows a view of the Communication Analysis platform independent metamodel containing the metaclasses that correspond to level L3.

In the metamodel, some types of requirements have been supported in two complementary ways: as textual requirements and within the construct to which they affect. For instance, the domain of a message structure data field can also be specified as a textual message requirement of the type contextual constraint. We recommend is that, as long as possible, a requirement is expressed within the construct itself (in the previous example, the domain of a data field should be specified using the attribute Domain of the metaclass DATA_FIELD. The reason is that textual requirements are not easily processed, either for verification or for model transformation purposes, so they should only be used when no other means is available.

The taxonomy of textual requirements is completed in level L4. Even so, it should only be regarded as an exemplification of the idea that all requirements in a requirements model should be categorised properly. We acknowledge that more research is needed to develop a taxonomy that fits Communication Analysis. This is part of our plans for future work. For the moment, given the limitations and inconveniences of the current state of the taxonomy and its support, we only provide these metaclasses in the platform independent metamodel (the complete metamodel is depicted in Section 4.5). In the platform specific metamodel, intended for implementing the tool, the taxonomy is replaced with a metamodel of such.

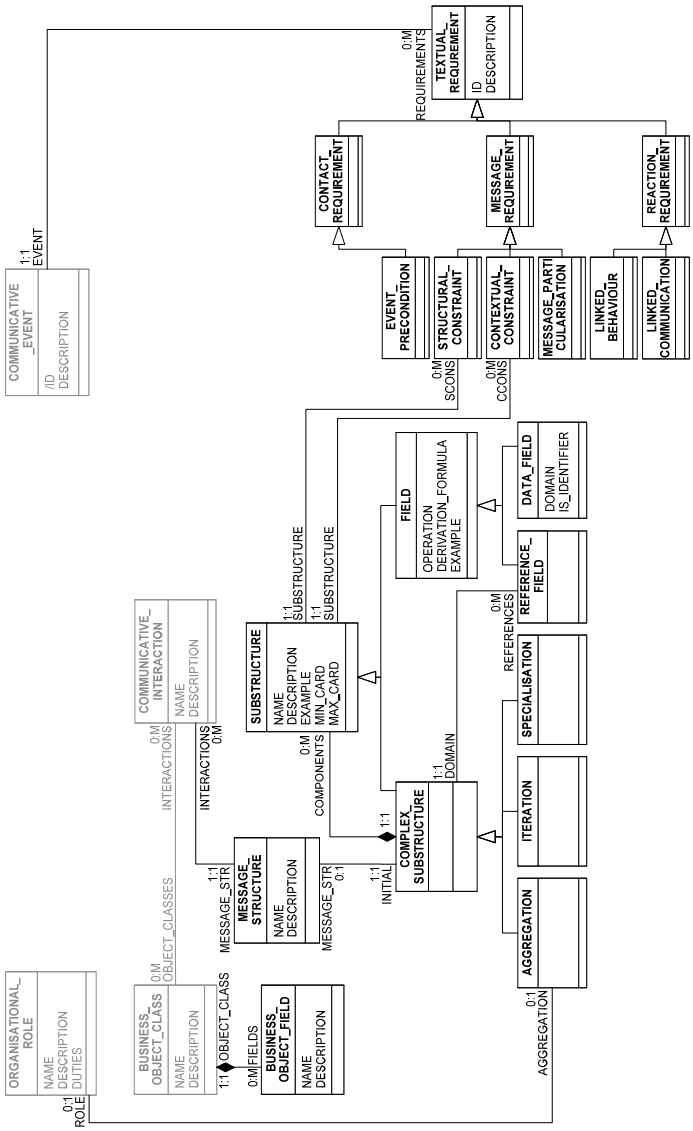


Figure 51. Platform independent metamodel view that corresponds to requirements level L3

For the description of the information system dynamic view, Communication Analysis proposes to specify the requirements related to communicative events by means of the *Event Specification Template* modelling technique.

> **Event Specification Templates** is a modelling technique intended for specifying the requirements associated to a given communicative event.

Def. 187

This modelling technique is specified in more detail in Section 4.3.3. It mainly consists of a textual template structured in four sections, one section that contain a general description of the event[49] and the organisational goals it serves, and one section for each category of requirements (contact, message and reaction). Within the message requirements, it is important to describe the message conveyed by the primary actor in the ingoing communicative interaction. To do so, a modelling technique named Message Structures is proposed.

Message Structures is a modelling technique that allows describing, by means of structured text, the message that is associated to a communicative interaction.

> **Message Structures** is a modelling technique intended for specifying the message that corresponds to a communicative interaction.

Def. 188

The technique is usually applied to specify the message associated to the ingoing communicative interaction of a communicative event; that is, the message conveyed the primary actor of the communicative event. In such cases, rigorously speaking, the resulting message structure specifies the regulated-transition information definition associated to the communicative event. However, it can as well be used to specify outgoing communicative interactions and, at lower requirements levels, it also plays a role on interface design.

Table 9 shows an overview of the grammatical constructs of this technique; for further details see Section 4.3.2.

At this point in time, Communication Analysis does not offer much support to mapping communicative events and organisational goals. Although the Event Specification Template accounts for this mapping, this support could be improved (we plan to overcome this limitation in the future).

For the description of the static view of the information system, Communication Analysis proposes to specify classes of business objects by means of a *Business Object Glossary*. As mentioned above, at the requirements level L3, the business objects are specified in detail, indicating their fields.

---

[49] Communicative events should always be summarised in plain text, taking into account that future readers of the requirements model may not be familiar with the organisational work practice.

Table 9. Overview of the grammatical constructs of the Message Structures modelling technique

Substructure				
Field		Complex substructure		
Data field	**Reference field**	**Aggregation substructure**	**Iteration substructure**	**Specialisation substructure**
a a represents a data whose domain is basic.	b b represents a reference to an already-known business object.	A=< a + b + C > A is composed of the data field a, the reference field b and the complex substructure C.	A = { a + b + C } A represents a set of substructures that, in turn, are composed of the data field a, the reference field b and the complex substructure C.	A = [a | b | C] A is composed of the data field a, or the reference field b, or the complex substructure C, exclusively.

Example

Table 10 shows the message structure associated to the communicative event *SALE 1*. Note that the structure of message fields lies vertically (at the right). Properties of the message fields are arranged horizontally; namely, the information acquisition operation, the data domain, and an example value provided by users. The description of the message fields is provided in Table 11.

The message structure itself specifies the composition of the message and it includes some structural constraints. For instance, the fact that a client order can have many destinations is already represented by the aggregation substructure DESTINATIONS (it is therefore redundant to include them as textual constraints, but it may be done to improve understandability). Some constraints, however, cannot be expressed in the message structure and are specified separately. For instance, the fact that an order line can only refer to a single client order cannot be expressed in the message structure; it can be specified as a textual constraint.

Also the message structure specifies some contextual constraints. For instance, the facts that the field Request date must contain a date and that the field Client refers to a previously registered client are specified by the field domain. Other constraints cannot be expressed in the message structure and are specified separately. For instance, the facts that users identify orders by means of the Order number and that the product price should be the current price of the product in the catalogue cannot be expressed in the message structure; they can be specified as a textual constraint.

Table 10. Message structure of the event *SALE 1. The client places an order*

FIELD	OP	DOMAIN	EXAMPLE VALUE
ORDER =			
< Order number +	g	number	10352
Request date +	i	date	31-08-2009
Payment type +	i	text	Cash
Client +	i	Client	56746163-R, John Papiro Jr.
DESTINATIONS =			
{ DESTINATION =			
< Address +	i	Client address	Blvd. Blue mountain, 35-14A, 2363 Toontown
Person in charge +	i	text	Brayden Hitchcock
LINES =			
{ LINE =			
< Product +	i	Product	ST39455, Rounded scissors (cebra) box-100
Price +	i	money	25,40 €
Quantity >	i	number	35
}			
>			
}			
>			

Table 11. Separate table defining the meaning of the fields that compose the message structure ORDER (see Table 10)

Field	**Description**
Order number	A sequential number that identifies the order.
Request date	The date in which the client places the order.
Payment type	Information about the payment type. Its value is normally either Cash, Credit or Cheque, but the salesman can freely indicate any other information here.
Client	The client that places the order.
Address	A client destination at which the products have to be delivered.
Person in charge	The name of the person that will receive the order at the destination. From one order to another, this person can be different.
Product	A product that is requested by the client.
Price	The price of the requested product.
Quantity	The amount of items of the product that the client requests at a specific destination.

Table 12 shows some structural and contextual constraints over the message structure of event *SALE 1*. Although we acknowledge that, in this case, the constraints are quite simplistic, they at least illustrate the intention of such requirements.

Table 12. Some textual requirements of level L3 concerning event *SALE 1*

Message requirements
Structural constraints
One order can have many destinations. One destination can have many lines. An order line only refers to a single client order.
Contextual constraints
Orders are identified by Order number. The product price in the line takes its value from the current price of the product in the catalogue.
Reaction requirements
Linked communications
The Sales Manager is informed of the order placement.

With regards to reaction requirements, whenever *SALE 1* occurs, the Sales Manager needs to be informed of the order placement because he is in charge of assigning the order to a supplier. Although this is already specified in the communicative event diagram by means of an outgoing communicative interaction (the red arrow from *SALE 1* to the Sales Manager receiver actor in Figure 50) this can also be expressed as a textual requirement (it is redundant but it can be done for the sake of understandability or in such cases where this type of requirement is especially important).

Event *SALE 1* has no linked behaviours, except for the fact that *SALE 2* is expected to occur (but this is already expressed by a precedence relation in the communicative event diagram). In the SuperStationery case, there is a linked behaviour in event *SALE 3* that has been expressed textually as follows: "In case the order is rejected, the Sales Manager has to reassign it to a different supplier."

4.2.4 L4. Usage Environment Level

This requirements level mainly deals with interface and information system memory design. This is not only relevant to computerised information systems; paper-based information systems do also have an interface and a memory that need to be designed. However, in the following, we will consider that computerisation is intended.

Since this requirements level is discarded in the integration with the OO-Method, the concept definitions that follow do not intend to be as rigorous as in previous levels. The concepts about the computerised information system interface are further explained in [España 2005]. Moreover, with regards to data modelling, Communication Analysis does not offer new relevant concepts; therefore, we do not provide many definitions on this topic.

Concepts

Crossing the frontier between levels L3 and L4 implies moving from the specification of organisational behaviour to the specification of how this behaviour is supported. It implies starting to focus on the technology, on a particular solution to the problems stated by the stakeholders. The usage environment is an abstract model of the information system in which the modeller starts looking inside the information system (e.g. communicative interactions are analysed from the point of view of Shannon's model, the system memory is designed, the information system interface becomes important).

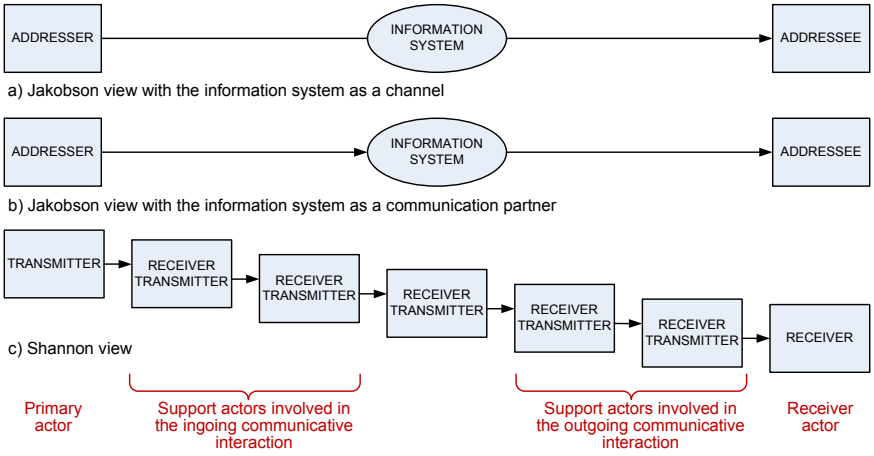


Figure 52. Three communicative views on information systems

In L4, the analyst trespasses the encapsulation of the communicative event and defines its internal composition. Now the communicative event is looked at as a set of actions related to information (acquisition, storage, processing, retrieval and/or distribution), which are carried out in a complete and uninterrupted way, on the occasion of an external stimulus [González 2004].

In fact, communicative interactions are often a complex process that involves other actors apart from primary and receiver actors. We refer as these actors as support actors because they make communication possible; they support message transfers in some way. A *support actor* is responsible of interacting with the information system in order to convey the message (e.g. the salesperson that fills the order form, the office boy that delivers internal mail).

A **support role** for a given communicative event is a set of organisational actors that, in the action context of that communicative event, participate as interacting partners, not being a primary actor or a receiver actor. For instance, the support actor can participate as the channel of a message transfer, or as an interface actor.	Def. 189
A **support actor** is an organisational actor playing a support role in the action context of a communicative event.	Def. 190

Note that the concepts of primary role (Def. 175) and receiver role (Def. 177) correspond to the addresser role and the addressee role (respectively) of Jakobson's model of the communicative act (see Section 2.2 for a brief explanation of this theory). Therefore, they can be identified in a completely computation-independent (or technology-agnostic) way; for instance, a client is always the source of the information when placing an order disregarding how the communicative interaction takes place and which is the supporting technology. In turn, support roles are related to Shannon's model of communication (see Section 2.2). They correspond to those sender and receiver roles that do not coincide with the addresser and addressee roles of the superjacent Jakobson's model. They are usually indentified by investigating how communicative interactions actually take place and this often implies already being aware of the underlying technology. Figure 52 offers three views on the communication that can take place in a communicative event.

There are some types of requirements that apply to communicative events but belong to the usage environment level. We provide just some examples:

- *Contact requirements* are related to the triggering of the event by an actor who wishes to communicate something to the information system.
  - *Communication channels* that the primary actor can use in order to communicate with the information system.

- *Availability requirements* and constraints that refer to the degree to which the information system is in a position to engage in the ingoing communicative interaction.
- *Medium requirements* that refer to the technology (this includes paper-based forms) that supports the ingoing communicative interaction.
- *Accreditation requirements* that refer to the protocols that the organisational system prescribes for each actor participating in the ingoing communicative interaction (i.e. informally, how to know that actors are who they say they are).
- *Verification requirements* that refer to a) ensuring that the provided documentation (if any) is not fraudulent, and b) confirming the influx of physical elements associated to the ingoing communicative interaction (e.g. stock coming into a warehouse).

- *Message requirements* specify the contents of the message being communicated to or from the information system. This category of requirements needs to include a description of the message conveyed by the primary actor to the information system.
  - *Derivation* of field values: derived fields are a design aspect that is ascribed to the usage environment.
  - *Message constraints* such as particularisation requirements for a specific organisational actor, etc.
- *Reaction requirements* are related to the information system reaction to the conveyed message. At the level L3, these requirements are related to processing the information, and updating the system memory.
  - *Data model view* related to the communicative event; it is the part of the memory of the information system that the communicative event contributes to build.
  - *Treatments* that define what changes occur in the information system as a result of the communicative event (e.g. what processing takes place, what information is stored). It involves defining how the information acquisition is related to the data model.

The computerised information system memory stores a representation of the subject system phenomena that the organisational system needs for taking action and making decisions. Current database technologies force the normalisation of data models so business object classes of level L3 need to be fragmented into normalised business object classes. A *normalised business object class* is a fragment of (part of) a business object class according to a given set of normalisation rules. In the area of object-oriented modelling this concept is referred to as a class (of objects); in the area of relational data modelling this concept is referred to as relation or table. In the following chapter we integrate Communication Analysis

with the OO-Method and the computerised information system memory is specified by means of classes of objects.

In the usage environment, the essential interactions are related to the *editing* of the messages that are associated to communicative events. This editing occurs in the information system interface and it is carried out by an interface actor (see Def 114). It is important to identify which organisational role is going to play the role of the interface actor for each communicative event; that is, who is in charge of editing input messages according to the structure and the codes required by the information system. It may be one of the support actors involved in the ingoing communicative interaction or the primary actor himself/herself.

In order to edit messages, the user needs to navigate through the interface and to *locate* the corresponding editorial environment, to *trigger* a start signal to begin the editing process, and to trigger an end signal that initiates the expected reaction of the system. Editing, location and trigger are the types of interactions that guide usage requirements elicitation. At this requirements level, interface design is done by means of abstract interface patterns. Also, requirements related to ergonomics are addressed (e.g. layout, look and feel).

In enterprise information systems, the simplest interface component that supports message editing and display is the elementary interface field. Interface fields are able to receive signals intended to edit or display their corresponding piece of data, as well as other control signals (e.g. a signal indicating the end of the editing process, and the subsequent triggering of the information system reaction).

An *abstract interface field* is a system component belonging to the information system interface that is intended to allow editing or displaying a piece of data. Abstract interface fields are related to message structure data fields.

However, although fields can be conceived to exist on their own, it is convenient to model them grouped by means of abstract interface structures.

An *abstract interface structure* is a set of abstract interface fields that is intended to allow editing or displaying a set of related pieces of data. Abstract interface structures are mapped to (parts of) complex message substructures. Three types are distinguished:

- *Registry abstract interface structures* correspond to aggregation substructures.
- *Registry-set abstract interface structures* correspond to iteration substructures.
- *Matrix abstract interface structures* correspond to two nested iteration substructures.

Abstract interface structures are often related to one another, forming complexly structured forms and listings.

An *abstract interface structure relationship* abstracts the data relation and behaviour of two abstract interface structures, one of them acting as the source and the other one acting as the target.

According to previous works [España 2005], several types of abstract interface structure relationships are distinguished in Communication Analysis, although further investigation is required to formalise their precise semantics:

- *Composition relationships* model a master-detail (a.k.a. header-lines) behaviour that corresponds to a complex message substructure having a nested iteration substructure; it abstracts the behaviour of editing or displaying several (parts of a) business objects that are related to a single (part of a) business object.
- *Indexation relationships* model a list-detail behaviour that corresponds to two fragments of a single iteration substructure; it abstracts the behaviour of editing or displaying details about (part of) a business object that is selected from a summarised list.
- *Synchronised indexation relationships* establish a functional relationship among two registry-set abstract interface structures.
- *Generation relationships* model a constructive relationship between two abstract interface structures, meaning that the data presented in the target abstract interface structure is derived from the data presented in (or selected from) the source abstract interface structure. The generation can either be total or partial, and closed or open (see [España 2005] for more details).
- *Multiple-composition relationships* represent purely associative relationships among registries; that is, relationships in which the participation of the involved elements is symmetric and the user may want to see the relationship from either side.
- *Population composition relationships* abstract the behaviour of two abstract interface structures that present data that includes identification of populations (as opposed to identification of individuals; e.g. several tanks of fruit juice are mixed to create different kinds of multi-flavour juices and the company wants to trace the origin and target of juices as well as the quantities).
- *Functional-composition relationships* allow to model complex behaviours that cannot be modelled using the previous abstractions. Complex relationships can be specified by means of functions, in terms of selection, grouping and projection of data.

Two communicative events are said to be *editorially compatible* if their message structures allow the reuse of abstract interface structures to edit their corresponding messages. That is, the editing that is necessary to build the messages that correspond to both events has enough similarities to be supported by the same set of interrelated abstract interface structures. The fewer

modifications needed to be applied to a set of abstract interface structures when shifting from one event to another, the higher the editorial compatibility of the events.

An *editing context* is the set of interrelated abstract interface structures that support a set of editorially compatible communicative events. An editing context will offer different modes of interaction (i.e. changes in the available interaction) to support the editing of messages for all the communicative events in the set.

The abstract interface structures of an editing context are encapsulated in abstract interface containers. An *abstract interface container* is an abstraction that represents a user interface screen (a.k.a. form or window) of a desktop application, or a web page of a web-based application. It groups several abstract interface structures together.

A *navigation link* is the definition of a path that the user can choose to follow to move from one abstract interface container to another. Navigation links are modelled by means of a navigation map.

When graphical user interfaces are implemented manually (i.e. not using code generation strategies), component reuse becomes crucial so as to improve efficiency during implementation and maintenance. The mechanism of graphical user interface reuse in Communication Analysis is the use of interface *settings*[50], by which we refer to distinct configurations within the interface that depend on the state of the information system (often on the state of a given business object). This way, the concept of setting can be applied to abstract interface containers and to abstract interface structures, providing two levels of reuse.

A *presentation* is an abstract interface container that is configured according to a given setting. A presentation contains one or several abstract interface structures (each in a given setting), and a set of triggers.

A *subpresentation* is an abstract interface structure that is configured according to a given setting. The setting defines the properties of the abstract interface fields contained in abstract interface structure the abstract interface structure (e.g. which fields are visible, which are editable, etc.), as well as which triggers are enabled.

---

[50] We avoid the term *mode* due to its bad reputation in the area of human computer interaction [Raskin 2000] (classic examples are the use of the "caps lock" and "insert" keys, which lead to error-prone interfaces). However, we believe that there exists a difference between using modes in a way that the user can forget about the system state or even perceive it as arbitrary, and using modes in which the interface setting depends on the state of a given business object (e.g. if a client order is already being produced, it cannot be further modified so some functionalities are disabled). In fact, this kind of interface configurations, as long as proper feedback is provided to the user, are not even considered as modes by Raskin [Raskin 2000].

An *interface behaviour rule* is a rule that specifies how the user and the interface can interact. It consists of the context (an abstract interface container, a presentation or a subpresentation) in which the rule applies, a logical trigger (i.e. a generalisation of the physical triggers that share the same condition and the same reaction), one or several physical triggers (i.e. stimuli that starts the interaction), a condition that must hold for the reaction to take place (e.g. a check on the state of a business object), the reaction that defines the behaviour of the software application to the trigger (it can be a change of presentation, a change of subpresentation, or a business logic function).

Modelling support

The reader should take into account that the view of the platform independent metamodel of level L4 shown in Figure 53 is far from complete. It only includes some of metaclasses that belong to this level. As mentioned above, the specialisation hierarchy is only an exemplification of the idea of having a requirements taxonomy.

In activity 7, the business object class information definitions that were specified during activity 5 are fragmented according to normalisation criteria (e.g. Boyce-Codd normal form) in order to allow their computerisation. In common practice of Communication Analysis, the computerised information system memory is specified in the *Logical data model*, which can be represented can be described using the Entity-Relationship Diagram, the UML Class Diagram, a graphical relational model, or an equivalent notation.

For the description of the interface (activity 6) Communication Analysis offers set of templates, a graphical notation and methodological guidelines [España 2005]. The procedure for designing the interface is as follows. Communicative events are organised in editorial environments since a system interface usually supports several external interactions combinedly (both for interface usability reasons and for component reuse). Communication Analysis considers the interface to be a message editor and displayer, and it offers techniques that allow reasoning the interface from the message structures: the message structures are fragmented according to different criteria (e.g. to apply a first normal form) and each fragment is assigned an abstract interface structure. Each editorial environment is then modelled in terms of abstract interface structures that are encapsulated in abstract interface containers. The navigation among containers is specified by means of a dialogue map.

The bidirectional arrow between activities 6 and 7 represents that interface design often originates new classes of objects, and that decisions taken during object class modelling influence interface design.

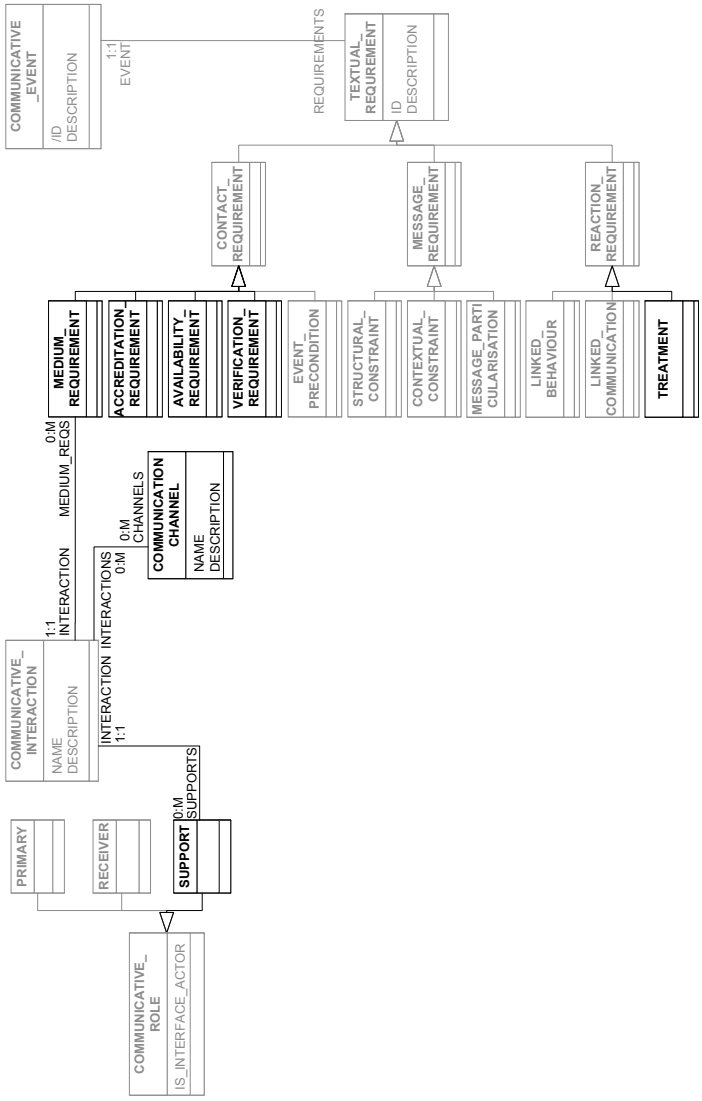


Figure 53. Platform independent metamodel view that corresponds to requirements level L4

Example

Table 13 presents some requirements of level L4 that correspond to communicative event *SALE 1*. With regards to contact requirements, the communication channel used by the client (the primary actor) is stated, and the salesman is identified as the interface actor. Some temporal requirements such as occurrence temporal constraints contribute to defining the precondition of the event (although this particular constraint is intended to regulate the service provided to clients and is probably not going to be projected onto the final software). Other temporal requirements, such as frequencies of occurrences are intended to state the conditions under which the system to be built needs to be able to operate; it could also be expressed as "the software application should be able to deal with 500 orders per week", In this case, the workload is not very high and this requirement will not pose any problem; however, this type of requirements is very important in real-time and safety-critical systems. Lastly, the reaction requirement describing the treatment of the information is very simple: the order must be recorded. In other cases, complex processing may be needed (e.g. calculating a payroll).

Table 13. Some textual requirements of level L4 concerning event *SALE 1*

Contact requirements
Actor responsibilities
Communication channel: In person, by phone, by fax **Interface actor**: Salesman
Temporal requirements
Occurrence temporal constraints: Only working days during reception hours (09:00-18:00) **Frequency of occurrence**: 500 orders per week.
Reaction requirements
Treatments
The order is recorded.

4.2.5 L5. Operational Environment Level

This level is very conditioned to the chosen implementation technology. The activities associated to this level are only mentioned for the sake of brevity, and because these activities are out of the scope of this thesis[51]. This level concerns the design of software components that support the interface, the reaction, and the system persistence. The software architecture is chosen depending on several factors; e.g. technological and budgetary constraints. Lastly, the software is implemented.

4.2.6 A comparison with Zachman enterprise architecture framework

The *Zachman framework* is a taxonomy for organising artefacts (e.g. graphical models, documents) concerning an enterprise architecture. It relates concepts that describe the real world to concepts that describe an information system and its implementation. The framework is typically represented as a 5 rows per 6 columns matrix, where the rows correspond to different perspectives intended for different stakeholders and the columns correspond to different aspects of the system under analysis or development.

Figure 54 shows how the pieces of a Communication Analysis requirements structure relate to the Zachman framework. In general terms, each row of the Zachman framework corresponds to one requirements level in Communication Analysis, except for the Business model row, that corresponds to two requirements levels; namely, L2. Process and L3. Communicative interaction. Also, Communication Analysis does not define any level for the row Detailed representation (since it does not actually correspond to requirements but to final artefacts).

Each cell contains the name of the Communication Analysis artefacts that correspond to the cell. Empty cells indicate that Communication Analysis provides no artefact for defining that specific aspect of the system at that specific perspective. For instance, Communication Analysis proposes specifying the people from the scope perspective by means of organisational units. Some Communication Analysis artefacts cover more than one cell. For instance, message structures (at analysis time) represent the information that is conveyed to the information system in each communicative event occurrence. Although

[51] Also, to focus the thesis on integration of Communication analysis and the OO-Method, software architecture (which will be covered by the OO-Method code-generation capabilities) and other development process activities -such as project management, testing and deployment- are not addressed.

that information ends up being stored as data, message structures actually serve to bridge the gap between function and data. In Communication Analysis, the static view of data is provided by the business object glossary.

Figure 54 also depicts some relations among artefacts (not all of them). For instance, a communicative event diagram defines precedence relations among communicative events; these relations actually define temporal constraints. Also, business indicators aggregate or derive data from the other cells; more specifically, the indicators are defined in terms of the model elements belonging to the other cells in the same row.

Whether the existence of empty cells indicates that Communication Analysis misses an important view of the system is still to be analysed in detail.

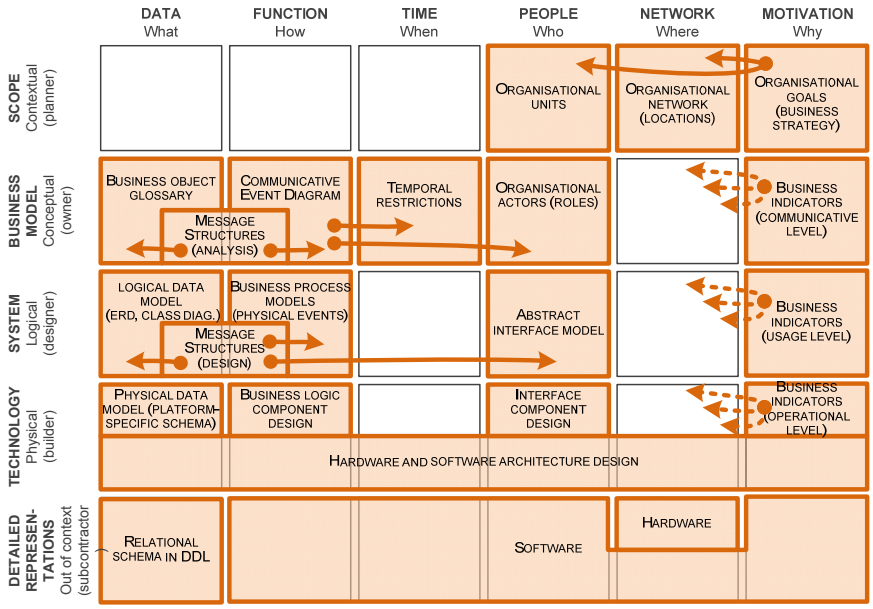


Figure 54. Communication Analysis artefacts projected onto the Kaplan enterprise architecture framework

4.3 Modelling techniques

This section presents several techniques for modelling the concepts defined in Section 4.2[52]; namely, the Communicative Event Diagram (see Section 4.3.1), Message Structures (see Section 4.3.2) and Event Specification Templates (see Section 4.3.3). These techniques are the result of evolving and adapting previously existing techniques to fit the communicational perspective that Communication Analysis imposes on information systems modelling. For instance, the communicative event diagram is a revision of business process modelling techniques (e.g. UML Activity Diagrams) so as to take into account communicative interactions, and Message Structures is a revision of the BNF notation and Structured Analysis data dictionaries. The notations have been tuned up according to our experience throughout the years, leaving out any superfluous details. Nevertheless, other notations can be used instead of the ones that we propose herein, as long as they are conveniently adapted according to the philosophy of the method; that is, as long as the Communication Analysis criteria and guidelines are preserved and the notation is added any lacking modelling primitives.

We place a special emphasis in the methodological guidelines. For instance, the unity criteria for business process modelling (see Section 4.3.1.2) guide business process model modularity. The guidelines for using Message Structures explain their distinct application during analysis than during design (see Section 4.3.2.3).

[52] Note that the business glossary for specifying business objects has been omitted. The reason is that there are plenty of methods that have already described how to define important terms used by an organisational system. For instance, the Rational Unified Process proposes distinguishing a glossary defining of important terms used in a project from a business glossary defining important terms used in the business modelling portion of the project [Rational 2003], and it offers guidelines to create these glossaries.

4.3.1 Communicative Event Diagram

The Communicative Event Diagram is intended to describe business processes from a communicational perspective.

4.3.1.1 Grammatical constructs

Each *communicative event* is represented as a rounded rectangle and is given an identifier and a descriptive name(see Def 172 and Figure 55). The *identifier* serves for traceability purposes and it is usually a code composed of a mnemonic (related to the organisational system or the business process to which the event is ascribed) and a number (e.g. *SALE 1*, where *SALE* is the acronym of the *Sales management process* and 1 is just a number that should be unique within this particular process).

Methodological guideline for naming communicative events:

- With regard to the *name*, we recommend to consistently use either an external nomination (primary actor + action + object + qualifier; e.g. "A client places an order") or an internal nomination (interface actor + action + object + qualifier; e.g. "The salesman receives an order"). For instance, in the SuperStationery case we have opted for an external nomination.

For each event, the organisational roles that play the different communicative roles are identified. Communication Analysis distinguishes several communicative roles (see Def 176-178) that have the following modelling primitives:

- *Primary actor role*. Organisational roles playing the primary role are modelled by means of a sticky figure. Since primary actors provide the conveyed input information, they are modelled as senders of ingoing communicative interactions. For instance, the client is the primary actor of event SALE 1.
- *Receiver actor role*. Organisational roles playing the receiver role are also modelled by means of a sticky figure. Since these are the actors that need to be informed of the occurrence on an event, they are modelled as receivers of outgoing communicative interactions. In order to truly understand the meaning of messages in organisations, it is necessary to analyse these actors. For instance, in SALE 1 the sales manager is informed of the order placement.
- *Interface actor role*. Organisational roles playing the interface actor role are specified textually at the bottom of the communicative event rounded rectangle. For instance, the salesman is the interface actor in SALE 1, because they fill the order form according to the client request. Although the concept of interface actor belongs to the requirements level L4. Usage environment, according to our experience it is often useful to identify interface actors when

discovering the business processes. Also, organisational actors often feel more motivated to participate in the analysis when they see their organisational role represented in the business process models.

With regards to support actors, these are not specified in the communicative event diagram, but in the event specification template (see Section 4.3.3).

Secondary notation guidelines for communicative roles:

- The sticky figures representing primary and receiver actors should be laid out in a way that communicative interaction arrows can be routed horizontally, either at the left or at the right of their corresponding communicative event rounded rectangles.
- As long as possible, sticky figures should be aligned in vertical columns, trying to minimise the number of columns, and trying to place the roles belonging to the organisational system in one side and the roles belonging to its environment in the other side.
- If it is perceived to improve diagram comprehensibility, for each organisational role a different vertical column can be created.

The communicative interactions associated to communicative events are represented by means of thick lines with a long filled arrowhead pointing at the receiver of the associated message. *Ingoing communicative interactions* depart from a primary actor sticky figure and arrive at the communicative event rounded rectangle. *Outgoing communicative interactions* depart from a communicative event rounded rectangle and arrive at a receiver actor sticky figure. Communicative interactions are given a name by labelling them. The name should coincide with the name of the associated message structure (unless there is a good reason not to do so) since this is believed to improve cognitive traceability[53].

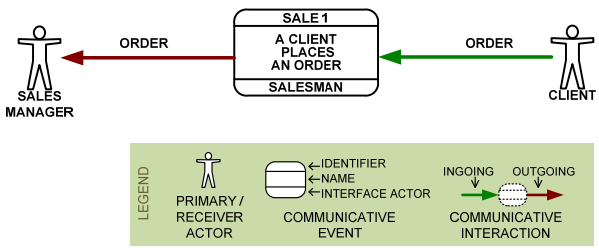


Figure 55. Example of communicative event from the SuperStationery case

[53] Message structures are later specified in detail in requirements level L3 (see Section 4.3.2).

Secondary notation guidelines for communicative interactions:

- Communicative interactions should be as straight and horizontal as possible.
- The colour of ingoing and outgoing communicative interactions should be different. Green and red are the author's choice, but different colours can be chosen. In any case, colour is just used as a visual cue and it is the direction of the arrow that indicates the type.

Communicative events are the main type of nodes in a Communicative Event Diagram. Other types of nodes are logical nodes (i.e. and-forks, and-joins, or-branches and or-merges), start nodes and end nodes.

Precedence relations (see Def 180) are modelled by means of thin lines with a short open arrowhead, departing from a source node and arriving at a target node. Communicative event A is said to be a precedent of communicative event B (and B a successor of A) if there is a precedence relation that departs from A and arrives at B, or if there is a sequence of precedence relations with only logical nodes in between.

A precedence relation that departs from a communicative event, e.g. A, and arrives at another communicative event, e.g. B, is a *loopback* if A is a precedent of B, and the depth of A is smaller than the depth of B (the depth of a communicative event being the length of the minimum path between the initial node and the event).

Secondary notation guidelines for precedence relations:

- Precedence relations should be drawn as vertical as possible, being successor events placed lower than precedent events.
- In general, straight lines are preferred to curved or angled lines. However, angled lines are preferred in the case of loopbacks.

Note that communicative interactions are placed in the horizontal axis, whereas the vertical axis is reserved for precedence relations.

Logical nodes have the following modelling primitives:

- *Or*. The or-merge indicates that only one of the incoming precedence relations needs to hold. Note that the or-branch is implicit and corresponds to event specialization. The or-merge is represented by a diamond shape containing the propositional connective symbol $\vee$.
- *And*. The and-fork and the and-join are implicitly represented by two or more precedence relations departing from or arriving to a communicative event, respectively; however, they can be explicitly drawn if needed to express complex logical expressions. The and-fork and the and-join are represented by a diamond shape containing the propositional connective symbol $\wedge$. Only one precedence arrow can arrive at an and-fork, but several arrows can depart from it. Several arrows can arrive at an and-merge, but only one arrow can depart from it.

Secondary notation for logical nodes:

- When or-merges are used in a diagram but no and-forks are used, then the propositional connective symbol can be omitted, as long as the legend consistent with this decision (see Figure 56).
- The symbols + and x could be used instead of $\vee$ and $\wedge$, as long as the legend reflects this decision.

Two special nodes can be used at the analyst's convenience.

- A *start node* symbol indicates that the communicative events to which it is connected do not have any precedent events; that is, they are globally initiatory events. The start node is represented by a filled black circle.
- An *end node* symbol represents that the communicative events to which it is connected do not have any successor events; that is, they are locally terminatory events. The end node is represented by two concentric circles, the external one filled in white and the internal one filled in black.

Globally initiatory events often result in the creation of a representation of a business object that does not refer to other business objects; for instance, *CLIE 1* creates a new client record, *SUPP 2* creates a new supplier record, and *PROD 2* creates the product records of the company catalogue.

We refer as *locally initiatory events* to communicative events that, having precedent events, result in the creation of a representation of a business object that refers to other business objects; for instance, *SALE 1* creates a representation of a new client order (which refers to existing clients and existing products).

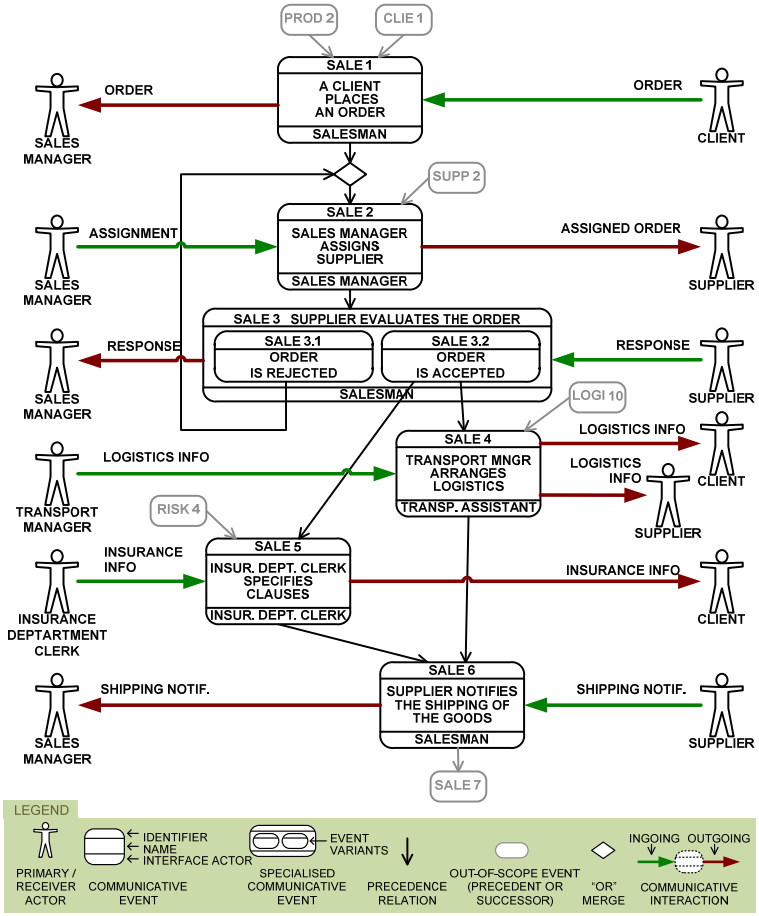


Figure 56. Communicative event diagram of part of the *Sales management* business process of SuperStationery case (also in Figure 50)

Secondary notation for start and end nodes:

- Start- and end-node symbols are often omitted for the sake of diagrammatic economy. Every communicative event not having any precedent event is implicitly connected to the start node. Every communicative event not having any successor event is implicitly connected to the end node.

Specialised events are represented by rounded rectangles that contain one smaller rounded rectangle per each of the event variants, as is *SALE 3* (see Figure 56). Several levels of specialisation are possible; that is, an event variant can, in turn, be specialised (see Figure 57). Both the specialised event and the event variants can be target and source of precedence relations. For instance, for *ALIE 5.2.2* to

occur, *ALIE 4* and *SPAC 4* need to have occurred first; after the occurrence, *ALIE 8*, *ALIE 10* or *ALIE 11* can occur. In every case, the captain will take note of the alien's decision.

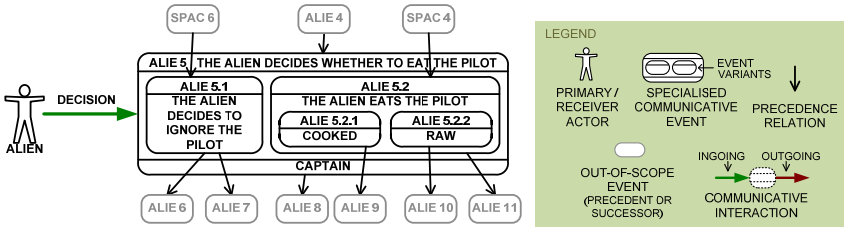


Figure 57. A specialised communicative event with two specialisation levels

Secondary notation for event specialisation:

- It is convenient to align horizontally all the event-variant rounded rectangles of the same level.

Methodological guidelines for event specialisation:

- Communicative events are specialised whenever each event variant leads to a different temporal path (i.e. distinct precedence relations depart from each variant).
- It must be avoided specialising an event as a result of different communication channels, since the message remains the same (e.g. a publishing house can order a report in person or by telephone).

The business processes of an organisational system can be specified in a single communicative event diagram or in several ones, depending on the size and complexity of the resulting diagrams. When the model denotation has been partitioned into several diagrams, there will probably exist precedence relations among communicative events of different diagrams. Although the preconditions of each communicative event are also described in event specification templates, it is important to make these precedence relations explicit in the diagrams. To do so, out-of-scope precedent events and *out-of-scope successor events* can be included in a communicative event diagram, representing them by means of a small grey rounded rectangle only containing the event identifier (they simply act as a reference). For instance, Figure 56 specifies part of the *Sales management* business process; that is, the process contains other communicative events concerning the reception of the shipping and the invoicing, but they have been omitted to keep the diagram simple. Note that *SALE 7* is left out of the scope of the diagram but it has been represented with the proposed symbol to indicate that the business process continues after the notification of the shipping departure. Also, all the direct precedents of the communicative events of the diagram have been included, even if they belong to different business processes; namely, *PROD 2*,

CLIE 1, *SUPP 2*, *LOGI 10* and *RISK 4*. Clie 1 is included in the *Client management* business process (see Figure 58). See another example of diagram partitioning in Section 4.3.1.3.

Secondary notation for diagram partition:

- Precedence relation arrows departing or arriving at out-of-scope events can be either black or grey, but the colour should be consistent within each diagram. This gives the arrows more or less relevance but does not change their meaning.

Methodological guidelines for diagram partition:

- The out-of-scope direct successors can be omitted but it is important to include all the out-of-scope direct precedents. The reason is that precedence relations are part of the precondition of the successor event. This way, it is important to represent in the diagram in Figure 56 that, for *SALE 2* to occur, *SUPP 2* must have occurred first. On the contrary, although it is convenient, it is not indispensable to represent in this diagram that after *SALE 6*, *SALE 7* can occur. It will be important, however, to represent this precedence relation in the second half of the model; that is, when *SALE 7* is included in a diagram, then *SALE 6* should also be included even in it is done by means of the out-of-scope symbol. Also note that, in Figure 58, it is indicated that communicative event *CLIE 3* has event *SALE 1* as precedence but it has not been considered important to specify that *SALE 1* has *CLIE 1* as precedence (*SALE 1* is the out-of-scope direct successor of *CLIE 1*).
- There exist guidelines that suggest partitioning a big business process diagrams. For instance, 7PMG recommends decomposing the diagram when it has more than 50 elements [Mendling, Reijers et al. 2010]. These guidelines should be followed, unless there is a strong reason not to do so.

Last but not least, it is strongly advised to add a legend to each and every communicative event diagram. This way, if the image is ever displayed out of its original context or presented to an audience that is unaware of the notation, the symbols would be easier to decipher.

- The legend should include each and every symbol used in the diagram. If a symbol does not appear (e.g. no specialised communicative events appear in Figure 55, so its symbol can be omitted in the legend).

For the Communicative Event Diagram, we recommend the notation described above. However, many aspects of the notation can be changed if needed (e.g. the shape for communicative events can be a circle instead of a rounded rectangle), as long as the changes are reflected in the legend.

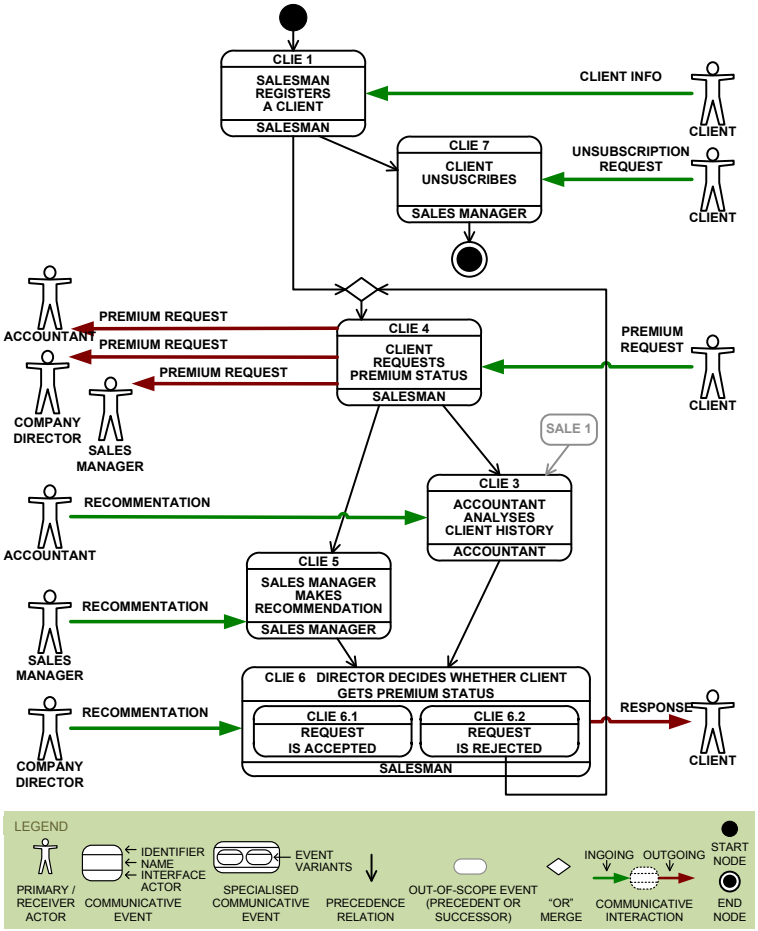


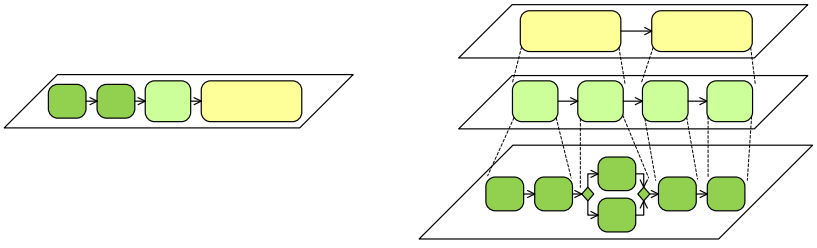
Figure 58. *Client management* business process of the SuperStationery case.

4.3.1.2 Guidelines for business process model modularity

Business process modelling methods provide notations that often include a means for activity refinement. However, most of the times the methods lack precise guidelines for model modularity. As a result, business process models end up mixing activities of different abstraction levels in the same diagram (we have tried to represent this in Figure 59.a). We claim that appropriate unity criteria aids analysts in creating modular models (see Section 3.4). We consider that a business process model has an appropriate modularity with respect to a given set of unity criteria when it fulfils the criteria; this may include having a structure of modelling layers and using stepwise refinement: each activity belongs to a specific modelling layer and can be refined in a subsequent layer if needed (we have tried to represent this in Figure 59.b).

The unity criteria that are proposed in this section are based on principles from Systems Theory and Communication Theory. The main principles of General Systems Theory have already been presented above (see Section 2.2) and the concepts of subject system, organisational system, information system and computerised information sub-system have already been defined (see Defs. 97, 95, 98, 100, respectively). In order to observe, control and influence the subject system, the organisational system performs business processes. Business processes demand much organisational communication; for this reason, organisations rely on information systems, which are often computerised for the sake of efficiency.

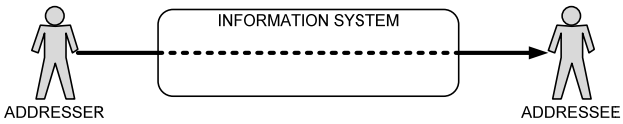
With regards to Communication Theory, we build upon two well-founded models; namely, Shannon's model of a general communication system and Jakobson's model of the communicative act (see Section 2.2 for both).



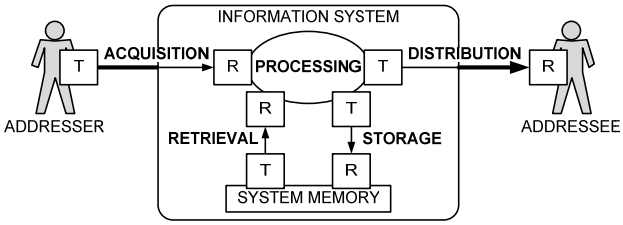
a) A business process model with inappropriate modularity

b) A modular business process model with clear criteria and stepwise refinement

Figure 59. Different communication levels in information systems



a) The information system viewed as a support for (Jakobson's) communicative acts (problem space)



b) Information system functions according to Lockemann; Shannon's roles are included (solution space)

Figure 60. Different communication levels in information systems

From a communicational perspective, an information system can be considered a communication channel among actors of the organizational system or its environment (the subject system). This vision is depicted in Figure 60.a. A communicative act takes place, via the information system channel, between an addresser and an addressee[54].

According to Lockemann and Mayr, information system functions are the acquisition, retrieval, processing, storage and distribution of information [Lockemann and Mayr 1986]. Figure 60.b refines Figure 60.a in order to reflect these functions. A similar model is included in the FRISCO report [Falkenberg, Hesse et al. 1998], and in the extension to that framework presented in Section 3.3. At this level of communicational abstraction, many message conveyances appear. Shannon's model is applicable since, in each message conveyance, a transmitter (T) and a receiver (R) intervene.

Needless to say, actual work practise can involve more complex communication procedures (such as re-encodings, translations, and chains of message conveyances that do not add new information). E.g. information acquisition may involve a client formulating a request to the clerk, and the clerk editing that message in order to convey it to the information system.

[54] Only the addresser, the channel and the addressee are depicted, the remaining elements from Jakobson's model are kept implicit

The existence of several communicational levels and complex communication procedures adds complexity to business process modelling, since business process encapsulation is different according to each level. The unity criteria defined next, clarify these levels (and, thus, facilitates business process modelling) by providing guidance for encapsulation.

We propose analysing organisations and their information systems following a systemic and communicational approach. From a systemic point of view, the focus is put on external interactions, in order to determine organisational behaviour. From the communicational point of view, unity criteria based on communication functions are proposed. Table 14 summarises modularity as applied to business process modelling.

The intention of modularity in business process modelling is to define capsules that correspond with external interactions. The content of such capsules (i.e. internal system composition) determines organisational reaction to external interactions. Business process modularisation allows distinguishing between problem space (i.e. organisational needs that are independent of any particular implementation) and solution space (i.e. the support that allows addressing organisational needs efficiently).

Table 14. Modularity in business process modelling

	Business Process Modelling
Intention	To distinguish and achieve independence between problem space and solution space (a.k.a. the 'what' and the 'how', the communicational need and the support to communication), in communication-oriented business process models.
Encapsulation	Unity criteria are based on the notion of complete communication; that is, trigger unity, communication unity, and reaction unity (see Table 15). Unitive conceptions (and, by means of metonymy, their representations) are referred to as communicative events[55].
Information hiding	When focusing on communicative events at the problem space, internal composition and internal reaction are disregarded. Business process models at this level specify organisational behaviour.

[55] Although we refer as communicative events to the unitive conceptions related to business process modelling that are obtained by using the proposed unity criteria and their representations, actually it is the criteria which are important, not the name. This way, as long as unity criteria are applied, capsules can be referred to as tasks, activities, processes, use cases, etc.

We refer as communicative event to a set of actions related to information, which are carried out in a complete and uninterrupted way, on the occasion of an external stimulus. Similar definitions can be found in [ISO 1987; Yourdon 1989; Falkenberg, Hesse et al. 1998]. In other words, an event happens in the environment of the information system, someone communicates this occurrence to the information system and a series of synchronous activities is triggered. Table 15 defines unity criteria that facilitate the identification of communicative events and the specification of business processes. Each criterion corresponds to one function of communication, as defined by Jakobson.

Table 15. Unity criteria to identify and encapsulate communicative events

Criterion (*Communication function*) Definition	**Type of communicative event**	
	Logical	**Physical**
Trigger unity (*Phatic function*) Trigger responsibility is external. This means that event occurs as a response to an external interaction and, therefore, some actor triggers it. This (primary) actor is the one that provides the information that is conveyed in the event.	✓	✓
Communication unity (*Referential function*) The input message must constitute a non-empty, complete unit. Firstly, this means that each and every event involves providing new meaningful information. Thus, an interaction needs to provide new facts in order to be considered an event. Input messages are representations of something that happens in the subject system. Secondly, this means that the message must be complete. Thus, it should be avoided to define several events for communicating parts of a message that only makes sense for the organisational system as a whole.	✓	
Reaction unity (*Connative function*) The event is a composition of synchronous activities; thus, these activities can communicate the information they need from each other. Events are asynchronous among each other; thus, events need a shared information system memory to communicate.		✓

These unity criteria allow marking out a frontier between problem space and solution space. They allow differentiating logical events and physical events[56]:

- *Logical events* correspond to Jakobson's communicative acts. They are ascribed to the problem space. Trigger unity and communication unity allow encapsulating this type of events[57]. Therefore, they are also referred to as communicative events. Logical events provide knowledge and their encapsulation disregards communicational support agents such as transmitters and receivers, only addressers and addressees are taken into account (see Figure 60.a). These events are of great importance[58].
- *Physical events* correspond to Shannon's message conveyances. They are ascribed to the solution space. Trigger unity and reaction unity allow encapsulating this type of events. Physical events are activities that provide concrete support to communication; thus, they depend on support agents such as transmitters and receivers (see Figure 60.b). These events can be changed without altering communicational content.

Starting with the idea that an occurrence takes place in the subject system, communication unity is closer to the external stimulus, and reaction unity is closer to information system reaction (communication comes before reaction).

Given a set of communicational needs of the organisational system, the set of logical events that satisfy those needs can be established objectively. On the other hand, physical events that support the logical ones depend on technological constraints, organisational responsibilities, budgetary constraints, etc.

For instance, physical events can depend on organisational responsibilities. The synchronism mentioned in the reaction unity criterion is conditioned to processing actors. Processing actors are those actors that are involved in a logical event but do not provide new information (only the primary actor does provide new information). Processing actors are typically in charge of carrying out Shannon-related activities (e.g. encoding, transmission, decoding) and Lockemann-related activities (e.g. acquisition, retrieval, processing, storage, distribution, see Figure 60.b). This way, an organisational system can decide to assign the responsibility of processing tasks to one or several workers. In case several workers are assigned the responsibility of processing tasks, this implies a change in the work environment; that is, the business process flows from one

---

[56] Physical vs. logical distinction in organisational system modelling already appears in DeMarco's work [1979].

[57] A logical event can fulfil the three criteria; this would result in a much optimised information system. Unfortunately, this is not always the case.

[58] Logical events (communicative events) subsume physical events and they are the main focus during business process modelling in Communication Analysis.

work environment to another. Then, several physical events appear, at least one for each work environment.

Communication Analysis business process unity criteria can be used to create business process models from scratch, but also to re-structure existing models (see Section 4.4.2.2.1)

Illustrative example of the application of unity criteria

The following is a case description that partly based on a real case; it is inspired by the work practice of a governmental institution (the name of the institution is fictitious) as performed some decades ago (processes have been improved over the years). The case illustrates the application of the unity criteria to unravel complex business processes by to separating the wheat from the chaff; that is, differentiating logical and physical events.

Valencia is a Spanish region that has an important agricultural sector (e.g. oranges). In case of hail or hard frost, the Valencia Institution of Agriculture and Farming (VIAF, a governmental institution) receives help requests from farmers. Each farmer goes to their corresponding town council and manually fills in a form with personal data, data about damaged cultivated plots of land, and the amount of money corresponding to damages in each plot. Farmers hand over the form to a council clerk. After the closing date, each council sends the received forms to VIAF. Then, a Technical Department clerk transfers the forms to a company that digitises data (message encoding is outsourced). The company returns digitised data and original forms to VIAF Technical Department, where digitised data is inspected and detected errors are corrected. Subsequently, help requests are recorded in the database. Then an agricultural technician visits the farmer's plots in order to assess the damages, take some photographs and handwrite some notes. Later, the agricultural technician issues an expert report. In view of the help request and the expert report, the Secretary of the Crisis Management Department resolves whether to grant economic aid and, if so, determines the amount. A Technical Department clerk sends a letter that informs the farmer about the resolution, and a copy of the resolution is sent by fax to the corresponding town council.

Figure 61 shows the above-mentioned process in terms of logical events, whereas Figure 62 further refines logical event *VIAF1* in order to depict its associated physical events[59]. Logical events *VIAF2* and *VIAF3* also have a similar model of their corresponding physical events, but these are not included herein for the sake of brevity.

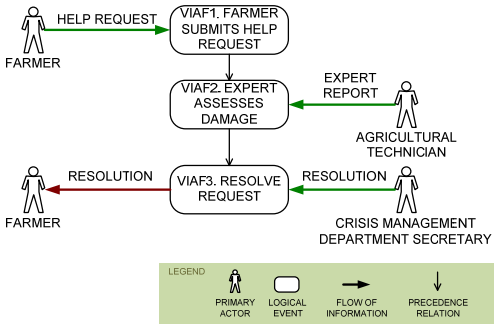


Figure 61. Logical events of the VIAF business process model for help requests

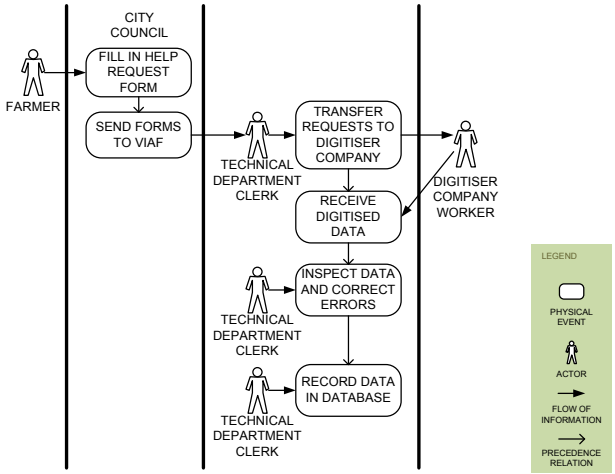


Figure 62. Physical events corresponding to logical event *VIAF1*. *Farmer submits help request*

[59] Purposely, a loose notation has been used in the models. They are only intended to clarify the notions of logical and physical events. We believe that these notions are applicable to many business process modelling notations.

Experience in the application of the unity criteria

The proposed unity criteria have been used in Requirements Engineering teaching and training for several years. Specifically, the criteria have been taught in the Computer Science degree and in the Master of Software Engineering, both at Universitat Politècnica de València (Spain). The former is a five year engineering degree; the unity criteria are part of an optional last-year course named Conceptual Modelling of Information Systems. The latter is a specialisation master aimed for industry practitioners which has been taught since 1997.

The proposed unity criteria have been adopted (a) by the Valencia Port Authority, (b) by the Infrastructure and Transport Ministry of the Valencian Regional Government[60], (c) and by Anecoop S. Coop, as part of the methodological guidelines of Communication Analysis.

Anecoop is a second degree cooperative and the major commercialisation agent in the Spanish fruit and vegetables sector. Anecoop intermediates between its associated cooperatives (ca. 100) and its worldwide clients. Therefore, its information system is highly complex, since it has to deal with disparate information needs (e.g. invoices conforming to different tax laws). Anecoop has adopted Communication Analysis entirely (concepts, criteria and notation). In fact, experience with this company constitutes the major feedback loop that allows the evolution of Communication Analysis. Anecoop has attained a successful software development process. A 2-6 people team elicits requirements. All the implementation effort is outsourced. They have implemented a software framework to support their internal business processes and they have also deployed a web service-based B2B solution to work with their associates. At the moment, Anecoop leads a big project aimed to unify the internal processes of their associates. Obviously, this success cannot be attributed only to the requirements practice; other practices as project management and risk management have also been undertaken.

The Infrastructure and Transport Ministry of the Valencian Regional Government adopted Communication Analysis but several adaptations to the method were made. Specifically, the organisation wanted to employ the Use Cases notation, so two kinds of use cases were defined: logical use cases and physical use cases (which correspond to logical events and physical events, respectively). This experience showed that the concepts and criteria could be used to extend existing notations. We believe that, similarly, notations such as BPMN or UML Activity Diagrams could also benefit from including the proposed unity criteria.

[60] (a) Autoridad Portuaria de Valencia; (b) Consellería de Infraestructuras y Transporte.

The use of unity criteria in Software Engineering

The concept of event in the context of information systems makes an appearance in 1978 with the Merise method [Collongues 1986]; its Conceptual Model of Treatments is the first event-oriented technique known in the area of information systems. Remora is a contemporary method that also adopts an event-driven approach [Foucaut and Rolland 1978]. The concept of operation or treatment in Merise is defined as a unitive concept or business process capsule, with the following unity criteria: an operation is a set of actions executed without interruption. In its turn, an event (*événement*) is either a trigger condition (pre-condition) or a post-condition. The difference with our unity criteria is that the trigger unity is fuzzy (the meaning of the term event is overloaded and also refers to non external stimuli) and that the Merise method lacks a communication unity criterion.

Most of the techniques which advocate using events employ directed hypergraphs to specify relationships among events. The Conceptual Model of Treatments in Merise, the scripts proposed in Taxis [Mylopoulos, Bernstein et al. 1980], workflows, activity diagrams and other business process modelling techniques explicitly use Petri nets [Yuditskii 2003].

Structured Systems Analysis did not incorporate the concept of event until its revision in 1989. An event is a stimulus that occurs in the outside world to which the organisation must respond [Yourdon 1989]. Yourdon insists in the external aspect of the event and, therefore, in the need of describing events from the point of view of the environment (external trigger unity criterion). Although Data Flow Diagrams (DFDs) were Petri nets [Wieringa 1998], authors did not take advantage of this. We argue that the main disadvantage of DFDs is related with their refinement; transform analysis and transaction analysis are the result of the ambiguity in the unity criteria, since these analyses are performed with the aim of identifying unitive concepts a posteriori (after an excessive refinement of processes has been done) [González 2004]. Contemporaneous techniques such as SADT, PSL/PSA, REVS or ISACS did not refer to events at all.

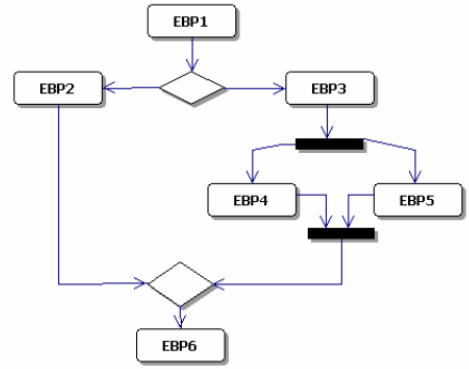
Two reports achieve quite terminological consensus in the area of information systems. In 1982, an ISO report [ISO 1987] defines an event as "the fact that something has happened in either the universe of discourse, or the environment, or the information system." In 1998 the FRISCO report defined an event as a meaningful occurrence in the environment that produces a change in the system state [Falkenberg, Hesse et al. 1998]. A reactive system is defined as an open active system where events cause impressions, which immediately cause system reaction, which in turn cause expressions. Its systemic view raises one grade above the ISO report, in the thirdness sense of Peirce [Peirce 1998]. Intentionality and communication are the key point and an explicit constructivist stance is adopted; the importance of communication is essential in the report.

Wieringa offers a detailed characterisation of reactive systems and transformational systems [Wieringa 2002]. The analysis of this comparison leads to the definition of criteria which have been influential in our approach [González 2004].

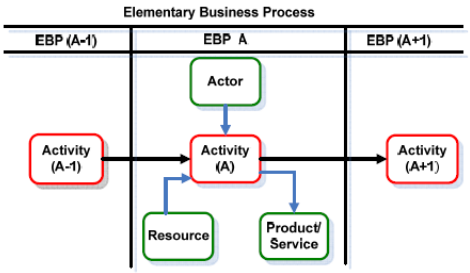
Merode [Snoeck, Michiels et al. 2003] proposes the use of events, which are attributed the following criteria: temporal synchronicity, significant communication and atomicity. "An event of a certain type occurs or is recognised at one point in time; its possible duration can be abstracted." "A business event matches something that happens in the real world within the universe of discourse". "An event is not decomposable, it cannot be split into sub-events."

In UML, events are descriptions of real world happenings, the specification of a meaningful happening that occurs in space and time [Booch, Rumbaugh et al. 1999]. However, object-oriented analysis does not define a clear capsule for business processes. Class diagrams allow assigning services to classes of objects, but business processes do not exclusively affect one class of objects. For the same reason, State Transition Diagrams are neither useful. "Higher level tasks involving operations from several objects, will not be visualised and can only be established by detailed reading of information from several parts of the model" [Høydalsvik and Sindre 1993]. Interaction Diagrams allow the representation of service composition and the description of the interaction between the user and the system, but there is still a need for a first-order element to adequately encapsulate a business process.

Use-Case Diagrams have become popular to represent system services [Dobing and Parsons 2006] even though there is a lack of a unified criterion to determine what is (and what is not) a use case. Jacobson offered as criterion that a use case must give a measurable value to a particular actor and a "twenty use cases per system" rule of thumb [Jacobson 2004]. Cockburn also proposes user goal-based unity criteria [Cockburn 1997]. In order to identify elementary business processes he proposes asking: "Can the primary actor go away happy after having done this?" "Does your job performance depend on how many of these you do today?" "After I get done with this, I can take a coffee break." Use case unity criteria makes them hardly objectivable, they are far from being an engineering principle. For this reason, many authors acknowledge lack of objectivity; e.g. "a use case is an actor-initiated, complete, system behavior that brings value to the actor. Sometimes it may be difficult to identify the set of use cases that our system offers" [Chonoles and Schardt 2003]. Other authors openly criticise the approach "A use case is not a specific or clear statement of such objectives. Instead, a use case is a representation that simply summarizes behavior traces; it augments and clarifies our understanding of the requirements, but a use case does not a requirement make" [Antón, Dempster et al. 2001].



a) UML Activity Diagram representing a flow of EBPs



b) Activities and objects needed for realizing an EBP

Figure 63. Modelling business processes in terms of elementary business processes (from [Kherraf, Lefebvre et al. 2008])

Recent works on unity criteria include an approach that extends use cases unity criteria in order to establish a so called Informational Level of Objectives for use case elicitation [Fortuna, Werner et al. 2008], the extension of BPMN with consecutive flows (a form of reaction unity) in order to achieve more homogeneous granularity in task descriptions [de la Vara and Sánchez 2009], an empirical study on the factors that influence business process modularity in practice [Ding and Jie 2008], an empirical assessment of the impact of modularity in model comprehensibility [Reijers and Mendling 2008], and a multi-layered framework for business process modelling that claims to enhance modularity [Bhat and Deshmukh 2005]. However, these proposals either are not soundly based on theory or no guidelines are offered to assist the modeller.

[Larman 1997] defines an elementary business process (EBP) as "a task performed by one person in one place at one time, in response to a business event, which adds measurable business value and leaves the data in a consistent state. e.g., Approve Credit or Price Order [original source lost]" (sic). But next he

clarifies that this definition should only be taken as an orientation to grasp the idea. Actually, a task can still be considered an elementary business process even if two people are required, or if a person has to walk around. The essence of the definition is that an elementary business process is not a single small step such as "delete a line item" or "print document" and it neither takes days and multiple sessions, like "negotiate a supplier contract", but it is a task done during a single session. The emphasis is placed in adding observable or measurable business value, as well as leaving the system and data in a stable and consistent state.

Larman proposes the concept of elementary business process for use case modelling; however, it can be used with other notations. For instance, [Kherraf, Lefebvre et al. 2008] propose to model business processes as a collection of elementary business processes "which are related according to a workflow specifying the handing over of the tasks from one actor to another;" for this purpose they use the UML Activity Diagram (see Figure 63.a). Then, for each elementary business process they identify both the activities (the dynamic aspect) and the objects (the static aspect) required for its realisation (see Figure 63.b); with regards to the objects, they distinguish the actors (responsible for the activities), the resources required (human, material, informational), and the products or services resulting from the activity.

Table 16 analyses Larman's criteria and compares them to Communication Analysis unity criteria. The conclusion is that some of Larman's criteria are not applicable ("one person in one place at one time"), others are not easily applicable to guide business process modularity (e.g. "measurable business value") and the rest are subsumed by Communication Analysis unity criteria. Moreover, Larman's criteria lacks a criterion that we find very useful in practice: *communication unity*. The allows discarding task encapsulations that do not provide new meaningful information to the organisational system (the input message must neither be empty, nor contain only previously-known information), as well as avoiding unnecessary fragmentations of task encapsulations (the message should be complete, thus avoiding two separate tasks, one for entering an order header and another one for entering item lines).

It should also be noted that Larman allows for use cases that represent reusable routines as "reasonable violations of the EBP guideline" (sic). In Communication Analysis we strongly advise against applying reuse criteria to guide modularity during information business process modelling and system analysis. Reuse should only be used as a guideline in design time.

The role of modularity in goal modelling has been debated in recent years, and the lack of modularity principles in i* has been demonstrated by [Estrada, Martínez et al. 2006]. However, Franch [2010] introduces support for modules within the i* framework.

Table 16. An analysis of the modularity criteria by Larman

Criterion	Comment
A task performed by one person in one place at one time	These criteria are relaxed by Larman because they are not applicable in practice (they are not fulfilled by many encapsulations that deserve being an elementary business process). The equivalent criterion in Communication Analysis is the *reaction unity* criterion. For a similar reason, it is a criterion that is only enforced for physical communicative events; logical communicative events do not necessarily fulfil it.
in response to a business event	The equivalent criterion in Communication Analysis is the *trigger unity* criterion; they are similar in that an external stimulus is required. What else Larman's criterion implies remains fuzzy (defining the concept of elementary business process in terms of business events does not solve the problem, since the concept of business event is not defined).
which adds measurable business value	The problem with this criterion is that business value hardly measurable, except when economic value is meant (but it is not the case). Non-economic value is often subjective and, thus, it is not a rigorous criterion for modularity.
and leaves the data in a consistent state.	This criterion, on its own, is not enough (e.g. deleting a line item can perfectly leave the state in a consistent state and still should not be considered an elementary business process). It seems a sensible criterion related to the information system reaction to the task, but there is a nuance in the concept of consistency that should be noted. Consistency is useful as a modularity criterion to avoid encapsulating in two separate tasks two parts of the same transactional algorithm; in such case, even having consistent input information, if only one task occurs, then the system could end up in an inconsistent state. But there is another source of inconsistency that cannot be avoided even by good business process modularity: having inconsistent input data or being requested a non-permitted action given the system state (e.g. receiving a demand for a refund on damaged goods related to an order that has not yet been produced). Communication Analysis provides for the first type of consistency by means of the *communication unity* criterion (i.e. the ingoing communicative interaction should provide a complete input message). The second type of inconsistency can only be dealt with in process execution/enactment time, by means of preconditions, postconditions and integrity constraints.

4.3.1.3 Diagram partitioning

The Communicative Event Diagram modelling technique allows interrelating two or more diagrams by using out-of-scope symbols and precedence relations. The out-of-scope symbol has been presented in Section 4.3.1.1, and guidelines for its usage have been provided. Whenever a communicative event from one diagram has a precedent event that belongs to a different diagram, this relation has to be specified (it is important to have readily at sight all the precedent events because precedence relations are part of an event precondition). Also, whenever a communicative event has a successor event that belongs to a different diagram, then this relation can be specified (although in this case it is not so relevant, it helps conceiving the big picture of the organisational work practice).

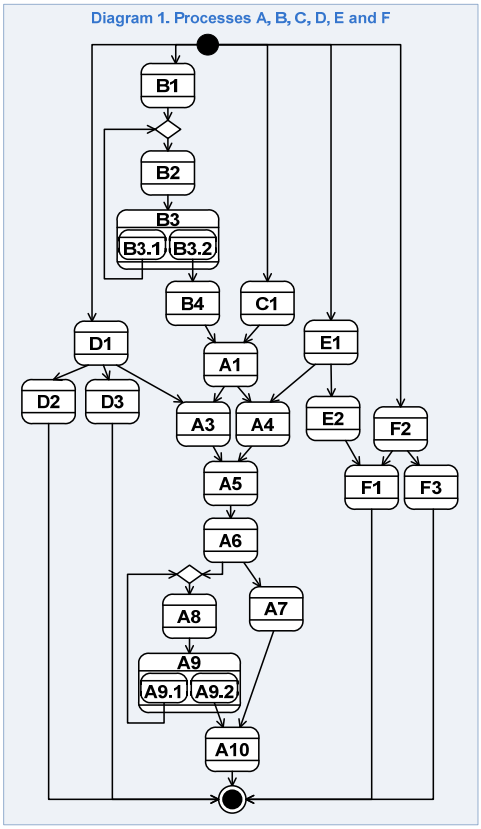


Figure 64. A diagram presenting a complete view of the processes of the whole business

By means of defining several interrelated communicative event diagrams, the analyst can partition the description of organisational work practice. The benefit is that cognitive complexity is reduced because the fragmented models are more manageable; this way the modelling technique becomes scalable. The trade-off lies in the fact that the out-of-scope precedence relations have to be kept consistent.

For instance, Figure 64 shows an illustrative (meaningless) example of a company's work practice. The work practice has been divided into several business processes, but all of them are depicted in the same communicative event diagram. The business process model can be partitioned into several diagrams, using different partitioning criteria, as shown in Figure 65. For instance, each business process has been modelled in a separate diagram. Also, business process A has been considered two big and has been further partitioned in two parts. All the diagrams in Figure 64 and Figure 65 are just views of the complete business process model (which, if supported by a CASE tool, can be stored in a repository).

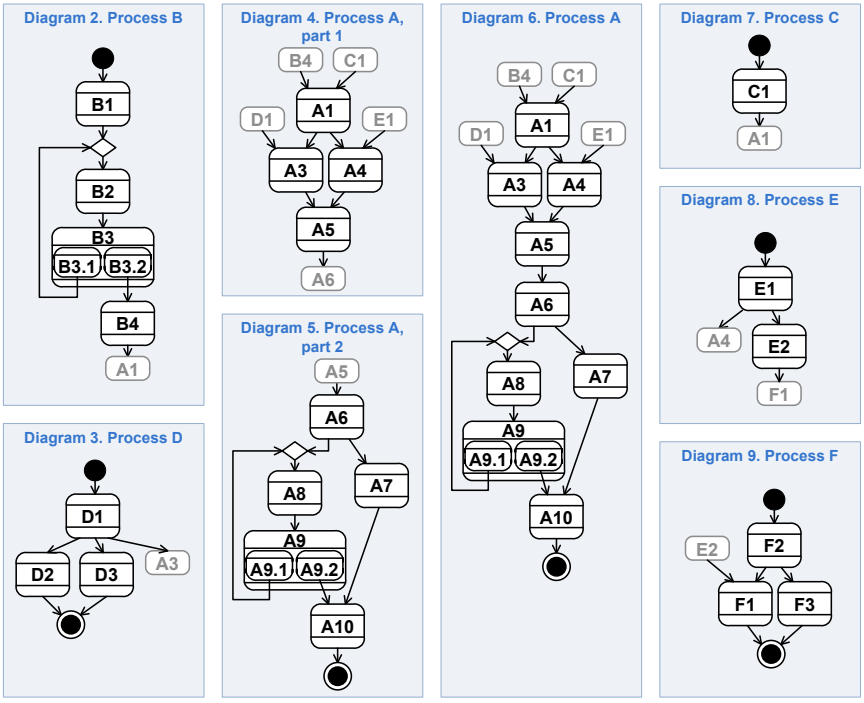


Figure 65. Several diagrams presenting partial views of the whole business process model

4.3.2 Message Structures

Message Structures is a specification technique that allows describing, by means of structured text, the message that is associated to a communicative interaction. Although message structures can be used to specify outgoing communicative interactions, we focus on the specification of ingoing communicative interactions, given their analytical interest. Table 17 presents an example[61] of the usage in analysis time; the ORDER message structure specifies a communicative interaction by which a client places an order[62].

Table 17. Example of a message structure in analysis time

FIELD	OP	DOMAIN	EXAMPLE VALUE
ORDER =			
< Order number +	g	number	10352
Request date +	i	date	31-08-2009
Payment type +	i	text	Cash
Client +	i	Client	56746163-R, John Papiro Jr.
DESTINATIONS =			
{ DESTINATION =	i	Client address	Blvd. Blue mountain, 35-14A, 2363 Toontown
< Address +	i	text	Brayden Hitchcock
Person in charge +			
LINES =			
{ LINE =	i	Product	ST39455, Rounded scissors (cebra) box-100
< Product +	i	money	25,40 €
Price +	i	number	35
Quantity >			
}			
>			
}			
>			

[61] This particular font, the colours and the capitalisation are a non-prescriptive convention that is intended to facilitate message structure comprehension. Feel free to configure these aspects.

[62] Most of the examples in this paper are taken from a requirements model that can be found in http://hci.dsic.upv.es/ca/SuperStationery-TR-v2.0.pdf

4.3.2.1 Grammatical constructs

The syntax of Message Structures can be described in terms of the following grammatical constructs.

We refer as *substructure* to an element that is part of a message structure. This way, LINE, Client and Payment type are substructures that are part of ORDER. There exist two classes of substructures: fields and complex substructures. We refer as *initial substructure* to the substructure that constitutes the first level of a message structure. For instance, ORDER=<Order number + Request date + Payment type + Client + DESTINATIONS >.

- *Field*. It is a basic informational element of the message; that which is not composed of other elements. There exist two types of fields.
  - *Data field.* It is a field that represents a piece of data with a basic domain[63]. For instance, Payment type is a data field that belongs to the message structure ORDER.
  - *Reference field.* It is a field whose domain is a type of business objects. E.g., Client references a client that is already known by the information system.
- *Complex substructure*. It is any substructure that has an internal composition. There exist three types of complex substructures.
  - *Aggregation substructure.* It specifies a composition of several substructures in a way that they remain grouped as a whole. It is represented by angle brackets < >. For instance, LINE=<Product+Price+Quantity> specifies that an order line consists of information about a product, its price and the quantity that the client requests.
  - *Iteration substructure.* It specifies a set or repetition of the substructures it contains. It is represented by curly brackets { }. For instance, an order can have several destinations and, for each destination, a set of order lines is defined. Both DESTINATIONS and LINES are iteration substructures. LINES={LINE=<Product+Price+Quantity>}
  - Specialisation substructure. It specifies one or more variants; that is, structural alternatives[64]. There is no example of specialisation substructure in Table 17; the message structure in Table 18 specifies that the assignment made by an student can be of type THEORY, in which case the fields Subject and Title characterise the work, or it can be of type PRACTICE, in which case the fields Programming language and Functionality characterise the work.

[63] Basic domains (e.g. numbers, text) are discussed below.

[64] It is more frequent to use specialisation with two or more variants. The usage with one variant represents the optionality of that variant; that is, a message might or might not include the variant.

Table 18. Fragment of a message structure that includes specialisation

FIELD	OP	DOMAIN
< ...		
Type of assignment +	i	[theo|prac]
TYPE =		
[THEORY =		
< Subject +	i	Subject
Title >	i	text
| PRACTICE =		
< Programming language +	i	Language
Functionality >	i	text
] + ...		
>		

For greater disambiguation, Table 19 presents the grammatical constructs of Message Structures using the Extended Backus-Naur Form notation (EBNF) [ISO/IEC 1996].

In practice, the syntax is more flexible: the names of complex substructures can be omitted, an iteration substructure also aggregates its own content (there is an implicit aggregation substructure), and each variant of a specialisation substructure also has an implicit aggregation. This 'syntactic sugar' allows adapting the notation to project contingencies and it facilitates the usage of the technique.

Table 19. EBNF grammar of Message Structures[65]

```
message structure
= structure name, '=', initial substructure;
initial substructure
= aggregation substructure | iteration substructure;
aggregation substructure
= '<', substructure list, '>';
iteration substructure
= '{', substructure list, '}';
specialisation substructure
= '[', substructure list,{ '|', substructure list },']';
substructure list
= substructure, { '+', substructure };
complex substructure
= aggregation substructure | iteration substructure
| specialisation substructure;
substructure
= substructure name, '=', complex substructure | field;
```

[65] The elements `structure name`, `substructure name` and `field` are character strings.

4.3.2.2 Field specification

To characterise a field, the following properties can be specified:

- *Name*. Each field must have a significant name (e.g. Request date).
- *Acquisition operation*. It specifies the origin of the information that the field represents.
  - *Input* i. The information of the field is provided by the primary actor.
  - *Generation* g. The information system can automatically generate the information of the field.
  - *Derivation* d. The information of the field is already known by the information system and, therefore, it can be derived from its memory; that is, it was previously communicated in a preceding communicative event. This operation can have an associated derivation formula.
- *Domain*. It specifies the type of information the field contains.
- *Example*. An example of a value for the field, provided by the organisation.
- *Description*. An explanation that helps the reader to understand the field meaning.
- *Label*. A brief text that describes the field when shown in a graphical interface.
- *Link with memory*. It specifies the correspondence between the field and a database table column or a class diagram attribute.
- *Compulsoriness*. It specifies whether the field necessarily takes value or not (i.e. whether the field is mandatory or not). It is also possible to specify that the field is not compulsory by using a one-variant (e.g. [a]).
- *Initialisation*. The value that the field is given by default can be specified by means of a function or a derivation formula.
- *Visibility*. It specifies whether the field is visible in a graphical user interface form.

It is recommended to lay the fields out vertically ad to specify the field properties horizontally (by means of columns). For reasons of space, the description of the fields can be done in a separate table. Message Structures can be extended with other field properties that a method designer or an analyst deem appropriate. However, as discussed below, not all properties are convenient at analysis time.

4.3.2.3 Usages of Message Structures

Table 20. Applicability of field properties to development stage

		Name	Acquisition operation			Domain	Example	Description	Label	Link with memory	Compulsoriness	Initialisation	Visibility
			i	g	d								
Analysis		++	++	++	--	++	++	++	--	--	--	--	--
Design	Memory	++	++	++	++	++	++	++	-	++	+	-	-
	Interface	++	++	++	++	++	++	++	++	++	++	++	+

++ highly recommended + recommended - not recommended -- discouraged

Message Structures can be applied for different purposes (from software development to adaptive maintenance) and in different stages of the software development life cycle (e.g. analysis, design). Depending on whether they are used in analysis or design time, syntactic and pragmatic differences have to be taken into account. Table 20 presents recommendations on the usage of field properties, depending on the development stage in which Message Structures are used.

Creation and usage of Message Structures in analysis time

In analysis time, Message Structures allow specifying in detail the communicative interactions that take place in the organisational work practice. This way, they offer a communicational perspective for business process modelling and they act as requirements for the information system. In the context of Communication Analysis, the new meaningful information that is conveyed to the information system in each communicative event is specified by means of a message structure.

In the following, we enumerate some sources of information and techniques for acquiring information and analysing the messages exchanged with the information system.

Organisational actors play an important role in information systems analysis, since they know organisational work practice first-hand. The analyst will employ elicitation techniques such as interviews or JAD sessions [August 1991]. It is crucial to ask the proper questions so as to define which information is conveyed in each communicative event, as well as to distinguish new information from derived information.

Business forms are a technological support for communicative interactions and, therefore, they are a major source for analysis. In this sense, the user interface screens from pre-existing software are equivalent. Forms can be used for entering

information (input forms), for presenting data (output forms), or for both purposes. In analysis time, input forms allow to identify communicative events that convey new information to the information system. See Section 4.4.2.1.1 for an explanation on how to analyse input forms to find out the communicative interactions they support and to specify their corresponding message structures.

If the organisation has *previous business process specifications* or quality procedures, then this documentation can also be used as input for the analysis.

Usage of Message Structures in design time

In design time, Message Structures allow establishing the traceability between analysis documentation, the specification of the information system memory (e.g. by means of a class diagram or a relational database schema), and the specification of the user interface. Moreover, it is possible to define techniques for deriving the memory if the information system from requirements models, as well as techniques for systematically reasoning the interface design.

The summarised procedure for the derivation of the information system memory is as follows[66]. First the communicative events are sorted according to their temporal precedence. Then the message structure of each event is processed in order to obtain a class diagram view. This way, the complete class diagram is iteratively created by integrating the class diagram views that correspond to all the communicative events. Figure 66 (right-hand side) shows the derivation of the class diagram view that corresponds to a communicative event in which a client places an order. A more detailed and bigger example is available online[62].

The summarised procedure to reason the user interface is as follows. First the interface style manual. Then the editing environments are identified (i.e. sets of forms or interface screens that support a set of editorially-compatible communicative events. Next, the message structures are fragmented (e.g. normalising them in first normal form) and the fragments are assigned to abstract interface structures (e.g. registry, set of registries). The abstract interface structures are encapsulated in forms. Each form is specified in detail, establishing the possible interaction and the editing facilities (filters, order criteria). The behaviour of the interface can be specified by means of trigger tables. Lastly, additional listings and printouts are specified. Figure 66 (left-hand side) shows how the information system interface is designed in terms of abstract interface patterns. Methodological guidelines are described in detail in [España 2005].

[66] We describe the derivation of class diagrams because this derivation technique is part of ongoing research. An analogous argumentation can be made for relational schemas.

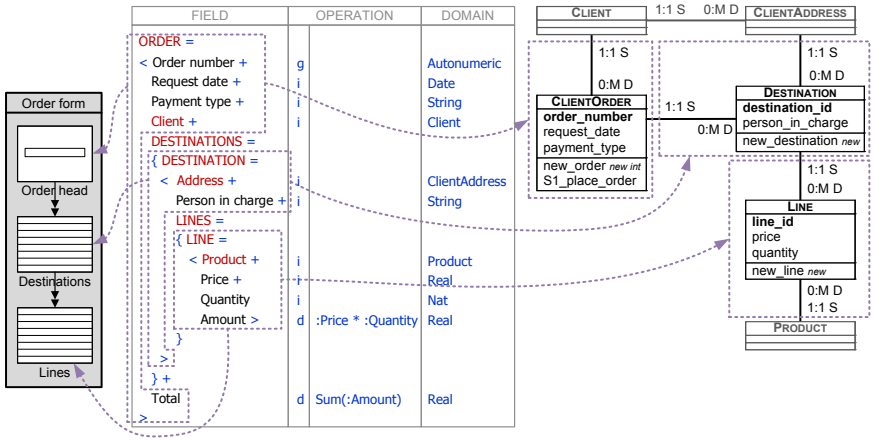


Figure 66. Derivation of a class diagram view and the interface design of a communicative event in which an order is placed

In design time, it is usual to specify derived fields (e.g. the total amount of each order line Amount d (:Price * :Quantity)). Other properties that specify aspects of design are also specified in this stage (e.g. the specification of the data field Request date could also include a formula that defines its initialisation value: today()).

4.3.3 Event Specification Templates

Communicative events that appear in the Communicative Event Diagram need to be described in detail. Requirements associated to an event can be structured by means of an event specification template. The template is composed by a header and three categories of requirements: contact, message and reaction requirements. These categories are related to Jakobson's phatic, referential and connative communication functions, respectively (see Section 2.2).

The *header* contains general information about the communicative event; that is, the event identifier, its name, a narrative description and, optionally, an explanatory diagram.

- *Event identifier*. Event identification needs to be kept consistent throughout the entire analysis and design specification in order to enhance requirements traceability. This way, the event identifier needs no to coincide with the one in the communicative event diagram. It normally consists of the acronym of the business process to which the event belongs and a sequential number.
- *Event name*. The name of the communicative event should clearly state the change in the subject system that is being reported in the event. See 4.3.1.1 for naming guidelines.
- *Narrative description*. Since requirements specifications are meant, first of all, to facilitate problem understanding, a narrative description of the event is strongly advised.
- *Explanatory diagram*. Also, whenever the event is complex, an explanatory diagram illustrating its associated flow of tasks shall be included. In such diagram, we recommend to focus only in physical communicative events; that is, actions related to information acquisition, re-encoding, and distribution that fulfil the trigger unity and the reaction criteria. In Section 4.3.1.2 the unity criteria are explained and Figure 62 presents an explanatory diagram depicting physical events.

Contact requirements are related to the conditions that are necessary in order to establish communication. For instance, the primary actor, possible communication channels (e.g. fax, email, in person), availability and temporal constraints (e.g. office hours for order reception), authentication requirements (e.g. in Spain, bureaucratic proceedings often require showing an identity card).

- *Primary actor*. The organisational role that is responsible for communicating with the information system to report a change in the subject system (see Def. 176).
- *Support actors*. The organisational roles that participate in message transfers but do not provide new information (see Def. 190); that is, the actors responsible for physical communicative events.

- *Interface actors.* The organisational roles that are in charge of editing input messages in the form and codes required by the information system (see Def. 114). In a paper-supported information system, the interface actor is the one that fills paper business forms; in a computerised information system, the interface actor is the one that interacts with the user interface of the software application.
- *Availability requirements* and constraints that refer to the degree to which the information system is in a position to engage in the ingoing communicative interaction.
- *Medium requirements* that refer to the technology (this includes paper-based forms) that supports the ingoing communicative interaction. In case scanned business forms or screenshots of a previous software application are available, those that apply to the communicative event (e.g. those that support the ingoing communicative interaction) can be included in this section. In case they are catalogues in a different document or repository, then a reference to them can be included.
- *Accreditation requirements* that refer to the protocols that the organisational system prescribes for each actor participating in the ingoing communicative interaction (i.e. informally, how to know that actors are who they say they are).
- *Verification requirements* that refer to a) ensuring that the provided documentation (if any) is not fraudulent, and b) confirming the influx of physical elements associated to the ingoing communicative interaction (e.g. stock coming into a warehouse).

Message requirements specify the message conveyed by the primary actor to the information system in a communicative event and related constraints (e.g. reliability: certifying that a diploma provided by a student is not fraudulent). With regard to the message, both metalinguistic aspects (e.g. message field structure, compulsoriness of fields) and linguistic aspects (e.g. field domains, example values) need to be specified. We propose Message Structures to specify the message (Table 21 presents a summary of the grammatical constructs, see Section 4.3.2 for more details), but the analysts can choose an equivalent notation. In any case, the following aspects are the most important and should be specified whatever notation is chosen.

- *Metalinguistic aspects*. They refer to the structure of the message, its editing and display.
    - *Structure of the message* that is conveyed to the information system; that is, its composition in terms of complex substructures and message fields.

Table 21. Summary of the grammatical constructs of Message Structures

Message Structure grammatical constructs
Aggregation A = < a + b + c > A is composed of fields a and b and c.
Alternative A = [a | b | c] A is either composed of field a or b or c, (only one of them).
Iteration A = { B } A is composed of several substructures of type B.

- *Acquisition operation*. Indicating whether the data contained by a field is new information for the organisational system or simply a recall of previously-reported information (this includes derived information such as total amounts) is important in order to analyse whether the event is actually a communicative event or it should be discarded for not providing new meaningful information.
- *Description of the fields*. A description should be provided for the sake of comprehensibility.

- *Linguistic aspects*. They refer to the content of the message and its meaning.
  - *Domains* of the message fields. During analysis, an orientation of the field content should be given (preferably not a programming-language data type).
  - *Example value.* One or several realistic values that the field can contain clarify its meaning; they should be provided by the organisational actors involved in the analysis (i.e. representative users).
  - *Derivation* of field values. Some fields (and even some complex substructures) are derived from already-known information. Specifying such derivation can be done textually or by means of formulas.
- *Message constraints*. Such as constraints over the structure of the message, over the domains of the message fields, etc.[67]

A communicative event cannot be fully understood until the structure of its ingoing message is defined in detail. Specifying with precision an event message structure forces and helps analysts and users to appropriately mark the boundary of an event and its meaning for the organisation.

[67] Sometimes the message (the structure or the field domains) needs to be particularised to organisational roles or actors depending on their characteristics (e.g. their duties, the information that concerns them, etc.). For instance, privacy constraints on the composition of the message determine which fields an actor can/cannot view (e.g. in a given company, only salespersons can view the fields 'discount' and 'commission' and not their assistants), constraints over the domain restrict the information domain to which an actor has access (e.g. a salesman can only view orders placed on its own region).

Reaction requirements describe how the information system reacts to the communicative event occurrence (i.e. to the conveyed message). Typically, the information system processes and stores the new information (updating the system memory), extracts all the necessary conclusions that can be inferred from new knowledge, and makes new knowledge and conclusions available to the corresponding actors (distributing the information to other actors so that they can act accordingly). Therefore, this category of requirements includes the treatment or processing of the information and the outgoing communicative interactions being generated by the event, among other requirements.

- *Data model view* related to the communicative event; it is the part of the memory of the information system that the communicative event contributes to build. This is an aspect of design that can be specified by means of linking each message field with tables and columns (in case the memory is being specified by means of a relational schema), with classes and attributes (in case object orientation is chosen), etc. In the context of this thesis, the data model view is actually a class diagram view (see Chapter 5).
- *Treatments* that define what changes occur in the information system as a result of the communicative event (e.g. what processing takes place, what information is stored). It involves defining how the information acquisition is related to the data model. In some cases is suffices to indicate that the information is stored. In other cases, complex processing needs to be done and, thus, an algorithm needs to be described with more or less level of detail (textually, with pseudocode, etc).
- *Linked behaviours* refer to how the occurrence of a communicative event affects future occurrences of events. For instance, the information provided in this communicative event can condition future behaviours of the system. This includes business rules or complex conditions (e.g. decision tables) that determine future reactions depending on the values provided in the current communicative event.
- *Linked communications* specify to whom the occurrence of the communicative event must be communicated; that is, which organisational roles (or specific organisational actors) need to know about the event occurrence and its associated information so as to take further actions (e.g. make decisions). In many cases, during analysis, it may suffice to state the actor who needs to be reported the occurrence; in such cases, linked communications can simply be expressed as outgoing communicative interactions in the communicative event diagram. If further details about the communication need to be specified, it can be done in this section (e.g. scanned output form, sketch).

In Figure 67, the event specification template corresponding to event *SALE 1* of the SuperStationery case is provided as an example.

SALE 1. A CLIENT PLACES AN ORDER

1 General information

Goals

The objective of the organisation is to attend the clients when they request goods.

From the point of view of the information system, the objective of this event is to record the order that the client places, and to let the Sales Manager know that a new order has arrived.

Description

Most clients call the Sales Department, where they are attended by a salesman. Then the client requests one or several products that are to be sent to one or many destinations. The salesman takes note of the order. Other clients place orders by email or by fax.

2 Contact requirements

Actor responsibilities

- **Primary actor**: Client
- **Communication channel**: In person, by phone, by fax
- **Interface actor**: Salesman

Temporal requirements

- **Occurrence temporal constraints**: Only working days during reception hours (09:00-18:00)
- **Frequency of occurrence**: 500 orders per week

Business forms

ORDER

Order number: 10352 **Request date:** 31-08-2009
Payment type: ☒ Cash ☐ Credit ☐ Cheque **Planned delivery date:** 05-09-2009

Client

VAT number: 56746163-R
Name: John Papiro Jr.
Telephone: 030 81 48 31

Supplier

Code: OFFIRAP
Name: Office Rapid Ltd.
Address: Brandenburgen street, 46, 2983 Millhaven

Destination: Blvd. Blue mountain, 35-14A, 2363 Toontown
Person in charge: Brayden Hitchcock

#	Code	Product name	Price	Q	Amount
1	ST39455	Rounded scissors (cebra) box-100	25,40 €	35	889,00 €
2	ST65399	Staples cooper 26-22 blister 500	5,60 €	60	336,00 €
3	CA479-9	Stereofoam cups box-50 (pack 120)	18,75 €	10	187,50 €
					1412,50 €

Destination: Greenhouse street, 23, 2989 Millhaven
Person in charge: Luke Padbury

#	Code	Product name	Price	Q	Amount
1	ST65399	Staples cooper 26-22 blister 500	5,60 €	30	444,50 €
2	CA746-3	Sugar lumps 1kg	2,30 €	3	6,90 €
					451,40 €
				Total	1863,90 €

Form 1. Example of an order form

Some parts of the form are not yet filled in this event.

3 Message requirements

Message structure

FIELD	OP	DOMAIN	EXAMPLE VALUE
ORDER =			
< Order number +	g	number	10352
Request date +	i	date	31-08-2009
Payment type +	i	text	Cash
Client +	i	Client	56746163-R, John Papiro Jr.
DESTINATIONS =			
{ DESTINATION =			
< Address +	i	Client address	Blvd. Blue mountain, 35-14A, 2363 Toontown
Person in charge +	i	text	Brayden Hitchcock
LINES =			
{ LINE =			
< Product +	i	Product	ST39455, Rounded scissors (cebra) box-100
Price +	i	money	25,40 €
Quantity >	i	number	35
}			
>			
}			
>			

Field	Description
Order number	A sequential number that identifies the order.
Request date	The date in which the client places the order.
Payment type	Information about the payment type. Its value is normally either Cash, Credit or Cheque, but the salesman can freely indicate any other information here.
Client	The client that places the order.
Address	A client destination at which the products have to be delivered.
Person in charge	The name of the person that will receive the order at the destination. From one order to another, this person can be different.
Product	A product that is requested by the client.
Price	The price of the requested product.
Quantity	The amount of items of the product that the client requests at a specific destination.

Structural constraints

One order can have many destinations.

One destination can have many lines.

One order is places by exactly one client

Contextual constraints

Orders are identified by Order number.

The product price in the line takes its value from the current price of the product in the catalogue.

4 Reaction requirements

Treatments

The order is recorded.

Linked communications

The Sales Manager is informed of the order placement.

Figure 67. Example of an event specification template (both pages)

4.4 Elicitation and analysis techniques

Requirements neither have the structural simplicity nor the homogeneity that most requirements engineering handbooks claim. Actually, requirements are related to different intentions, they are formulated/specified from a given point of view and they are defined in different systemic levels and with different levels of detail. With regards to information systems requirements, the following aspects have to be taken into account:

- *Inputs* and *outputs*. Information system requirements are essentially structured around the input and output messages that the organisational system needs carry out the tasks (which are aimed to fulfil the mission of the organisation).
- *Events* and *objects*. The requirements of an information system can be derived from expressions that describe reality from two perspectives. Objects: expressions that describe world phenomena by means of properties or characteristics. Events: expressions that describe the world phenomena in terms of changes that occur to those objects.
- *Requirements levels*. User requirements have a complex structure, so flattening them out in a list is not a good approach. Requirements should be defined at different requirements levels. Every requirement has to be ascribed to a type of systemic encapsulation. Three requirements levels are of utmost importance during the requirements engineering stage: business areas (a.k.a. sub-systems), business processes and communicative interactions.

Since information systems are a support for organisational communication, requirements engineering should be focused on discovering communicational needs of the organisational actors (since they will be the users of the software application). We herein present an analytical approach that is specifically designed for this purpose: analysis of communicational behaviour. The aim is to:

1. Define the series of communicative events and outgoing communicative interactions that the organisation requires for each business process.
2. Depict how communicative events are organised in time (mainly, by defining precedence relations among them).
3. Define the essential content of each and every communicative interaction.

In order to determine the communicative events, the following techniques can be used: generative analysis and revision of communicational behaviour. Basically, generative analysis is done to create a first version of the business process models; then the revision of communicative behaviour allows eliciting unnoticed requirements and completing the business process models.

Generative analysis is based on determining communicational needs, taking as input the structure of input messages of an information system. The analyst

builds a business process model. In Communication Analysis, these models are represented as Communicative Events Diagrams.

There are several ways of applying generative analysis:

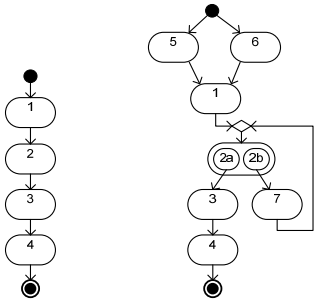
- Starting from business forms in order to classify them, determine their characteristics and identify the communicative events in which they are filled in (see Section 4.4.2.1).
- Starting from the documentation of business processes (if any). If an organisation has achieved a quality certification, it will surely have process specifications (see Section 4.4.2.2).
- Starting from informal user descriptions, in those organisations that lack the proper documentation (see Section 4.4.2.3).

By doing generative analysis, the analyst creates a business process model. However, the analysis does not end at this point. Once the business process model has been defined, it is advisable to do a **revision of communicational behaviour** in order to improve business processes and discover more communicational needs. In particular, the analyst can perform the following investigations:

- There is more than meets the eye. Users normally describe "normal" chains of events, how things happen nearly every working day. The analyst should investigate for exceptions and alternative paths (see Section 4.4.3.1).
- Users do not always bring up basic data maintenance during interviews. These communicative events have to be considered, although they do not usually entail much difficulty. Every object that the information system wants to keep track of needs to be registered first. This applies especially to basic and mediator objects (see Section 4.4.3.2).
- In some communicative events the primary actor requires previous information in order to take decisions. Also, after a communicative event, the primary actor may require a document that certifies the occurrence of the event. It can also be necessary to inform a third actor of the event occurrence and its associated information. These three requirements have in common that they are outgoing communications that are linked to a communicative event; the analyst should investigate them (see Section 4.4.3.3).
- The organisation may require auditing outgoing communicative interactions; either to monitor their issuance or to certify that the addressee has received the information (see Section 4.4.3.4).
- The organisation may as well require auditing communicative events; either their occurrence, their content, or with the intention to ensure that there are no delays (see Section 4.4.3.5).

While revising communicational behaviour, new communicative events will probably arise (see Figure 68). These communicative events have to be included

in the Communicative Event Diagram and then be properly specified by means of templates. Exceptionally, the analyst may want not to specify with much detail some simple data-maintenance events (for simple create, read, update and delete events it may suffice to specify the corresponding business form).



a) Before revision **b)** After revision

Figure 68. When revising communicational behaviour new events can arise

Before going into detail about the several techniques that the analyst can use to define communicational behaviour, it is important to comment that two of the main requirements discovery techniques are the *interview* [Stewart and Cash 2002] and cooperative workshops such as *participatory design* sessions [Schuler and Namioka 1993] and *joint application design* sessions [Dennis, Hayes et al. 1999]. However, since there is a large amount of literature on these topics and Communication Analysis does not contribute any improvement to their practice, we omit discussing them. New approaches such as creativity workshops can also prove valuable to encourage creative thinking about requirements [Maiden, Manning et al. 2004].

4.4.1 Starting business process engineering

Business processes are abstract conceptions of how the members of an organisation work jointly to achieve organisational goals. Davenport [1993] defines the term business process as "a structured, measured set of activities designed to produce a specific output for a particular customer or market. It implies a strong emphasis on how work is done within an organization, in contrast to a product focus's emphasis on what. A process is thus a specific ordering of work activities across time and space, with a beginning and an end, and clearly defined inputs and outputs: a structure for action."

Business processes can be conceived (and, thus, specified) from different perspectives. From a communicational perspective, the following abstractions are essential.

- The communicative interactions that characterise the process.
- The business objects that are manipulated by the process.
- The business indicators that offer the managerial staff a global vision of the running processes and allow them to have them under control.

Elements of a business process specification

In order to define business processes, the following elements have to be specified (see Figure 69):

- **Name** and **identifier** of the process.
- **Reasons for change**. When a reengineering project is undertaken, the rationale behind the decision of changing process specifications has to be declared. The analyst notes down the problems that have been detected in current work practice, and the organisational needs that current business procedures cannot fulfil.
- **Expected improvements**. The projected state of affairs that the organisation intends to achieve. In case of a reengineering, this refers to the improvements that are expected in relation to profit, projected image, as well as the reduction of cost, resources, time... It is just a generic, narrative description; these improvements are later specified with more precision in section *Process goals*.
- **Project scope**. A description of what falls in and out of the scope of the project; what is going to be undertaken and what is going to be left for future projects. Also known as design charter.
- **Sources of information**. It has to be defined who will take part in the analysis (that is, who will participate in interviews, JAD sessions, validation, etc.). Also, the current business forms that are being employed are enumerated and catalogued.

  - **User in charge**. The user that has the main responsibility in this process definition.
  - **Representative users**. Users that get involved in analysis.
  - **Paper and software forms and listings**. A catalogue of paper and software forms and listings that currently support the organisational work practice. Scanned images are very convenient.
- **Interfaces**. The analyst defines relations with other sub-systems that receive or offer information to the process. If this process is part of a bigger sub-system for which the interfaces have already been defined, this section does not apply.
- **Business objects**. Identify essential requirements about the entities of the domain that the organisation is interested in observing. Define a glossary of objects and their main properties.
  - **Basic objects**. The elements that are affected by the process being defined. Raw materials, goods and services. These objects are mainly perceived from a static perspective. E.g. wood lath, cardboard package, orange, t-shirt, camping slot, water supply.
  - **Mediator objects**. The elements that participate in the process and the information system is interested in. Mediator objects comprise human resources (e.g. clerk, salesman, customer, provider, mechanic, surgeon) and means of production (chainsaw, car wash machine, truck, rotary tool, silkscreen printer).
  - **Management objects**. The processes themselves (in their static and dynamic perceptions). The elements that allow the organisation to structure its procedures (e.g. the mental conception of the "order request" task), or to support them (e.g. the request form). The analyst should determine the main stages in procedure management. For instance, request (orders, budgets, approvals, etc.), planning (resource allocation), supply and purchases, production and manufacturing, execution, transformation, distribution and transportation, invoicing and payments.
- **Process goals**. They define what the organisation intends to achieve by implementing/re-engineering this process.
  - **Goal**. A declaration of an intention of improvement. Each goal is related to one or several process indicators.
  - **Process indicators**. Formulas that allow calculating values from running processes and allow monitoring them. For instance, they can be related to workload (e.g. orders received per week), effectiveness (e.g. customer claims per month), time efficiency (e.g. mean time to serve order) or resource efficiency (e.g. wood waste per year).
  - **Current level**. An estimation of the current value for the process indicator.

  - **Target levels**. An estimation of the expected value for the process indicator after the improvements are applied.
  - **Associated listings**. Listings and output forms that allow knowing whether the goal is being satisfied. If the listing has been scanned and stored in a separate catalogue, the catalogue code of the listing suffices.
- **Communicative Event Diagram**. It is a graphical business process model. It depicts a temporal ordering of the communicative interactions that correspond to the process. It allows investigating the information needs in a systematic and methodical way.
- **List of communicative events**. A list of all the communicative events and outgoing communicative interactions[68] that appear in the Communicative Event Diagram.

The Communicative Event Diagram, the list of communicative events (and outgoing communicative interactions) and the process goals are the result of several interviews. Therefore, the process specification document requires several iterations to be completed.

During the first interview, the main skeleton of the process model should be outlined. It is important to take into account the maturity of the organisation and the business and technological expertise of the users.

When users *suffer* software applications that are not well adapted to their needs, they have problems with the support to their daily work practice. For instance, they have to input the same data several times when filling current business forms. Or they do not have the appropriate information available when carrying out sales management (e.g. the listing of shipments planned for the day is not available from the screen that salesmen are filling while talking with the customer by phone; therefore, they cannot estimate when it will be delivered until they 'close' the order). In these cases, it is important to analyse how to solve their operational problems before undertaking tactical or strategic problems. Once users have their operational needs solved, they start being able to expressing tactical or strategic needs. In fact, when the organisation has obtained certain technological maturity, they start asking for more complex listings and indicators.

[68] Remember that listings are considered outgoing communicative interactions.

PROCESS SPECIFICATION

PROCESS

NAME		ID	

REASONS FOR CHANGE

EXPECTED IMPROVEMENTS

PROJECT SCOPE

SOURCES OF INFORMATION

User in charge				
Name	email	Telephone	Department	Responsibility

Representative users				
Name	email	Telephone	Department	Responsibility

Paper and software forms and listings		
Form name	Code	Observations

INTERFACES

BUSINESS OBJECTS

Basic objects

Mediator objects

Management objects

PROCESS GOALS

Goal	Process indicator	Current level	Target level	Assoc.listings

COMMUNICATIVE EVENT DIAGRAM

LIST OF COMMUNICATIVE EVENTS

File: Documento1 Printed: 16/08/2009 19:26:00 1 of 1

Figure 69. Process specification template

4.4.2 Generative analysis

The aim of generative analysis is to specify the temporal sequence of input messages. Temporal sequence of messages is also called communicational behaviour, since it represents the behaviour of the organisation from a communicational point of view. Generative analysis is a *direct* strategy. In order to carry out generative analysis the analyst can rely on existing business forms (see 4.4.2.1) or on process specification documents (see 4.4.2.2). However, in case no documentation is available, the analyst can still define and analyse the processes together with the users (see 4.4.2.3).

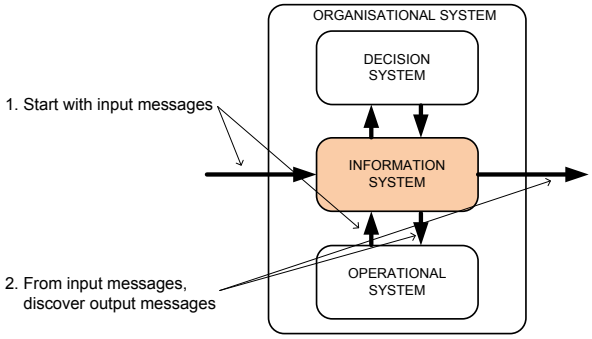


Figure 70. Generative analysis is a direct strategy (input message-driven)

4.4.2.1 Form analysis

Business forms are a technological support for message conveyances. Business forms can be supported in paper (a paper record) or they can be computerised (a software screen). There are different types of business forms and they are used either for data input (input forms), data output (output forms) or both. It is a good practice to focus first in input forms, in order to identify communicative events.

Business forms must be exhaustively analysed in order to obtain all the information that is needed to specify communicative events in detail. The analyst has to fully understand each and every form that is used by the organisation. Special attention has to be paid to data input forms (i.e. forms used to input information). Perry Edwards [1985] defines different uses of organisational forms. Based on his work, we propose the following classification:

- **Input forms**. Input forms are intended for conveying information to the information system; i.e. they are associated to information acquisition. These forms are the starting point in Communication Analysis.
  - Data input forms
  - Rountrip forms
- **Output forms**. The main purpose for issuing them is consulting information. They are a support for outgoing communications, for output messages. In many cases, they are discovered by pulling the thread when a communicative event is analysed; that is, listing forms are often linked to communicative events.
  - Strategic and tactical level listings
  - Listings for operational support
  - Intermediate results form
  - External listings
  - Listings and indicators for audit and control

We now describe each sub-kind of forms.

Data input forms

A data input form is a business form mainly intended to provide new information to the organisational system (although they often contain redundant information). They normally correspond to ingoing communicative interactions; however, as explained in Section 4.4.2.1.1, the relation among business forms and communicative interactions is many to many. See an example in Figure 71.

ORDER

Order number: 10352 **Request date:** 31-08-2009
Payment type: ☒ Cash ☐ Credit ☐ Cheque **Planned delivery date:** 05-09-2009

Client

VAT number: 56746163-R
Name: John Papiro Jr.
Telephone: 030 81 48 31

Supplier

Code: OFFIRAP
Name: Office Rapid Ltd.
Address: Brandenburgen street, 46, 2983 Millhaven

Destination: Blvd. Blue mountain, 35-14A, 2363 Toontown

Person in charge: Brayden Hitchcock

#	Code	Product name	Price	Q	Amount
1	ST39455	Rounded scissors (cebra) box-100	25,40 €	35	889,00 €
2	ST65399	Staples cooper 26-22 blister 500	5,60 €	60	336,00 €
3	CA479-9	Stereofoam cups box-50 (pack 120)	18,75 €	10	187,50 €
					1412,50 €

Destination: Greenhouse street, 23, 2989 Millhaven

Person in charge: Luke Padbury

#	Code	Product name	Price	Q	Amount
1	ST65399	Staples cooper 26-22 blister 500	5,60 €	30	444,50 €
2	CA746-3	Sugar lumps 1kg	2,30 €	3	6,90 €
					451,40 €

Total 1863,90 €

Figure 71. Data input form (the client order form of the SuperStationery case)

If an organisational system lacks data input forms, it is necessary to design them in collaboration with the users, by means of simple prototypes (mock-ups). Forms are the best tool for specifying and validating communication requirements that an analyst can use. Users understand them very well and they are often able to reason on them.

Roundtrip forms

Roundtrip forms are those forms that the system generates automatically, by means of an outgoing communicative interaction of a given communicative event, and they are expected to come back to the information system later in time (by means of the ingoing communicative interaction of a successor event), in order to continue with the business process.


Printed: 21/05/2004

Delivery Note R001010000318

Sly Development Ltd
3620263
15 The Avenue
Tel: B27 6NG
Fax: 0121 707 0060

Customer: P R Webber & Sons Ltd
53 Station Road

Delv Date: 21/05/2004

Code	Description	Qty
0000000001076	Bright Rhodium Plate Snake Chain Pendant Set	15
0000000001069	17 inch Colour Monitor	10

Figure 72. Roundtrip form[69]

For instance, when a person registers in a driving school, the secretary issues a payment document. This person goes to the bank and makes a deposit for the amount of the enrolment. Then this person returns a stamped copy of the payment document to the secretary in order to prove the payment. A similar treatment can be seen in many payment managements, and in sending and returning delivery notes.

The most important aspect of roundtrip forms is identification control; that is, how to identify the form in order to recognise it when it returns. Since the organisation is the one that issues the document in the first place, it is possible for the information system to generate identifiers that facilitate this control. For instance, an internal code can be generated (payment document number). Even more, some types of identifier allow rapid recognitions. For instance, barcodes (see Figure 72), QR codes, etc.

Roundtrip forms are associated to several communicative events. At least, one event for issuing the form and another event for receiving it back.

Strategic and tactical level listings

These forms can be the most complex of all. To understand them (and, needless to say, to design them), a deep knowledge of the business management is needed. In many cases these forms are introduced in the organisation quite late in time, when the organisation starts to gain experience with the information system.

These forms are a sort of managerial requirements. Managerial staff defines the business indicators that they need to have a clear view of the situation and to be at the helm of the business.

[69] Image from WinTill® Small Business Manager (wholesale edition) http://upd.wintill.co.uk

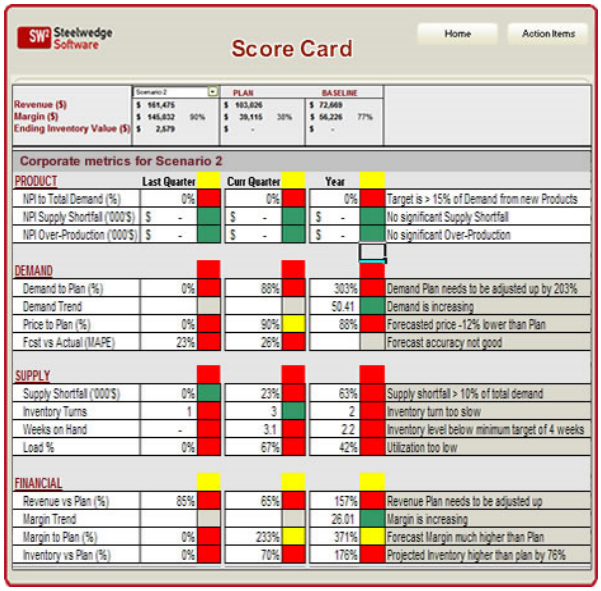


Figure 73. Scorecards are strategic listings[70]

These are not usually linked to communicative events. They are fairly simple in structure and layout. However, the generation of the information they carry can be very sophisticated. In many cases they involve a big design effort; it may even be necessary to undertake data warehousing or data mining.

Also, their related performance and graphical design requirements are important. Always treat well management people; they pay for the information system development.

These forms are often the most evolved and mature part of an information system.

Listings for operational support

These are normally printouts and listings that are linked to communicative events; that is to say, they are necessary in order to carry out daily operations of the organisation. Their complexity is medium. Normally, graphical design is not a critical requirement, but often it is important that their layout is appropriate for good performance (after all, they support common, repetitive tasks). For the same reason, the performance of the associated reaction needs to be as high as possible.

[70] Image from Steelwedge Software Performance Management http://www.steelwedge.com

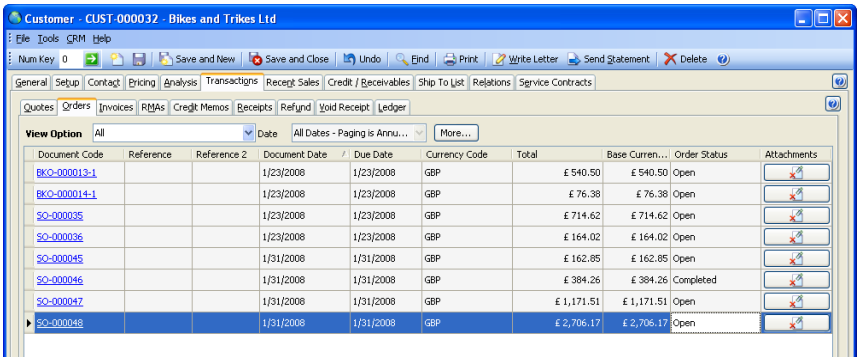


Figure 74. A listing for operational support

However, the most important aspect of operational-support listings is the information they provide. For instance, a list of client orders is a listing that supports many activities of a company that sells goods.

When analysing communicative events, it is useful to ask the users what information they need in order to carry out their tasks or to make decisions. For instance, if a user has to select the truck that will deliver a shipment, s/he may need to know the availability of the truck on the delivery date, the other deliveries that the truck has been scheduled for that day, in order to optimise logistics. This information will be displayed in listings for operational support.

Intermediate results form

Many organisations have information systems that are not entirely computerised. These information systems involve many manual operations. In these cases, the analyst can find some forms that seem input forms because they are filled in manually, but are actually output forms and listings.

These forms usually contain summarised information and calculated fields. This information is redundant (the data where it comes from is already known by the system and is stored elsewhere); however, intermediate results forms are (manually) updated in order to have useful information readily available.

For instance, products come in and go out of a warehouse. When information is managed manually, the persons in charge of the warehouse usually have daybook forms where they note down every input and output of goods (or they have one form for inputs and another form for outputs, which after all is the same). But it is also common to have an inventory list where the current amount of each product is shown; every time a product comes in or goes out, it is noted down on the daybook and the inventory list is updated.

WAREHOUSE DAYBOOK			
Week: 16			
Date	**Product**	**IN**	**OUT**
13/04/2009 08:30	t-shirt cotton orange	400	
13/04/2009 08:35	peaked cap cotton white	100	
13/04/2009 09:17	t-shirt cotton chartreuse green		55
13/04/2009 09:17	t-shirt cotton orange		195
13/04/2009 09:46	belt vinyl pink cats	320	

INVENTORY LIST	
Product	**Amount**
t-shirt cotton chartreuse green	45
t-shirt cotton dark yellow	38
t-shirt cotton orange	205
t-shirt cotton white	982

Figure 75. The inventory list is an intermediate results form

External listings

External listings are those listings that go beyond the organisation, to external users. Their appearance is important because it affects the corporate identity (the image of the organisation). Performance requirements are not critical unless the amount of external users is big; they depend on the expected frequency of issuance and workload.

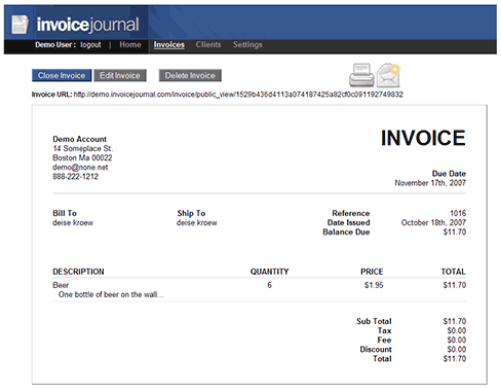


Figure 76. An invoice is an external listing[71]

[71] Image from Invoice Journal http://www.invoicejournal.com

For instance, an invoice is a document that the information system generates and is given to the client (see Figure 76). Other examples are catalogues, budgets, certificates and diplomas.

Listings and indicators for audit and control

Those that facilitate auditing the usage of the information system.

They can be oriented towards controlling staff with regards to their organisational responsibility: who is making use of the information system functions, when, and how. In this case, the corresponding input information is associated to a communicative event; that is, each time a communicative event takes place, the information system can automatically store a timestamp and data about the user (see Figure 77).

They can also be oriented towards finding out whether a particular event has taken place or not. For instance, has the invoicing of last month's orders actually been triggered?

They can be used to detect normative events that have not yet occurred. A normative event is a communicative event that necessarily has to occur, according to organizational rules; normally within a time frame (see Section 4.4.3.5.3 for more details on normative events). For instance, the return of a book that has been lent, the acknowledgement of receipt of a delivery (before a reasonable time limit).

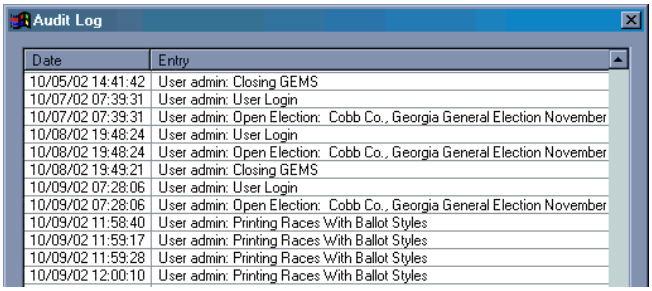


Figure 77. Audit logs are listings for auditing system information system usage[72]

[72] Image borrowed from http://www.wildboar.net/politics/voting/articles/scoop/S00065.htm

4.4.2.1.1 Analysis of input forms

Each and every input form has to be investigated. The analyst has to carry on, along with the users, the following investigations:

- Whether the form is filled in one go or incrementally in different moments in time. The corresponding communicative events are identified (each moment in time will probably correspond to a different communicative event, but the unity criteria must be applied to be sure). For instance, the client order form shown in Figure 78 is affected by more than one communicative event: the request of the order (*SALE 1*), the assignment to a supplier (*SALE 2*), and the supplier response (*SALE 3*). Each communicative event adds more information to the form[73].
- Which the temporal order in which communicative events occur is. In the case of the SuperStationery order form, the order of events is *SALE 1*, *SALE 2*, *SALE 3*. In any case, the order is not necessarily a sequence; complex behaviour in terms of workflow patterns should be carefully analysed and specified by means of a communicative event diagram and event preconditions.
- Who the primary actors of the communicative events are (those that are the source of the information the form is filled which). For instance, in the SuperStationery case, three different organisational actors provide the information to fill the client order form. It is the client who provides the information about the requested products (*SALE 1*), the Sales Manager decides which supplier is assigned the order (*SALE 2*), and the supplier, in case of acceptance, provides the planned delivery date (*SALE 3*).
- What message is conveyed in each communicative event (the fields of the form that are affected by the event). Observe that the message structure in Table 17 does not include fields such as Supplier or Planned delivery date, which correspond to later communicative events.

Using the above-mentioned sources and techniques, analysts identify communicative events and they specify, by means of message structures, the new meaningful information communicated in each event. An initial working artefact can a copy of a real business form provided by the organisation; e.g. marking out the set of fields that are acquired in each event, noting down who is responsible for each event, and indicating the order of occurrence so as to later specify the temporal sequence of events in the diagram.

---

[73] The analyst has to be careful not to think in an object-oriented way and mistake the fact that a form is structured in several parts with fact that there exist distinct events; for instance, a customer request form can be composed of general data, client data, and a list of product lines, but it all corresponds to the same communicative event.

The information conveyed in each event is mainly provided by the primary actor; in case the information of a field is provided by the primary actor, then it is attached an input *acquisition operation* (e.g. the quantity of items of a certain product that the client requests: Quantity i). Some informational elements may not be provided by the primary actor, but generated by the information system itself; in that case the acquisition operation is generation (e.g. order numbers, as well as invoice numbers, are usually pre-printed in form booklets: Order number g).

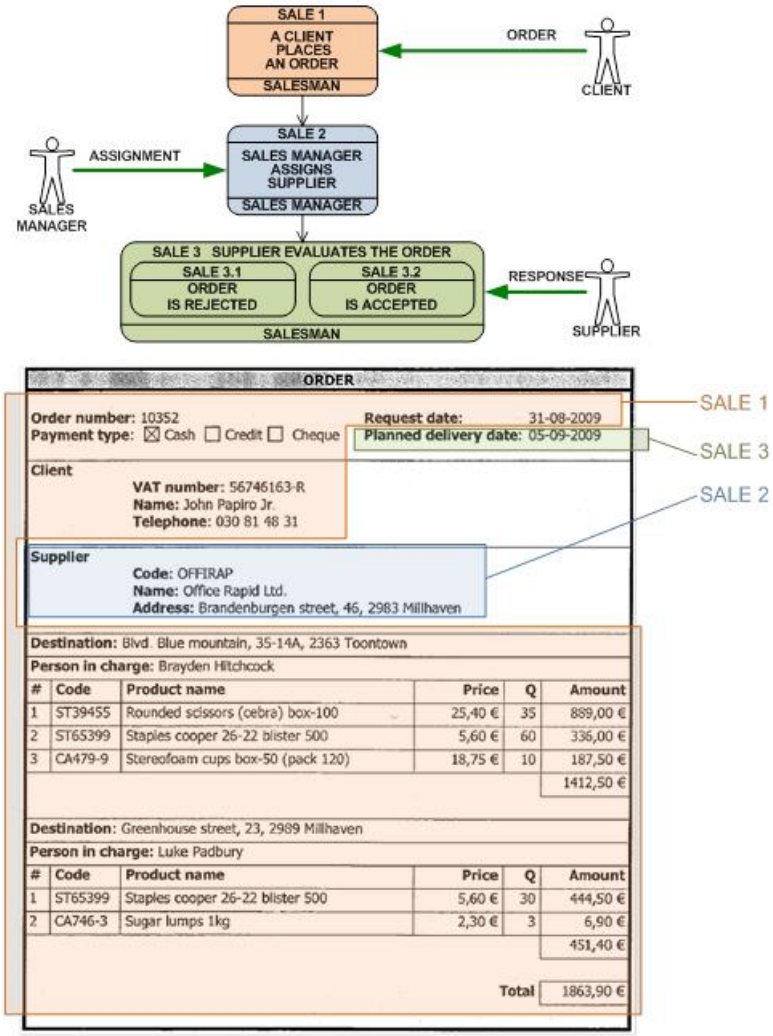


Figure 78. The SuperStationery order form is filled in incrementally in three steps

With regards to the *domain*, in analysis time it is advised to use five basic types for data fields: text, number, money, date, time. For instance, the domain of Quantity is number. For reference fields, the domain is the type of business object. For instance, the field Client identifies a client that has already been registered in the information system; thus, its domain is Client (similarly, Address refers to a client address and Product identifies a product of the catalogue). Also, if the values that the field can take are restricted to a predefined set and there is not much prospect that the set will be updated, then the domain can be expressed by means of a specialisation substructure (e.g. the domain of Type of work in the example in Table 18 is [theo|prac]).

Also, in order to enhance the comprehensibility of analysis specifications, an *example* is provided for each field (e.g. Request date, 31-08-2009), as well as a *description*.

In analysis time, it is discouraged to specify derived fields (data fields whose acquisition operation is d), as well as any other field property that is an aspect of design (see Table 20). Derived fields do not convey new information and they contribute to an unnecessary over-specification. It is convenient to postpone specifying derivation until the design stage. In order to avoid specifying derived fields, each time it is needed to identify a business object that is already known by the information system (e.g. a client), the guideline is to include a reference field (e.g. Client i Client), and to avoid including the information that characterises the object (e.g. the client name). Equally, the analyst will avoid including fields whose content can be derived from the rest of the information in the message (e.g. total amounts such as Amount or Total).

In any case, notice that the message structure in Table 17 includes a field named Price that corresponds to the price of the product. In fact, this field is infringing the above-mentioned methodological guidelines. The field actually refers to information that is already known by the information system (the price is part of the company catalogue). The analyst has included this field with a design solution in mind: the information system is intended to register the Price of the product at the time of the order request so as to guarantee that, even though the prices are yearly updated, the information on the orders and invoices will remain unchanged. However, this is only one possible design solution among others (e.g. a archival storage of prices). In case analysts decide to include design aspects in analysis time, then they should be well aware and justify their decision.

Summing up, forms are a very useful tool for analysis, but requirements investigation has to be guided by the search of communicative events. The existence of forms that do not correspond to data input (see intermediate results forms and listings forms, above) implies that the analyst should not investigate each form separately. The analyst should, in fact, investigate communicative

events in order to discover their temporal schema. It is necessary to ask the user for the communicative events that start a business process, and then add the subsequent events until the process is completed.

4.4.2.1.2 Analysis and design of output forms

Analysing listings and printouts in detail is simplified when the system data model is known. This does not mean that listings do not have to be documented until all the input forms have been analysed and the data model has been designed. The analyst can discover output information needs as they come up during the interviews with users. For each listing and printout, the analyst should specify their requirements regarding structure, appearance, and performance. However, output forms cannot be precisely defined until the data model is designed.

Also, designing listings and printouts often involves denormalising the data model (e.g. incorporating redundancy to improve the performance of listings, denormalising specialisations)

From a generative point of view, the complexity of the listing has to be assessed. Initially, it is enough to study whether the form is generable by using SQL or it needs an algorithm. In case an algorithm is needed, it is necessary to study the complexity of the algorithm.

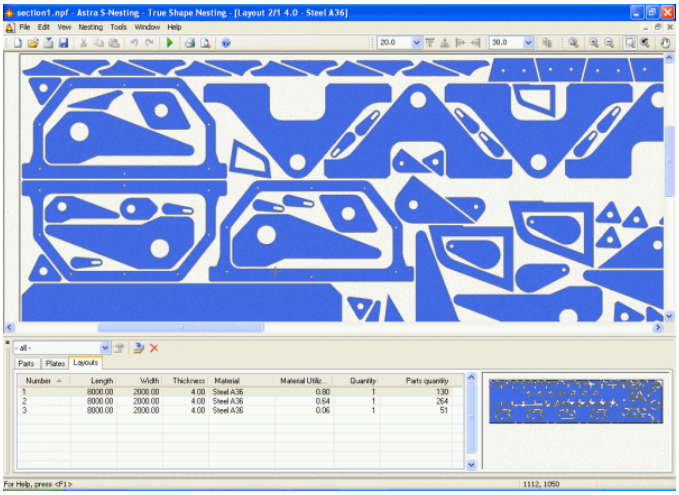


Figure 79. An optimised is an output form based on an algorithm[74]

[74] Image from Astra S-Nesting software by Technos http://www.techno-sys.com

For instance, a company that cuts wood and metal supplies pieces to customer companies. The company buys big boards to a timber yard and metal sheets to a metalworking manufacturer and, depending on the customer orders, cuts the material using mechanical saws and plasma cutters. The company has an operator that plans the cuts. Currently, the waste is in the order of 10%. The company wants to halve the waste.

This problem cannot be solved by means of a simple SQL query. The optimised cutting pattern is an 'output form' that requires a detailed algorithmic study; it is a problem of optimisation (e.g. the knapsack problem).

4.4.2.2 Analysis of process specification documents

Communicative events can be defined by analysing organisational business processes. When workflow specifications or quality proceedings exist, this documentation can be used as input for the analysis.

Normally this documentation describes the procedures that are followed in the organisation. It is oriented towards actor responsibilities; that is, it describes, for each procedure, what each organisational actor has to do: what responsibilities they have and how they have to act.

This documentation may not be well structured and it may contain inaccuracies, ambiguities and outdated specifications. In that case, it is convenient to reengineer the specification so as to update it and structure it properly (see Section 4.4.2.2.1); also, the analyst will check whether certain events related to the state of business objects are missing (see Section 4.4.2.2.2).

4.4.2.2.1 Reengineering specifications to adopt a communicational orientation

The available documentation is often subject to improvements. This does not necessarily mean that business processes (a.k.a. organisational procedures) have to be changed, but the structure of the business process models can be improved. For instance, logical events (communicative events) can be separated from physical events, giving the model a layered structure. To do this, the analyst interviews the users and asks the appropriate questions in order to identify communicational unity: the initial trigger that establishes contact, the conveyed message, and the synchronism in organisational reaction (see Section 4.3.1.2). This way, it is possible that something that is documented as a single activity actually corresponds to two or more communicative events. Or perhaps two or more activities are actually one single communicative event. If this happens, the specification of business process models can be changed accordingly (see Figure 80).

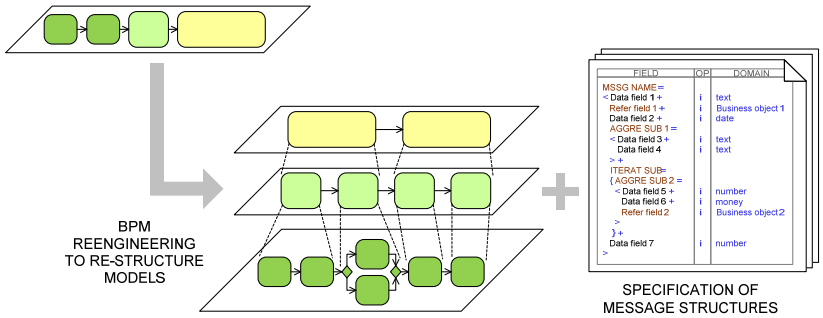


Figure 80. Reengineering of business process models to provide a communicational orientation

It is of utmost importance that the message structures that correspond to each of the logical events are specified. This gives the business process models a communicational orientation.

In most cases, the observation network[75] that supports communicative events is already deployed. The analyst just has to reconsider whether the instant when communicative events are triggered is appropriate, and whether the content of the conveyed messages fulfils organisational needs.

If the observation network were not deployed (for the any organisational process or just a part of one), the analyst would have to design it (see 4.4.2.3).

4.4.2.2.2 Analysis of state observation

The structure of the message associated to many communicative events is trivial. The information that is being reported is just the fact that a change has occurred in the subject system (or an actor has the intention to produce some change in it). The analyst has to determine whether that information is actually relevant for the organisation. These events can affect basic or mediator elements, contributing to increase their value. Every time something happens in the organisation the analyst has to consider whether this occurrence changes the state of any basic or mediator element.

For instance, a company makes kitchen furniture on demand. A customer places an order, production is planned, the order enters production, the order is

[75] The observation network is the set of interconnected agents (human actors, sensors, etc.) that act as informants of the information system. For instance, an operator that reports to the information system that a truck full of raw material has arrived at the dock, or a sensor that reports to the information system that a delivery has been palletised (i.e. the articles that constitute a delivery have been wrapped, packaged, labeled and piled up in pallets).

moved to the loading dock, the order is shipped out to the customer. Therefore, the order goes through different states.

The analyst has to model these states and then check whether there is any missing state that the company would be interested in knowing, in order to include the corresponding communicative event (see Figure 81).

For instance, it may happen that some orders have been completed but they still remain in the production warehouse until someone picks them up. The fact that the production has been completed does not guarantee that the order is in the shipping department, at the loading dock. The pieces of furniture of an order have a different value, a different state, depending on whether they are at the loading dock or not. Therefore, the state of an order is changed by the event that reports that the order has been moved to the loading dock. The structure of the message is minimal: the order identifier[76]. The important fact to know is that the order is already at the loading dock and, thus, it can be loaded by the transporter. Not having this information available may imply losing time and money.

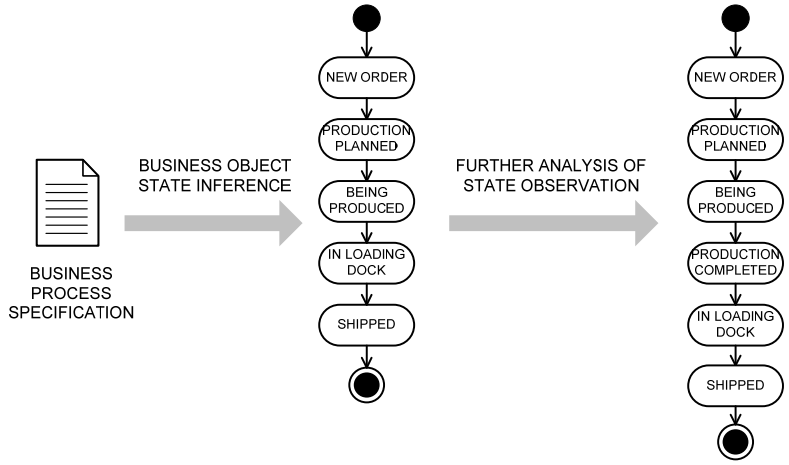


Figure 81. Business object states can be inferred from process specifications; further analysis can be done

[76] Additionally, the company may want to know the date and time when the order has been moved to the loading dock, for auditing purposes (e.g. in order to calculate business indicators such as the mean production time or the mean shipping time).

4.4.2.3 Generative analysis in the absence of documentation

Sometimes organisational procedures are not documented, it is necessary to elicit them via interviews and to analyse the gathered information in order to identify communicative events.

The analyst should define processes and design forms together with representative users (e.g. Figure 82 shows a sketch of a form created by an analyst along with a user). It is not a single-handed task. Organisational actors such as managers, department heads and experts in all areas within the scope of the project have to be involved and encouraged to think carefully about the structure of the information they need for their daily tasks, as well as about the processes that are to be followed. Otherwise, there are many threats to the validity of the outcomes of the analysis; i.e. it may not fulfil the organisational needs, and later the implemented system will probably have hidden defects and faults that will cost money.

In some cases, the analyst has to design the observation network for a particularly complex part of the business processes. The analyst has to decide which instant is more appropriate for gathering information, the mechanism by which the information is actually acquired and reported to the organisation, and the various alternatives for supporting the acquisition. The objective is to minimise the use of resources and guaranteeing the best possible adaptation to organisational procedures (e.g. not to disturb the operations). However, the users have to fully understand and validate the analyst proposal.

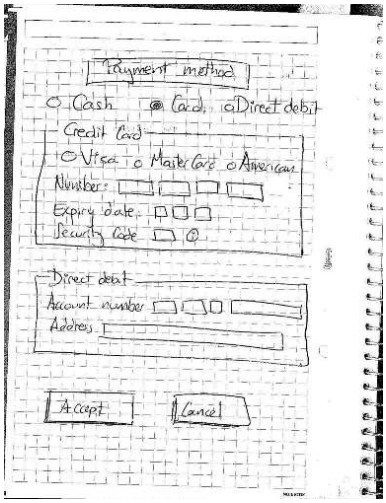


Figure 82. Users are able to design many of the forms they need

4.4.3 Revision of communicative behaviour

All the communicative events of an information system are important. However, not all events entail the same complexity. It is common to focus the attention first in analysing those business processes that the organisation considers more relevant.

For instance, in a tailoring & dressmaking company, they may perceive as important the reception of a client order, the request of the cloth and buttons, the reception of the material, the actual tailoring, the picking of the order by the client, and the payment of the order.

However, there are other communicative events that remain hidden or, at least, they do not appear in such an evident way. For this reason, the analyst should revise the process specifications together with the users, in search of new events and requirements.

This way, the interviewed user of the tailoring & dressmaking company may have not expressed during the first interviews that (i) the company wants to register regular clients to avoid having to include their details each time they place an order; (ii) the fact that, in order to reduce the risk of non-payment, the manager wants to personally approve (or reject) those orders that amount more than a certain quantity; (iii) the provider might not deliver the requested cloth on time and, thus, the client has to be informed of the delay; (iv) the client might require taking the dress in at the waist.

The following sections explain what aspects of the process definition can be revised and how the analyst can proceed during revision.

4.4.3.1 Analysis of exceptions and alternatives

There are several acknowledged problems in requirements elicitation: (1) the fact we, human beings, have limitations concerning information processing, (2) our tendency to make a biased selection of information, and (3) our limitations concerning problem solving [Davis 1982]. In our industrial practice with Communication Analysis, we have realised that, commonly, interviewed users only report "normal" events during the first discussion about a business process; that is, the come up first with those events that occur more frequently. They tend to give little importance to exceptional events. This related to the second of the above-mentioned problems. As a result of this phenomenon, the analyst has to further investigate exceptions and alternatives because, although proportionately irrelevant, they must be discovered and specified or else the information system under development will not provide support for them.

4.4.3.1.1 Exceptions in internal treatments

There may appear exceptions in internal treatments; that is, exceptions in the organisational reaction, in events that occur inside the Organisational System. They are usually events that are related to approvals or decision taking. The analyst has to check for the possibility of rejections or alternative decisions.

Questions do not have to be asked in positive, but in negative terms. If the analyst asks –Are expense reports always authorised?-, the user can still respond –Yes- without thinking too much. However, if the analyst asks –Is it absolutely impossible that an expense report is rejected?-, then the chances of finding an exceptional behaviour are increased. It is important to find out these behaviours or otherwise the implemented information system will not fully support these contingencies.

In the illustrative example, the analyst should also investigate whether the employee could insist in case of rejection; for instance, by providing new proofs of the expenses and a letter justifying the total amount, or by lowering the total amount. In such case, a loopback in the process would appear.

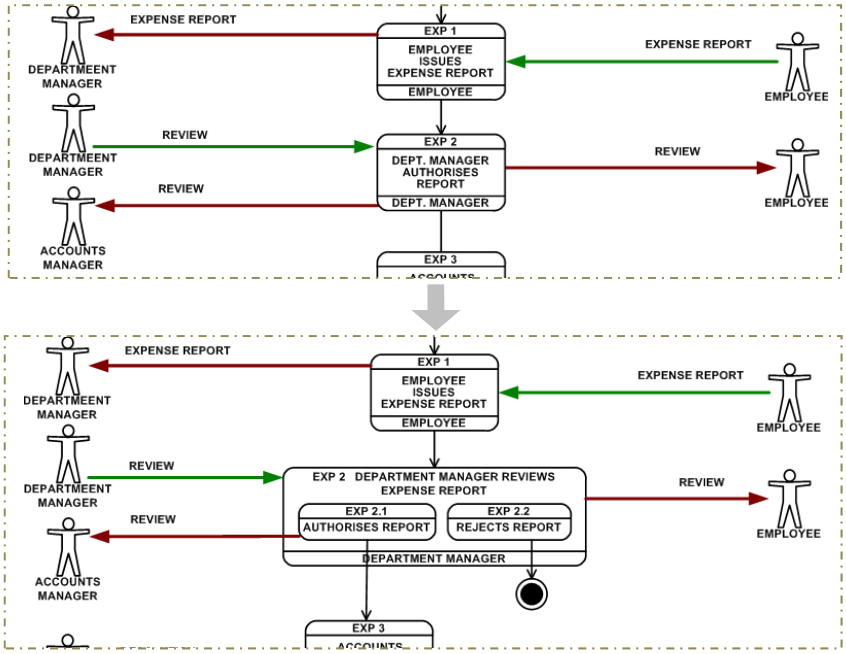


Figure 83. Exceptions in internal treatments have to be discovered and specified

4.4.3.1.2 Deviations from optimist assumptions

Another kind of exception that is worth considering is related to deviations from optimist assumptions. For instance, the organisation could, under certain circumstances, run out of resources. In processes that use resources (e.g. planning, allocations, etc.), the analyst has to investigate whether it is possible that these resources are not available due to saturation (i.e. the organisation is already making use of all resources).

Saturation can affect mediator objects that are responsible of a specific management (e.g. telephone operators of a technical assistance service, sales department clerks that are responsible of managing transportation and shipments).

Saturation can also affect basic objects that are being consumed in a production process (e.g. the warehouse uses up all raw material –wood boards, screws or hinges- while manufacturing furniture) or in serving a request (e.g. no available camping plots, production department runs out of red t-shirts while packaging a customer order).

4.4.3.1.3 Exceptions in external treatments

There may appear exceptions in external treatments; that is, exceptions in events that occur in the Subject System but outside the range of action of the Organisational System (i.e. in the environment, outside the organisation).

For instance, when analysing shipping management, it is necessary to investigate possible incidences. What happens if the truck has a breakdown? What happens if the client rejects the shipped goods? One way of investigating these incidences is imagining that something happens between two events that are expected to be consecutive (e.g. between the consignment of a delivery –the moment the truck leaves the loading dock- and the acknowledgement of receipt -the moment the customer informs that the delivery has been received-).

The possibility of things happening in a different way may lead to the definition of new communicative events that are related to specialised behaviour: the exception or incidence turns into an alternative path; this path had not been initially considered.

4.4.3.2 Basic data maintenance

When analysing a particular communicative event, other communicative events that are linked to the former can appear. This is the case of **conditioned registries**, which are communicative events that (conditionally) need to be triggered before another communicative event in order to register some information (the condition is that this information is not already in the information system memory).

For instance, let's consider a video club (see Figure 84; we use the Data Flow Diagram notation [Yourdon 1989] to clarify the concepts presented herein). A client wants to borrow a film; if this person is not already a member of the video club, then the clerk registers this person as a member before lending him the film.

Furthermore, it is typical that some basic and mediator objects are managed with simple **CRUD operations** (create, read, update and delete). For instance, it may be the case of the customers of a company; the information system is expected to offer a means for managing them (e.g. registering a new customer, updating the data whenever a change is notified, deleting the data in case the customer demands so). The set of CRUD operations that affect a business object can be seen as a sub-system that contains several communicative events. The analyst can opt not to include these events in the Communicative Event Diagram; in fact, it is recommended not to include them unless their inclusion really clarifies the way the organisation works, in order to keep the diagram as simple as possible. Also, the events can be specified in a concise way instead of separately. A single communication structure can serve the purpose of specifying several create and update events related to the same business object.

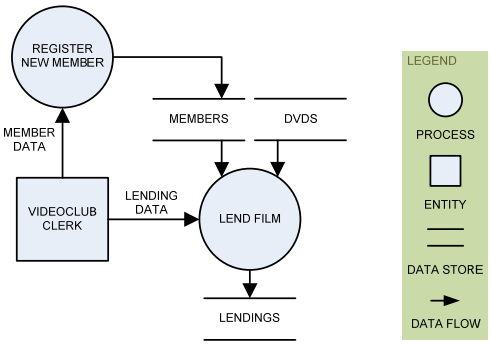


Figure 84. Conditioned registry

Table 22. One communication structure specifying two CRUD-related communicative events

FIELD	DOMAIN	CREATE	UPDATE	MEMORY	EXAMPLE VALUE
CLIENT RECORD =				CLIENT	
< Client code +	number	i		< Code +	C11836
VAT number	text	i	i	VAT +	84776963-Y
Client name +	text	i	i	Name +	Lola Castañuela
Telephone +	text	i	i	Phone +	086 775 36 37
Registration date	date	i		RegDate	12/09/1989
>				>	

4.4.3.3 Analysis of linked communications

Communicative events serve as source of information to infer the need of linked outgoing communications. These communicative interactions are referred to as *linked* communications because, during the analysis, they appear linked to a communicative event (i.e. they are normally discovered while analysing a communicative event). Moreover, in design time, they should also be linked to the communicative event: the software form or screen that supports the communicative event should allow triggering the output messages that correspond to the linked communications.

The needs for linked communications may arise as the communicative events associated to a business form are analysed in detail.

For instance, when a citizen places a request for state subsidised housing, s/he requires certain information in order to formulate the request, such as the conditions of the request of the list of available houses; these are referred to as prior outputs (see Section 4.4.3.3.1).

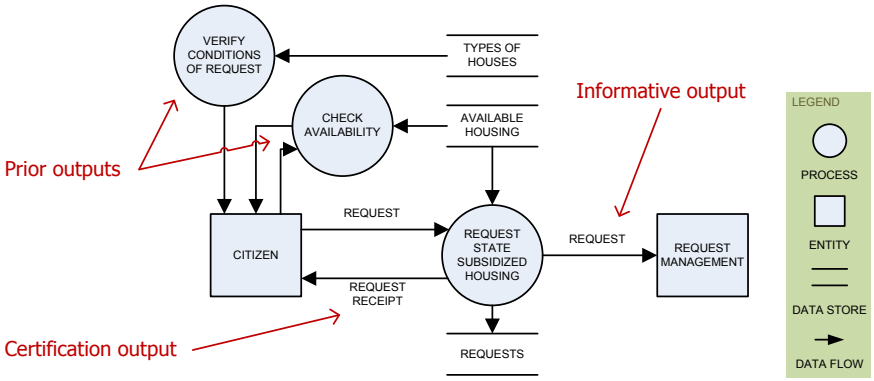


Figure 85. Outgoing communications linked to a communicative event

When the request is placed, the citizen will receive a receipt that certifies the placement; this is referred to as a certification output (see Section 4.4.3.3.2). Then the request should be sent to the appropriate department; this is referred to as an informative output (see Section 4.4.3.3.3).

4.4.3.3.1 Prior outputs

In order to formulate the input message, primary actors need prior information that allows them deciding the data of the message. For instance, allocating staff to a project may require a listing that shows the availability of the staff.

In order to discover prior outputs, the analyst can ask the user about the need of prior information for each communicative event. The questions to be asked are, what information do you need in order to carry out this task? What information could the information system provide you in order to facilitate your job?

A prior output allows deciding which the most suitable selection for the communicative event input data is. When the communicative event usually involves taking a decision, making an election, an estimation, a judgement, or a diagnosis, then the prior output aids in this task.

A communicative event can require more than one prior output. For instance, when citizens want to request a state subsidised housing, they first need to know the terms and conditions of the request, as well as the housing availability (see Figure 85).

4.4.3.3.2 Certification outputs

These outgoing communicative interactions are given to the primary actor in order to certify that a communicative event that they have triggered has been completed (i.e. that its corresponding transaction has been performed successfully). It is a printout that usually contains the input data (a.k.a. hard copy). For instance, a receipt so a citizen can certify that s/he has requested housing within the time frame (see Figure 85), or a copy of an order that is issued for a client for future references.

4.4.3.3.3 Informative outputs

The aim of informative outputs is to communicate a member of the organisation the occurrence of a communicative event. That is, some communicative events imply reporting to a member of the organisation the fact that the communicative event has occurred, as well as the content of the input message.

In order to discover these outputs, the analyst asks the users about which organisational actor should know that a specific communicative event has

occurred, who has to react to that event taking into account the new information. It is convenient to find out for whom that information is important and for whom it is necessary, since informative outputs can have two different purposes:

- To inform an organisational actor of the occurrence of a communicative event so s/he can trigger another communicative event in response.
- To inform of the state of affairs to an organisational actor who does not have an immediate responsibility but should be aware of it.

4.4.3.4 Audited outputs

Outgoing communicative interactions (e.g. listings, printouts, output forms) mainly imply showing or distributing information that the information system already knows. That is, the information system has a memory of facts that have been previously communicated; an actor provides some sort of selection criteria (e.g. "I want to consult all the wholesale warehouses that sell long-sleeved shirts") and the system shows the requested information. There is no provision of new meaningful information (by new and meaningful information we mean facts that the organisation is interested in knowing and the information system is not aware of them yet).

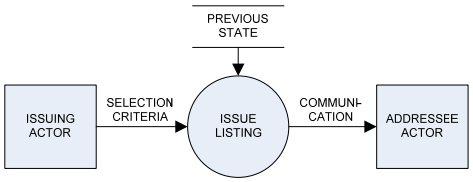
However, outgoing communicative interactions involve information input in the case that the organisation wants them to be audited (see a generic example in Figure 86).

4.4.3.4.1 Issuance audit

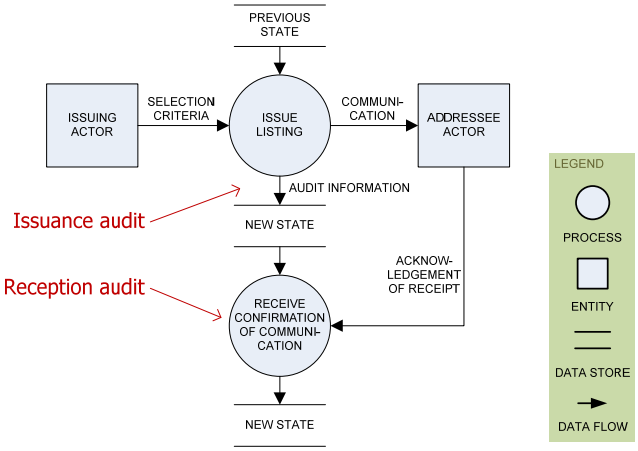
The organisation wants to control or monitor their occurrence. It may be necessary to introduce a communicative event for the issuance (if it had not been defined already). For instance, each time a specific document is issued, a timestamp and the identifier of the user are recorded. This kind of audit is particularly important in the case of confidential information.

4.4.3.4.2 Reception audit

The organisation wants to control that the addressee actually receives the information. Issuance audit only monitors that the information has been sent, but it does not guarantee that the addressee has received it. It is necessary to introduce a communicative event for this purpose. For instance, the addressee confirms the reception of the document.



a) Before auditing the listing



b) After auditing the listing

Figure 86. Input information related to a listing

4.4.3.5 Audited inputs

In some cases, inputs are audited. The analyst should investigate which communicative events require some sort of audit. Audit of inputs is typically required when trustworthiness or reliability issues exist, or when business indicators are required.

4.4.3.5.1 Audit of occurrence

This is done when some members of the organisation have to know that a communicative event has occurred. In this case, it is only needed to know about the occurrence, not about the content of input the message.

The organisation probably wants to audit the occurrence of a communicative event when the organisation suspects that the actor that is responsible for triggering the event may fail to do it, or may be acting a bit unhurriedly (i.e. too calmly). Also, when a member of the organisation is interested in knowing the activity rates of one or more communicative events.

For instance, the organisation may ask themselves the following questions, which lead to the definition of indicators:

- Whether the timing of events is changing. Is the mean elapsed time between packaging and truck loading increasing?
- Whether the frequency of a certain event is decreasing. Do we have the same amount of customer orders this month than the same month last year?

4.4.3.5.2 Audit of message content

The analyst should investigate whether the organisation really trusts the information that a person provides in a communicative event, or whether there are risks that the business is not willing to take. In case trust issues or risks associated to the content of the message exist, someone will have to supervise the content (it needs to be audited).

Audit of message content allows defining business rules that affect the organisational reaction to a communicative event. Sometimes, these rules lead to the definition of new communicative events.

For instance, the following issues related to trust need to be taken into account: supervision of valuations and appraisals, verification of estimates and statements.

Valuations or appraisals that an actor does and the organisation wants to limit or supervise. For instance, if a lecturer requests a software application that has a license that exceeds 200,00€ then the purchase has to be first approved by the faculty dean (see Figure 87). This sets a restrictive block and a new communicative event for authorisation has to be defined. The business rules for the block also have to be specified.

Verification of **estimates or statements** that an actor makes. For instance, when registering an application an applicant for a job claims to have ten-year experience of Oracle database management. This should be verified and, thus, a communicative event for reviewing the applicant's merits is defined. Also, some insurance claims are assessed by an expert to determine whether the actual cause of the damage corresponds to the cause that the client reports.

When analysing the audit of message content, two requirements investigations need to be carried out.

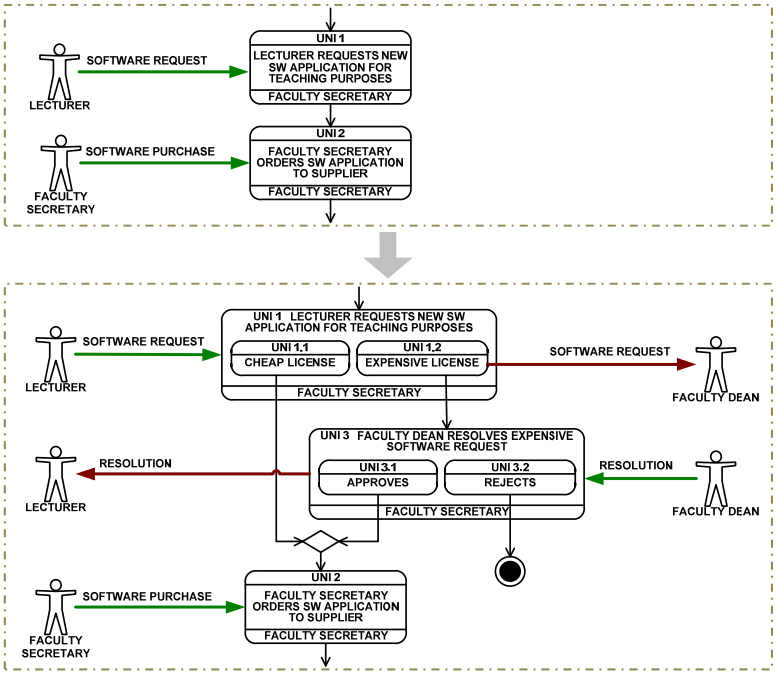


Figure 87. Authorisation event to supervise message content

The analyst investigates the most convenient way of informing the "auditor" that the condition associated to the business rule has been satisfied[77]. If the auditor uses the software system frequently, the analyst/designer can propose a listing that shows all pending audits each time the auditor logs into the system (or first thing in the morning, or on user's request). If the auditor does not use the system frequently, it is necessary to forward this information to him/her with due speed (e.g. using the email system).

If it has been decided that a communicative event for approving/rejecting an operation is needed, then the analyst/designer has to investigate the most convenient way of supporting this event. Auditing the message content (actually, performing the corresponding verification or resolution) has to facilitated. The same mean of communication that is used to inform the auditor of a pending audit has to allow him/her to react (i.e. to respond, to perform the audit, to resolve). For instance, if a listing of pending audits is provided, then the same window/screen should allow the reaction.

[77] It could also be said that the business rule has been transgressed, depending on how it is formulated and the meaning it has.

4.4.3.5.3 Audit of normative events

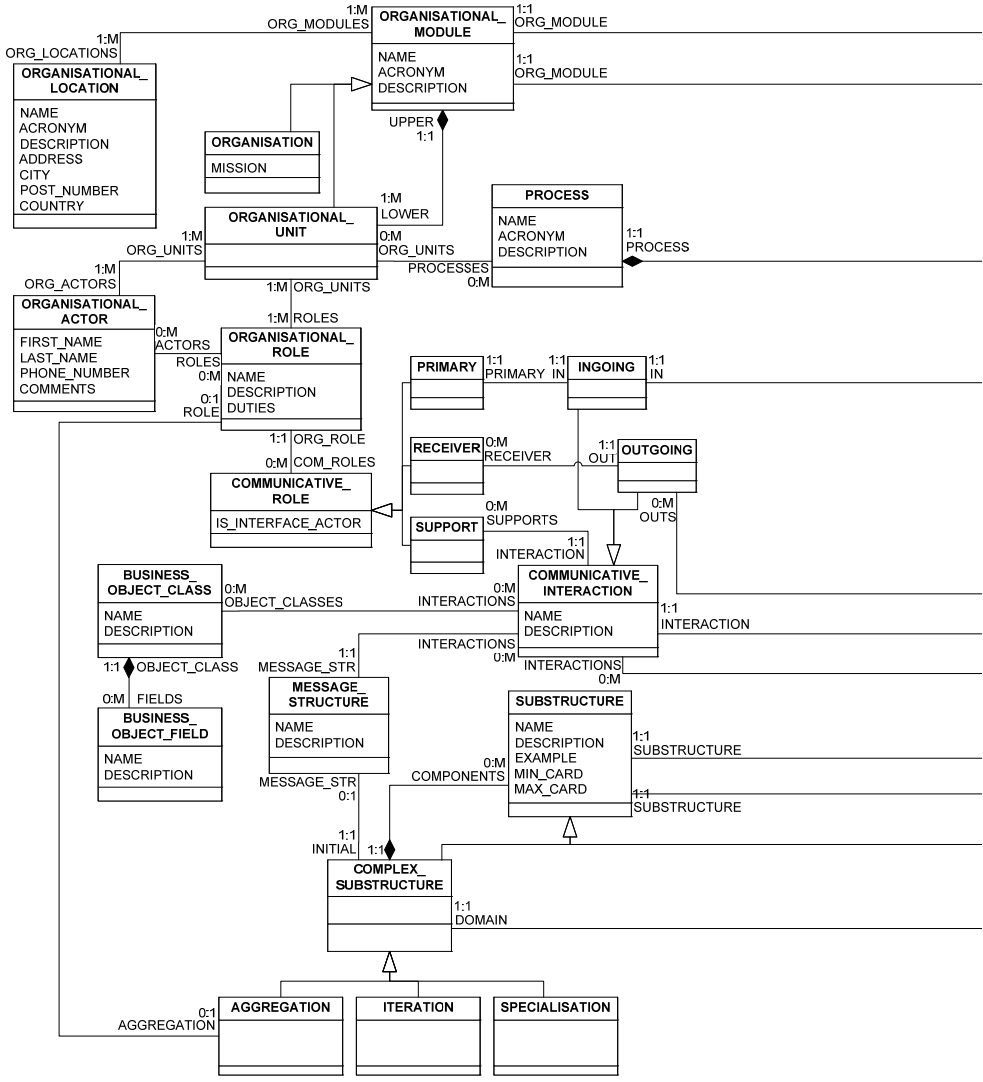
Business process models often specify normative sequences of events. When a normative sequence of events is defined, the users expect that, once a given event has occurred, the subsequent event should occur within a reasonable time frame. In these cases it is important to know whether the event has occurred or not. For instance, it is normative that, after a plane has taken off, it should land in a reasonable amount of time. It is also normative that, if a client places an order and the goods are delivered, the order will be invoiced in a reasonable amount of time.

Depending on the seriousness that is attributed to the events, different types of audits are required. Outputs related to the audit of normative events can be supported using different communication protocols. If the non-occurrence of a normative event is really serious, the audit can be synchronous in order to communicate the situation to the corresponding auditor as soon as possible (e.g. by SMS, by a pop-up window, a sound alarm). Otherwise, the audit can be asynchronous; that is, the information is displayed on demand, when the auditor decides 'to take a look at the state of things' (e.g. when the auditor opens a listing of incidences).

4.5 Communication Analysis platform-independent metamodel

As part of the method definition, we propose a metamodel that contains many of the constructs that are defined by the method specifications in previous sections. The metamodel is depicted in Figure 88. The metamodel includes the metaclasses intended for modelling the dynamic view of the information system. It mainly covers the three first requirements levels: L1 concerned with organisational modelling and problem decomposition, L2 concerned with business process modelling from a communicational perspective, and L3 concerned with communicative event specification.

We have omitted some metaclasses intended for modelling the static view of the information system because this aspect is expected to be covered by metaclasses of the OO-Method after the integration of both methods (e.g. metaclasses related to normalised business object class information definitions; that is, those metaclasses intended for modelling the computerised information system memory).


ORGANISATIONAL_ MODULE
NAME
ACRONYM
DESCRIPTION
ORG_MODULES
ORG_MODULE
ORG_MODULE
UPPER
ORGANISATIONAL_ LOCATION
NAME
ACRONYM
DESCRIPTION
ADDRESS
CITY
POST_NUMBER
COUNTRY
ORG_LOCATIONS
ORGANISATION
MISSION
ORGANISATIONAL_ UNIT
LOWER
ORG_UNITS
PROCESSES
PROCESS
NAME
ACRONYM
DESCRIPTION
PROCESS
ORG_ACTORS
ORGANISATIONAL_ ACTOR
FIRST_NAME
LAST_NAME
PHONE_NUMBER
COMMENTS
ACTORS
ROLES
ORGANISATIONAL_ ROLE
NAME
DESCRIPTION
DUTIES
ROLE
PRIMARY
PRIMARY
IN
INGOING
IN
RECEIVER
RECEIVER
OUT
OUTGOING
OUTS
ORG_ROLE
COM_ROLES
COMMUNICATIVE_ ROLE
IS_INTERFACE_ACTOR
SUPPORT
SUPPORTS
INTERACTION
BUSINESS_ OBJECT_CLASS
NAME
DESCRIPTION
OBJECT_CLASSES
COMMUNICATIVE_ INTERACTION
NAME
DESCRIPTION
INTERACTIONS
INTERACTION
OBJECT_CLASS
FIELDS
BUSINESS_ OBJECT_FIELD
NAME
DESCRIPTION
MESSAGE_STR
MESSAGE_ STRUCTURE
NAME
DESCRIPTION
SUBSTRUCTURE
NAME
DESCRIPTION
EXAMPLE
MIN_CARD
MAX_CARD
COMPONENTS
SUBSTRUCTURE
SUBSTRUCTURE
INITIAL
COMPLEX_ SUBSTRUCTURE
DOMAIN
AGGREGATION
AGGREGATION
ITERATION
SPECIALISATION

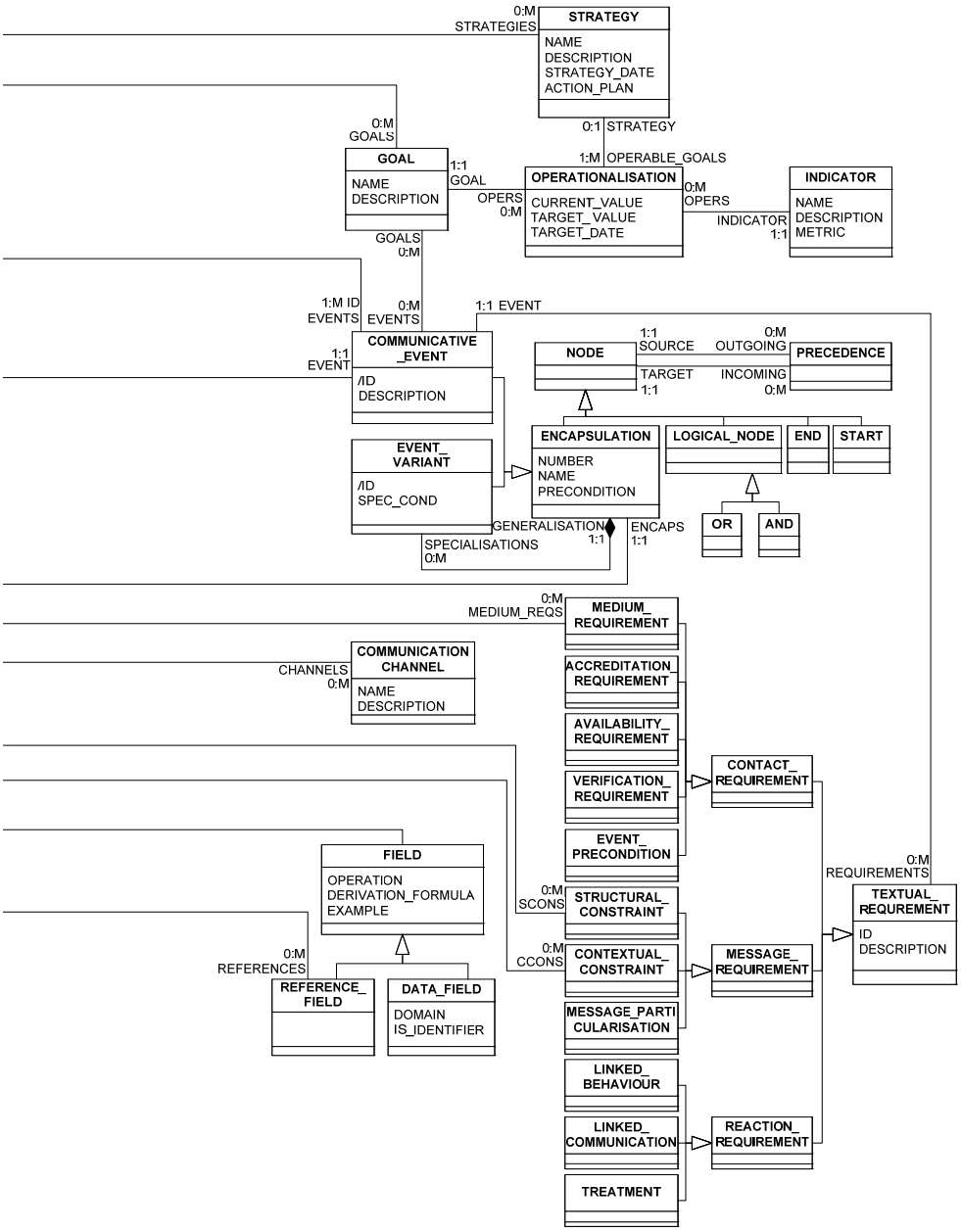


Figure 88. Communication Analysis platform-independent metamodel

Obtaining a complete requirements taxonomy is an ongoing long-term research project. There are some prior attempts (e.g. [White and Edwards 1995; Gorschek and Wohlin 2005; Myers and Hathaway 2005; Volere 2006]), but defining a well-founded taxonomy for information system requirements and implementing a proper tool support is still an open challenge. As discussed above (see page 136), the taxonomy of textual requirements is just provided as an exemplification of our vision that every requirement in a requirements model should be categorised properly. For implementation purposes, this specialisation hierarchy could be replaced by a more flexible way of supporting the taxonomy, similar to the one shown in Figure 89. Since it is also arguable whether textual requirements only apply to communicative events, we provide for a means to associate them to any qualifiable element of the requirements. However, this solution would require populating the class REQUIREMENT_TYPE with the taxonomy and finding a means for preventing some nonsensical applications of the taxonomy (e.g. a message requirement of type structural constraint does not apply to organisational units). After all, it remains an open challenge.

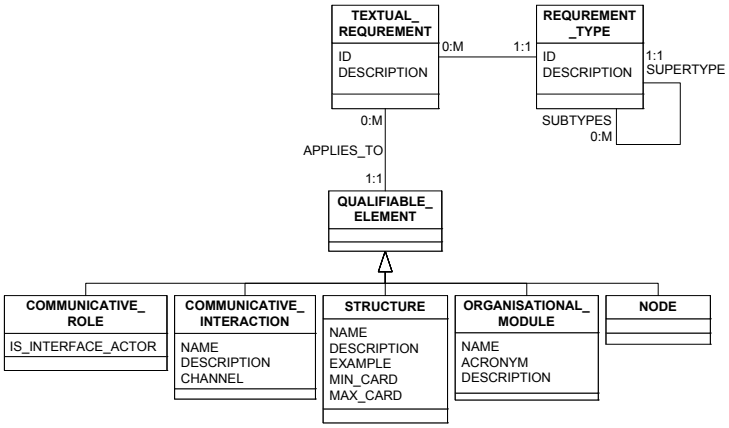


Figure 89. A more flexible support for the requirements taxonomy

Table 23 presents two formulas corresponding to the derived attributes that calculate the identifier of communicative events and event variants. This way, events of the *Sales management* process will have identifiers of the form *SALE 1*, *SALE 2*, *SALE 3*, etc. Also, the variants of the specialised event *SALE 3* have a compound identifier of the form *SALE 3.1* and *SALE 3.2*.

Table 23. Derivation formulas for the Communication Analysis PIM metamodel

Derivation Formulas	
F1	COMMUNICATIVE_EVENT.Id = concat(GENERALISATION.Id, IntToStr(Number))
F2	EVENT_VARIANT.Id = concat(PROCESS.Acronym, IntToStr(Number))

The following constraints restrict what can be expressed using the metamodel in Figure 88. These rules are part of the grammars of Communication Analysis modelling languages.

Table 24. Constraints for the PIM metamodel of Communication Analysis

Constraints	
C1	A start node must not have incoming precedence relations.
C2	An end node must not have outgoing precedence relations.
C3	An and node must have two or more incoming precedence relations and only one outgoing precedence relation. Take into account that the and node represents the and-join, since the and-split is implicit (see Section 4.3.1.1).
C4	An or node must have only one incoming precedence relation and two or more outgoing precedence relations. Take into account that the or node represents the or-merge, since the or-branch is implicit in communicative event specialisation (see Section 4.3.1.1).
C5	Within each business process, each communicative event must have a distinct number.
C6	Within each specialised communicative event, each event variant must have a distinct number.
C7	A formula is either a specialisation condition, an initialisation formula or a derivation formula, but not several of these at the same time (this means that an instance of FORMULA can only have a link via one of the relationships).
C8	The initial substructure of a message structure cannot be a specialisation substructure. Thus, given an instance of COMMUNICATIVE_EVENT, it cannot be linked via Message_Structure to a COMPLEX_STRUCTURE of type SPECIALISATION.
C9	The minimum cardinality of a message structure field (an instance of FIELD) is 0 or 1, and the maximum cardinality of a field is 1.
C10	The minimum and the maximum cardinality of an aggregation substructure (an instance of SUBSTRUCTURE) is 1.
C11	The minimum and the maximum cardinality of a specialisation substructure (an instance of SPECIALISATION) is 1.

The constraint C8 could have been expressed by specialisation on the metamodel (see Figure 90), but we ruled it out for the sake of simplicity.

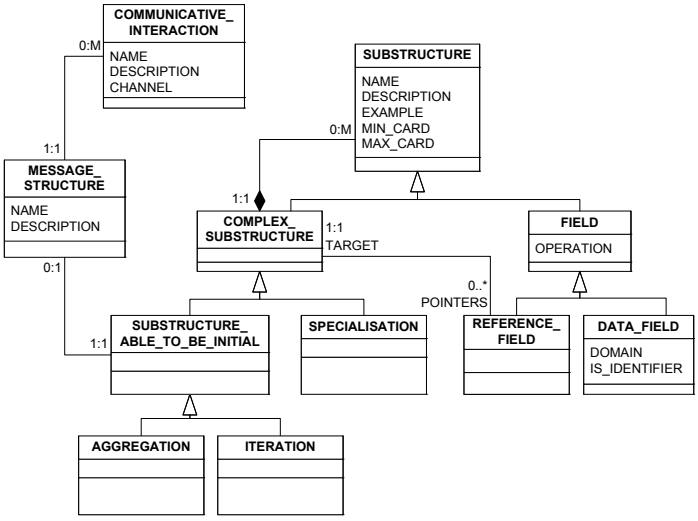


Figure 90. An option for expressing constraint C8 directly in the metamodel

4.6 Related works

In the following we provide a discussion that compares the presented approach with existing methods. It does not intend to be an exhaustive and systematic comparison but our aim is to help the reader understand the similarities with other methods, as well as the distinctive features of Communication Analysis.

Position of Communication Analysis within the area

There exist distinct orientations with regard to requirements elicitation for information system development. *Goal-oriented approaches* intend to identify stakeholders' necessities, modelling them as goals, where a "a goal is an objective the system under consideration should achieve" [van Lamsweerde 2001]. E.g. Map [Rolland 2007], i* [Yu and Mylopoulos 1994] and KAOS [Dardenne, van Lamsweerde et al. 1993]. Among *agent-oriented approaches*, which design the system as a set of autonomous and automatable agents, Tropos includes a goal-oriented requirements stage [Castro, Kolp et al. 2002]. *Usage-based approaches* describe the interaction between the user and the software system under development. E.g. Use Cases [OMG 2010b] and Info Cases (an extension of the

former) [Fortuna, Werner et al. 2008]. *Value-oriented approaches* identify and model value object exchanges [Weigand, Johannesson et al. 2006]. E.g. e3-value [Gordijn and Wieringa 2003]. *Aspect-oriented approaches* apply the separation of concerns principle [Dijkstra 1976] to requirements engineering. E.g. Early Aspects [Rashid, Sawyer et al. 2002] and Theme/DOC [Baniassad and Clarke 2004]. Some organisational modelling approaches propose modelling and integrating multiple views of the system [Bubenko, Brash et al. 1998; Scheer 2000; de la Vara, Sánchez et al. 2008]. There also exist *communicational approaches*. In this field, a widely extended orientation is the Language Action Paradigm (LAP) [Flores and Ludlow 1980; Winograd and Flores 1987], which is mainly based on the work of Austin [Austin 1962] and Searle's speech act classification [Searle and Vanderveken 1985]. Communicative Action Paradigm (CAP) is an evolution of LAP that extends the paradigm to non-verbal communication [Dietz, Goldkuhl et al. 1998]. Several approaches stem from LAP, such as Action Workflow [Medina-Mora, Winograd et al. 1992], SAMPO [Auramäki, Lehtinen et al. 1988], Business Action Theory [Goldkuhl 1996], DEMO [Dietz 1999] and Cronholm and Goldkunhl's Communication Analysis [Cronholm and Goldkuhl 2004]. SANP [Chang and Woo 1994] adopts a similar approach, but it is based on Ballmer and Brennenstuhl's speech act classification [Ballmer and Brennenstuhl 1981]. *Semiotic approaches* to organisational modelling have also been proposed [Stamper 1997].

This thesis presents a communicational approach. Communication Analysis does offer support for goal modelling but the conceptual framework regards this aspect as relevant for organisational modelling and we plan to investigate on how to integrate the goal-orientation with our communicational perspective. Likewise, value network modelling has not been considered, but the method is usually put into practice in existing organisations with well established businesses, not with the intention to support an "innovative e-commerce idea" [Gordijn and Wieringa 2003]. In any case, information system development is better tackled with a contingent approach, so we are open to integrating the value perspective with Communication Analysis in the future as well.

Some features give advantage to our proposal over other methods. Unlike the above-mentioned approaches, Communication Analysis places the emphasis on specifying the messages that correspond to ingoing communicative interactions; to that end, we propose Message Structures. Also, the communicative roles defined above (see Defs. 175-178 and Defs. 189-190) distinguish Communication Analysis from other approaches. Many business process modelling techniques use support actors and/or reaction processors as criteria for organising processes in swimlanes but (communicatively speaking) primary actors are disregarded (e.g. [de la Vara, Sánchez et al. 2008]). However, primary actors are central to our

approach[78]. Furthermore, most requirements engineering approaches do not specify communicational content of interactions (or message specification is mixed with system usage description, as in Use Cases). Info Cases is an exception, since this technique proposes a structured text specification of messages. However, Message Structures have greater expressiveness (e.g. specialisation, iteration, information acquisition operation).

With regard to communicational approaches, we share with them the communicational perspective and many foundations borrowed from Communication Theory. However, Communication Analysis differs in several conceptual foundations, in the underlying requirements structure, and in the business process modularity guidelines. Communication Analysis does not necessarily preconceive a specific speech acts classification nor assumes conversational patterns, as LAP-based approaches do. An analyst following our approach does not impose patterns on the organisation, but confines to discovering the communication needs of the organisational stakeholders, shedding light on their work practice and identifying possible improvements. Our proposal coincides with the one by Cronholm and Goldkuhl in the communicational perspective. Also, both proposals consider organisational documents (e.g. business forms) invaluable sources of information. However, Cronholm and Goldkuhl choose existing documents as a starting point in requirements engineering process and their modelling notation (the Document Activity Diagram) is document-centred; that is, communicative interactions are subordinate to documents. We choose communicative interactions as a starting point and the modelling notation that we propose in this paper (the Communicative Event Diagram) is communicative interaction-centred. We argue that communicative events represent pure work practice and that it is possible to discover and describe them independently of their associated documents. Documents are a specific technological support[79] (solution space) for a communicative event (problem space); that is, documents are the result of a previous information system implementation.

A blend of the constructivist and the mechanistic perspectives

With regard to the conceptual framework for understanding business processes by Melão and Pidd [Melão and Pidd 2000], our proposal combines two of the four perspectives; namely the constructivist and mechanistic perspectives. Both perspectives are reflected in the underlying ontology. Business processes are

[78] Organisations can replace many support agents with computer interfaces (e.g. clerks vs. web-based forms) and IS reaction processors are typically automated. Primary actors, however, are irreplaceable because they are the ultimate source of information.

[79] We do not necessarily refer to computer technology; paper is an ancient form of technology.

defined both from the mechanistic view as predefined flows of actions (along with all the necessary statements about the involved roles, the resources, timing constraints, etc.) and from the constructivist view that considers them as socially constructed conceptions, norms that should (but do not always) guide actions. The first view coincides with that of Davenport and Short [Davenport and Short 1990], Hammer and Champy [Hammer and Champy 1993], Armistead and Rowland [Armistead and Rowland 1996], and Kock and McQueen [Kock and McQueen 1996]. The second is in line with the FRISCO stance and Galliers [Galliers 1994], Patching [Patching 1995], Chan & Choi [Chan and Choi 1997] and Mumford [Mumford 1994]. Both perspectives are also reflected in the method:

- In order to discover business processes, a constructivist stance is adopted: business processes are considered a social construct that is agreed among stakeholders, and the requirements engineer acts as facilitator in this agreement.
- In order to describe business processes, a mechanistic stance is adopted: the use of models that are based on actors, events and messages allows creating a requirements specification that serves as a starting point for later software design.

Discovering requirements allows their description and, conversely, information system description provides feedback to discovery by allowing new interactions with stakeholders (e.g. to formulate new questions). Both perspectives are intertwined. The combination of hard (mechanistic) and soft (constructivist) approaches is not new to the information systems scene [Brown, Cooper et al. 2006], but Communication Analysis contributes a requirements structure and communication-oriented modelling techniques that do not appear in previous proposals.

Another feature of modelling techniques that has implications of their expressiveness and usage is whether it models business behaviour procedurally or declaratively (see Table 25). According to the features of the Communicative Event Diagram, it stands somewhere in between both paradigms, but closer to the procedural one. We provide no explicit language for expressing business concerns and business rules. Communication Analysis can enforce certain business rules using control-flow-based modelling primitives, as other procedural process languages do (e.g. BPMN [OMG 2011], BPEL [OASIS 2007], UML Activity Diagram [OMG 2010b]); however, declarative modelling has the advantage of being able to formulate the rues independently from the process models in which they are to be enforced, preventing redundancies. With respect to modality, we depart from the usual *necessity modality* of procedural process models. The semantics of precedence relationships is not as the token flow semantics of BPMN or Petri Nets [Peterson 1981], but it specifies part of the precondition of each encapsulation of the diagram (a communicative event or an

event variant). It allows for other modalities like intention (what ought), possibility (what can) and certainty (what is); however, to fully take advantage of this potential, the language should be extended to make it more expressive.

Table 25. Procedural versus declarative process modelling (from [Goedertier and Vanthienen 2007])

	Procedural modelling	**Declarative modelling**
Business concerns	implicit	explicit
Execution scenario	explicit	implicit
Execution mechanism	state-driven	goal-driven
Modality	what must	what must, ought, can
Rule enforcement	procedural (what, when, how)	declarative (what)
Communication	explicit (how)	implicit (what)

With regards to communication, the very nature of the method enforces the modelling of the ingoing communicative interactions and their associated message structures; it is also recommended modelling outgoing communicative interactions in the diagrams, but this is left to the analyst criteria. However, a communicative event diagram does not depict communication logic in a procedural manner; it does not specify how and when communication occurs. In this sense, the method is closer to the declarative paradigm; according to [Goedertier and Vanthienen 2007], the declarative modelling style enhances modelling-time flexibility, as it allows to model business processes irrespective of the communication channels and protocols: messages can be "sent by the producer (push model), retrieved by the consumer (pull model) or via a publish-subscribe mechanism" but this is not specified in the process model.

With regards to Message Structures

A notable ancestor of Message Structures is Backus-Naur Form (BNF). BNF is a notation for context-free grammars that was proposed during the development of the compiler Algol 60 [Backus, Bauer et al. 1963]. The grammatical constructs are the sequence, represented by contiguity, and the alternative, represented by a vertical bar |. Later extensions incorporate the curly brackets { } for repetitions, and the square brackets [ ] for the alternative. Structured Analysis adapts BNF for the specification of data structures [DeMarco 1979]. Fortuna et al. [Fortuna and Borges 2005] propose extending UML Use Cases with a BNF-based notation to specify informational interfaces, with a similar intention to Message Structures.

However, no previous notation includes an operator to explicitly parenthesise sequences. This prevents the analyst from including substructures within

substructures and forces the fragmentation of complex substructures. For instance, the first structure (a) presents an ambiguity that can only be avoided by fragmenting the structure in two parts (b) or, as proposed by Communication Analysis, parenthesising the aggregation (c). An advantage of (c) over (b) is that (c) preserves the unity of the message.

a) Vehicle=NumberPlate+Brand+Model+Motor=CubicCapacity+Valves+Fuel+Colour
b) Vehicle=NumberPlate+Brand+Model+Motor+Colour
Motor=CubicCapacity+Valves+Fuel
c) Vehicle=NumberPlate+Brand+Model+Motor=<CubicCapacity+Valves+Fuel>+Colour

Other techniques that allow modelling the interaction between the organisational actors and the information system can be used instead of Message Structures; for instance, UML Sequence Diagrams and ConcurTaskTrees [Panach, España et al. 2008]. However, our experience with these techniques has shown us the convenience of using a lightweight notation, such as Message Structures, which has the advantage of allowing a fast, textual specification.

Notations are contingent

Our last remark concerns notations; especially the Communicative Event Diagram. The proposed business process modelling technique consists of a set of interrelated concepts, criteria plus other methodological guidance, and notations. We believe that, in general, it is the concepts and criteria that we propose what matter the most (not the notations). We actually acknowledge that some practitioners would be more comfortable using other notations for business process modelling (e.g. BPMN, Activity Diagrams, Use Cases). There is no problem with that, as long as the notation is adapted to support the communicational perspective. In fact, this has been done before (see Section 6.3.2).

4.7 Summary

This chapter has presented a detailed specification of Communication Analysis, with the intention to pave the way for its integration with a model-driven development method; namely, the OO-Method. Communication Analysis provides a framework for eliciting, analysing and modelling information systems from a communicational perspective. It proposes a requirements structure with five levels (L1 to L5). Each level has been explained, its underlying concepts have been defined on top of the FRISCO 1.1 ontology (presented in Chapter 3), and some examples of its application have been provided. L1. System/Subsystems (see Section 4.2.1) deals with the structure and the goals of the organisational system. L2. Process (see Section 4.2.2) deals with discovering and modelling business process models in terms of communicative events. L3. Communicative

Interaction (see Section 4.2.3) specifies in detail each communicative event, including its associated message. L4. Usage Environment (see Section 4.2.4) specifies the interface and the memory of the information system. L5. Operational Environment (see Section 4.2.5) enters architectural design. With regards to Communication Analysis, both L4 and L5 are out of the scope of this thesis, because the OO-Method will cover these levels once both methods are integrated (this is done in Chapter 5). To clarify the scope of the proposed requirements structure, it has been projected onto the well-known Zachman framework for enterprise architecture (see Section 4.2.6).

The chapter has also presented with full detail the modelling techniques that Communication Analysis proposes for requirements specification:

- Communicative Event Diagram, a business process modelling technique that adopts a communicational perspective and facilitates the development of an information system that will support those business processes (see Section 4.3.1). The most important feature of the technique is a set of unity criteria that guides model modularity (see Section 4.3.1.2), since it can be applied to other modelling notations as well.
- Message Structures, a modelling technique for the specification of messages communicated with (and within) the organisation (see Section 4.3.2).
- Event Specification Templates, a textual requirements specification technique intended to support the description of communicative events (see Section 4.3.3).
- Also, several techniques for requirements elicitation and requirements analysis are presented. Generative analysis (see Section 4.4.2) is a collection of techniques for discovering requirements based on the analysis of existing business forms and existing business process specifications. Once an initial version of the requirements model is created, a revision of the communicative behaviour can be performed (see Section 4.4.3) to discover missing requirements and, therefore, improve model completeness.
- In order to facilitate the integration of Communication Analysis into a model-driven development framework, a metamodel of the method has been provided (see Section 4.5). The metamodel presented in this chapter is platform independent, in the sense that it does not yet regard any CASE-tool development platform and it purely focuses on clarifying the constructs of the modelling techniques (a metamodel that is specific for the Eclipse Modeling Framework is presented in Section 8.2).
- Lastly, we have positioned Communication Analysis in the business process modelling and requirements engineering arena by comparing it to other approaches (see Section 4.6).

Chapter 5
Integration of Communication Analysis and the OO-Method

"Adding wings to caterpillars does not create butterflies,
it creates awkward and dysfunctional caterpillars.
Butterflies are created through transformation."

Stephanie Pace Marshall

5.1 Motivation

Model-driven software development methods have evolved since the advent of the Model-Driven Architecture paradigm [OMG 2003], but the support for the requirements engineering stage is still immature (see Section 2.3). There is still a need for rigorously-defined model transformations that bridge the gap between the business requirements and the information system conceptual model. Also, better support for requirements traceability is needed; as well as sound empirical validations that demonstrate the benefits of the model-driven approach.

Moreover, not many requirements engineering approaches actually focus on the essence of information systems; that is, providing support to organisational communication. Picking up these problems, this chapter presents the integration of Communication Analysis, a communication-oriented requirements engineering method, and the OO-Method, an object-oriented conceptual modelling method. The focus is put on combining both perspectives appropriately and on providing a systematic derivation of conceptual models from requirements models. This chapter aims at providing a set of derivation guidelines that can be understood and manually applied by an analyst. The implementation of model transformations is considered in Section 8.3.

The remainder of the chapter is organised as follows. Section 5.2 explains the way we approach the method integration. Section 5.3 summarises the most important concepts of the OO-Method. Section 5.4 presents an overview of the derivation strategy. Section 5.5 provides the guidelines that allow deriving the OO-Method Object Model (an extended UML Class Diagram) and the Dynamic Model (a set of UML State Machines) from Communication Analysis requirements models. Section 5.6 concludes with a summary.

5.2 *Approaching the integration the MDA way*

The Model-Driven Architecture paradigm promoted by the Object Management Group (OMG), besides providing a framework for structuring in several layers the models of a system under development, it also distinguishes several approaches to model transformation. The MDA Guide [OMG 2003] refers to the specification of a model transformation as MDA mapping. In our conceptual framework for model-driven system development defined in Section 3.5, we consider an MDA mapping to be a set of model transformation rules (see Def. 150). According to the MDA, the following types of mappings can be distinguished:

- *Model type mappings*. A model type mapping specifies a mapping from any model built using types specified in the source language to models expressed using types from a target language.
  - *Metamodel mappings*. It can be defined whenever the types of model elements in the source and the target models are both specified as metamodels that conform to the Meta Object Facility [OMG 2006b] (in this context, MOF can be seen as a language for expressing metamodels).
  - *Other type mappings*. It accounts for non MOF-based metamodels.

In any case, properly speaking, any kind of model type mapping is a metamodel mapping, since any modelling language has an implicit metamodel even if it is not expressed according to the MOF. From our point of view, what the OMG is distinguishing is simply metamodel mapping with and without *the MOF brand*.

- *Model instance mappings*. Another approach to mapping models is to identify model elements in the source model which should be transformed in particular way, given a specific target modelling language.
  - Marks. Model instance mappings can be based on the definition of marks. A mark is a qualification of an element of the source model by which the element is related to a concept that belongs to the target modelling language. A mark is an input to decision making within model transformation rules (i.e. how the marked element is to be transformed).

OMG acknowledges, however, that most mappings consist of some combination of the above approaches. Otherwise, it is difficult to provide enough expressiveness to a mapping. A model type mapping is only capable of specifying transformations in terms of deterministic rules whose behaviour (e.g. decisions) is restricted to the information contained in the source metamodel.

In Section 5.5, we describe a set of transformation rules (we refer to them as derivation guidelines). The derivation guidelines use a combination of model type mappings and model instance mappings. Whenever a mark type needs to be used, we provide an explanation and examples of its application. Since it is convenient to structure and constrain marks in order to systematise their usage [OMG 2003], we have metamodelled the mark types and provided guidance for their application and interpretation.

5.3 Brief reminder of the OO-Method

OO-Method is a model-driven object-oriented software development method that focuses on modelling enterprise information systems at the conceptual level [Pastor, Gómez et al. 2001]. The Integranova technology framework supports modelling and, by means of a *Model Compiler*, fully-functional three-layered software code is generated automatically [CARE Technologies]. This feature adds extra value to deriving a conceptual model from a requirements model.

The OO-Method Conceptual Model consists of four complementary views (also referred to as models):

- *Object Model*. An extended UML 2.0 Class Diagram that specifies the memory of the information system.
- *Dynamic Model*. It specifies the valid lifecycles of objects and the interaction between them by means of a collection of state-transition diagrams (that are compliant with the UML 2.0 State Machine Diagram).
- *Functional Model.* It describes the reaction of the information system by means of specifying changes in the objects state using an abstract pseudocode.
- *Presentation Model*. It models the user interface by means of abstract patterns.

Object Model

The modelling primitives of the Object Model include those expected from a UML Class Diagram: classes have attributes and services, and they are interrelated by means of structural relationships. In any case, some primitives that extend the UML definitions or that are particularly relevant to the derivation are explained next:

A *class* is a "structural and behavioural abstraction that is shared by a set of objects" [Pastor and Molina 2007]. Graphically, classes are represented by a rectangular box with three areas: heading, attributes and services. The class name is shown in the heading area and it must be unique within the context of the conceptual model.

An *attribute* is an structural property of a class; they represent a piece of data about an object. The properties of an attribute its name (unique within the context of the class), the attribute type (either "Constant", "Variable" or "Derived"), a flag indicating whether the attribute is part of the class identifier (either "Yes" or "No"), its data type (always a simple data type and not an object; allowed data types are Nat, Int, Real, Autonumeric, String, Text, Time, Date, DateTime, Real, Bool, Image and Blob[80]), its size (in case the data type is String), a default value (a well-formed formula), a flag indicating whether the attribute is required upon creation, a flag indicating whether null values are accepted[81].

A *service* is "a processing unit", "the basic components associated with the specification of the behaviour of a class" [Pastor and Molina 2007]. A service name must be unique within the context of the class. Other optional properties of a service are its alias, the comments and the help message. Services can be of three types; namely, atomic services[82], transactions and operations (operations are out of the scope of this thesis, but the other two are defined below). Each service has one or more arguments.

An *argument* is a basic property of a service. The argument name must be unique within the context of the service and it cannot coincide with the name of any argument of the class. Services also have a data type, a size (in case the data type is String), a default value, a flag indicating whether null values are accepted. Optional properties of arguments are its alias. There are two kinds of arguments; namely, inbound arguments (their value is set by the user at the moment the

[80] We clarify the meaning of some data types that are not straightforward: *Nat* stands for natural number, *Int* for integer, *Autonumeric* refers to a sequence of natural numbers that is incremented automatically every time an instance is created, *String* refers to an array of characters of a specified length, *Text* refers to a multi-line text, *Bool* stands for Boolean, *Blob* stands for binary large object. Take into account that the data type Autonumeric should only be used for constant attributes (e.g. class identifiers).

[81] Many primitives also have the following optional properties: an alias (an alternative to the attribute name that does not have to conform to the strict naming conventions such as no_spaces_allowed), the comments (for model documentation purposes), and a help message (to offer hints to the user on runtime). However, we ommit them for the sake of brevity.

[82] Atomic services are often referred in the OO-Method as events. However, we have reserved this term for the Communication Analysis constructs (communicative, logical and physical events).

service is triggered and they constitute the input to the service reaction) and outbound arguments (their value is a consequence of the service reaction and it is determined by a well-formed formula).

An *atomic service* is an execution unit that represent the abstraction of a state change that occurs instantly. Several types of atomic services are distinguished.

- A *creation service* (a.k.a. new) creates new instances of a class and initialises its attributes.
- A *destruction service* (a.k.a. destroy) is in charge of destroying an instance.
- An *edit service a*ffects a single instance by changing the values of its attributes.
- A *shared service* is shared by two classes between which a structural relationship exists. This type of service is in charge of creating (insertion shared service), destroying (deletion shared service) or changing (change shared service) links between instances.

A *transaction* is a molecular execution unit that composes several atomic services from one or several classes. Their behaviour is specified by means of a transaction formula and their execution in the Integranova framework fulfils the ACID properties [Haerder and Reuter 1983].

An *operation* is a molecular execution unit, identical to a transaction except for the fact that its execution needs not fulfil the atomicity (all-or-nothing rule) and the isolation properties.

A *precondition* is a well-formed formula that is associated to a class service. The precondition must hold at the moment that a service is triggered, in order to allow its execution.

An *integrity constraint* is "a semantic condition that must be preserved in every valid state of an object" of a given class. Besides from a well-formed formula that specifies the condition, a error message must be specified; it will be displayed to the user at runtime, whenever the integrity constraint is violated.

An *agent class* (or simply *agent*) is a special type of class that represents a role played by the users of the software system.

An *agent relationship* is a special kind of mapping between agents and attributes or services of other classes. They grant rights to the users to view information or trigger information system functionality.

A *structural relationship* indicates that two classes are related; reflexive relationships in which only one class intervenes are also possible. In theory, the OO-Method allows specifying the meaning of the relationship by indicating the relationship type (either association, aggregation or composition); each type has its own graphical representation. However, in practice (when using the Integranova technology), the type of relationship can be disregarded because it does not have any implications in the compilation of the model into generated code. Each class participating in a relationship is said to have a role in that

relationship; in the case of reflexive relationships, one class plays both roles, one at each side of the relationship. Structural relationships have minimum and maximum cardinalities on each role side. The role names indicate how the instance (or instances) of one of the classes in the relationship are identified when seen from the an instance on the other side of the relationship. The temporality indicated whether the relationship is static or dynamic. Temporality is a directed property; one side of the relationship is static when, once the link between two instances has been created, it cannot be changed or destroyed unless the instance in the opposite side is also destroyed. The relationship can also denote identification dependency, which specifies that the identifier of the component class includes the identifier of the composite class (the one at the side with cardinality 1:1).

A *specialisation relationship* relates a parent class with a child class to indicate that the child class inherits the attributes and services from the parent class, being able to define other attributes and services of its own.

- For each specialisation relationship, at least one service of the parent class must be designated the *carrier service*, meaning that whenever an instance of the parent class successfully executes this service, then it also becomes an instance of the child class.
- A service of the child class (including those services inherited from the parent class) can be designated the *liberator service*, meaning that whenever an instance of the child class successfully executes this service, then it stops being an instance of the child class and remains being only an instance of the parent class.

A *cluster* is the means for partitioning the Object Model into groups of classes, according to any arbitrary criteria determined by the analyst.

Dynamic Model

The Dynamic Model consists of a state transition diagram per each class in the Object Model. The state transition diagram imposes constraints on the triggering of class services by means of the following modelling primitives[83].

A *state* represents a situation in which a class instance can find itself during its lifetime. There exist three kinds of states in the Dynamic Model.

- A *pre-creation state* represents the theoretical situation in which objects are immediately before they are created. Only creation services (new) depart from states of this kind. Pre-creation states cannot have incoming transitions. As state transition diagrams necessarily has one and only one pre-creation state.

[83] An informal explanation that clarifies the meaning of each primitive within the OO-Method is provided. For a more formal specification of the State Transition Diagram modelling technique see [Pastor and Molina 2007].

- A *destruction state* represents the theoretical situation in which objects are immediately after they are destroyed. Only destruction services (destroy) come into states of this kind. Destruction states cannot have outgoing transitions. A state transition diagram can have zero or one destruction state.
- An *intermediate state* represents a situation in which an instance has been created and not yet destroyed. An intermediate state can have both incoming transitions and outgoing transitions. The class services that apply to transitions are edit services and shared services. A state transition diagram can have one or many intermediate states.

A *transition* represents a change of state of an object, caused by the execution of a class service, triggered by an agent. A transition departs from a source state and arrives at a target state. Transitions are characterised by the following elements:

- The *service* that causes the change of state. The service must be part of the class for which the state transition diagram is defined. Graphically, the service labels the transition.
- A *control condition* is a well-formed formula used to determine the destination state that is reached after executing a service that labels multiple transitions having the same source state and different target states. The formula is constructed using constant values, object attributes (including those visible in related objects) and service arguments.
- The *list of agents* that are entitled to trigger the transition.

Functional Model

An *evaluation* is a rule that is used to specify the reaction of atomic services. Evaluations are characterised by the following elements:

- The *atomic service* to which the evaluation corresponds.
- The class *attribute* to which the evaluation applies. It must be of type Variable, and it must belong to the same class as the service.
- The *evaluation effect* is a well/formed formula that determines the value of the attribute whenever the service occurs. The formula can involve constants, functions, attributes of the same and related classes and arguments of the atomic service.
- An *evaluation condition* is a well-formed formula that specifies in which cases the evaluation effect has to be applied. This element is optional; in case it is not defined, then the evaluation is always applied when the service is triggered. However, it is useful in order to define several effects upon which one should be selected each time the service is triggered.

According to [Pastor and Molina 2007], the transaction formula belongs to the Object Model. However, in this thesis, we treat the transaction formula as if it

were ascribed to the Functional Model, since evaluations and transaction formulae specify the reaction of atomic and molecular services, respectively.

5.4 *Overview of the derivation strategy*

As shown by Figure 91, the Object Model is derived from the elements that appear in the Communicative Event Diagram and the Event Specification Templates. The Dynamic Model is initially derived from the Communicative Event Diagram; then some transitions are added when processing the Event Specification Templates. The Functional Model is derived from the Event Specification Templates. Although the Presentation Model can also be reasoned from the information contained in the Requirements Model (preliminary work can be found in [España 2005]), this thesis focuses on the derivation of the Object Model (i.e. the reasoning of the class diagram). However, some guidelines for the derivation of the Dynamic Model are also proposed.

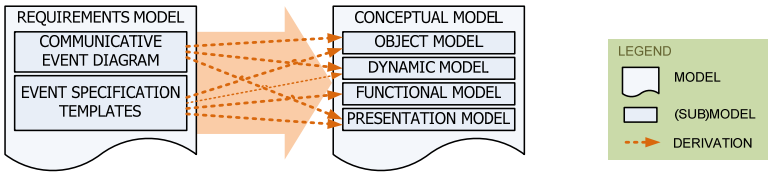
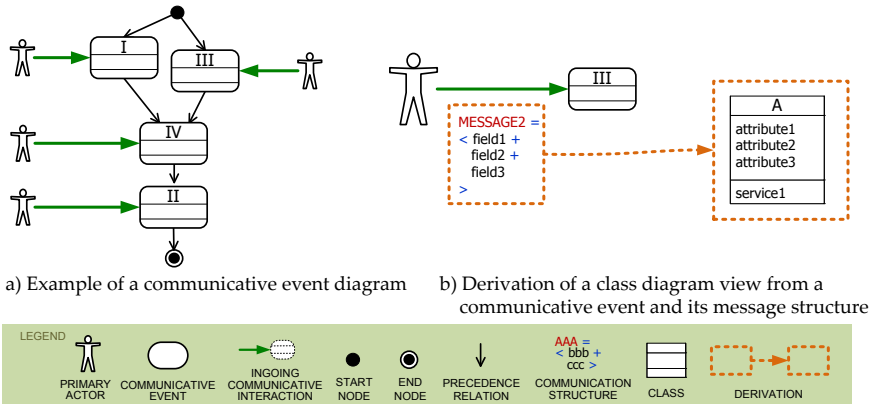


Figure 91. Derivation strategy from Communication Analysis to OO-Method models

5.4.1 Outline of the derivation of the Object Model

The Object Model specifies the memory of the information system by means of an extended UML class diagram. In order to obtain the Object Model, the communicative events of the Communicative Event Diagram are processed. Requirements related to communicative events are specified in detail in the Event Specification Templates; therefore these templates contain relevant information for the derivation procedure, as well. Specifically, for each communicative event, the message structure corresponding to its ingoing communicative interaction is processed in order to derive a class diagram view. We refer as *class diagram view* to a portion of the class diagram that includes one or several portions of classes and structural relationships among them.



a) Example of a communicative event diagram

b) Derivation of a class diagram view from a communicative event and its message structure

Figure 92. Simplified example of the derivation of a class diagram view from a communicative event

By portion of a class we mean some or all of the attributes and services of the class (including their properties). The concept of class diagram view with regard to conceptual models is analogous to the concept of relational view with regard to relational database schemas[84]; relational views allow displaying different perspectives of the same database.

Figure 92.a presents a simple example of a communicative event diagram that is used for illustration purposes[85]. Figure 92.b informally represents how a class diagram view is derived from a communicative event. Two dashed-line rectangles and an arrow represent that a class diagram view (on the right) is obtained from processing the message structure (on the left) that specifies the message conveyed by the primary actor of the communicative event III.

Each communicative event derives a different class diagram view. The derived class diagram views are integrated to obtain the complete class diagram, the same way that different relational views are integrated to obtain a single database schema [Batini, Lenzerini et al. 1986].

[84] In some works, a relational (or database) view is considered to be a virtual relation derived from base relations using a set of view defining operations such as projection, join, and others [Codd 1974]. However, other works consider views as relational schemas (or even conceptual models or data-oriented requirements specifications) that correspond to part of a complex domain [Batini, Lenzerini et al. 1986]. Our definition of class diagram view, at the point of conceptual model derivation, is closer to the latter notion.

[85] Some details of the communicative event diagram have been omitted for the sake of simplicity.

This integration can be approached in two ways; we opt for the second:

- *Post-process view integration*. First, the communicative events are processed so as to obtain their corresponding class diagram views. The events can be processed in any order. Then, all the class diagram views are integrated into a single class diagram. It is similar to the traditional relational database view-integration strategy; structural conflicts can happen. Figure 93 exemplifies this approach.

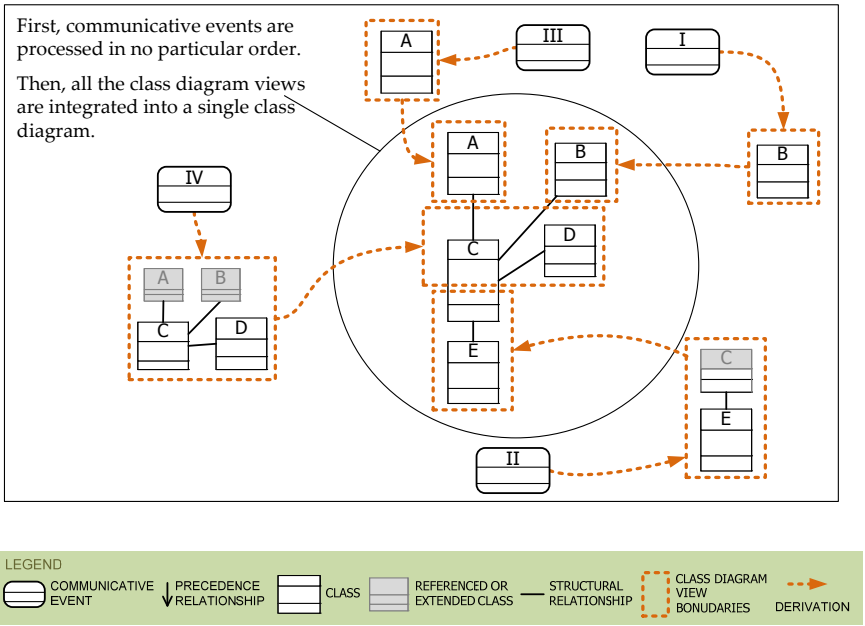


Figure 93. Construction of the class diagram by means of post-process view integration

- *Incremental view integration*. First, the communicative events are sorted. Then, communicative events are processed in order, one by one. The class diagram view that results from processing each communicative event is integrated into the class diagram under construction. This way, new classes can be added to the class diagram, class attributes and services can be added to existing classes, and structural relationships[86] among existing classes can be added to the class diagram. Figure 94 exemplifies this approach.

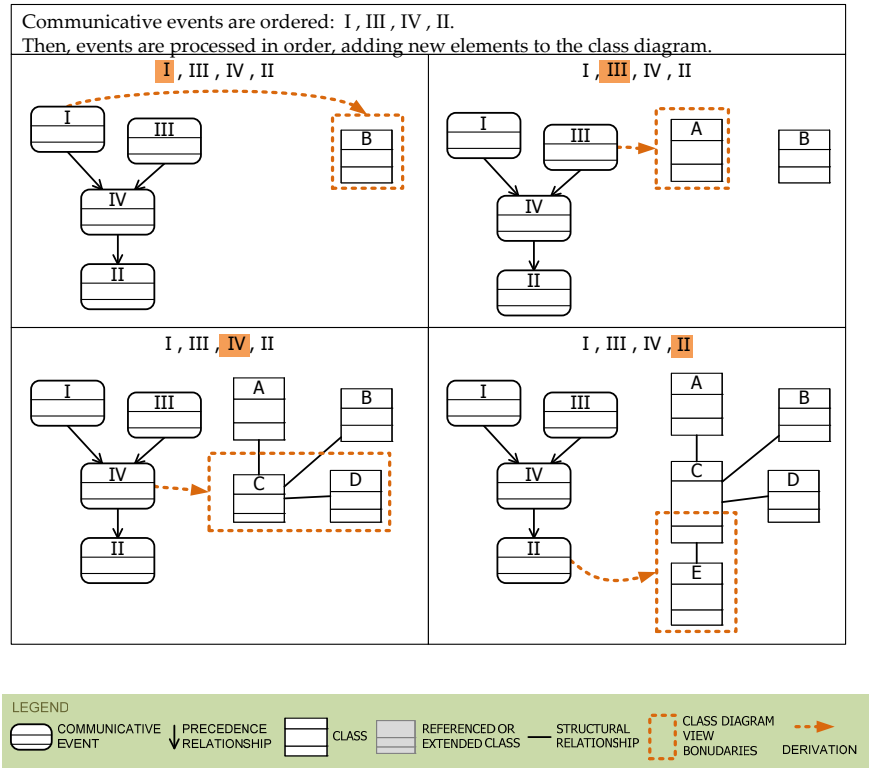


Figure 94. Construction of the class diagram by means of incremental view integration

[86] We refer as structural relationships to semantic relationships between classes as defined by Pastor and Molina [2007, pp. 80-95]. Although the OO-Method defines several types of structural relationships (namely, association, aggregation and composition), only associations are derived for the moment.

By performing an incremental view integration and by processing the communicative events in a predefined order, we intend to prevent a situation in which a newly-derived class diagram view refers to a class diagram element that has not yet been derived (e.g. trying to add an attribute to an inexistent class). In any case, if the requirements model suffers incompleteness, this situation cannot be fully prevented.

Figure 95 informally represents the correspondences between the communicative events of a business process and the class diagram views. Note that the relation among communicative events and classes has a *many-to-many* cardinality:

- One communicative event can derive one class (e.g. communicative events I and II). It is the case of communicative events with a simple message structure, such as the event SUPP 2 in the SuperStationery case [España, González et al. 2011].
- One communicative event can affect[87] several classes (e.g. communicative event III). Communicative events with more complex message structures result in class diagram views that contain several classes and relationships among them, as in SALE 1 (see Figure 104).
- One class can be affected by several communicative events (e.g. class C is affected by events IV and II). This is normally due to the fact that the occurrence of one communicative event creates a business object that is later updated by the occurrence of another communicative event, as it happens with events SALE 1 and SALE 2 (see Figure 104 and Figure 105 respectively).

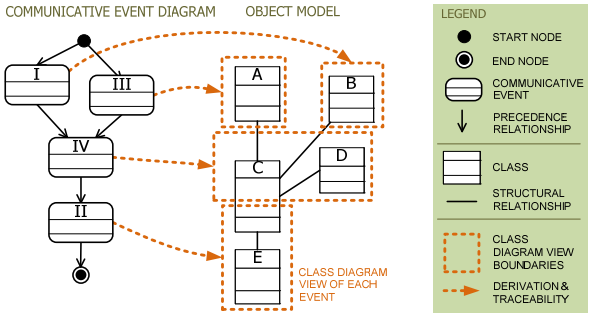


Figure 95. Simplified example of communicative events and their corresponding class diagram views

[87] When processing a communicative event for derivation purposes, a class can be created for the first time or it can extended with new attributes or services. In either case, the class is said to be part of the class diagram view of that particular communicative event. Also, the class is said to be *affected* by that communicative event.

5.4.2 Outline of the derivation of the Dynamic Model

5.4.2.1 Shifting from process-oriented to object-oriented behavioural specifications

Most process-oriented modelling techniques typically allow defining the behaviour of an organisation from a global point of view (i.e. many business objects may be involved in one process); this facilitates information system analysis and it enhances communication with users. On the other hand, object-orientation typically brings about the fragmentation of system behaviour, adopting a point of view that is local to each class; this facilitates the later implementation of the proper mechanisms to guarantee the fulfilment of the behaviour in run-time.

We have investigated several mechanisms provided by object-orientation for specifying business process-related system behaviour locally. There exist alternative mechanisms, depending on the model in which the dynamic constraints are specified and on how these are linked to other models of the system under construction. Figure 96 illustrates three mechanisms available in the OO-Method, using a simplistic example in which a business process is composed by three communicative events that must occur sequentially (see Figure 96.a).

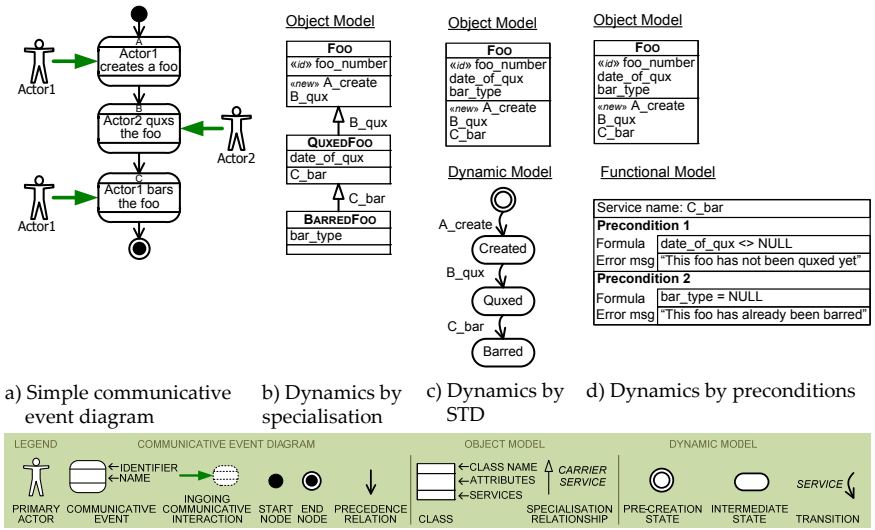


Figure 96. Different alternatives for specifying class dynamics

Dynamics by specialisation

The constraints can be specified in the Object Model using specialisation by successive classes; that is, a sequence of specialised classes that correspond to the states in which the business object can be found. *Carrier services* and *liberator services* perform and undo specialisations, respectively. As shown in Figure 96.b, each specialised class can add new attributes and services (also structural relationships). When an object is an instance of class FOO (i.e. after executing the creation service A_create), it can only trigger the service B_qux. When this carrier service is executed, then the object becomes an instance of class QUXEDFOO and the service C_bar becomes available.

On the one hand, this visual representation enhances model comprehensibility: each class corresponds to a possible state of the business object; the state is reached when a carrier service is triggered. On the other hand, the expressiveness is limited to sequences of states and decisions such as acceptance/rejection. State cycles lead to loosing information (once a specialisation is undone, the object looses the attributes and relationships that correspond to the specialised class). And-forks, and-joins, or-branches and or-merges are neither supported.

Dynamics by state-transition diagrams

System behaviour can also be specified in the Dynamic Model using state-transition diagrams. As shown in Fig. 3.c, class FOO is now designed disregarding its behaviour. Its corresponding state-transition diagram defines the possible states in the life of an instance, and the valid transitions among them. For instance, after an object is created (by executing A_create), the object is in the state Created. Note that, in this state, there is no transition defined for C_bar so this service is not available yet. After executing the service B_qux, the object moves to the state Quxed.

Firstly, this mechanism is concise and highly visual; a single model specifies the whole dynamics of the class. Secondly, the state-transition diagram is expressive enough to account for transition cycles and other complex behavioural patterns[88].

Dynamics by preconditions

The Functional Model offers the possibility of defining preconditions to each class service. They can be defined to ensure that the service is only executed when the instance is in the desired state (e.g. by checking attribute values). For instance,

[88] There is, however, a limitation that is due to the fact that the OO-Method state-transition diagrams do not include pseudostates (e.g. fork, join). But this limitation can be overcome by creating states that make the pseudostates explicit, or by using transition guards.

Figure 96.d shows two preconditions of C_bar that ensure that this service can only be executed after date_of_qux has been given a value and before bar_type has been given a value.

On the one hand, this mechanism is as expressive as the state-transition diagram because the same constraints that are specified in an state-transition diagram can be defined by means of preconditions. It is also possible to add an attribute to a class so as to store the state of an object (e.g. a string attribute named state that can be given the values "created", "quxed" and "barred"), and then the preconditions would check the value of this attribute. On the other hand, the preconditions are scattered across the services so no integrated view of the business object behaviour is provided, hindering the comprehensibility of the conceptual model.

Given the previous analysis, we opt for the second mechanism. The Dynamic Model provides an integrated view of the dynamic behaviour of a class and the state-transition diagram offers enough expressivity for our purposes. Other works have taken a similar path (e.g. [de la Vara and Sánchez 2010])

5.4.2.2 State-transition diagrams as the chosen alternative

The Dynamic Model specifies the valid lifecycles of objects and the interaction between them by means of a collection of state-transition diagrams (that are compliant with the UML State Machine Diagram). Each class of an Object Model has its own state-transition diagram in the Dynamic Model. The Dynamic Model can be obtained mainly by processing the Communicative Event Diagram. However, some elements can be added to the Dynamic Model as a result of processing the Event Specification Templates.

Two types of state-transition diagrams can be derived for a class, depending how many communicative events affect the class:

- A class is affected by only one communicative event. For each class in the conceptual model that is affected by only one communicative event, a basic state-transition diagram is derived by default. The basic state-transition diagram represents the essential valid lifetimes for the instances of a class. The diagram includes three states; namely a pre-creation state, an intermediate state and a destruction state. The diagram also includes one transition for each creation service, departing from the pre-creation state and arrives to the intermediate state; one transition for each destruction service, departing from the intermediate state and arriving to the destruction state; one transition for each one of the rest of services of the class, departing from and arriving to the intermediate state [Pastor and Molina 2007, pp. 120-121]. To derive the basic state-transition diagram, only the Object Model is needed.

- A class is affected by more than one communicative event. For each class in the conceptual model that is affected by more than one communicative event (see class C in Figure 95), a more complex state-transition diagram is derived (see Figure 97). The main derivation guideline is the following. Communicative events are converted into transitions (e.g. IV, II) and precedence relationships are converted into states (e.g. IVed, IIed[89]). Additional transitions can be added to the state transition diagram; for instance, transitions that correspond to atomic edition and destruction services (e.g. edit and destroy, respectively), as well as transitions that correspond to atomic services that take part in a transaction (these transitions appear as a result of processing the event specification templates).

A more illustrative example is shown in Section 5.5.3.

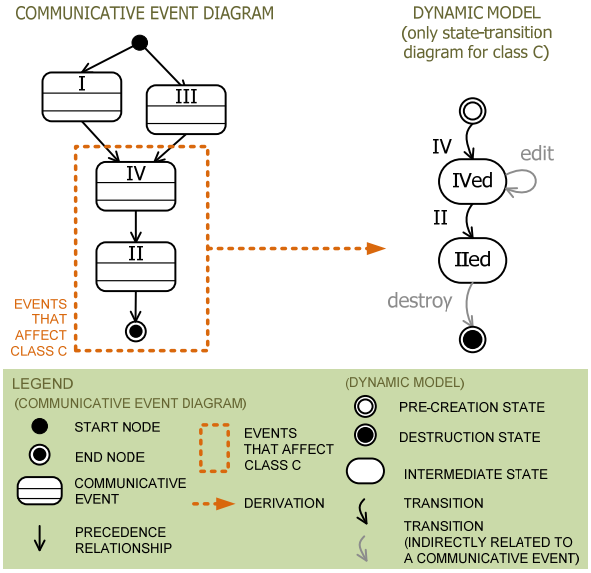


Figure 97. Simplified example of the derivation of the structural and behavioural views of the conceptual model

[89] To facilitate the example understandability, precedence relations and transitions have been labelled with a name that is derived from the name of the precedent communicative event. The English participle suffix *–ed* is added to the event name. This intuitively represents the state in which an object remains after an event. For instance, after an occurrence of the event IV the affected object remains in the state IVed. It makes more sense with real words: after an occurrence of the communicative event "A customer submits a claim" an object of the class CLAIM remains in the state Submitted; after an occurrence of the communicative event "A clerk classifies the claim" the object remains in the state Classified.

5.5 *Derivation guidelines*

In the following, we specify the derivation guidelines (a.k.a. transformation rules) by means of a template with the following structure. Each rule has a unique sequential identifier. The rule is given an explanatory name. In some cases, a pattern matching expression is provided, so as to explain to which elements of the requirements model the rule is to be applied. Then we enumerate the set of preconditions that need to hold in order to apply the rule (so as to ensure a correct application order). The rule is described textually as a set of numbered steps. Lastly we describe how to record traceability information (this aspect has been separated from the rule steps for the sake of comprehensibility).

The rules are exemplified by describing their application to the SuperStationery Co. case; the complete derivation of this case is described in [España, González et al. 2011].

We provide the following rules for the derivation of the Object Model (rules prefixed with **OM**), the Functional Model (rules prefixed with **FM**) and the Dynamic Model (rules prefixed with **DM**).

Index of rules for the derivation of the Object Model

Index of rules for the derivation of the Functional Model

Index of rules for the derivation of the Dynamic Model

The rules provide model-based mappings according to the MDA terminology (see Section 5.2). However, we combine the metamodel mapping with model instance mappings supported by marks. Figure 98 offers a view of the marks that will be used to facilitate the transformations. These marks do not belong to the computation independent viewpoint but to the platform independent viewpoint, so they are provided in a different metamodel.

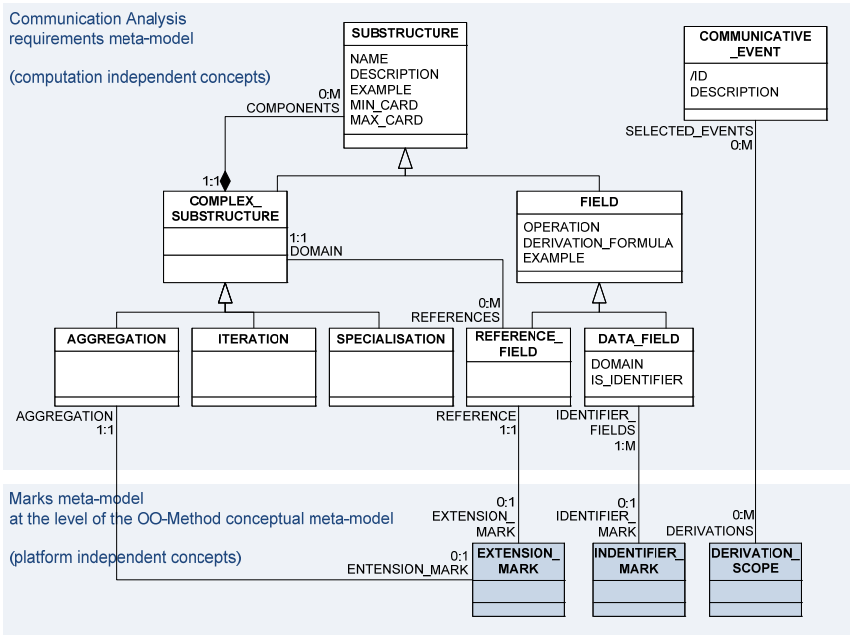


Figure 98. Support for model instance mappings by means of marks

5.5.1 Derivation of the Object Model

In this section, we provide the derivation guidelines intended to create a first version of the OO-Method Object Model (i.e. the class diagram) by means of processing the requirements model.

However, before applying the derivation guidelines that add new elements to the Object Model, the scope of the derivation needs to be determined.

In many cases, having a Communication Analysis requirements model, the analyst will want to derive the OO-Method conceptual model that supports the whole work practice described in the requirements model. However, we provide for the case in which the analyst decides to define a narrower scope for the derivation. In such case, the analyst can select a subset of all the communicative events in the requirements model, in a way that the derived conceptual model will specify an information system that supports the selected subset. In later derivations, the analyst can select the remaining communicative events and derive their corresponding view of the conceptual model. This way, an iterative and incremental approach can be followed. It can also be used as a form of requirements triage [Davis 2003a].

In any case, the analyst cannot choose and discard whatever communicative events s/he wishes, because the precedence relations among events impose dependency constraints on their processing. Thus, after the analyst selects a subset of communicative events, the subset is extended with other communicative events, according to rule **OM1**. A rooted directed graph is obtained[90].

Let us suppose that, in the SuperStationery Co. case, the analyst decides to derive the conceptual model that corresponds to part of the *Sales management* (SALE) business process; namely, communicative events *SALE 1* to *SALE 6*. Then step 5 of rule **OM1** is repeated until no remaining precedent events exist. The result is shown in Figure 99, which extends the original diagram in Figure 50 with the communicative events that are needed to obtain a rooted directed graph. The communicative event *PROD 2* belongs to the *Product management* business process, *CLIE 1* belongs to *Client management*, *SUPP 2* belongs to *Supplier management*, *LOGI 10* belongs to *Logistics*, and *RISK 4* belongs to *Risk management*.

Despite the guidelines for including start and end nodes in communicative event diagrams (see page 157 and also rule **OM1**, step 8), the start note and end nodes can be omitted for the sake of simplicity (i.e. to avoid many line crossings). They can be left implicit.

---

[90] A *rooted directed graph* is a directed graph in which there is a node designated as the root, from which there is at least one path to every other node.

Rule OM1. Determination of the scope of the derivation	
Pattern matching	
1	The complete requirements model.
Preconditions	
	None.
Rule steps	
1	Create a new communicative event diagram. We will refer to it as the transformation-scope (communicative event) diagram.
2	Decide whether to derive the conceptual model for the complete requirements model or just for a portion of it. In case the derivation for the complete requirements model ins intended then...
3	Add all the communicative events of the requirements model to the transformation-scope diagram.
	Else...
4	Select the subset of communicative events for which information system support is intended.
5	Include any communicative event that, not being included in the selected subset of communicative events, it precedes a communicative event which is included in the subset. Include the event even if it belongs to a different business process.
6	Repeat step 3 until no more events are included.
7	Add all the communicative events of the subset to the transformation-scope diagram.
8	Add a start node to the transformation-scope diagram.
9	Add precedence relations from the start node to the globally initiatory events (i.e. the events that do not have a precedent event)
10	Designate the start node as the root of the graph, in order to obtain a rooted directed graph.

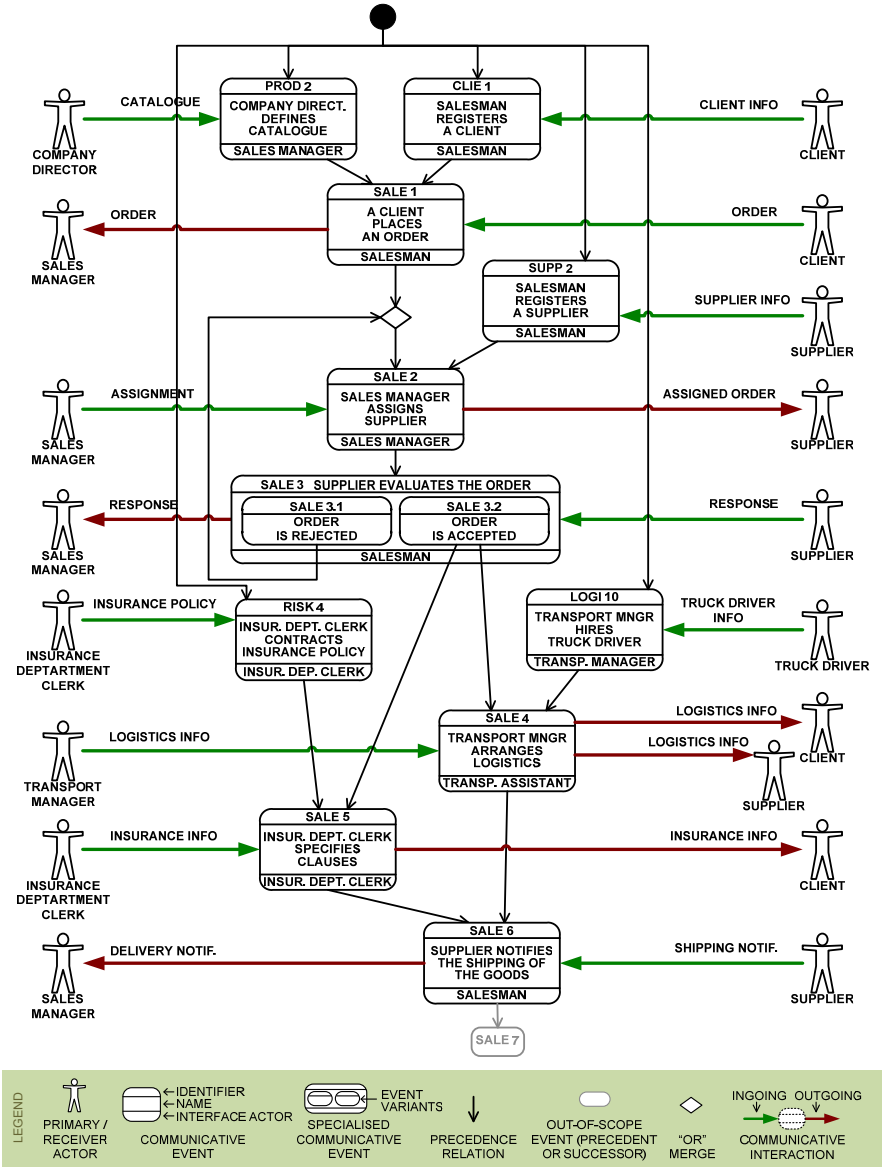


Figure 99. Transformation-scope diagram that includes communicative events from other business processes

Before applying the derivation guidelines that add new elements to the Object Model, the analyst needs to pre-process the requirements model (at least the part of it within the scope of the derivation), by marking some reference fields with an *extension mark* (see rule **OM2**). These marks will facilitate the derivation process later on.

An extension mark in a reference field contained in the message structure of a communicative event is a means the following: by the time the communicative event occurs, the information system already has some information about the business object referred by the reference field and, during the communicative event occurrence, the primary actor is providing some extra information about the business object.

These marks can be supported by means of a new Boolean property for reference fields; we suggest adding a column named *Extends business object* to the message structure description. Its default value is False, and the extension mark consists of setting the value to True. Only one reference field per aggregation substructure can be marked with an extension mark.

We refer as *extension aggregation substructure* to an aggregation substructure that contains a reference field that has been marked with the extension mark. That is, an aggregation substructure that contains a reference field whose property *Extends business object* has been set to True.

Rule OM2. Marking of reference fields	
Pattern matching	
1	The part of the requirements model that corresponds to the transformation-scope diagram.
Preconditions	
1	Rule **OM1** has already been applied to the requirements model.
Rule steps	
1	Add a new Boolean property (i.e. a new column) named *Extends business object* to every message structure of the requirements model. The new property only applies to reference fields.
2	Mark each reference field that indicates that an aggregation substructure is describing the provision of new information about an already-known business object. The extension mark consists of setting the new property *Extends business object* to True.
Notes	
1	Only one reference field per aggregation substructure can be marked. It is not necessary to mark as False the remaining reference fields (False is considered the default value).

When this rule is applied to the communicative event *SALE 1. A client places an order*, no changes are made, because the event does not extend a business object but actually creates a new one (namely, a client order). Therefore, we focus on communicative event *SALE 2. Sales Manager assigns supplier*.

Given the meaning of the communicative event (its pragmatic consequences), the reference field Order actually specifies that the Sales Manager selects an existing client order form so as to make a decision. He selects a supplier that should be able to serve the order and he adds this information to the form.

Table 26. Message structure of the communicative event *SALE 2* marked according to rule **OM2**

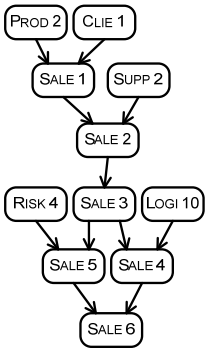
FIELD	OP	DOMAIN	EXAMPLE VALUE	EXTENDS BUSINESS OBJECT
ASSIGNMENT = < Order + Supplier + Assignment date >	 i i i	 Client order Supplier date	 2008-00352 OFFIRAP, Office Rapid Ltd. 01-09-2009	True

Therefore, the reference field Order is referencing a business object being extended with new information; namely, a client order. According to this derivation rule, the analyst should set the property *Extends business object* of the field Order to True, as shown in Table 26. We can then say that ASSIGNMENT is an extension aggregation substructure.

Similarly, the analyst marks the reference fields Order in *SALE 3*, Order and Destination in *SALE 4*, Order in *SALE 5*, and Order in *SALE 6*.

As mentioned above, the communicative events are processed in a predefined order, to avoid referencing a not-yet-derived element during the application of the derivation rules. Thus, the communicative events depicted in the transformation-scope diagram need to be ordered according to the precedence relations (see rule **OM3**). For this purpose, a rooted directed graph is obtained by removing the loopbacks that might appear in the transformation-scope diagram (i.e. the feedback arc set is removed). The remaining precedence relations define a strict partial order among the events. Any known procedure for topological sorting can be used for obtaining the ordered list of events; for instance, the algorithm proposed by Kahn [1962]. Often many solutions (a.k.a. linear extensions) are possible; any solution is suitable for derivation purposes and the choice is not expected to influence model transformation efficiency; moreover, the same conceptual model will be produced as output.

Rule OM3. Event ordering	
Pattern matching	
1	The transformation-scope diagram.
Preconditions	
1	Rule **OM1** has already been applied to the requirements model.
Rule steps	
1	Make a copy of the transformation-scope communicative event diagram. In the rest of the rule we will refer to this copy simply as the diagram.
2	Remove the loopbacks that might appear in the diagram and remove every other model element except for precedence relationships and communicative events.
3	Create an empty list of communicative events.
4	Remove the start node and the precedence relations that depart from it.
5	While there are still communicative events left in the diagram do...
6	Select an event that has no incoming precedence relations.
7	Add its identifier to the list.
8	Remove the event from the diagram, as well as all the precedence relations that depart from it.



PROD 2, CLIE 1, SALE 1, SUPP 2, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
CLIE 1, PROD 2, SALE 1, SUPP 2, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
PROD 2, CLIE 1, SUPP 2, SALE 1, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
CLIE 1, PROD 2, SUPP 2, SALE 1, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
PROD 2, SUPP 2, CLIE 1, SALE 1, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
CLIE 1, SUPP 2, PROD 2, SALE 1, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
SUPP 2, PROD 2, CLIE 1, SALE 1, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
SUPP 2, CLIE 1, PROD 2, SALE 1, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
PROD 2, CLIE 1, SALE 1, SUPP 2, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
CLIE 1, PROD 2, SALE 1, SUPP 2, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
PROD 2, CLIE 1, SUPP 2, SALE 1, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
CLIE 1, PROD 2, SUPP 2, SALE 1, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
PROD 2, SUPP 2, CLIE 1, SALE 1, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
CLIE 1, SUPP 2, PROD 2, SALE 1, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
SUPP 2, PROD 2, CLIE 1, SALE 1, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
SUPP 2, CLIE 1, PROD 2, SALE 1, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
PROD 2, CLIE 1, SALE 1, SUPP 2, SALE 2, SALE 3, LOGI 10, RISK 4, SALE 5, SALE 4, SALE 6

a) Poset of communicative events **b)** Some suitable ordered lists of events

Figure 100. Obtaining ordered lists of events from the transformation-scope communicative event diagram

After applying rule **OM3**, step 2, to the transformation-scope diagram in Figure 99, we obtain the partially ordered set[91] of communicative events shown in Figure 100.a; note that the loopback from event *SALE 3* to *SALE 2* has been removed. It is also convenient to simplify the diagram removing any remaining logical nodes (simply draw the precedence relation directly from source events to target events, disregarding any additional logic).

Any ordered list of events that is compatible with the partial order is suitable for the conceptual model derivation (see Figure 100.b). In [España, González et al. 2011], the following ordered list of events has been chosen for pedagogical purposes:

SUPP 2, PROD 2, CLIE 1, SALE 1, SALE 2, SALE 3, LOGI 10, SALE 4, RISK 4, SALE 5, SALE 6

Note that a list starting with *SUPP 2, PROD 2, SALE 1, CLIE 1,...* would not be suitable because *SALE 1* cannot appear before *CLIE 1* (there is a precedence relation between them in the opposite direction).

The remaining derivation rules process the event specification templates of the communicative events. The events in the ordered list are processed one by one. All rules whose precondition holds need to be applied to each communicative event before proceeding with the following event.

First of all, the message structure of a communicative event is processed. Each aggregation substructure that does not contain a reference field that has been marked according to **OM2**, leads to the derivation of a new class (see rule **OM4**). The name of the aggregation substructure can be used to name the class; however, the analyst can decide to give the class a different name.

[91] A partially ordered set (a.k.a. poset) consists of a set together with a binary relation that indicates that, for certain pairs of elements in the set, one of the elements precedes the other. In our case, the binary relation is defined by the precedence relationships. If a communicative event A precedes another communicative event B then we can consider that A<B. The start node (even if it is implicit) is the *least element* of the poset.

Rule OM4. Derivation of a new class from an aggregation substructure	
Pattern matching	
1	Each aggregation substructure within the message structure of the event being processed that *is not* an extension aggregation substructure.
Preconditions	
1	Rule **OM2** has already been applied to the requirements model.
Rule steps	
1	Create a new class.
2	Assign to the name of the class the name of its corresponding aggregation substructure, replacing spaces by underscores and using uppercase letters. In any case, the analyst can choose to give a different name to a class.
Traceability information	
1	Create a trace link of type Class_Derivation between the communicative event and the newly-created class (this can be done after step 2).
2	Create a trace link of type Class_Derivation between the aggregation substructure and the newly-created class (this can be done after step 2).
3	If the aggregation substructure describes an organisational role, then create a trace link of type Agent between the organisational role and the newly-created class (this can be done after step 2).
Notes	
1	In case the analyst is creating the conceptual model using Integranova Modeler, then we advice not using *extended creation* facility optionally offered by the tool (which automatically creates an autonumeric identifier attribute, a creation service, a destruction service, and an editing service). In case this facility is used, the Analyst should be aware of the model elements automatically added by the tool and remove those that are not needed.

For instance, the message structure of event *SALE 1* consists of an initial aggregation structure named ORDER that includes an iteration substructure named DESTINATIONS, which includes another iteration substructure named LINES. Both iteration substructures have an aggregation substructure inside of them (DESTINATION and LINE respectively). The three aggregation substructures lead to the derivation of the classes CLIENTORDER, DESTINATION and LINE, respectively.

Then, according to rule **OM5**, for each data field of the substructures, an attribute is added to the corresponding class (e.g. the data field Order number leads to the derivation of the class attribute order_number)[92]. Figure 101 shows the derived classes and attributes.

Rule OM5. Derivation of an attribute from a data field	
Pattern matching	
1	Each data field of each aggregation substructure.
Preconditions	
	One of the following preconditions applies, depending on whether a new class is being created (precondition 1) or an existing class is being extended (precondition 2):
1	In case the substructure *is not* an extension aggregation substructure, then rule **OM4** has already been applied to the substructure.
2	In case the substructure *is* an extension aggregation substructure, then rule **OM15** has already been applied to the substructure.
Rule steps	
1	Select the class to which the attribute is going to be added; simply follow the trace link of type Class_Derivation or the trace link of type Class_Extension from the aggregation substructure containing the data field.
2	Add an attribute to the class.
3	Assign a name to the attribute according to the name of its corresponding data field (i.e. the data field from which the attribute is derived), replacing spaces by underscores and using lowercase letters.
4	In any case, the analyst can choose to give a different name to an attribute.
Traceability information	
1	Create a trace link of type Attribute_Derivation between the communicative event and the newly-created attribute (this can be done after step 4).
2	Create a trace link of type Attribute_Derivation between the data field and the newly-created attribute (this can be done after step 4).

The data field Order number leads to adding the attribute order_number to the class CLIENT_ORDER; the same is done for the data fields Request date and Payment type.

[92] Reference fields (i.e. fields of the message structure that reference business objects, such as Client) do not lead to the derivation of attributes but to the derivation of structural relationships, but this is explained later on.

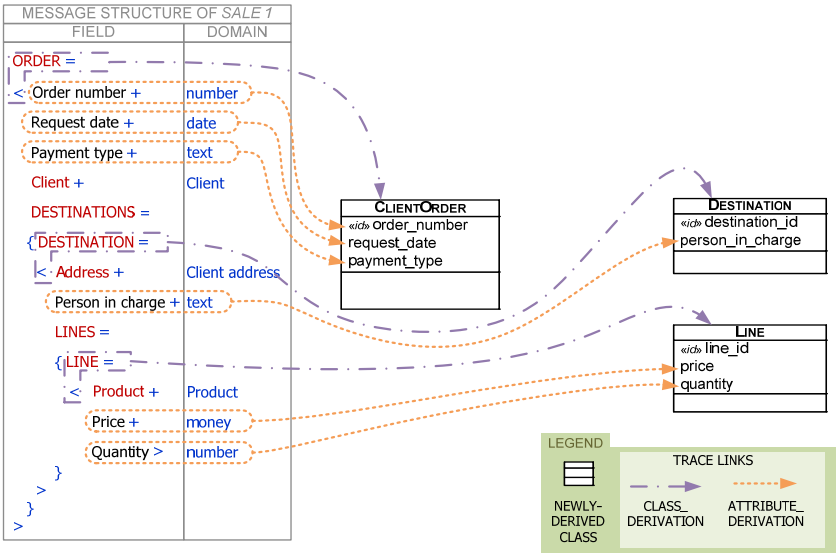


Figure 101. Derivation of classes and attributes (class diagram view of communicative event *SALE 1*)

The information contained in the event specification templates (e.g. message structures, structural restrictions and contextual restrictions) is used to derive the class identifier and to derive other properties such as the attribute data types, their sizes, whether they allow null values, etc. Table 27 presents the specification of the attributes of the class CLIENT_ORDER after processing the event *SALE 1*. In the following, we explain how this information is derived.

Table 27. Specification of the attributes of the class CLIENT_ORDER after processing the event *SALE 1*[93]

Attribute name	Id	Attribute type	Data type	Size	Requested	Null allowed
order_number	yes	Constant	Autonumeric	-	yes	no
request_date	no	Constant	Date	-	yes	no
payment_type	no	Variable	String	30	yes	yes

[93] These tables are not part of the requirements model; they are part of the conceptual model. At the beginning of the derivation process, such tables do not exist. Whenever a class is created (by means of applying rule **OM4**), an empty table is created. Later, whenever a class attribute is derived (by means of applying rule **OM5**), an empty row in the corresponding table is created. As the properties of the attributes are derived, the columns are filled.

The requirements model should specify how the organisational actors identify business objects of each kind. If available, this information is specified as a contextual restriction and it offers guidance for selecting which attributes constitute the class identifier (see **OM6**). In the absence of such restrictions, then the analyst should ask the users or decide relying on his/her own judgement.

Rule OM6. Selection of class identifier	
Pattern matching	
1	Each aggregation substructure that *is not* an extension substructure.
Preconditions	
1	Rule **OM5** has already been applied to all the data fields of the aggregation substructure.
Rule steps	
1	Select the class to which the attribute is going to be added; simply follow the trace link of type Class_Derivation or the trace link of type Class_Extension from the aggregation substructure.
2	If there exists a contextual restriction in the event specification template that indicates which data fields (or set of fields) the organisation uses (or intends to use) to identify a given business object (or a part of it) then...
3	Use the information in the contextual restriction to select which attributes constitute the class identifier.
	Else (in the absence of such restriction)...
4	The analyst should ask the users or decide relying on his/her own judgement. An option is to create an artificial autonumeric identifier and give it the name class_name_id (i.e., the class name with the suffix _id).
Traceability information	
1	In case the identifier has been derived from a contextual restriction, then create a trace link of type Identifier_Selection between the contextual restriction and the attributes selected as class identifier (this can be done after step 3).

According to the event specification template (see Figure 67), salespersons in SuperStationery use to identify orders by means of an order number. Thus, the attribute order_number is designated the identifier of the class CLIENT_ORDER (according to rule **OM6**, steps 2-3); this has been specified in Table 27, column **Id**. With regards to the aggregation substructures DESTINATION and LINE, since there are no contextual constraints specifying how these parts of the business object should be identified, then two artificial autonumeric attributes have been added to their corresponding classes, in order to act as identifiers (according to rule **OM6**, step 4). Class DESTINATION has been added an identifier named id_destination and class LINE has been added id_line.

Rule OM7. Specification of attribute type	
Pattern matching	
1	Each data field of an aggregation substructure.
Preconditions	
1	In case the aggregation substructure *is not* an extension substructure, then rule **OM6** has already been applied to the aggregation substructure.
Rule steps	
1	Select the attribute related to the data field; simply follow the trace link of type Attribute_Derivation from the data field.
2	If the attribute is part of the class identifier then...
3	Set the attribute type as constant.
	Else...
4	If the class is being extended (i.e. the aggregation substructure is an extension substructure) then...
5	Set the attribute type as variable.
	Else...
6	The analyst should ask the users the following question "Consider that this piece of information about a business object (indicate the attribute name) is set a value in a given moment; is it possible that this information needs to be changed later?"
7	If the response is "Yes, it should be possible to change this piece information later" then the attribute type is variable.
8	If the response is "No, the initial value should never be changed" then the attribute type is constant.
9	In case of doubt set the attribute type as variable.
Notes	
1	In case the analyst is creating the conceptual model using Integranova Modeler, then the tool performs step 3 automatically.

In the OO-Method, there are three types of class attributes; namely, constant, variable and derived attributes [Pastor and Molina 2007]. Once a constant attribute has been initialised, its value cannot be modified. In contrast, variable attributes can be later modified, as long as the proper class services are defined[94]. Derived attributes represent data that can be calculated in terms of the values of other attributes. Class identifiers must be defined as constant according to the

[94] Take into account that, according to Integranova, constant attributes can only be initialised by means of the creation service; therefore, an attribute not being requested upon creation and of type constant is useless. This is a restriction of the tool and it does not apply to the OO-Method. According to the method, a constant attribute can have its value initialised in an edition service as well.

OO-Method. Attributes that are part of the extension of a class must be defined as variable. With regards to the remaining attributes, their type should be asked to the users (formulating the appropriate questions and providing examples); however, we suggest defining them as variable since it is the less restrictive option (see rule **OM7**).

For instance, the attribute order_number is part of the class identifier. This implies that the attribute is of type Constant (according to rule **OM7**, steps 2-3). The remaining attributes have been defined of type variable by default. In any case, the analyst should ask the users whether the information related to these attributes needs to be changed later or not. The types of the attributes of class CLIENTORDER have been specified in Table 27, column **Attribute type**).

The data types of the attributes determine what kind of information the attributes can store. Attribute data types are derived from the domains of their corresponding message structure data fields (see rule **OM8**). Communication Analysis prescribes a few basic domains for data fields which merely serve the purpose of clarifying the meaning of the messages, whereas the OO-Method offers a wider selection of data types for attributes because code generation is intended.

Table 28. Conversion table for attribute data types

Communication Analysis Data field domain	**OO-Method Attribute data type**
number	Nat, Int, Real, Autonumeric
text	String, Text
date, time	Time, Date, DateTime
money	Real
not considered	Bool, Image, Blob

Table 28 offers some guidance for the conversion by defining a correspondence between Communication Analysis data field domains and OO-Method attribute data types.

Rule OM8. Specification of attribute data type	
Pattern matching	
1	Each data field of an aggregation substructure.
Preconditions	
1	Rule **OM5** has already been applied to the data field.
Rule steps	
1	Select the attribute that has been derived from the data field (i.e. follow the trace link of type Attribute_Derivation).
2	Assign the attribute *Data type* according to the conversion table for attribute data types (see Table 28).
3	Some data field domains can be converted to several attribute data types; in these cases, the analysts should apply their criteria. However, the data types that are underlined are the ones that are recommended in case of doubt.
4	In case a String data type is chosen, then the *Size* (the maximum amount of characters that the string attribute will allow to store) needs to be defined. For this purpose, the analyst asks the users or decides relying on his/her own judgement.

In order to derive the data type of the attribute order_number of class CLIENT_ORDER, we first follow the trace link of type Attribute_Derivation from the attribute back to the message structure data field from where the attribute has been derived: it is Order number (see Figure 101). The domain of Order number is number (see Figure 67). This domain is looked up in the first column of Table 28 (it is located in the first row) and one of the corresponding OO-method attribute data types can be selected (either Nat, Int, Real or Autonumeric). Autonumeric is chosen because, in this case, an automatic sequence of natural numbers is convenient for identifying order forms. Finally, this property of the attribute order_number is specified in Table 27, column **Data type**.

Similarly, the data field that corresponds to the attribute request_date is Request date. The domain of this data field is date, so a DateTime data type is chosen. With regards to the attribute payment_type, it has been derived from the data field Payment type, which has a text domain; therefore a data type String is chosen. Since this data type needs a size, the analyst asked the users how long was the longest text they intended to write in the payment type field of order forms: an appropriate size is 30 characters. The size is specified in Table 27, column **Size**.

An attribute can be requested upon creation or not. In case an attribute is requested upon creation, it will have a corresponding argument in the creation service of the class; otherwise, that piece of information will be provided by means a service different from the creation service. According to rule **OM9**, in a newly-derived class, all the attributes are set as requested at creation time; in

turn, when the class is being extended with new attributes, these attributes should not be requested at creation time (because the information they contain is not provided when the instance is created, but in a later moment).

Rule OM9. Decision on whether an attribute is requested at creation time	
Pattern matching	
1	Each data field of an aggregation substructure.
Preconditions	
1	Rule **OM5** has already been applied to the data field.
Rule steps	
1	Select the attribute that has been derived from the data field (i.e. follow the trace link of type Attribute_Derivation).
2	If it is a newly-derived class that has been created during the processing of the current communicative event (i.e. the aggregation substructure to which the data field belongs *is not* an extension substructure) then...
3	Set the value of the property *Requested* upon creation to "Yes".
	Else (the class is being extended during the processing of the current communicative event)...
4	Set the value of the property *Requested* upon creation to "No".

There are actually two ways of checking whether the class has been derived from the aggregation substructure. One is mentioned in rule **OM9**, step 2, and it consists in checking whether the aggregation substructure to which the data field belongs has a reference field that has an extension mark. The other way is to check whether the trace link from the aggregation substructure towards a class is of type Class_Derivation or of type Class_Extension.

Since the class ORDER_FORM is created during the processing of the event *SALE 1*, therefore all of the attributes derived from the message structure of *SALE 1* are set as *Requested upon creation*. This is specified in Table 27, column **Requested**.

By means of the property Null allowed, class attributes can allow null values or not. Necessarily, all attributes that are part of the class identifier should not allow null values. Whether the remaining attributes should allow null values or not needs to be asked to the users or the analyst should decide relying on his/her own judgement (see rule **OM10**).

Rule OM10. Decision on whether an attribute allows null values	
Pattern matching	
1	Each data field of an aggregation substructure.
Preconditions	
1	In case the aggregation substructure *is not* an extension substructure, then rule **OM6** needs to already have been applied to it.
Rule steps	
1	Select the attribute that has been derived from the data field (i.e. follow the trace link of type Attribute_Derivation).
2	If the attribute is part of the class identifier then...
3	Set the attribute to not allow null values.
	Else...
4	The analyst should asks the users or decide relying on his/her own judgement. In case of doubt, set the attribute to allow null values because this is less restrictive.

Since the class identifier is order_number, then this attribute is set to not allow null values. With regards to request_date, although it is not part of the class identifier, the analyst decides to make it compulsory based on the comments by the users. The attribute payment_type is set to allow null values, because it is not mandatory that the client decides *a priori* how the order will be paid. These properties are specified in Table 27, column **Null allowed**.

Messages are often complex in structure and this has an effect on the class diagram. Complex message structures nest two or several substructures. Since aggregation substructures lead to the derivation of classes (see **OM6**), the nesting of aggregation substructures leads to the derivation of structural relationships between those classes (rule **OM11**). It is possible to derive some of the cardinalities of the structural relationships from the message structure itself, whereas other cardinalities depend on structural constraints that (may) appear in the event description templates [95].

[95] We acknowledge that cardinality is often referred to as multiplicity since UML became a de-facto standard; however, OO-Method still uses this term [Pastor and Molina 2007]. Also, the notation in Integranova departs from the UML. Some differences:

Integranova CLASSA minA:maxA --- minB:maxB CLASSB, the cardinality many is expressed M

UML CLASSA minA..maxA --- minB..maxB CLASSB, the cardinality many is expressed *

Rule OM11. Derivation of a structural relationship from nested substructures	
Pattern matching	
1	Each pair of nested aggregation substructures that are either adjacently nested (e.g. A and C in A=<a+b+C=<d+e>+f>) or have an iteration substructure in between (e.g. A and C in A=<a+b+G={C=<d+e>}+f>). In both cases, we can refer to A as the *outer substructure* and to C as the *inner substructure*.
Preconditions	
	For each of the substructures involved (the outer and the inner), one of the following preconditions needs to be fulfilled:
1	In case the aggregation substructure *is not* an extension substructure, then rule **OM4** has already been applied to the substructure.
2	In case the aggregation substructure *is* an extension substructure, then rule **OM15** has already been applied to the substructure.
Rule steps	
1	Select the *referencing class* by following the trace link of type Class_Derivation or the trace link of type Class_Extension from the outer aggregation substructure.
2	Select the *referenced class* by following the trace link of type Class_Derivation or the trace link of type Class_Extension from the inner aggregation substructure.
3	Create a structural relationship between the referencing class and the referenced class.
4	If the two aggregation substructures are adjacently nested then...
5	Set the maximum cardinality on the side of the referenced class to 1.
	Else (i.e. there is an iteration substructure in between)...
6	Set the maximum cardinality on the side of the referenced class to M (many).
7	Assign the rest of the cardinalities according to structural constraints included in the event description template. In the absence of such constraint, the analyst should ask the users or decide relying on his/her own judgement.
8	If the maximum cardinality in the role side of the referenced class is M then...
9	Set the relationship as dynamic
	Else...
10	The analyst should decide whether the relationship is dynamic or static. In case of doubt, it should be defined dynamic, since it is less restrictive.
11	Set the name of the structural relationship to the concatenation of the names of the related classes. In any case, the analyst can choose a different name.

12	If the maximum cardinality at the side of the referenced class is 1, then...
13	Set the name of the role at the side of the referenced class to the name of the referenced class itself.
	Else...
14	Set the name of the role at the side of the referenced class to the name of the referenced class itself, but in plural form.
15	If the maximum cardinality at the side of the referencing class is 1, then...
16	Set the name of the role at the side of the referencing class to the name of the referencing class itself.
	Else...
17	Set the name of the role at the side of the referencing class to the name of the referencing class itself, but in plural form.
18	If the relationship is dynamic then...
19	If the cardinality at any of the role ends is 1:1 (i.e. both the minimum and maximum cardinalities are 1 at any of the two sides of the relationship) then...
20	Create a shared service in both the referenced class and the referencing class.
21	Name the service change_rolename (i.e. the name of the role where the cardinality is 1:1 with the prefix change_).
22	Set the type of shared service to "Change".
	Else...
23	Create a shared service in both the referenced class and the referencing class.
24	Name the service ins_rolename (i.e. the name of the role on the side of the referenced class with the prefix ins_).
25	Set the type of shared service to "Insert".
26	Create a shared service in both the referenced class and the referencing class.
27	Name the service del_rolename (i.e. the name of the role on the side of the referenced class with the prefix del_).
28	Set the type of shared service to "Delete".
29	For each shared service created above do...
30	Add an object-valued inbound argument.
31	Name the argument p_thisClass (i.e., the name of the class to which the service belongs with the prefix p_this).
32	The data type of the argument is the class to which the service belongs. It does not accept null values.
33	Add an object-valued inbound argument.
34	Name the argument p_evcOppositeClass (i.e., the name of the class at the other side of the structural relationship with the prefix p_evc).
35	The data type of the argument is the class at the other side of the

	structural relationship. It does not accept null values.
Traceability information	
1	Create a trace link of type Structural_Relationship_Derivation between the communicative event and the newly-created structural relationship (this can be done after step 3).
2	Create a trace link of type Structural_Relationship_Derivation between the inner aggregation substructure and the newly-created structural relationship (this can be done after step 3).
3	In case any cardinality is derived from a structural constraint, then create a trace link of type Cardinality_Derivation between the structural constraint and the newly-created structural relationship (this can be done after step 7).
4	Create a trace link of type Shared_Service_Derivation between the communicative event and each newly-created shared service (this can be done after steps 20, 23 and 26).
5	Create a trace link of type Shared_Service_Derivation between the inner aggregation substructure and each newly-created shared service (this can be done after steps 20, 23 and 26).
Notes	
1	In case the analyst is creating the conceptual model using Integranova Modeler, then steps 18 to 35 are automatically performed by the tool. At most, the analyst can opt to change the names of the shared services.

Since the substructure DESTINATION is part of the substructure ORDER, then the corresponding classes (CLIENTORDER and DESTINATION, respectively) are related by means of a structural relationship. Furthermore, since DESTINATIONS (the substructure that relates ORDER and DESTINATION) is an iteration substructure, then the structural relationship has maximum cardinality M on the side of class DESTINATION (the iteration specifies that one client order can have *many* destinations). Since no cardinalities can be derived from the structural constraints that appear in the event description template of *SALE 1*, then the analyst decides that the cardinalities 1:1 --- 0:M are appropriate and will later validate this decision with the users. See the structural relationship and its cardinalities in Figure 102.

Note that a possible structural constraint could have been "One destination corresponds to one and only one order". This restriction would have led to the same cardinalities.

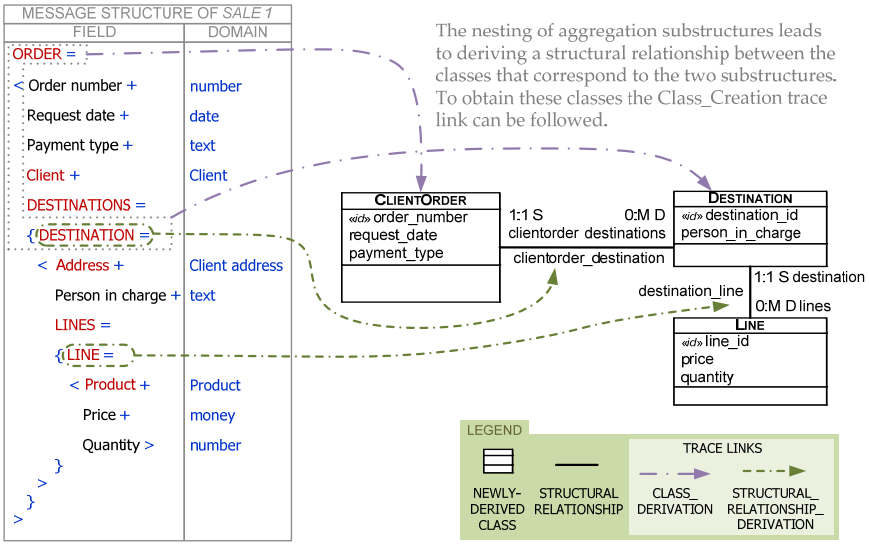


Figure 102. Derivation of structural relationships (class diagram view of communicative event *SALE 1*)

Similarly, since LINE is part of the substructure DESTINATION, then the corresponding classes (DESTINATION and LINE, respectively) are related by means of a structural relationship. Furthermore, since LINES (the substructure that relates DESTINATION and LINE) is an iteration substructure, then the maximum cardinality is M on the side of class LINE (the iteration specifies that one destination can have *many* lines). Again, the rest of the cardinalities depend on structural constraints, on the user response, or on the analyst criteria.

Reference fields lead to the derivation of structural relationships between the class that corresponds to the complex substructure containing the reference field and the class that corresponds to the reference field (see **OM12**). The reference field is expected to reference to an aggregation substructure that belongs to a precedent communicative event (otherwise there is either an invalidity in the precedence relations or the requirements model suffers incompleteness –typically, either a precedence relation or a communicative event are missing-). Also, the precedent communicative event is expected to have been already processed by the time this rule is applied (otherwise an error has been made during event sorting –rule **OM3**- or while following the order). Note that, as long as the communicative events have been properly sorted and processed in order, the referenced class has already been derived during a previous event processing. Otherwise, the derivation will run into an unexpected error, since there will be no class to define the structural relationship with.

Rule OM12. Derivation of a structural relationship from a reference field	
Pattern matching	
1	Each reference field of an aggregation substructure.
Preconditions	
	One of the following preconditions applies, depending on whether a new class is being created (precondition 1) or an existing class is being extended (precondition 2):
1	In case the aggregation substructure *is not* an extension substructure, then rule **OM4** has already been applied to it.
2	In case the aggregation substructure *is* an extension substructure, then rule **OM15** has already been applied to it.
Rule steps	
1	Select the *referencing class* by following the trace link of type Class_Derivation or the trace link of type Class_Extension from the aggregation substructure containing the reference field.
2	Select the *referenced class* as follows. The domain of the reference field points to another aggregation substructure that belongs to a precedent event (we will refer to this substructure as the *referenced aggregation substructure*). Find the referenced aggregation substructure and follow the trace link of type Class_Derivation or the trace link of type Class_Extension.
3	Create a structural relationship between the referencing class and the referenced class.
4	Assign the maximum cardinality 1 on the role side of the referenced class. Note that, if a higher cardinality had been intended, an iteration substructure should have been used instead of a reference field.
5	In case of a class creation (i.e. the reference field *does not* belong to an extension aggregation substructure) then...
6	Assign the rest of the cardinalities (i.e. the minimum cardinality on the role side of the referenced class and the minimum and maximum cardinalities on the role side of the referencing class) according to structural constraints included in the event description template. In the absence of such constraints, the analyst should ask the users or decide relying on his/her own judgement.
7	The analyst should decide whether the relationship is dynamic or static. In case of doubt, it should be defined dynamic because it is less restrictive.
8	In case of a class extension (i.e. the reference field belongs to an extension aggregation substructure) then...
9	Assign the minimum cardinality 0 on the role side of the referenced class (the moment the instance is created this link is not created; it occurs later).

10 Assign the rest of the cardinalities (i.e. the minimum and maximum cardinalities on the role side of the referencing class) according to structural constraints included in the event description template. In the absence of such constraints, the analyst should ask the users or decide relying on his/her own judgement.
11 Set the relationship as dynamic.
12 Set the name of the structural relationship to the concatenation of the names of the related classes. In any case, the analyst can choose a different name.
13 Set the name of the role at the side of the referenced class to the name of the reference field.
14 If the maximum cardinality at the side of the referencing class is 1, then...
15 Set the name of the role at the side of the referencing class to the name of the referencing class itself.
Else...
16 Set the name of the role at the side of the referencing class to the name of the referencing class itself, but in plural form.
17 If the relationship is dynamic then...
18 If the cardinality at any of the role ends is 1:1 (i.e. both the minimum and maximum cardinalities are 1 at any of the two sides of the relationship) then...
19 Create a shared service in both the referenced class and the referencing class.
20 Name the service change_rolename (i.e. the name of the role where the cardinality is 1:1 with the prefix change_).
21 Set the type of shared service to "Change".
Else...
22 Create a shared service in both the referenced class and the referencing class.
23 Name the service ins_rolename (i.e. the name of the role where the cardinality is 1:1 with the prefix ins_).
24 Set the type of shared service to "Insert".
25 Create a shared service in both the referenced class and the referencing class.
26 Name the service del_rolename (i.e. the name of the role where the cardinality is 1:1 with the prefix del_).
27 Set the type of shared service to "Delete".
28 For each shared service created above do...
29 Add an object-valued inbound argument.
30 Name the argument p_thisClass (i.e., the name of the class to which the service belongs with the prefix p_this).
31 The data type of the argument is the class to which the service

	belongs. It does not accept null values.
32	Add an object-valued inbound argument.
33	Name the argument p_evcOppositeClass (i.e., the name of the class at the other side of the structural relationship with the prefix p_evc).
34	The data type of the argument is the class at the other side of the structural relationship. It does not accept null values.
Traceability information	
1	Create a trace link of type Structural_Relationship_Derivation between the communicative event and the newly-created structural relationship (this can be done after step 3).
2	Create a trace link of type Structural_Relationship_Derivation between the reference field and the newly-created structural relationship (this can be done after step 3).
3	Create a trace link of type Referenced_Class between the reference field and the referenced class (this can be done after step 2).
4	In case any cardinality is derived from a structural constraint, then create a trace link of type Cardinality_Derivation between the structural constraint and the newly-created structural relationship (this can be done after steps 6 and 10).
5	Create a trace link of type Shared_Service_Derivation between the communicative event and each newly-created shared service (this can be done after steps 19, 22 and 25).
6	Create a trace link of type Shared_Service_Derivation between the aggregation substructure containing the reference field and each newly-created shared service (this can be done after steps 19, 22 and 25).
7	Create a trace link of type Shared_Service_Derivation between the reference field and each newly-created shared service (this can be done after steps 19, 22 and 25).
Notes	
1	In case the analyst is creating the conceptual model using Integranova Modeler, then steps 17 to 34 are automatically performed by the tool. At most, the analyst can opt to change the names of the shared services.

Provided that the requirements model is complete and the communicative events have been processed in the prescribed order, at the moment of applying this rule, the processing of previous communicative events has already led to the derivation of several classes (we depict them in grey in Figure 103).

The reference field Client belongs to the complex substructure ORDER, whose corresponding class is CLIENTORDER; we consider CLIENTORDER the referencing class. The domain of Client is Client; this acts as a pointer towards the aggregation substructure CLIENT INFO. During the processing of communicative event *CLIE 1*,

CLIENT INFO led to the derivation of the class CLIENT; now CLIENT is considered the referenced class. By applying rule **OM12**, step 3, a structural relationship is defined between the classes CLIENTORDER and CLIENT. A similar reasoning is done for the reference fields Address and Product. They lead to the derivation of structural relationship between the classes DESTINATION and CLIENTADDRESS, and LINE and PRODUCT, respectively.

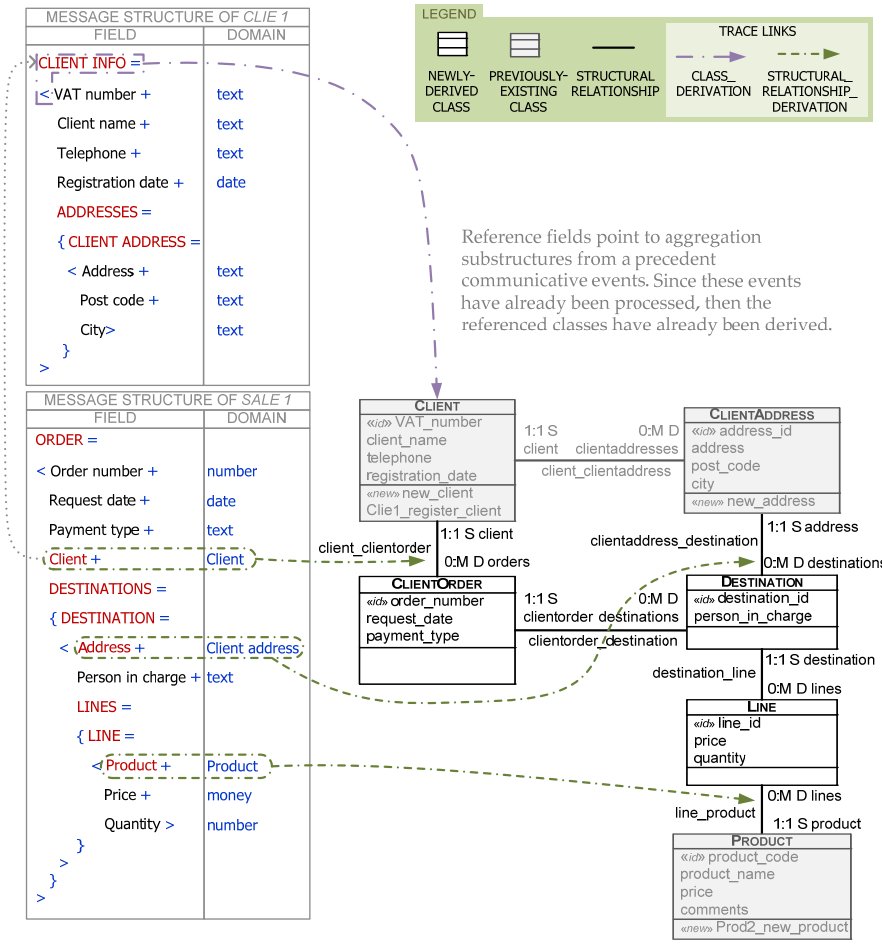


Figure 103. Derivation of the structural relationships from reference fields (class diagram view of communicative event *SALE 1*)

With regards to the cardinalities, the reference field Client refers to the client that places the order; note that a message field represents only one piece of data (i.e. an order is placed by at most one client), as opposed to when an iteration substructure is used. Thus, according to step 4, the maximum cardinality on the side of class CLIENT is 1. Also, there is a structural relationship in the event specification template stating that "One order is places by exactly one client" (the communicative event template is shown in Figure 67). Thus, according to step 6, the minimum cardinality on the side of class CLIENT is 1, as well. Although there is no explicit structural restriction concerning the cardinalities on the role side of the class CLIENTORDER, step 7 has been applied; relying on her knowledge of the field, the analyst has opted for cardinalities 0:M.

The analyst has decided that the structural relationship is static, according to her knowledge of the business (once an order has been placed by a client, the order remains always of the same client). Since the relationship is static, no shared services need to be defined (an object-valued inbound argument of the creation service will serve the same purpose).

A similar reasoning applies to the cardinalities of the relationship between DESTINATION and CLIENTADDRESS, and to the cardinalities of the relationship between LINE and PRODUCT. The derivation of the above-mentioned structural relationships is depicted in Figure 103.

In order to create instances of classes and provide the values of some of their attributes, creation services are needed. Therefore, each newly-derived class needs to have a creation derived (see rule **OM13**). It is also necessary to define the arguments of the creation services (these arguments will convey the values for the attributes). To do this, the analyst uses the fields of the message structures. Data fields derive data-valued inbound arguments, whereas reference fields derive object-valued inbound arguments.

According to the OO-Method and its CASE tool support Integranova, atomic creation services do not need valuation rules, since their semantics is implicitly defined: for each data-valued argument its corresponding attribute is initialised to the argument value, for each object-valued argument a link is created between the instance created as a result of the creation service and the instance referenced by the object-valued argument.

Rule OM13. Addition of a creation service to a newly-created class	
Pattern matching	
1	Each aggregation substructure that *is not* an extension substructure.
Preconditions	
1	Rules **OM5** to **OM12** have already been applied to the fields of the aggregation structure.
Rule steps	
1	Select the class to which the service should be added. For this purpose, simply follow the trace link of type Class_Derivation from the aggregation substructure
2	Add a creation service to the class.
3	The analyst can freely decide the name of the creation service, although it is recommended to give it the name new_class_name (i.e., the class name with the prefix new_).
4	For each data field contained in the aggregation substructure do...
5	Add a data-valued inbound argument to the creation service.
6	The analyst can freely decide the name of the data-valued inbound argument, although it is recommended to give it the name p_atrdatafield (i.e., the data field name with the prefix p_atr).
7	The properties of the data-valued inbound arguments coincide with the properties of their corresponding attributes. This means that the data type, size and compulsoriness are the same that were specified for the attributes.
8	For each reference field contained in the aggregation substructure that has led to the derivation of the service do...
9	Add an object-valued inbound argument to the creation service.
10	The analyst can freely choose the name of the object-valued inbound argument, although it is recommended to give it the name p_agrreferencefield (i.e., the reference field name with the prefix p_agr).
11	Obtain the referenced class (i.e. the class to which the reference field refers). For this purpose, follow following the trace link of type Referenced_Class from the reference field.
12	The argument data type is the referenced class.
13	Obtain the structural relationship derived from the reference field by following the trace link of type Structural_Relationship_Derivation.
14	If the minimum cardinality of the relationship on the side of the referenced class is 0 then...
15	Set the property *Null allowed* of the argument to "yes".
	Else (if it is 1)...
16	Set the property *Null allowed* of the argument to "no".
17	If the aggregation substructure is nested inside another aggregation

	substructure (referred to as the outer substructure), then...
18	Add an object-valued inbound argument to the creation service.
19	Obtain the outer class (i.e. that corresponds to the outer substructure). For this purpose, follow the trace link of type Class_Derivation or the trace link of type Class_Extension from the outer substructure.
20	The analyst can freely choose the name of the object-valued inbound argument, although it is recommended to give it the name p_agrOuterClass (i.e., the name of the outer class with the prefix p_agr).
21	The argument data type is the outer class.
13	Obtain the structural relationship between the outer class and the current class being extended by following the trace link of type Structural_Relationship_Derivation from the aggregation substructure being processed.
14	If the minimum cardinality of the relationship on the side of the outer class is 0 then...
15	Set the property *Null allowed* of the argument to "yes".
	Else (if it is 1)...
16	Set the property *Null allowed* of the argument to "no".
Traceability information	
1	Create a trace link of type Creation_Service_Derivation between the communicative event and the newly-created creation service (this can be done after step 2).
2	Create a trace link of type Creation_Service_Derivation between the aggregation substructure and the newly-created creation service (after step 2).
3	Create a trace link of type Argument_Derivation between the communicative event and each newly-created data-valued argument (after step 5).
4	Create a trace link of type Argument_Derivation between the newly-created data-valued argument and its corresponding data field (after step 5).
5	Create a trace link of type Argument_Derivation between the communicative event and each newly-created object-valued argument (after step 9).
6	Create a trace link of type Argument_Derivation between the newly-created object-valued argument and its corresponding reference field (after step 9).
Notes	
1	In case the analyst is creating the conceptual model using Integranova Modeler, then only steps 1 to 3 are needed. The tool automatically adds a data-valued argument for each attribute whose value *Requested upon creation* is set to "yes". It also adds an object-valued argument for each structural relationship in which the cardinality at the opposite role side is either 0:1 or 1:1.

For instance, a service named new_order is added to the class CLIENTORDER (see Figure 104). Similarly, the classes DESTINATION and LINE are added creation services named new_destination and new_line, respectively.

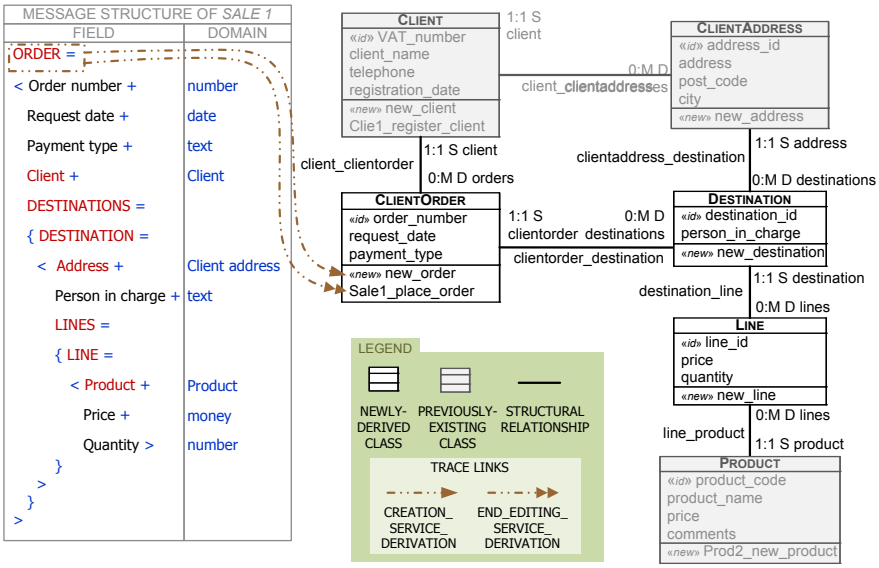


Figure 104. Derivation of creation and end-of-editing services (class diagram view of communicative event *SALE 1*)

Data-valued arguments of creation services are derived from data fields of message structures. For instance, the data field Order number has led to the derivation of the attribute order_number (by applying the rule **OM5**). Now it leads to the derivation of a data-valued argument named p_atrorder_number, which is added to the creation service new_order. A similar reasoning applies to data fields Request date and Payment type. The data-valued arguments and their properties are specified in Table 29. Notice that its properties are the same that its corresponding attributes (see Table 27).

Object-valued arguments of creation services are derived from reference fields of message structures. For instance, the reference field Client has led to the derivation of a structural relationship between classes CLIENTORDER and CLIENT. Now it leads to the derivation of an object-valued argument named p_agrClient, which is added to the service new_order. This attribute defines the client who places the order. The data type of the argument is the class CLIENT. Since the minimum cardinality of the above-mentioned structural relationship on the side of class CLIENT is 1 (see Figure 104) then the inbound argument does not allow

null values. The object-valued arguments and their properties are specified in Table 29.

Table 29. Specification of inbound arguments of service new_order

Argument name[96]	Data type	Size	Null allowed
p_atrorder_number	Autonumeric	-	no
p_atrrequest_date	Date	-	no
p_atrpayment_type	String	30	yes
p_agrClient	Client	-	no

When a communicative event affects several classes (i.e. it has either added or extended two or more classes), then, apart from the creation or editing service for each of its constituent classes, another service needs to be added to the *main* class of the business object (see rule **OM14**). We refer to this kind of service as end-of-editing service because, by means of executing it, the user indicates that s/he has finished editing the message corresponding to the communicative event and that the information system reaction should be triggered. End-of-editing services need no valuation rules; they simply act as a signal.

In the case of *SALE 1*, the client order is a complex business object. Therefore, apart from the creation service new_order, an *end of editing* service named Sale1_place_order is added to the class CLIENTORDER (see Figure 104). This service is triggered by the salesperson whenever s/he has finished introducing the information if the order; that is, after introducing the destinations and the lines. Only after this service is executed, the order is considered to be placed. Table 30 shows the details of the inbound argument (according to the OO-Method guidelines for this kind of services, it only needs to define an argument for the client order which is being placed, see rule **OM14**, steps 5 to 8).

Table 30. Specification of inbound arguments of service Sale1_place_order

Argument name	Data type	Size	Null allowed
p_thisClientOrder	ClientOrder	-	no

[96] Prefixes in the argument names follow Integranova naming conventions.

Rule OM14. Addition of an end-of-editing service for complex messages	
Pattern matching	
1	A message structure that contains one or more nested complex substructures (e.g. A=<a+b+C=<d+e>+f> matches this condition, but A=<a+b+f> does not match it).
Preconditions	
	One of the following preconditions applies, depending on whether a new class is being created (precondition 1) or an existing class is being extended (precondition 2):
1	In case the aggregation substructure *is not* an extension substructure, then rule **OM13** has been applied to the aggregation substructure.
2	In case the aggregation substructure *is* an extension substructure, then rule **OM16** has been applied to the aggregation substructure.
Rule steps	
1	Find out how many classes the communicative event has affected. For this purpose, follow the trace links of type Class_Derivation or those of type Class_Extension.
2	In case the communicative event has affected more than one class, then...
3	Add a service to the main class (the class that was derived or affected in the first place; that is, the class mapped to the initial aggregation substructure of the message structure).
4	The analyst can freely choose the name of service, although it is recommended to give it the name eventid_verb_businessobjectname (i.e., the event identifier followed by the action and the business object, according to the event name).
5	Add an object-valued inbound argument to the service.
6	Name the argument p_thisClassName (i.e., the name of the class prefixed with p_this).
7	The argument data type is the class to which the service has been added.
8	Set the property Null allowed of the argument to "no".
Traceability information	
1	Create a trace link of type End_Editing_Service_Derivation between the communicative event and the newly-created end-of-editing service (this can be done after step 3).
Notes	
1	In case the analyst is creating the conceptual model using Integranova Modeler, then the tool automatically performs steps 5 to 8.

Extending classes with new attributes, relationships and services

As seen above, some communicative events add new classes to a class diagram. There are other communicative events, however, that extend previously-derived classes by adding new attributes, services and structural relationships them. The events that extend existing classes are those whose message structure includes a reference field that has been marked according to rule **OM2**. This way, whenever we encounter a reference field whose property *Extends business object* has been set to True, we need to select a previously-derived class (see rule **OM15**) and then extend this with new attributes (the attribute is created by means of rule **OM5**) and/or structural relationships (see rule **OM11**, that processes nested substructures, and rule **OM12**, that processes reference fields).

Rule OM15. Selection of a class being extended	
Pattern matching	
1	Each aggregation substructure within the message structure of the event being processed that *is* an extension substructure.
Preconditions	
1	Rule **OM2** has already been applied to the requirements model.
Rule steps	
1	To identify the class that needs to be extended, first take the reference field whose property *Extends business object* is True. The domain indicates the (part of the) business object being extended.
2	Find the aggregation substructure that corresponds to the creation of this (part of a) business object; it will belong to a precedent communicative event (otherwise there is either an invalidity in the precedence relations or an error has been made during the event sorting, rule **OM3**).
3	Then obtain the class that was derived from this aggregation substructure; to do this, simply follow the trace link of type Class_Derivation.
Traceability information	
1	Create a trace link of type Class_Extension between the communicative event and the selected class (this can be done after step 3).
2	Create a trace link of type Class_Extension between the extension aggregation substructure and the selected class (after step 3).
3	Create a trace link of type Class_Extension between the reference field whose property *Extends business object* is True and the selected class (after step 3).

After an order is placed, the Sales Manager assigns the order to one of the many suppliers that work with SuperStationery. Thus, communicative event *SALE 2* affects the same business object as communicative event *SALE 1*; namely, the client order. The message structure of communicative event *SALE 2* includes a

reference field that, according to rule **OM2**, has been marked as extending a business object; namely, Order (see Table 26). Now, according to rule **OM15**, the event leads to extending an existing class. The domain of Order is Order (i.e. a client order); therefore, class CLIENTORDER is extended. Figure 105 depicts how to select the class being extended.

The data fields in the message structure of *SALE 2* lead to adding new attributes to this class (see rule **OM5**); for instance, the field Assignment date leads to adding an attribute named assignment_date to the class CLIENTORDER. Rule **OM6** does not apply to class extension because class identifiers are defined during class creation (i.e. attributes added during a class extension are not part of the identification function). However, other rules related to attribute properties do apply to the case of class extension. According to rule **OM7**, attributes added while extending a class are necessarily of type variable. Rule **OM8** determines the data type of the attribute; in the case of assignment_date, since the domain of its corresponding data field is date, the selected data type is Date. According to **OM9**, attributes of a class extension should not be requested upon creation. According to rule **OM10**, attributes of a class extension should allow null values. Figure 105 depicts this derivation and Table 31 specifies the details of the new attribute.

As a rule of thumb, note that attributes of a class extension should always be of type variable, not requested upon creation and should allow null values.

In turn, the reference fields in the message structure lead to adding new structural relationships between class CLIENTORDER and other classes that (presumably[97]) already exist in the class diagram under construction (see rule **OM12**).

Table 31. Specification of the new attribute added to class CLIENTORDER

Attribute name	Id	Attribute type	Data type	Size	Requested	Null allowed
assignment_ date	no	Variable	Date	-	no	yes

[97] As argued above, as long as the communicative event diagram has been properly extended so as to include all the precedent communicative events and the requirements model is complete, the classes to be related already exist in the class diagram. In case of incompleteness, the classes may not exist.

With regards to reference fields, the field Supplier references a business object that was processed in the communicative event *SUPP 2*. Therefore, a structural relationship is defined between the class CLIENTORDER and the class SUPPLIER. The maximum cardinality in the side of SUPPLIER is 1 (according to rule **OM12**, step 4); the minimum cardinality is necessarily 0 because the orders are not assigned to suppliers when they are placed, but it occurs in a later moment in time. For the same reason, the relationship is dynamic (see rule **OM12**, step 11).

According to rule **OM12** (steps 22-27), two shared services are included in each of the two classes due to the cardinality of the structural relationship, and the fact that it is dynamic; namely, an insertion shared service named ins_supplier and a deletion shared service named del_supplier. Rule **OM12** (steps 28-34) indicates how the inbound arguments of these services are derived; the following tables specify them.

Table 32. Specification of inbound arguments of service CLIENTORDER.ins_supplier

Argument name	Data type	Size	Null allowed
p_thisClientOrder	ClientOrder	-	no
p_evcSupplier	Supplier	-	no

Table 33. Specification of inbound arguments of service SUPPLIER.ins_supplier

Argument name	Data type	Size	Null allowed
p_thisSupplier	Supplier	-	no
p_evcClientOrder	ClientOrder	-	no

Table 34. Specification of inbound arguments of service CLIENTORDER.del_supplier

Argument name	Data type	Size	Null allowed
p_thisClientOrder	ClientOrder	-	no
p_evcSupplier	Supplier	-	no

Table 35. Specification of inbound arguments of service SUPPLIER.del_supplier

Argument name	Data type	Size	Null allowed
p_thisSupplier	Supplier	-	no
p_evcClientOrder	ClientOrder	-	no

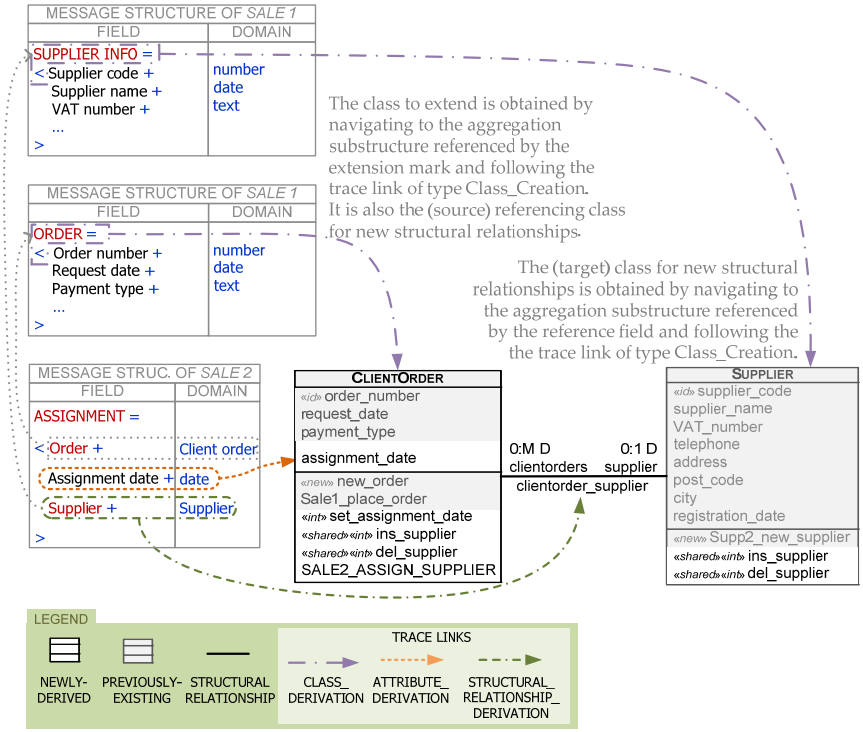


Figure 105. Extension of a class (class diagram view of communicative event *SALE 2*)

Whenever new attributes are added to an extended class, a service needs to be added in order to set their values (see rule **OM16**). According to the OO-Method, these attributes are said to be *own services* (a.k.a. editing services), what means that they are neither creation services nor destruction services.

These services need to have arguments and valuation rules defined, even if the Integranova Modeler is used as tool support. Arguments are also derived by applying rule **OM16**. Valuation rules belong to a different model of the OO-Method conceptual model; namely, the Functional Model. Thus, following the principle of separation of concerns [Dijkstra 1974], their derivation has defined in a different set of rules (see Section 5.5.2). Specifically, the valuation rules that correspond to the services derived by rule **OM16** can be derived by applying rule **FM1**.

Rule OM16. Addition of an editing service to an extended class	
Pattern matching	
1	Each aggregation substructure that *is* an extension substructure.
Preconditions	
1	Rule **OM5**, and rules **OM7** to **OM12** have already been applied to the fields of the aggregation structure.
Rule steps	
1	Select the class to which the service should be added. For this purpose, follow the trace link of type Class_Extension from the communicative event being processed.
2	Add a service to the class.
3	If the class has been extended with just one attribute then...
4	Set the name of the service to set_attribute_name (i.e. the name of the attribute with the prefix set_). In any case, the analyst can choose a different name.
	Else...
5	Set the name of the service to set_aggregation_name (i.e. the name of the aggregation substructure with the prefix set_). In any case, the analyst can choose select a more appropriate name that represents all the information being provided.
6	Add an object-valued inbound argument to the newly-added service.
7	Name the argument p_thisClassName (i.e., the name of the class prefixed with p_this).
8	Set argument data type to the class to which the service has been added.
9	Set the property *Null allowed* of the argument to "no".
10	For each data field contained in the aggregation substructure do...
11	Add a data-valued inbound argument to the newly-added service.
12	The analyst can freely decide the name of the data-valued inbound argument, although it is recommended to give it the name pt_datafield (i.e., the data field name with the prefix pt_).
13	The properties of the data-valued inbound arguments coincide with the properties of their corresponding attributes. This means that the data type, size and compulsoriness are the same that have been specified for the attributes.
Traceability information	
1	Create a trace link of type Extension_Service_Derivation between the communicative event and the newly-created editing service (this can be done after step 2).
2	Create a trace link of type Extension_Service_Derivation between the aggregation substructure and the newly-created editing service (after step 2).

3	Create a trace link of type Argument_Derivation between the communicative event and each newly-created data-valued argument (after step 11).
4	Create a trace link of type Argument_Derivation between the newly-created data-valued argument and its corresponding data field (after step 11).
Notes	
1	In case the analyst is creating the conceptual model using Integranova Modeler, then the tool automatically performs steps 6 to 9.

By applying rule **OM16** to event Sale 2 (properly speaking, to the aggregation substructure ASSIGNMENT), a service is added to the class CLIENTORDER so as to inform of the value of the attribute assignment_date; the service is named set_assignment_date. Table 36 specifies the arguments of the service.

Table 36. Specification of inbound arguments of service set_assignment_date

Argument name	Data type	Size	Null allowed
p_thisClientOrder	ClientOrder	-	no
pt_assignment_date	Date	-	no

Note that steps 6 to 9 create an inbound argument that refers to the instance for which the service is triggered (in the example this argument is p_thisClientOrder). This addition of an argument of this type is prescribed by the OO-Method for every service that is not a creation service.

When the reaction to a communicative event requires the execution of several services of the same class, then it is convenient to create a transaction is in order to ensure the atomic execution of all these services. Otherwise, the user could inadvertently forget (or deliberately omit) executing one of the services and the system would remain in an inconsistent state. Rule **OM17** creates the transaction and sets the atomic services as internal. The transaction formula belongs to the Functional Model and it is derived by rule **FM2**.

Rule OM17. Definition of a transaction for complex reactions	
Pattern matching	
1	Each aggregation substructure that *is* an extension substructure.
Preconditions	
1	Rule **OM16** has already been applied to the aggregation substructure.
Rule steps	
1	Obtain a list of all the services that need to be encapsulated in the transaction. For this purpose, follow the trace links of type Extension_Service_Derivation and Shared_Service_Derivation (but consider only those of type "Insert" or "Change", not those of type "Delete").
2	Obtain the class that corresponds to the aggregation substructure by following the trace link of type Class_Extension.
3	Add a transaction to the class.
4	Add an object-valued inbound argument to the newly-added service.
5	Name the argument p_thisClassName (i.e., the name of the class prefixed with p_this).
6	Set the argument data type to the class to which the service has been added.
7	Set the property *Null allowed* of the argument to "no".
8	For each of the services in the list do...
9	For each argument of the service do...
10	If the name of the argument *is not* p_thisClassName (i.e. the name of the class to which the transaction is being added prefixed with p_this) then...
11	Add an identical argument to the transaction (same name, same data type, same value for the property *Null allowed*).
12	Set all the services of the list as internal (i.e. set the property *Internal use* to "yes").
Traceability information	
1	Create a trace link of type Transaction_Derivation between the communicative event and the newly-created transaction (this can be done after step 3).
2	Create a trace link of type Transaction_Derivation between the aggregation substructure and the newly-created transaction (after step 3).
3	Create a trace link of type Argument_Derivation between the communicative event and each newly-created argument (after step 11).
Notes	
1	In case the analyst is creating the conceptual model using Integranova Modeler, then the tool automatically performs steps 4 to 7.
2	Also, if the analyst wants to avoid having to define each argument of the transaction (that is, step 11) then see note 1 in rule **FM2**.

For instance, in order to convey the message associate to event *SALE 2. Sales Manager assigns supplier*, two services are derived: ins_supplier is a shared service that assigns the order to a supplier by linking an instance of CLIENTORDER with an instance of SUPPLIER, set_assignment_date provides the value of the date in which the supplier is assigned the order. Both services need to be executed; it should not be allowed to only set the date but not assign the order. In order to preserve data integrity, a transaction named S2_ASSIGN_SUPPLIER is defined. This transaction will execute atomically the services set_assignment_date and ins_supplier (but the corresponding transaction formula is derived by applying rule **FM2**). Figure 105 shows the derived transaction; note also that the two atomic services have been set as internal.

Table 37. Specification of inbound arguments of the transaction S2_ASSIGN_SUPPLIER

Argument name	Data type	Size	Null allowed
p_thisClientOrder	ClientOrder	-	no
pt_assignment_date	Date	-	no
pt_p_evcSupplier	Supplier	-	no

Regardless of how many services have been derived while processing a communicative event and of which types (creation, extension, transaction, end-of-editing):

- One of them corresponds to the establishment of contact with the information system with regards to the communicative event; that is, the service that starts the editing of the message associated with the event. We refer to this service as the contact service of the event.
- One of them (it could be the same as the contact service) actually triggers the organisational reaction to the event occurrence. We refer to this service as the reaction service of the event. It is convenient to prefix the name of the reaction service with the identifier of the communicative event, so as to spot it visually.

These services are determined (and the corresponding trace links are created) by means of rule **OM18**.

Rule OM18. Determination of the contact and reaction events	
Pattern matching	
1	Each communicative event within the transformation-scope diagram.
Preconditions	
	For all the aggregation substructures contained in the message structure of all the communicative events within the transformation-scope diagram, one of the following preconditions needs to apply:
1	In case the aggregation substructure *is not* an extension substructure, then rule **OM13** and (if applicable) rule **OM14** have already been applied to the substructure.
2	In case the aggregation substructure *is* an extension substructure, then rule **OM16** and (if applicable) rule **OM14** have already been applied to the substructure.
Rule steps	
1	If the communicative event has an associated end-of-editing service (try to follow the trace link of type End_Editing_Service_Derivation from the event), then...
2	Determine that the end-of-editing service is the reaction service of the event.
	Else...
3	If the communicative event has an associated transaction (try to follow the trace link of type Transaction_Derivation from the event), then...
5	Determine that the transaction is the reaction service of the event.
	Else...
6	If the communicative event has an associated extension service (try to follow the trace link of type Extension_Service_Derivation from the event), then...
7	Determine that the extension service is the reaction service of the event.
	Else...
8	Determine that the creation service is the reaction service of the event (to obtain the service, follow the trace link of type Creation_Service_Derivation from the event).
9	Change the name of the reaction service in order to prefix it with the identifier of the communicative event as in EventId_servicename. Remove any spaces that the identifier might have.
10	If the communicative event has an associated creation (try to follow the trace link of type Creation_Service_Derivation from the event), then...
11	Determine that the creation service is the contact service of the event.
12	Else...
13	If the communicative event has an associated transaction (try to follow

	the trace link of type Transaction_Derivation from the event), then...
14	Determine that the transaction is the contact service of the event.
15	Else...
16	If the communicative event has an associated extension service (try to follow the trace link of type Extension_Service_Derivation from the event), then...
17	Determine that the extension service is the contact service of the event.
Traceability information	
1	Create a trace link of type Reaction_Service between the communicative event and the service determined as reaction service (this can be done after step 9).
2	Create a trace link of type Contact_Service between the communicative event and the service determined as contact service (after step 17).

We now offer an example of each of the cases considered in rule **OM18**. In the case of the communicative event *SALE 1. The client places an order*, an end-of-editing service was derived. Therefore, this service is considered the reaction event and its name is prefixed the event identifier: Sale1_place_order. In the case of *SALE 2. Sales manager assigns supplier*, no end-of-editing service was derived, but a transaction was derived in order to ensure the atomic execution of the services set_assignment_date and ins_supplier. Therefore, the transaction is considered the reaction event and its name is prefixed the event identifier: SALE2_ASSIGN_SUPPLIER. In the case of *SALE 6. Supplier notifies the shipping of the goods*, the extension service is the reaction event: Sale6_notify_shipping. In the case of *PROD 2. Company director defines catalogue*, the creation service is the reaction event: Prod2_new_product.

Figure 106 shows the class diagram (Object Model) that results from processing the remaining events. The Dynamic Model, whose derivation is explained in Section 5.5.3, specifies system behaviour, so attributes such as order_state are not strictly necessary for this purpose.

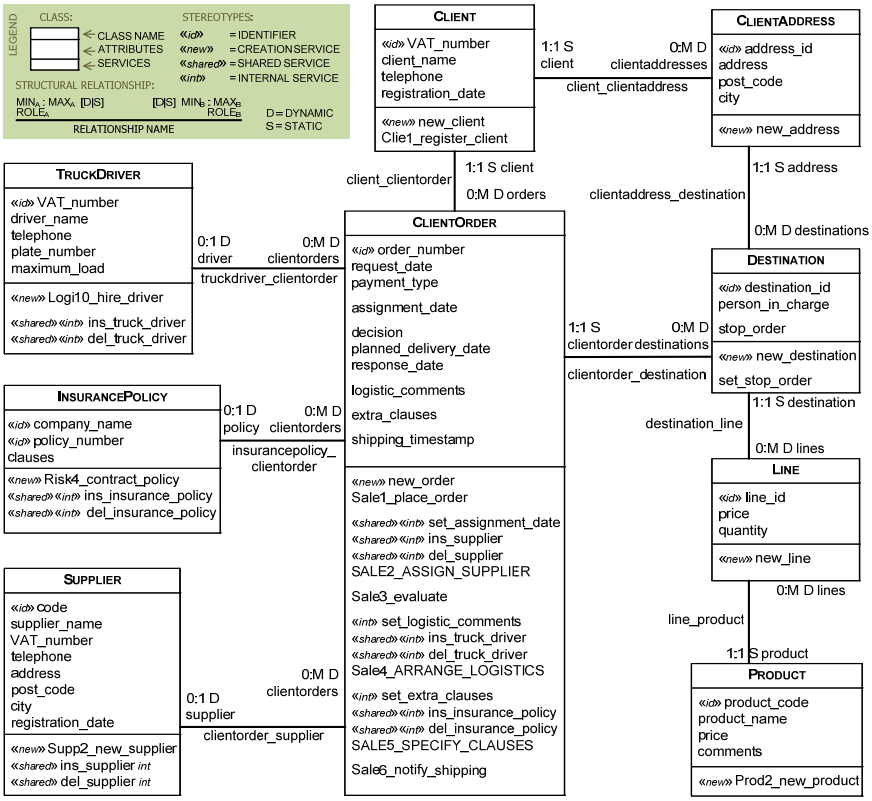


Figure 106. Class diagram that corresponds to the *SALES* business process of SuperStationery Co.

Defining agents of the system (modelling software users)

The OO-Method defines user permissions by means of defining agent classes and agent relationships. An agent class is simply a class that represents a type of user of the software system (properly speaking, an organisational role). Later on, users are defined as instances of agent classes, what allows them to log in and access the functionality of the application. Agent relationships are a special kind of relationship that relates an agent class with the services (possibly belonging to other classes) that a user of such kind should have permission to trigger. Agent relationships restrict the interaction of users with the application.

Agent classes can contain attributes. Table 38 suggests some attributes for agent classes; but the analyst should discover the attributes necessary in each specific project.

Table 38. Specification of the suggested attributes for agents (i.e. users of the software system)

Attribute name	Id	Attribute type	Data type	Size	Requested	Null allowed
id_user	yes	Constant	Autonumeric	-	yes	no
first_name	no	Constant	String	50	yes	no
last_name	no	Constant	String	50	yes	no
telephone	no	Variable	String	24	yes	yes
email	no	Variable	String	50	yes	yes

For each organisational role that plays the interface role in any communicative event, an agent class should be derived (see **OM19**).

Note that some aggregation substructures include fields representing pieces of information about an actor playing a given organisational role. For instance, the message structure of the communicative event *CLIE 1. Salesman registers a client* corresponds to the client record. When such an aggregation substructure is processed (by means of rule **OM4**), it leads to the derivation of a class that may later be defined as agent (the clients of SuperStationery are not expected to interact with the software, but it could be the case if a reengineering was performed and a web interface was provided to clients). It is convenient to reuse the previously-derived class as an agent class instead of duplicating it when applying rule **OM19**. Class duplication is prevented provided:

1. The analyst has related the organisational role with the message structure (more specifically, with the *main* aggregation substructure that contains the fields describing the actor) during requirements modelling. The platform independent metamodel contains a relationship intended for this purpose: ORGANISATIONAL_ROLE 0:1 --- 0:1 AGGREGATION.
2. The traceability information of rule **OM4** is properly recorded (especially, the trace link of type "organisational_role2class").
3. This traceability information is consulted when applying rule **OM19** (step 1). In case the class already exists, it is not created again; at most, it is added some attributes.

Rule OM19. Addition of agent class from organisational role	
Pattern matching	
1	Each organisational role acting the interface role in at least one communicative event within the transformation-scope diagram.
Preconditions	
	For all the aggregation substructures contained in the message structure of all the communicative events within the transformation-scope diagram, one of the following preconditions needs to apply:
1	In case the aggregation substructure *is not* an extension substructure, then rule **OM13** and (if applicable) rule **OM14** have already been applied to the substructure.
2	In case the aggregation substructure *is* an extension substructure, then rule **OM16** and (if applicable) rule **OM14** have already been applied to the substructure.
Rule steps	
1	Find out whether a class has been already derived for this organisational role. Try following the trace link of type Agent. If no class has yet been derived then...
2	Create a new class.
3	Assign to the name of the class the name of the organisational role, replacing spaces by underscores and using uppercase letters. In any case, the analyst can choose to give a different name to a class.
4	Add a creation service to the class.
5	The analyst can freely decide the name of the creation service, although it is recommended to give it the name new_class_name (i.e., the class name with the prefix new_). For each of the attributes that are considered necessary for a computerised information system user (see Table 38 for a suggested set of attributes and their properties) do...
6	Add the attribute and set its properties.
	Else...
7	In case the class had been previously derived, it may already have some of the attributes defined. For each of the attributes that are considered necessary for a computerised information system user (see Table 38 for a suggested set of attributes and their properties) and have not yet been defined do...
8	Add the attribute and set its properties.
9	For each attribute that has been added to the class during the application of this rule do...
10	Add a data-valued inbound argument to the creation service.
11	The analyst can freely decide the name of the data-valued inbound

12	argument, although it is recommended to give it the name p_atrdatafield (i. e., the data field name with the prefix p_atr). The properties of the data-valued inbound arguments coincide with the properties of their corresponding attributes. This means that the data type, size and compulsoriness are the same that were specified for the attributes.
Traceability information	
1	Create a trace link of type Agent between the organisational role and the newly-created class (this can be done after step 2).

Salespersons play the interface role in *CLIE 1, SUPP 2, SALE 1, SALE 3* and *SALE 6*. When rule **OM19** is applied to the organisational role Salesman, a class named SALESMAN is derived. Similarly, a class named SALESMANAGER is derived because the Sales manager plays the interface role in *PROD 2* and *SALE 2*. Figure 107 depicts these agent classes. Now rule **OM20** defines the agent relationships that grant salespersons enough permissions within the application so as to be able to carry out their duties (i.e. they should be able to trigger those services that have been derived while processing the communicative events in which they are the interface actor).

Rule OM20. Derivation of agent relationships from interface roles	
Pattern matching	
1	Each communicative event within the transformation-scope diagram.
Preconditions	
	For all the aggregation substructures contained in the message structure of all the communicative events within the transformation-scope diagram, one of the following preconditions needs to apply:
1	In case the aggregation substructure *is not* an extension substructure, then rule **OM13** and (if applicable) rule **OM14** have already been applied to the substructure.
2	In case the aggregation substructure *is* an extension substructure, then rule **OM16** and (if applicable) rule **OM14** have already been applied to the substructure.
3	Also, rule **OM19** has been applied to each organisational role acting the interface role in at least one communicative event within the transformation-scope diagram.
Rule steps	
1	Create a list with all the services derived from the communicative event. For this purpose, follow the trace links of type Shared_Service_Derivation, Creation_Service_Derivation, End_Editing_Service_Derivation and Extension_Service_Derivation, from the communicative event.
2	For each of the services in the list do...
3	Create a list with all the agent classes corresponding to the interface roles of the communicative event. For this purpose, go each of the organisational roles playing the interface role in the event and then follow the trace link of type Agent.
4	For each of the agent classes in the list do...
5	Create an agent relationship that grants permissions to the agent class to execute the service.
Traceability information	
1	For each of the agent classes in the list create a trace link of type Agent between the communicative event and the agent class (this can be done after step 5).

When applied to communicative event *SALE 1*, salespersons are granted permission to record the placement of an order; this implies triggering the following services: CLIENTORDER.new_order, DESTINATION.new_destination, LINE.new_line and CLIENTORDER.Sale1_place_order. When applied to *SALE2*, the Sales manager is granted permission to assign the order to a supplier; this implies triggering the transaction CLIENTORDER.SALE2_ASSIGN_SUPPLIER. Note that, since users can only access to the functionality provided by services that are not

internal, internal services do not take part in agent relationships. Figure 107 depicts the agents SALESPERSON and SALESMANAGER, as well as the agent relationships that correspond to communicative events *PROD 2*, *CLIE 1*, *SUPP 2*, *SALE 1* and *SALE 2*.

There are other kinds of agent relationships that need to be defined in order to ensure a usable interaction. Agents need to be granted permission to view class attributes; otherwise the software application interface will not display this information to them. Similarly, agents need to be granted permission to traverse structural relationship roles; otherwise the interface will not allow them to navigate properly. However, these agent relationships fall out of the scope of this thesis, since they are interaction aspects related to outgoing communications. They will be considered in future work. On the contrary, the agent relationships derived by means of rule **OM20** allow conveying messages to the information system; they are related to ingoing communicative interactions.

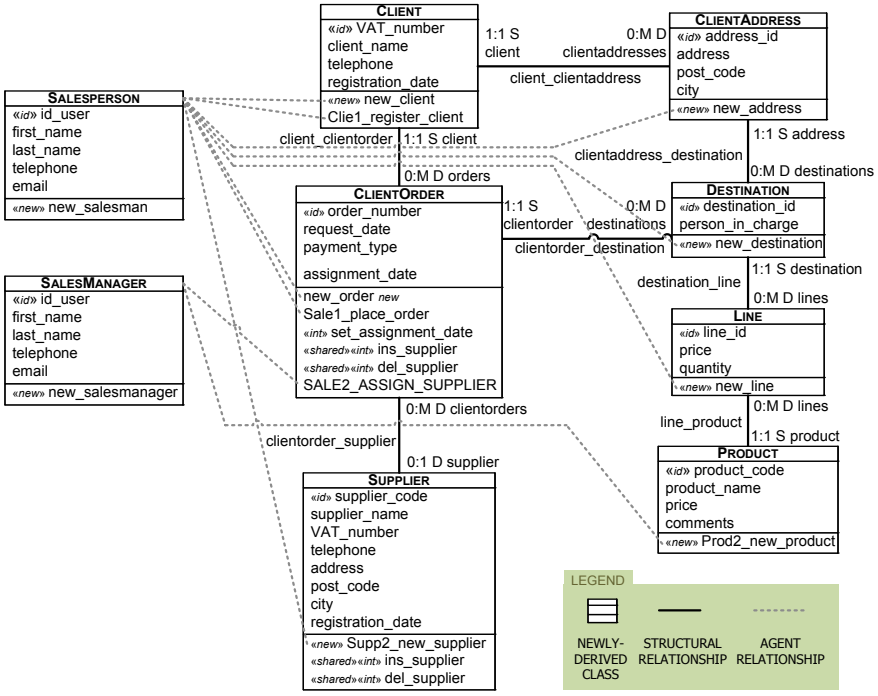


Figure 107. Agents and agent relationships are derived from organisational roles playing the interface role in events

Rule OM21. Creation of an agent of the whole system	
Pattern matching	
1	The requirements model.
Preconditions	
1	Rules **OM19** and **OM20** have been applied to all communicative events.
Rule steps	
1	Create a new class.
2	Assign to the name of the class Administrator.
3	Designate the class as agent of the whole system.
4	Add a creation service to the class.
5	The analyst can freely decide the name of the creation service, although it is recommended to give it the name new_class_name (i.e., the class name with the prefix new_).
6	For each of the attributes that are considered necessary for a computerised information system user (see Table 38 for a suggested set of attributes and their properties) do...
7	Add the attribute and set its properties.
8	For each attribute that has been added to the class during the application of this rule do...
9	Add a data-valued inbound argument to the creation service.
10	The analyst can freely decide the name of the data-valued inbound argument, although it is recommended to give it the name p_atrdatafield (i. e., the data field name with the prefix p_atr).
11	The properties of the data-valued inbound arguments coincide with the properties of their corresponding attributes. This means that the data type, size and compulsoriness are the same that were specified for the attributes.
Notes	
1	Creating the agent relationships manually for the agent of the whole system is unmanageable. Therefore, when the OO-Method conceptual model is created without a proper tool support, simply add an annotation or a (preferably) a stereotype to the class. Instead, in Integranova Modeler, defining a class as agent of the whole system is easily done with a contextual menu.

5.5.2 Derivation of the Functional Model

As discussed above, creation services do not need to have valuations defined because their semantics is implicitly defined in the OO-Method, as well as in the Integranova Modeler. However, editing services that have been derived as a result of a class extension (see rule **OM16**) do need valuation rules to be defined in order to specify the semantics of the service.

Since each data field of an extension substructure has led to the derivation of an inbound argument, now rule **FM1** creates a valuation rule for each data field. Each valuation rule takes the value of an argument and uses this value to initialise its corresponding attribute.

Rule FM1. Derivation of valuation rules from the data fields of an extension substructure	
Pattern matching	
1	Each aggregation substructure that *is* an extension substructure.
Preconditions	
1	Rule **OM16** has already been applied to the aggregation structure.
Rule steps	
1	For each data field contained in the aggregation substructure do...
2	Add a valuation rule to the functional mode.
3	The service to which the valuation rule refers is the editing service resulting from the class extension; to obtain it, simply follow the trace link of type Extension_Service_Derivation from the aggregation substructure.
4	The attribute of the valuation rule is the attribute that corresponds to the data field; simply follow the trace link of type Attribute_Derivation from the data field.
5	The valuation rule is defined of category *State*.
6	The valuation condition is left empty.
7	The formula that corresponds to the event effect simply contains the name of the argument that corresponds to the data field. To obtain the argument follow the trace link of type Argument_Derivation from the data field.

When this rule is applied to the message structure of communicative event SALE 2, it derives the valuation rules of the service set_assignment_date. In this case, there is a single attribute that needs to take value. The valuation rule is shown in Table 39.

Table 39. Valuation rule of the service set_assignment_date

Service	Category	Attribute	Effect	Condition
set_assignment_date	State	assignment_date	pt_assignment_date	-

When this rule is applied to the message structure of communicative event *SALE 3*, it derives the valuation rules of the service Sale3_evaluate. In this case, there are several attributes that need to be set a value, as shown in Table 39.

Table 40. Valuation rule of the service Sale3_evaluate

Service	Category	Attribute	Effect	Cond.
Sale3_evaluate	State	decision	pt_decision	-
Sale3_evaluate	State	planned_delivery_date	pt_planned_delivery_date	-
Sale3_evaluate	State	response_date	pt_response_date	-

Rule FM2. Definition of a transaction formula for complex reactions	
Pattern matching	
1	Each aggregation substructure that *is* an extension substructure.
Preconditions	
1	Rule **OM17** has already been applied to the aggregation substructure.
Rule steps	
1	Obtain a list of all the services that need to be encapsulated in the transaction. For this purpose, follow the trace links of type Extension_Service_Derivation and Shared_Service_Derivation (but consider only those of type "Insert" or "Change", not those of type "Delete").
2	Obtain the class that corresponds to the aggregation substructure by following the trace link of type Class_Extension.
3	For each of the services in the list do...
4	Add a call to the service in the transaction formula. To do this, write the name of the service, followed by an opening parenthesis.
5	For each argument of the service do...
6	If the name of the argument is p_thisClassName (i.e. the name of the class to which the transaction is being added prefixed with p_this) then...
7	Write THIS in the transaction formula (it is a reserved word that refers to the instance for which the transaction is triggered).
	Else...
8	Write the name of the argument in the transaction formula.
9	If it is not the last argument, then write a comma in the formula.
10	Write a closing parenthesis in the transaction formula.
11	If it is not the last argument write a period.

Notes	
1	If the analyst wants to avoid having to define each argument of the transaction in rule **OM17**, then s/he should use the wizard for adding services to the transaction formula and mark the checkbox "Add arguments to definition". This supersedes steps 4 to 10 of rule **FM2**.

As discussed above, in order to preserve data integrity, the transaction named S2_ASSIGN_SUPPLIER should execute atomically the services set_assignment_date and ins_supplier. The transaction formula derived for S2_ASSIGN_SUPPLIER according to rule **FM2** is shown in Table 41:

Table 41. Transaction formula of S2_ASSIGN_SUPPLIER

Transaction formula
ins_supplier(THIS, pt_p_evcSupplier) . set_assignment_date(THIS, pt_assignment_date)

5.5.3 Derivation of the Dynamic Model

In this section, we provide the derivation guidelines intended to create a first version of the OO-Method Dynamic Model (i.e. the collection of state transition diagrams) by means of processing the requirements model. Broadly speaking, communicative events derive transitions and precedence relations derive states.

The first step is to obtain, for each class in the Object Model, a communicative event diagram that contains only the communicative events that affect the given class (see Rule **DM1**). Also, specialised communicative events are flattened and loopbacks are identified. We refer to this diagram as the class-related encapsulation diagram (where class should be the name of the class). We use the generic term encapsulation to refer to communicative events and to event variants.

The derivation of the Object Model from the requirements model produces traceability links that specify which classes have been created or extended by which communicative events (see Table 42). The communicative events that should be included in such diagram are selected by following the Class_Derivation and Class_Extension trace links from the class back to the communicative events.

The loopbacks need to be marked. This is known as the feedback arc set problem and there exist numerous algorithms for this purpose (e.g. [Saab 2001]); however, for most business process models, we expect that analyst will be able to easily spot the loopbacks without applying any graph algorithm.

When flattening the diagram, some precedence relations are duplicated.

Rule DM1. Production of the class-related encapsulation diagram	
Pattern matching	
1	Each class in the class diagram.
Preconditions	
1	The Object Model has already been created (i.e. all rules related to the derivation of the class diagram have already been applied).
Rule steps	
1	Create a new communicative event diagram. We will refer to it as the *class-related encapsulation diagram* (in a specific diagram, *class* should be replaced by the class name).
2	Add to the class-related encapsulation diagram every communicative event that, being part of the transformation-scope diagram, affects the class; that is, each communicative event selected for transformation for which there is a trace link of type Class_Derivation or one of type Class_Extension between the event and the class. Add to the class-related encapsulation diagram those precedence relationships that depart from and arrive to communicative events that have been included in the class-related encapsulation diagram.
3	Identify the loopback precedence relationships of the class-related encapsulation diagram (if any) and mark them as loopbacks.
4	Flatten specialised communicative events of the class-related encapsulation diagram in a way that, instead the specialised communicative events, only the event variants are kept in the diagram. Each event variant should inherit the precedence relations of its outer encapsulations (both the incoming and the outgoing precedence relations).

The flattening of specialised communicative events implies that precedence relations have to be rearranged. Figure 108 shows the flattened version of the diagram in Figure 57 (supposing that all events affect a given class and that the events *ALIE 6* to *ALIE 11* have no other incoming precedence relations). Note that incoming precedences that arrive at an outer encapsulation (e.g. the relation from *ALIE 4* to *ALIE 5*) are multiplied, one arriving at each leaf variant (e.g. one relation from *ALIE 4* to *ALIE 5.1*, one from *ALIE 4* to *ALIE 5.2.1* and another from *ALIE 4* to *ALIE 5.2.2*). These precedences are combined via an and-join with the rest of precedences that the variants had, although the and-join is often left implicit (e.g. the precedences arriving at *ALIE 5.1* specify that, in order to occur *ALIE 5.1*, *SPAC 6* and *ALIE 4* have to occur first). Outgoing precedences that depart from an outer encapsulation (e.g. the precedence from *ALIE 5.2* to *ALIE 11*) are also multiplied, one departing from each leaf variant (e.g. one relation from *ALIE 5.2.1* to *ALIE 11* and another from *ALIE 5.2.2* to *ALIE 11*). These precedences are combined via an or-merge with the rest of precedences that the successor had (e.g. the precedences

arriving at *ALIE 11* specify that, in order to occur *ALIE 11*, either *ALIE 5.2.1* or *ALIE 5.2.2* have to occur first).

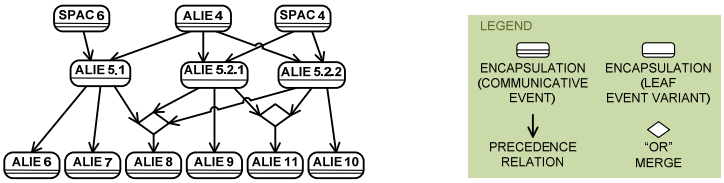


Figure 108. Flattened version of the diagram in Figure 57

Following the SuperStationery case, the transformation-scope diagram is shown in Figure 99 and the derived class diagram is shown in Figure 106. We will focus on the derivation of the state-transition diagram for class CLIENTORDER, since its dynamics is more complex that other that of other classes. First, a communicative event diagram that exclusively contains the events that affect class CLIENTORDER is obtained. Figure 109 offers a view of how the communicative events are selected by following the Class_Derivation and Class_Extension trace links from class CLIENTORDER.

Figure 110 shows the CLIENTORDER-related encapsulation diagram that is obtained by flattening the specialised communicative events. Note that the precedence relation from *SALE 2* to *SALE 3* has been duplicated: one relation going from *SALE 2* to *SALE 3.1* and another one from *SALE 2* to *SALE 3.2*. Also, the precedence relation going from *SALE 3.1* to *SALE 2* has been marked as a loopback.

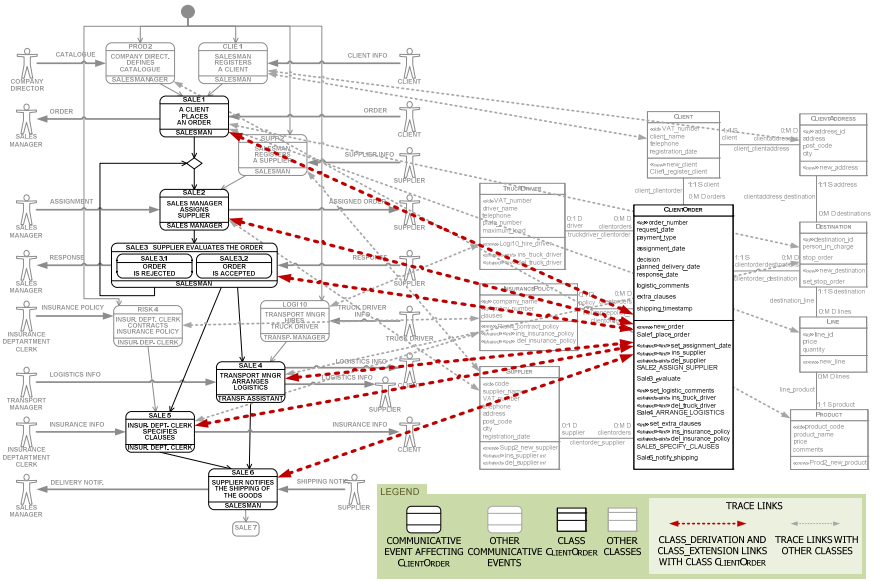


Figure 109. Selection of the communicative events that affect class CLIENTORDER

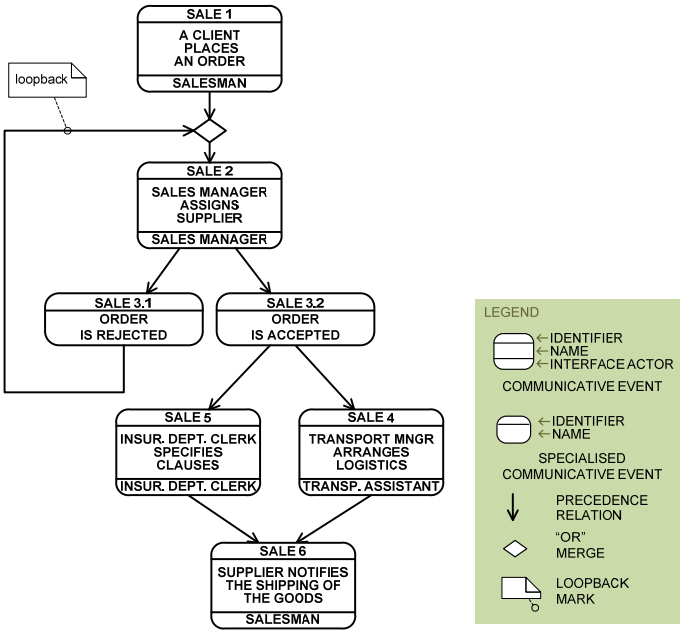


Figure 110. CLIENTORDER-related encapsulation diagram

Then an empty state-transition diagram is created and a pre-creation state is added (see rule **DM2**).

Rule DM2. Initialisation of the state-transition diagram	
Pattern matching	
1	Each class-related event diagram.
Preconditions	
1	None
Rule steps	
1	Create an empty state-transition diagram.
2	Add a pre-creation state to the state-transition diagram.

Rules **DM3** and **DM4** are the main derivation guidelines that produce intermediate states and transitions. Intuitively, rule **DM3** transforms encapsulations into states. For each encapsulation, the analyst creates a state in which a class affected by the event remains after the event occurrence; we refer to this state as the post-state. Precedent encapsulations also have left the class in a given set of states from which the current encapsulation can be allowed to occur; we refer as these pre-states. Therefore, one or several transitions are created to represent the encapsulation occurring in the different pre-states and arriving to the post-state.

Each transition corresponds to the execution of a class service. Therefore, during the derivation, each newly-derived transition is associated to the class service defines the reaction of the information system to the communicative event. When the diagram is represented graphically, the service name appears in the transition label. To obtain the service, simply follow the trace link of type Reaction_Service from the communicative event.

Rule DM3. Transformation of an encapsulation	
Pattern matching	
1	An encapsulation of the class-related encapsulation diagram.
Preconditions	
1	All the precedent encapsulations have already been processed with rule **DM3**.
2	If two or more precedence relations arriving at the encapsulation are joined in an and-join, then the and-join has already been applied rule **DM4**.
Rule steps	
1	Obtain the *contact service* of the encapsulation. To do this, in case the encapsulation is a communicative event directly follow the trace link of type Contact_Service from the encapsulation In case the encapsulation is an

	event variant, then first obtain the communicative event to which the variant belongs and then follow the trace link of type Contact_Service from the communicative event.
2	Obtain the *reaction service* that corresponds to the encapsulation. To do this, in case the encapsulation is a communicative event directly follow the trace link of type Reaction_Service from the encapsulation. In case the encapsulation is an event variant, then first obtain the communicative event to which the variant belongs and then follow the trace link of type Reaction_Service from the communicative event.
3	If the contact service and the reaction service are not the same service, then...
4	Add a state to the state-transition diagram. We refer to this state as the *editing state*.
5	Name the editing state Editing_EventId (i.e. Editing_ followed by the identifier of the encapsulation).
6	Add a state to the state-transition diagram. We refer to this state as the *post-state*.
7	Name the post-state with the identifier of the encapsulation plus the suffix –ed.
8	Create an empty list of states; we will refer to this list as the *pre-state list*.
9	Obtain a list of the encapsulations that are precedents of the current encapsulation, but do not follow the loopbacks. We refer to this list as the *precedence list*.
10	If the precedence list is empty (i.e. the current encapsulation does not have any precedent encapsulation) then...
11	Insert the pre-creation state of the state-transition diagram into the pre-state list.
	Else...
12	For each and-join joining precedence relations that arrive at the current encapsulation, do...
13	Process the and-join (see rule **DM4**, it returns a state we refer to as *joint-state*) and insert the resulting joint-state to the pre-state list.
14	For each precedent encapsulation in the precedence list do...
15	Obtain the states that correspond to the precedent encapsulation. For this purpose, follow the links of type State_Derivation from the precedent encapsulation. Add these states to the pre-state list.
16	For each of the pre-states in the pre-state list do...
17	If the contact service and the reaction service are not the same service, then...
18	Add a transition to the state-transition diagram; the transition departs from the pre-state and it arrives to the editing state.
19	Associate the transition to the contact service.

	Else...
20	Add a transition to the state-transition diagram; the transition departs from the pre-state and it arrives to the post-state.
21	Associate the transition to the reaction service.
22	If the current encapsulation is an event variant, then...
23	Add a control condition to the transition; the condition is derived from the specialisation condition that corresponds to the event variant (instead of message structure fields, their corresponding class arguments are used).
24	For each agent class of the reaction service (the agents are obtained by following the trace link of type Agent), do...
25	Add the agent class to the list of agents of the transition.
26	If the contact and the reaction services are not the same service, then...
27	Add a transition to the state-transition diagram; the transition departs from the editing state and it arrives to the post-state.
28	Associate the transition to the reaction service.
29	For each agent class of the reaction service (the agents are obtained by following the trace link of type Agent), do...
30	Add the agent class to the list of agents of the transition.
Traceability information	
1	Create a trace link of type Post_State_Derivation between the encapsulation and the post-state (this can be done after step 6).
2	In case the encapsulation is a variant, then create also a trace link of type Post_State_Derivation between its corresponding communicative event and the post-state (this can be done after step 6).
	In case an editing state has been derived, then create a Editing_State_Derivation between the encapsulation and the editing state (this can be done after step 4).
2	In case the encapsulation is a variant, then create also a trace link of type Editing_State_Derivation between its corresponding communicative event and the editing state (this can be done after step 4).
3	For each of the newly-added transitions create a trace link of type Transition_Derivation between the encapsulation and the transition (this can be done after steps 18, 20 and 27).
4	In case the encapsulation is a variant, then create a trace link of type Transition_Derivation between its corresponding communicative event and the transition (this can be done after steps 18, 20 and 27).
Notes	
We suggest a naming convention for the states, however, the analyst can choose more significant names for them.	

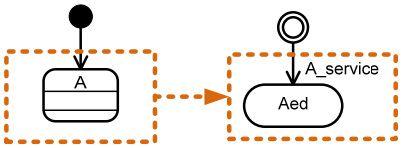


Figure 111. A locally initiatory communicative event leads to a transition departing from the pre-creation state

Rule **DM3** accounts locally initiatory encapsulations; that is, communicative events or event variants that do not have any precedences in the class-related encapsulation diagram. These encapsulations lead to transitions departing from the pre-creation state (see an illustrative example in Figure 111; the reaction service of event A is assumed to be named A_service).

When rule **DM3** is applied to encapsulations that have precedence relations combined in an or-merge, one transition is derived for each precedent encapsulation (see an illustrative example in Figure 112). The transitions depart from the states related to the precedent encapsulations and arrive at the state related to the current encapsulation. All of the derived transitions are mapped to the reaction service of the communicative event.

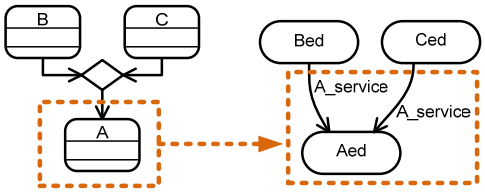


Figure 112. Or-merges imply deriving several transitions

Rule **DM3** also accounts for specialised communicative events. These events usually have departing precedence relations that take divergent paths in the business process (see an illustrative example in Figure 113). Similarly, the processing of such events leads to defining different paths in the state-transition diagram. All of the derived transitions are mapped to the reaction service of the specialised communicative event, but each transition has a different control condition that is derived from the specialisation condition of the corresponding event variant. According to the OO-Method, when the state-transition diagram is represented graphically, both the service name and the condition appear in the transition label: service_name when control_condition.

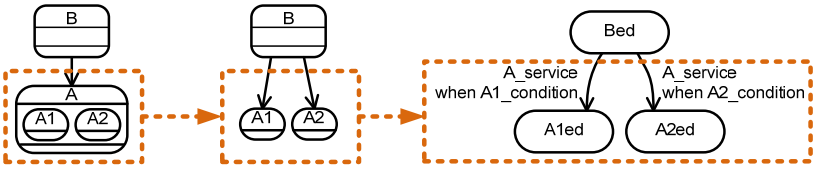


Figure 113. Example of transformation of a specialised communicative event

Rule **DM4** accounts for and-joins. This implies the creation of auxiliary states that constitute a logical network. One of the auxiliary states is returned as the pre-state to be used in rule **DM3**, step 13.

Rule DM4. Transformation of an and-join	
Pattern matching	
1	An and-join of the class-related encapsulation diagram (even if implicit) that joins two or more precedence relations. We refer to the encapsulations that are the source of these relations as *precedent encapsulations*.
Preconditions	
1	All the precedent encapsulations have already been applied rule **DM3**.
Rule steps	
1	Enumerate all the possible permutations of precedent encapsulations as columns in a matrix. In each row an encapsulation is appended in a way that each matrix cell contains the set of encapsulations that have occurred in that particular step of the sequence.
2	Links from one cell to another indicate the encapsulation that occurs. Given a link, we refer to the cell from where it departs as the *source cell* and to the cell where it arrives as the *target cell*.
3	Simplify the matrix is by merging those cells that contain an identical set of encapsulations (i.e. disregarding the order in which they appear). The corresponding links are redirected to the new target cell (the cell that has been merged). The cells that are merged necessarily belong to the same row. The last row will always be simplified to a single cell.
4	For each cell in the matrix do derive one state in the state-transition diagram (if the state does not exist yet).
5	Add a state to the state-transition diagram.
6	To name the state, we suggest adding the suffix –ed to the identifier of each of the encapsulations included in the cell and concatenating them. Remove any spaces and, optionally, insert an underscore between the identifiers.
7	For each link between cells in the matrix do...
8	Add a transition to the state-transition diagram; the transition departs from the state that corresponds to the source cell of the link to the state

	that corresponds to the target cell of the link (if the transition does not exist yet).
9	Associate the transition to the reaction service that corresponds to the encapsulation in the link. To do this, in case the encapsulation is a communicative event directly follow the trace link of type Reaction_Service. In case the encapsulation is an event variant, then first obtain the communicative event to which the variant belongs and then follow the trace link of type Reaction_Service from the communicative event.
10	If the encapsulation associated to the link is an event variant, then...
11	Add a control condition to the transition; the condition is derived from the specialisation condition that corresponds to the event variant (instead of message structure fields, their corresponding class arguments are used).
12	For each agent class of the reaction service (the agents are obtained by following the trace link of type Agent), do...
13	Add the agent class to the list of agents of the transition.
14	Return the state that corresponds to the last row of the matrix. We refer to this state as the *joint-state*.
Traceability information	
1	For each encapsulation in the cell, create a trace link of type Post_State_Derivation between the encapsulation and the newly-derived state (this can be done after step 5).
2	In case the encapsulation is a variant, then create a trace link of type Post_State_Derivation between its corresponding communicative event and the post-state (this can be done after step 5).
3	For each of the newly-added transitions create a trace link of type Transition_Derivation between the encapsulation and the transition (this can be done after step 9).
4	In case the encapsulation is a variant, then create also a trace link of type Transition_Derivation between its corresponding communicative event and the transition (this can be done after step 9).

The OO-Method uses Harel Statecharts, which allow for concurrency. However, the Integranova Dynamic Model does not allow for concurrency. This implies that, whenever there is an and-join, the analyst needs to do take one of the following approaches:

1. Creating additional states and transitions that represent possible permutations of occurrences (the approach chosen in **DM4**, see illustrative examples in Figure 114 and Figure 116).
2. Creating additional transitions, auxiliary attributes, extra valuations in the reaction services and control conditions to some transitions (see illustrative examples in Figure 115 and Figure 118).

The problem with the first approach is that, with and-joins joining many precedence relations, a state explosion occurs. However, most of the times, the behaviour of a class will not involve complex and-joins.

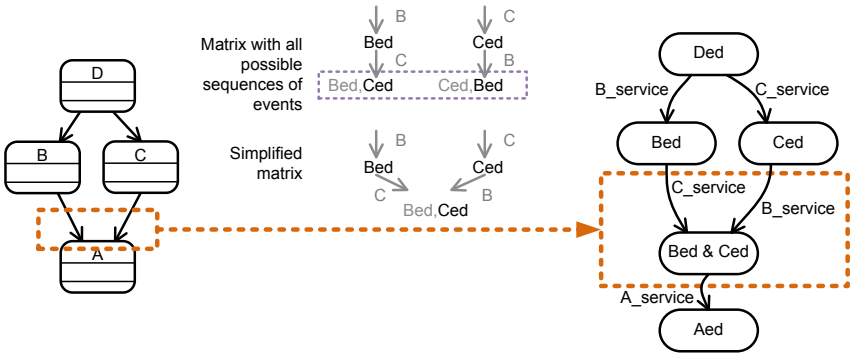


Figure 114. Example of transformation of an and-join joining two precedences, according to rule **DM4**

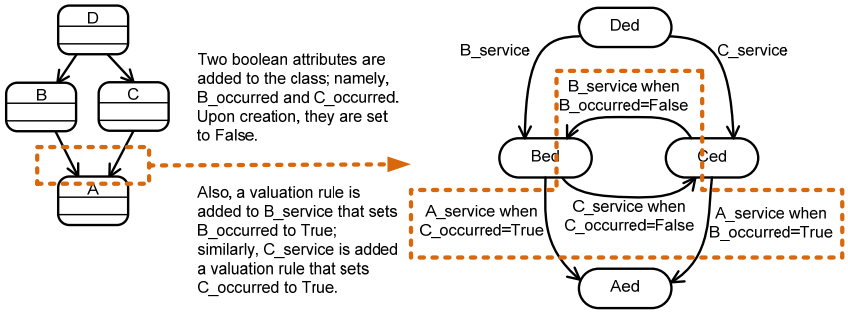


Figure 115. Another possible transformation of an and-join joining two precedences

The problem with the second approach is that, with and-joins joining many precedence relations, a transition explosion occurs. The subgraph that corresponds to events involved in the and-join is actually a complete directed graph; this means that an and-join joining N precedences will lead to the derivation of N(N-1) transitions.

Actually, both approaches are inconvenient, but we have chosen the first one for it leads to a state-transition diagram in which it is easier to perceive that all the events involved in the and-join need to occur. Compare the equivalent state-transition diagrams in Figure 116 and Figure 118.

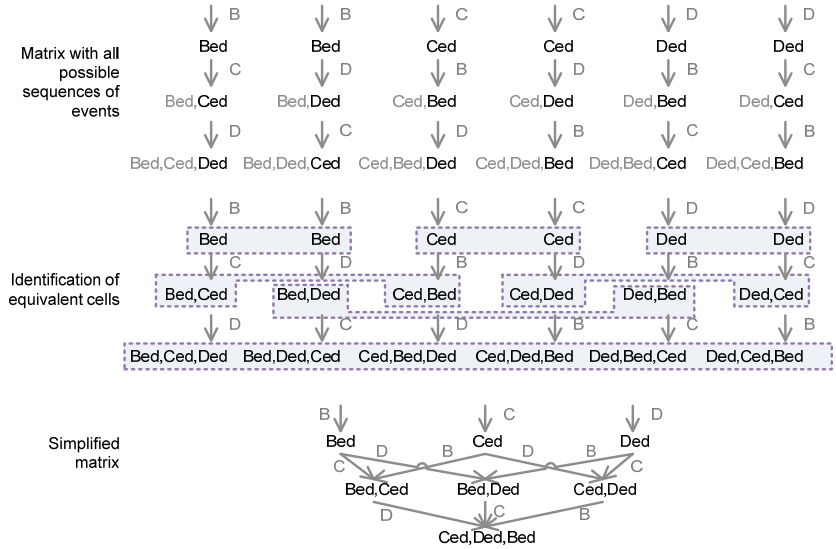


Figure 116. Simplification of the matrix corresponding to Figure 117

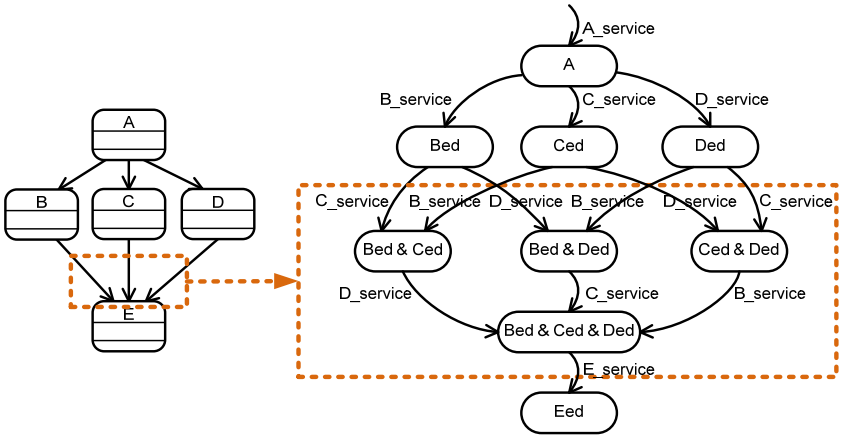


Figure 117. Example of transformation of an and-join joining three precedences, according to rule **DM4**

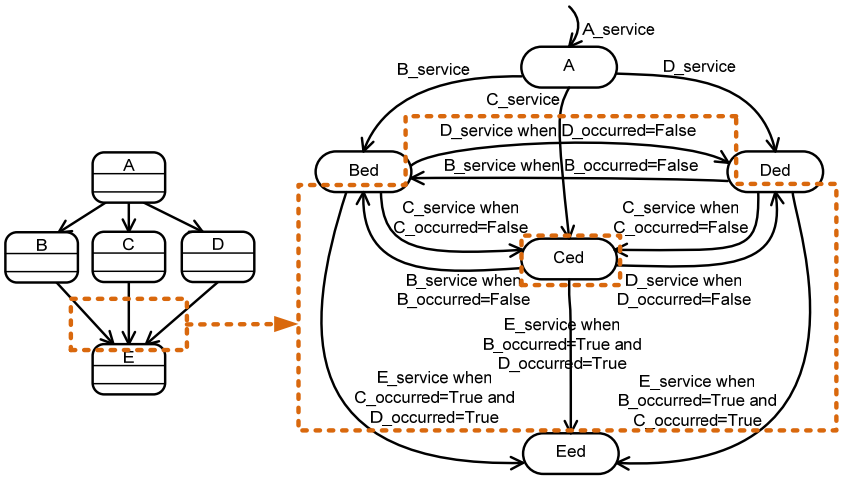


Figure 118. Another possible transformation of an and-join joining three precedences

In the first stage of the derivation of the state-transition diagram, processing loopbacks is avoided because considering them as a precedence relationship leads to rule precondition deadlocks. Once all the encapsulations i the class-related encapsulation diagram have been processed by applying rules **DM3** and **DM4**, loopbacks can be processed. Rule **DM5** derives (at least) one transition for each loopback. The transition corresponds to the target encapsulation of the loopback.

Rule DM5. Transformation of loopbacks	
Pattern matching	
1	A loopback of the class-related encapsulation diagram; that is, a precedence relation that relates an encapsulation back with one of its precedent encapsulations. We refer to the encapsulation from where the precedence relation departs as the *source encapsulation* and to the encapsulation at which it arrives as the *target encapsulation*.
Preconditions	
1	Both the source and the target encapsulations of the loopback have already been applied rule **DM3**.
Rule steps	
1	Obtain the *reaction service* that corresponds to the target encapsulation. To do this, in case the encapsulation is a communicative event directly follow the trace link of type Reaction_Service. In case the encapsulation is an event variant, then first obtain the communicative event to which the variant belongs and then follow the trace link of type Reaction_Service from the communicative event.
2	Obtain a list with the states of the state transition diagram that correspond to the source encapsulation. For this purpose, follow the trace links Post_State_Derivation from the source encapsulation. We will refer to this list as the *pre-state list*.
3	Obtain the states of the state transition diagram that correspond to the target encapsulation. For this purpose, follow the trace links Post_State_Derivation from the target encapsulation. We will refer to this list as the *post-state list*.
4	For each of the pre-states in the pre-state list do...
5	For each of the post-states in the post-state list do...
6	Add a transition to the state-transition diagram; the transition departs from the pre-state and it arrives at the post-state.
7	Associate the transition to the reaction service.
8	If the target encapsulation is an event variant, then...
9	Add a control condition to the transition; the condition is derived from the specialisation condition that corresponds to the event variant (instead of message structure fields, their corresponding class arguments are used).
10	For each agent class of the reaction service (the agents are obtained by following the trace link of type Agent), do...
11	Add the agent class to the list of agents of the transition.
Traceability information	
1	For each of the newly-added transitions create a trace link of type Transition_Derivation between the target encapsulation and the transition

2	(this can be done after step 7). In case the target encapsulation is a variant, then create also a trace link of type Transition_Derivation between its corresponding communicative event and the transition (this can be done after step 8).

Figure 119 shows an illustrative example of a loopback departing from an event variant named *C2* (the source encapsulation) and arriving at a communicative event named *B* (the target encapsulation). Rule **DM5** derives a transition that departs from the post-state corresponding to *C2* (i.e. the state C2ed), and arrives at the post-state corresponding to *B* (i.e. the state Bed). As explained above, the derivation of the transition is possible because both states had been already derived. The service associated to the transition is the reaction service corresponding to the target encapsulation (i.e. the class service that triggers toe information system reaction of the communicative event *B* is assumed to be named B_service).

Rule **DM6** is only necessary when the analyst is manually the model using Integranova Modeler. Whenever there is a transition related to a transaction, for each service that is invoked within the transaction formula, another transition needs to be created (departing from and arriving at the same states as the transition related to the transaction). The inclusion of these transitions is not mandatory according to the OO-Method, but they are prescribed by the Integranova Modeler.

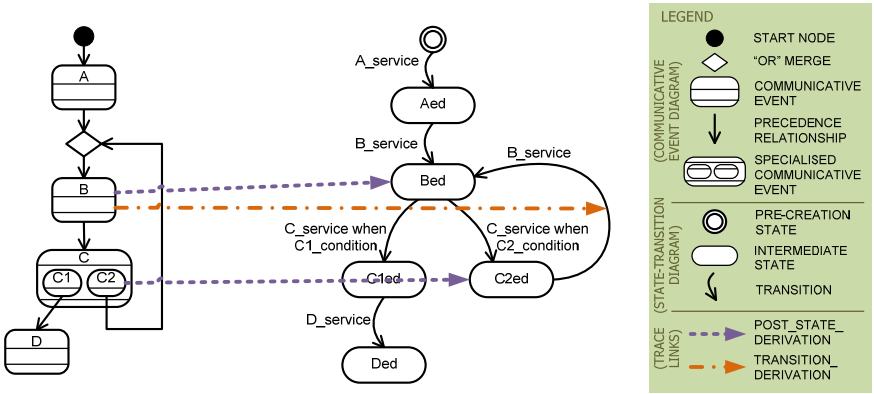


Figure 119. Transformation of a loopback into a transition

Rule DM6. Addition of atomic-service transitions to transaction transitions	
Pattern matching	
1	A transition of the state-transition diagram whose associated service is a transaction.
Preconditions	
	None
Rule steps	
1	Obtain the state from which the transition departs. We refer to this state as the *pre-state*.
2	Obtain the state at which the transition arrives. We refer to this state as the *post-state*.
3	Analyse the transaction formula and create a set of services that are included within the formula (i.e. services that are invoked by the transaction).
4	For each invoked service of the set do...
5	Add a transition to the state-transition diagram; the transition departs from the pre-state and it arrives at the post-state.
6	Associate the transition to the invoked service.
Traceability information	
1	Obtain the encapsulation that corresponds to the transaction transition. To do this, follow the trace link of type Post_State_Derivation from the transition back to an encapsulation (this can be done after step 2).
	For each of the newly-added transitions do...
2	Create a trace link of type Transition_Derivation between the encapsulation and the transition (this can be done after step 6).
Notes	
1	This rule needs only be applied in case the analyst is using Integranova Modeler to create the conceptual model.

With regards to the SuperStationery case, in the following, we describe how the rules are applied to the transformation-scope diagram in Figure 99 to derive the state-transition diagram for the class CLIENTORDER (see Figure 120). We choose this class because it has a more complex dynamics than the other classes.

As a result of applying rule **DM1**, the CLIENTORDER-related encapsulation diagram in Figure 110 is obtained; it contains the encapsulations *SALE 1*, *SALE 2*, *SALE 3.1*, *SALE 3.2*, *SALE 4*, *SALE 5* and *SALE 6*. Then, as a result of applying rule **DM2**, the state-transition diagram is created and the pre-creation event is added.

When applying rule **DM3** to communicative event *SALE 1*, the condition in step 3 holds: *SALE 1* has led to the derivation of a creation service (new_order, which acts as a contact service) and an end-of-editing service (Sale1_place_order, which

acts as a reaction service). Therefore, an editing state named Editing has been created (steps 4 to 5)and a transition associated to new_order departs from the pre-creation state and arrives at the state Editing (steps 18 to 19).Then, state Placed is added as a result of applying rule **DM3** to communicative event *SALE 1*.

State Assigned is added as a result of applying rule **DM3** to communicative event *SALE 2*. And a transition associated to SALE2_ASSIGN_SUPPLIER (the reaction service of *SALE 2*) is created from state Editing to state Assigned. The transitions that correspond to atomic services that are involved in the transaction (e.g. set_assignment_date and ins_supplier are part of the transaction formula, which is shown in Table 41) have been added by applying rule **DM6**.

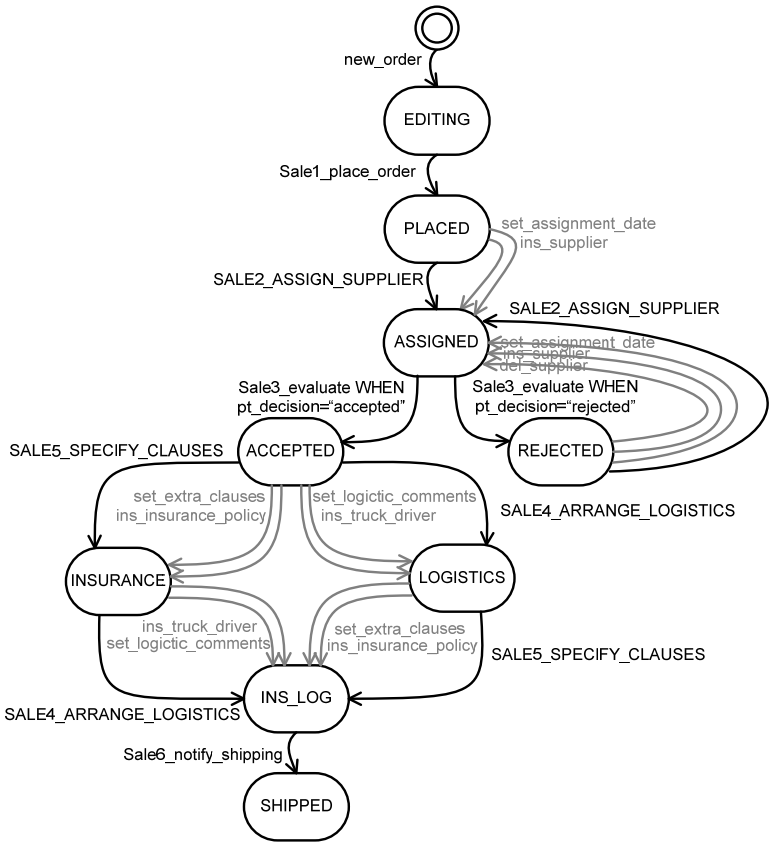


Figure 120. State-transition diagram of class CLIENTORDER

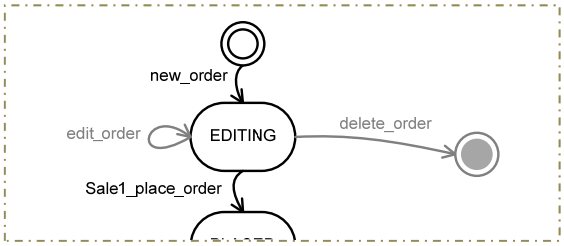


Figure 121. Fragment of Figure 120 that improves the editing of client orders

Since communicative event *SALE 3* is specialised, it was previously flattened in the CLIENTORDER-related encapsulation diagram. Thus, rule **DM3** is applied independently to the event variants *SALE 3.1* and *SALE 3.2*, leading to the derivation of states Accepted and Rejected, respectively. States Insuranc and Logistic are derived as a result of applying rule **DM3** to communicative events *SALE 5* and *SALE 4*, respectively. Before rule **DM3** can be applied to *SALE 6*, the implicit and-join must be processed by applying rule **DM4**: the auxiliary state Ins_Log corresponds to the state of an order that has been added both the insurance clauses and the logistic information.

Figure 120 also shows control conditions. For instance, the transition from state Assigned to state Accepted corresponds to the occurrence of event variant *SALE 3.2*. A control condition is defined (see rule **DM3**, steps 22 to 23); the original specialisation condition in the requirements model (i.e. Decision=accepted) is defined in terms of message structure fields, whereas the control condition is defined in terms of class attributes and allowed data-typed values (i.e. pt_decision="accepted") . Although the list of agents for each transaction is not shown, they have been derived accordingly. In the case of *SALE 3.2*, only the salesman can execute the service Sale3_evaluate, so class Salesman is the only agent allowed to trigger the transition.

Figure 120 shows in grey transitions that correspond to atomic services that take part in a transaction. These transitions are not mandatory according to the OO-Method but need to be included in the Integranova model if an effective code generation is intended.

Other transitions corresponding to edit_order and delete_order could be added manually, as well as the destruction state (see Figure 121). In any case, we do not provide for their derivation yet.

Table 42. Traceability relationships

Traceability relationship	Communication Analysis			OO-Method	Rules that create links
	Source class	Card.		Target class	
Class_Derivation	COMMUNICATIVE_EVENT	0:1	0:M	CLASS	**OM4**
Class_Derivation	AGGREGATION	0:1	0:1	CLASS	**OM4**
Agent	ORGANISATIONAL_ROLE	0:1	0:1	CLASS	**OM4, OM19**
Attribute_Derivation	COMMUNICATIVE_EVENT	0:1	0:M	ATTRIBUTE	**OM5**
Attribute_Derivation	DATA_FIELD	0:1	0:1	ATTRIBUTE	**OM5**
Identifier_Selection	TEXTUAL_REQUIREMENT	0:1	0:M	ATTRIBUTE	**OM6**
Structural_Relationship_ Derivation	COMMUNICATIVE_EVENT	0:M	0:M	AGGREGATION	**OM11, OM12**
Structural_Relationship_ Derivation	AGGREGATION	0:1	0:1	AGGREGATION	**OM11**
Cardinality_Derivation	TEXTUAL_REQUIREMENT	0:1	0:M	AGGREGATION	**OM11, OM12**
Shared_Service_Derivation	COMMUNICATIVE_EVENT	0:1	0:M	SERVICE	**OM11, OM12**
Shared_Service_Derivation	AGGREGATION	0:1	0:M	SERVICE	**OM11, OM12**
Shared_Service_Derivation	REFERENCE_FIELD	0:1	0:M	SERVICE	**OM12**
Structural_Relationship_Der vation	REFERENCE_FIELD	0:1	0:1	AGGREGATION	**OM12**
Referenced_Class	REFERENCE_FIELD	0:M	0:1	CLASS	**OM12**
Creation_Service_ Derivation	COMMUNICATIVE_EVENT	0:1	0:1	SERVICE	**OM13**
Creation_Service_ Derivation	AGGREGATION	0:1	0:1	SERVICE	**OM13**
Argument_Derivation	COMMUNICATIVE_EVENT	0:1	0:M	ARGUMENT	**OM13, OM16, OM17**
Argument_Derivation	DATA_FIELD	0:1	0:1	ARGUMENT	**OM13, OM16**
Argument_Derivation	REFERENCE_FIELD	0:1	0:1	ARGUMENT	**OM13**
End_Editing_Service_ Derivation	COMMUNICATIVE_EVENT	0:1	0:1	SERVICE	**OM14**
Class_Extension	COMMUNICATIVE_EVENT	0:M	0:M	CLASS	**OM15**
Class_Extension	AGGREGATION	0:M	0:M	CLASS	**OM15**
Class_Extension	REFERENCE_FIELD	0:M	0:1	CLASS	**OM15**
Extension_Service_ Derivation	COMMUNICATIVE_EVENT	0:1	0:M	SERVICE	**OM16**
Extension_Service_ Derivation	AGGREGATION	0:1	0:1	SERVICE	**OM16**
Transaction_Derivation	COMMUNICATIVE_EVENT	0:1	0:1	TRANSACTION	**OM17**
Transaction_Derivation	AGGREGATION	0:1	0:1	TRANSACTION	**OM17**
Reaction_Service	COMMUNICATIVE_EVENT	0:1	0:1	SERVICE	**OM18**
Contact_Service	COMMUNICATIVE_EVENT	0:1	0:1	SERVICE	**OM18**
Agent	COMMUNICATIVE_EVENT	0:M	0:M	CLASS	**OM20**
Post_State_Derivation	ENCAPSULATION	0:1	0:M	STATE	**DM3, DM4**
Editing_State_Derivation	ENCAPSULATION	0:1	0:1	STATE	**DM3, DM4**
Transition_Derivation	ENCAPSULATION	0:1	0:M	TRANSITION	**DM3, DM4, DM5**

5.5.4 Traceability relationships

During the derivation of the Object Model, the Functional Model and the Dynamic Model, elements of the Communication Analysis requirements model are mapped to elements of the OO-Method conceptual model. Table 42 presents the traceability relationships that have been defined, specifying which primitives of the requirements model are mapped to which primitives of the conceptual model (this is done by indicating their corresponding metaclasses of the platform independent metamodel presented in Section 4.5). Also, the cardinalities of each traceability relationship are specified. Last column indicates the rules that create trace links of each kind of traceability relationship.

Take into account that in the Communication Analysis metamodel, the metaclass AGGREGATION refers to the aggregation substructure, whereas in the OO-Method metamodel, the metaclass AGGREGATION refers to the structural relationship among classes.

5.6 Summary

This chapter has presented the integration of Communication Analysis and the OO-Method. For this purpose, we have defined a conceptual model derivation technique (see an overview in Section 5.4). By means of this technique, Communication Analysis requirements models can be processed in order to create an initial version of the OO-Method conceptual model. The derivation technique covers three of the four views of the OO-Method conceptual model; namely, the Object Model (an extended UML Class Diagram that specifies the information system memory), the Dynamic Model (a collection of UML State-Transition Diagrams that represents the valid lifecycles of objects and the interactions among them), and the Functional Model (a description of the reaction of the computerised information system using an abstract pseudocode). Given the complexity of deriving a Presentation Model (a model of the user interface by means of abstract patterns) from the requirements model, we have left this part of the derivation out of the scope of the thesis; we plan to address it in future works.

We have first defined the derivation guidelines in a way that a human analyst can apply them manually without tool support (see Section 5.5); the implementation of the model transformation rules that automate the derivation procedure is addressed in Section 8.3.

In Section 5.5.1 we have presented the derivation of the OO-Method Object Model from a Communication Analysis requirements model. The derivation technique offers a systematic way of tackling the derivation problem by sorting the communicative events and processing them in order. For each communicative

event, a class diagram view is derived. This is achieved by processing the message structure that corresponds to the communicative event. Each class diagram view constitutes a portion of the class diagram, so the class diagram is incrementally constructed by integrating the views. Agents (classes that represent users of the computerised information system under development) and agent relationships (permissions that allow the users to execute the class services) are also derived.

In Section 5.5.2 we have presented the derivation of the OO-Method Functional Model. This model contains valuation rules and transaction formulas that provide a precise semantics to class services. By means of the derivation rules, the attributes of the classes are set their value, and the invocation of several services can be encapsulated in a transaction to ensure their atomic execution. The derivation of services, their signatures and their reaction specification is an important contribution of this work. There are few approaches that produce this kind of output; however, most of them only produce creation, destruction and shared services (e.g. [Sendall and Strohmeier 2000]). A notable exception is [Albert, Cabot et al. 2011], which presents the generation of services and their specification taking as input extended UML class diagrams. Since our approach takes as input business process models, organisational behaviour that cannot be expressed in a class diagram can be taken into account in the derivation rules.

In Section 5.5.3 we have presented the derivation of the OO-Method Dynamic Model. While the Object Model is created, many traceability relationships are created; these relationships are used to obtain the communicative events that affect each class. A class-related encapsulation diagram is created and transformed into a state-transition diagram. Broadly-speaking, the derivation guidelines transform events into transitions and precedence relations into states. This way, the Dynamic Model constrains the possible life-cycles of an instance. The derivations provide enough information (structural and behavioural aspects of the system) to the conceptual model to allow for automatic code generation with the Integranova Model Compiler. Furthermore, since the Integranova Modeler offers the possibility offered of creating a default user interface, generated applications are fully functional (see Figure 122).

The validation of the derivation technique is discussed in Chapter 7. It will allow identifying the strengths and weaknesses of our approach.

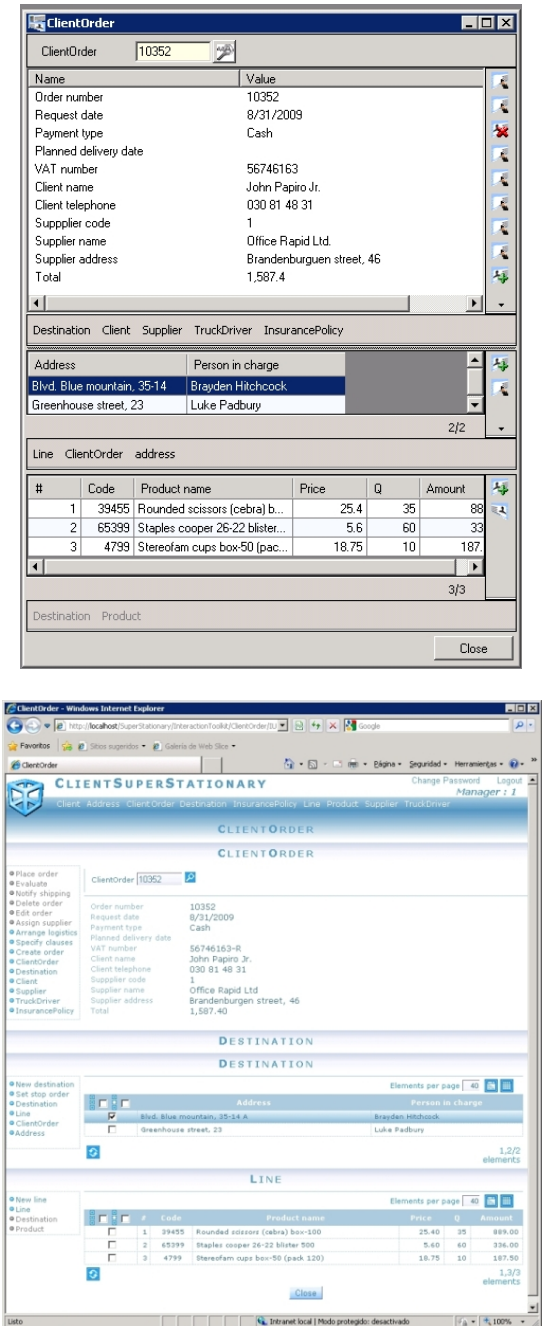


Figure 122. Snapshots of generated applications for desktop and web platforms

PART III.
SOLUTION VALIDATION

Chapter 6
Validation of Communication Analysis

"What counts is not what sounds plausible, not what we would like to believe, not what one or two witnesses claim, but only what is supported by hard evidence rigorously and skeptically examined."

Carl Sagan

6.1 Motivation

Requirements engineering (RE), in spite of being a relatively young discipline, has achieved an ever growing body of knowledge. Numerous RE methods have been proposed over the last decades. Also, it is widely acknowledged that requirements engineering has a big impact in software quality [Boehm, McClean et al. 1975]. However, most authors act as designers and propose new RE methods, while few authors act as real researchers validating that their (or other authors') proposals actually improve RE practice; although there is a growing concern for validating RE proposals, empirical evaluations are a strong need in the area [Wieringa and Heerkens 2004] [Wieringa and Heerkens 2006]. A grand challenge for software research is to develop an understanding of which software methods work better and why [Höfer and Tichy 2007], and it is likely that no standard method will suit all situations [Iivari and Kerola 1983]. There exist works that address comparisons of RE approaches based on industrial experience [Dieste, Lopez et al. 2008]. However, seldom do we see work intended to compare features of RE approaches within an experimental context.

In order to carry out method evaluations and comparisons, many evaluation techniques are available [Siau and Rossi 1998]. Some evaluation techniques are theoretical (e.g. ontological analysis) and others are empirical (e.g. laboratory

experiments, action research); each type having its strengths and weaknesses. For instance, laboratory experiments provide high level of control and they are powerful in determining causality (internal validity is their strength), while the artificial environment in which they are carried out compromises the generalisation of the results (external validity is their weakness) [Siau and Rossi 1998; Wohlin, Runeson et al. 2000].

This chapter presents several efforts that we have done with the intention of validating Communication Analysis. Section 6.2 presents a theoretical validation, based on an ontological analysis of the method. Section 6.3 discusses several lab demos; that is, manageable cases that we have solved using Communication Analysis. These cases are descriptions of fictional enterprises and their work practice. From these descriptions, we have created a Communication Analysis requirements model. The lab demos have allowed us to ascertain the feasibility of our proposals and have provided valuable feedback that has allowed us to refine the method specification. We have also used these cases as instrumentation for experiments and as teaching material. Section 6.4 describes an empirical framework that can be used to design experiments to investigate the impact of requirements modelling methods in the quality of the resulting models. We adopt (and adapt) the Method Evaluation Model (MEM), which defines a theoretical model and associated measurement instruments for evaluating information system design methods [Moody 2003]. The MEM has been combined with the framework for conceptual modelling quality by Lindland et al. [1994]. In turn, the quality framework has been extended with metrics for measuring model completeness and assessing model modularity. Section 6.5 describes a controlled experiment that evaluates and compares Communication Analysis and Use Cases [Cockburn 2000] with regards to the quality of the resulting requirements models. Strictly speaking, we compare the resulting use case diagrams and communicative event diagrams (and their corresponding textual descriptions) with regards to their completeness and modularity. Also, a perception-based comparison is carried out. Section 6.6 presents a summary of the chapter.

6.2 *Theoretical validation*

When it comes to theoretically validating and comparing software development methods, one comes across several problems. First of all, methods are often loosely defined, their definitions based on common sense and intuition. Also, there is a lack of widely-used standard techniques for evaluating methods. Ontological analysis offers strong theoretical foundations to analyse methods; it requires a reference ontology and a procedure to perform the evaluation.

In general, the evaluation procedure consists of establishing a mapping among the concepts of the ontology and the concepts of the modelling technique (often the modelling primitives or the metaclasses are employed). Then the evaluators assess to which extent the modelling technique covers the concepts of the ontology, and vice versa. In theory, the wider the coverage, the better. However, as seen below, ontological analyses usually involve a more complex investigation of the mappings and their outcomes are more detailed.

6.2.1 Related works

In [Wand and Weber 1990; Wand, Monarchi et al. 1995; Weber 1997; Weber 1999] the authors proposed the use of ontologies to evaluate information modelling methods. The basic idea is to evaluate the constructs of a modelling method (a.k.a. method concepts, modelling primitives) by matching them with the constructs of the ontology (a.k.a. ontological concepts). To do so, they extend Bunge's ontology and define the Bunge-Wand-Weber (BWW model). They introduced the notions of construct overload, construct redundancy, construct excess, and construct deficit [Wand and Weber 1993]. These notions are exemplified in Figure 123.

- *Construct overload*. A conceptual modelling grammar allows one modelling primitive to be used to represent more than one ontological concept.
- *Construct redundancy*. A conceptual modelling grammar allows one ontological concept to be represented by more than one modelling primitive.
- *Construct excess*. A conceptual modelling grammar has a modelling primitive that has no real-world counterpart; that is, it matches no ontological concept.
- *Construct deficit*. An ontological concept does not have any corresponding modelling primitive.

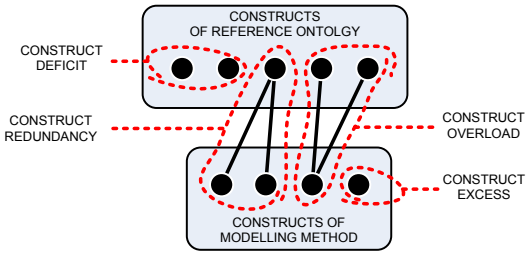


Figure 123. Ontological evaluation of a modelling technique

Many ontological analyses have been done based on the BWW model, mainly focused on object-oriented modelling languages [Opdahl and Henderson-Sellers 1999] [Opdahl, Henderson-Sellers et al. 2000] [Opdahl and Henderson-Sellers 2002].

Milton, Kazmierczak et al. [2001] propose Chisholm's ontology as a reference. Besides providing a mapping of constructs, their ontological analysis method also aims at providing qualitative information, which includes identifying partial supports for ontological constructs [Milton and Kazmierczak 2004].

Guizzardi, Pires et al. [2005] propose a framework for the evaluation and redesign of modelling languages. They define four properties (some of them related to the notions above) that a model should hold with respect to a domain abstraction; namely, lucidity, soundness, laconicity and completeness. An ontological evaluation of the UML is provided, using the Unified Foundational Ontology [Guizzardi 2005] as the reference ontology.

In [Pastor, González et al. 2007] and [Pastor, España et al. 2008] a procedure for ontological evaluation is proposed; it is referred to as conceptual alignment. It has been used to perform the ontological evaluation of the OO-Method, the Rational Unified Process [Rational 2003], and Wisdom [Nunes and Cunha 2000] using FRISCO as a reference ontology (the evaluations are compiled in [España 2008]).

6.2.2 Ontological evaluation of Communication Analysis

In the following we use FRISCO 1.1, the framework for information system concepts presented in Section 3.3, as the reference ontology. Table 43 provides a mapping between the constructs of Communication Analysis and the constructs of the FRISCO 1.1 ontology.

With regards to the constructs of Communication Analysis, we distinguish between meta-class (of the method metamodel) and concept (of the method); this provides an additional analysis of the extent to which the particular metamodel (one among the many possible supports to the method) covers the concepts of the method. We use as the reference column the Communication Analysis concept; the concepts appear in the same order as they are defined in Section 4.2. Additionally, the requirements layer to which the concept belongs is indicated. Note that some concepts are considered out of the scope of the Communication Analysis metamodel; these cases are explained below. Also note that, for the sake of brevity, we refer to some Communication Analysis concepts within the FRISCO 1.1 definitions; in such cases we use apostrophes (as Opdahl et al. do in [1999]). For instance, we refer as "business object" to a predicated thing which is not a predicator, a relationship or a state, and it is part of the domain model.

Table 43. Mapping among constructs of Communication Analysis and of the reference ontology

Communication Analysis construct			**FRISCO 1.1 construct**
Concept		**Metaclass**	**Concept**
L1	organisational system (adopted)	ORGANISATION	organisational system
	organisational unit	ORGANISATIONAL_UNIT	sub-system of the organisational system
	organisational location	ORGANISATIONAL_LOCATION	--
	organisational goal	GOAL	goal of an organisational system
	business indicator	INDICATOR	--
	organisational strategy	STRATEGY	composite thing mainly consisting of a set of operationalised goals
L2	organisational actor	ORGANISATIONAL_ACTOR	goal-pursuing actor
	organisational role	ORGANISATIONAL_ROLE	type of goal-pursuing actors
	business process	PROCESS	sub-system of the subject system, often focused on actions and actors
	communicative interaction	out of scope[a]	sequence of message transfers
	ingoing communicative interaction	INGOING	sequence of message transfers; related to the formulation of regulated transition information
	outgoing communicative interaction	OUTGOING	sequence of message transfers; related to the formulation of recalled facts
	communicative event unity criteria	out of scope[b]	set of rules which determine how achieve modular business process models (it corresponds to the concept of unity criteria of the conceptual framework for model modularity)
	communicative event	COMMUNICATIVE_EVENT	composite transition
	communicative event occurrence	out of scope[c]	transition occurrence
	event variant	EVENT_VARIANT	transition taking part of a state-transition structure of type choice that defines a communicative event

	primary role	PRIMARY	type of goal-pursuing actors, more specifically a set of regulated-transition observers
	primary actor	out of scope[c]	goal-pursuing actor acting as a regulated-transition observer
	receiver role	RECEIVER	type of goal-pursuing actors, more specifically a set of recalled facts receivers
	receiver actor	out of scope[c]	goal-pursuing actor acting as a recalled facts receiver
	precondition	not a metaclass, but an attribute of one: ENCAPSULATION.precondition	norm that determines permissible transition occurrences
	precedence relation, which is part of the precondition	PRECEDENCE	norm that determines permissible transition occurrences
	business object	out of scope[c]	predicated thing which is not a predicator, a relationship or a state, and it is part of the domain model
	business object class	BUSINESS_OBJECT_CLASS	a type of "business object"; it is part of the language of the domain model
L3	message structure	MESSAGE_STRUCTURE	type of messages; it is an input actand of a composite transition (i.e. the "communicative event")
	business object class property	BUSINESS_OBJECT_FIELD	predicator of "business objects"; it acts as a norm that is part of the language of the domain model
	business object representation	out of scope[c]	representation that is part of the domain model denotation
L4	support role	SUPPORT	type of goal-pursuing actors
	support actor	out of scope[c]	goal-pursuing actor

-- means that no mapping could be established

[a] Abstract metaclasses are out of the scope of this analysis

[b] Not a modelling primitive but a modelling guideline

[c] Not part of an information systems metamodel

We first investigate potential misalignments of the Communication Analysis ontology and the FRISCO 1.1 ontology, and then potential misalignments of the Communication Analysis ontology and the metamodel.

6.2.2.1 Potential construct excesses

There are some cases in which it was not possible to map a Communication Analysis concept to a FRISCO 1.1 concept (look for -- in the rightmost column of Table 43). We now investigate these potential construct excesses.

Organisational location

The Communication Analysis concept organisational location has no counterpart in FRISCO 1.1. However, this is due to the fact that FRISCO 1.1 does not include spatial concepts. However, it is widely acknowledged that such concepts are important for organisational modelling; organisational systems are often settled in specific locations that may be of interest in order to analyse communicational needs. Thus, we argue that it is not a case of construct excess.

Business indicator

The Communication Analysis concept business indicator has no counterpart in FRISCO 1.1. However this is due to the fact that FRISCO 1.1 does not include concepts related to metrology. However, business indicators are essential to monitor the performance or an organisational system. Moreover, they are closely related to information system development because indicators need to be analysed and designed in conjunction with the rest of the information system concepts. Indicators usually aggregate facts from the information system memory in such a way that the design of the memory influences the design of the indicators and vice versa. Thus, we argue that the construct is indeed needed and it is not a case of construct excess.

We advocate that the authors of the ontological analysis described in [Opdahl, Henderson-Sellers et al. 2000], under similar circumstances, considered that some constructs were not "truly excessive" when they are not meant to represent something within the problem domain covered by the reference ontology. In our case, the problem domain covered by FRISCO 1.1 are mainly information systems, whereas the method intends to also support the modelling of certain aspects of the organisation that are disregarded by the ontology. Whether it is an incompleteness of the ontology or not is a matter that would require further investigation.

6.2.2.2 Potential construct deficits

In case a FRISCO 1.1 concept did not have an associated Communication Analysis concept we would face potential construct deficit. We have omitted this interesting analysis setting our sight on the integration in an MDD framework. Ultimately, the aim is to integrate Communication Analysis and the OO-Method. As long as both methods are properly aligned, whether the OO-Method has construct deficits with regards to a reference ontology is out of the scope of this work. So the main purpose of the alignment above is to pave the way for the alignment with the OO-Method.

6.2.2.3 Potential construct overloads

There is no case in which a Communication Analysis concept is related to several FRISCO 1.1 concepts. Although several concepts may appear in the FRISCO 1.1 column, there is always a main one and the others just provide qualification or context to clarify the meaning. Therefore, there is no construct overload.

6.2.2.4 Potential construct redundancy

There are some cases in which a FRISCO 1.1 concept is related to several a Communication Analysis concepts. However, in most cases they are still essentially different concepts, built upon the same FRISCO 1.1 concept (a sort of specialisation of concepts).

Two types of communicative interactions

For instance ingoing communicative interaction and outgoing communicative interaction are both a sequence of message transfers but they differ in the communicative goal. In the former, the goal is to convey regulated transition information; that is, to provide new facts to the information system. In the latter, the goal is to obtain recalled facts; that is, to consult the information system memory. The same occurs with the concepts concerning roles and actors; it is by analysing the qualification included in the definition that we realise they are not redundant

Preconditions and precedence relations

The only case in which we have found an actual redundancy is between precondition and precedence relation. The precedence relation is actually part of the precondition so they are redundant ways of expressing the same thing; namely, that one communicative event must necessarily occur before another. At

the conceptual level, we do not see any inconvenient as long as this redundancy is clearly expressed to the method users (i.e. analysts).

6.2.2.5 Potential misalignment of the metamodel

When a Communication Analysis concept does not have an associated meta-class it is potentially a case of lack of support of the metamodel to the method.

In some cases, the Communication Analysis concept allows reasoning about other concepts but it does not belong to an information system metamodel. This implies that, although these concepts do not have an associated meta-class in the Communication Analysis metamodel, and that this should not be considered a lack of support (there is simply no need to support them).

Instances

It is the case of concepts related to particular instances, such as communicative event occurrence, primary actor, receiver actor, support actor, business object and business object representation. They all define conceptions; if they were to be included in the information system metamodel they would require using advanced metamodelling techniques such as powertyping. Figure 124 illustrates the point with the business object concept. The moment this concept is included in the metamodelling framework inconsistencies arise. These inconsistencies are due to the use of a strict metamodelling multi-level hierarchy. Instead, if powertypes are used, then box can become a clabject and the framework remains consistent. The use of powertypes can be considered as part of future work, but has been ruled out for the moment.

Note that the reasoning has been done with conceptions instead of meta-classes, to be faithful to the constructivist stance. Meta-classes are the representations of the conceptions. Sometimes two or more different representations can refer to the same conception. For instance, an organisational role can be expressed as a stereotyped class or as an element of an organisational chart. Similarly, a business object can be represented in several ways; in Figure 124, the domain model denotation contains two equivalent representations of the same concept (i.e. a class instance and a filing card).

Unity criteria

Another potential misalignment is the case of Communicative event unity criteria; it is a concept related to a set of methodological guidelines, a set of rules according to FRISCO 1.1. Thus, it does not correspond to an element of the metamodel (so no meta-class needs to be provided) but to the documentation that accompanies the metamodel. It should not be considered a lack of support.

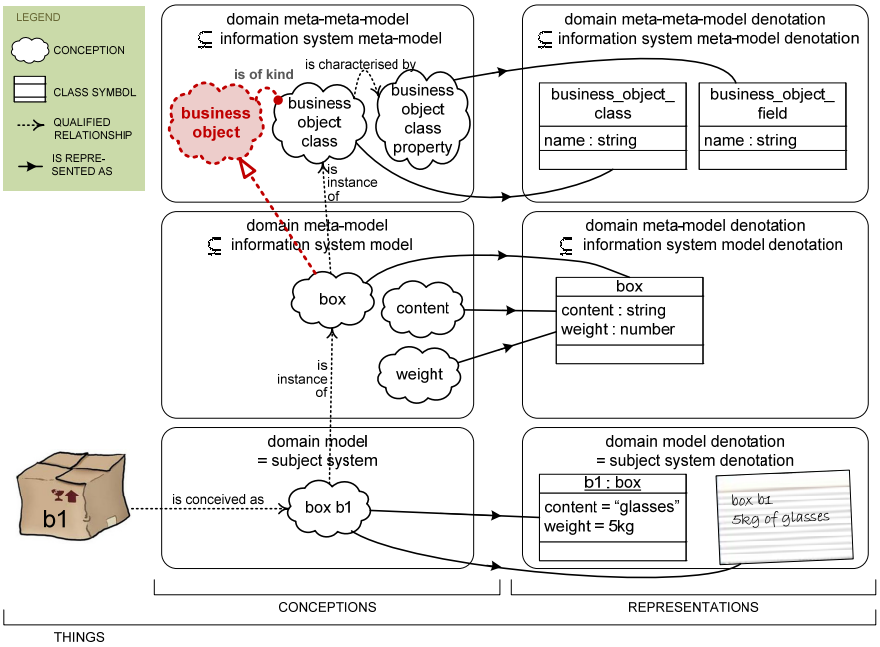


Figure 124. The need of powertypes in order to solve inconsistencies

Non-leaf metaclasses

In the cases where a specialisation hierarchy appears, we only consider the leaf meta-classes, for the sake of a simpler alignment. The non-leaf meta-classes are indeed abstract, which means that all their instances need to be specialised. This way, meta-classes such as ORGANISATIONAL_MODULE, ENCAPSULATION and COMMUNICATIVE_INTERACTION are disregarded (note, however, that COMMUNICATIVE_INTERACTION is aligned with the Communication Analysis concept communicative interaction and with the FRISCO 1.1 concept message transfer.

Preconditions and precedence relations

The redundancy related to preconditions identified above affects the metamodel as well. Even more, not only is it is the PRECEDENCE that is redundant with the ENCAPSULATION.precondition, but also the rest of the meta-classes related to specifying complex precedences (namely, OR, AND, END and START) and the EVENT_VARIANT.specialisat_cond (i.e. the specialisation condition of specialised communicative events). Figure 125 depicts this redundancy. All the elements in red could be removed and the metamodel would still support specifying all the necessary information about the business processes.

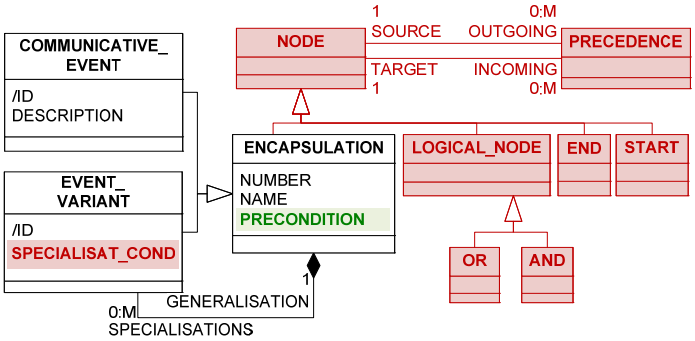


Figure 125. Redundancy related to the precondition of communicative events

We opt for keeping the metaclasses, since they will allow offering mechanisms for different specifying the preconditions that have different features. The precedence relations and logical nodes are mainly intended to be graphical, whereas the precondition is intended to be an algebraic formula that facilitates model verification. We now exemplify the relation (and redundancy) between precedence relations, specialisation conditions and preconditions. Whenever a fraudster spots a naive victim (a wretch) and scams him, he reports his fraud to the criminal organisation he belongs to (see Figure 126). The fraud can either be little (in case the swindled amount is lower than 1000€) or big (in case the swindled amount is equal or bigger than 1000€). The diagram shows some precedence relations. For instance, for *E5* to occur, *E2* has to have occurred first (e.g. the victim is first investigated and relevant data is recorded); for *E5.2* to occur, *E2*, and either *E3* or *E4* need to have occurred first. Additionally, the big con cannot happen if *E7* has already occurred (e.g. the criminal boss has told the fraudster to stay under the radar for a while); this has been informally expressed with doted lines, since the Communicative Event Diagram does not provide any modelling primitive to express such condition.

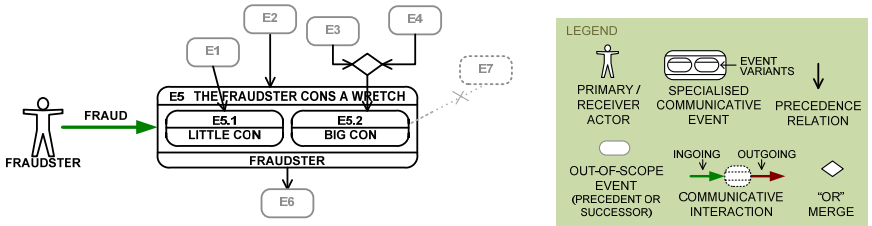


Figure 126. Example related to fraud to illustrate preconditions

Table 44. Message structure of communicative event *S5* from the fraud example

FIELD	OP	DOMAIN	SPECIALISATION CONDITION	RELATED EVENT VARIANT
FRAUD =				
< Some a +	i	A		
b +	i	date		
n +	i	money		
[LITTLE =			n<1000€	E5.1
< The c +	i	C		
d >	i	text		
| BIG =			n>=1000€	E5.2
< One e +	i	E		
The f +	i	F		
g >	i	text		
]				
>				

The information reported by the fraudster to the information system of the criminal organisation is specialised, depending on the type of fraud. The message structure is shown in Table 44. We have included two columns to show the specialisation condition and the event variant to which each part of the specialised substructure corresponds.

The preconditions of each of the encapsulations are shown in Table 45. Note that occ(*EI*) stands for "encapsulation *EI* has occurred." It can be seen how the precondition of each variant inherits the precondition of its upper encapsulation (the event variant *E5.1* inherits the precondition of the communicative event *E5*; that is, occ(*E2*)) and, additionally, includes its own specialisation condition (usually expressed in terms of message fields; e.g. n<1000€). Unlike the Communicative Event Diagram, which is intended to be kept a lightweight notation, the language for preconditions should allow for enough expressiveness to represent complex business rules by means of logical formulas.

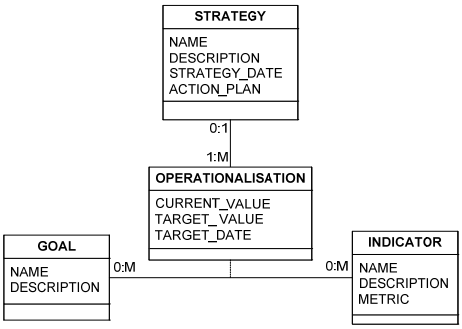
All in all, we acknowledge that this issue requires further investigation that will be carried out when the method is applied in practical settings which require such formality in precondition specification (until now, preconditions have been expressed in natural language).

Table 45. Preconditions of the fraud example

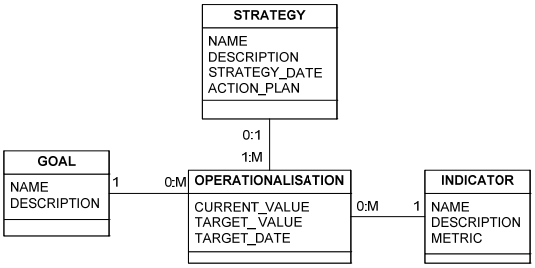
Precondition
precond(*E5*) = occ(*E2*)
precond(*E5.1*) = occ(*E2*) and occ(*E1*) and n<1000€
precond(*E5.2*) = occ(*E2*) and (occ(*E3*) or occ(*E4*)) and not(occ(*E7*)) and n>=1000€

Operationalisation

When a meta-class does not have a clear correspondence with a Communication Analysis concept it is potentially a case of an ungrounded modelling primitive. The meta-class OPERATIONALISATION is seemingly unaligned with the Communication Analysis ontology. However, it is due to the fact that the meta-class should actually be an association class between classes GOAL and INDICATOR (see Figure 127.a). However, we have chosen to specify the association as a class (see Figure 127.b) for several reasons: (i) although the semantics is slightly different, we consider that the practical implications for communicating the method are negligible; (ii) the OO-Method Object Model, the UML-compliant language chosen to represent the metamodel, does not offer the association class modelling construct; (iii) we plan to later specify Communication Analysis by means of EMOF (an Eclipse metamodelling facility), but the EMOF metamodel neither supports association classes.



a) OPERATIONALISATION as an association class



b) OPERATIONALISATION as a class (how it is currently supported)

Figure 127. OPERATIONALISATION should actually be an association class

6.2.3 Discussion

The result from the ontological evaluation of Communication Analysis shows that the method is well founded and properly aligned with FRISCO 1.1. This should not be surprising since the concepts of the method have been carefully built upon this conceptual framework. It would be interesting to perform an ontological evaluation using a different reference ontology. The Bunge-Wand-Weber model (BWW-model) is a candidate. However, we anticipate a clash of epistemological stances, such as the authors of [Opdahl, Henderson-Sellers et al. 2000] reported. The BWW-model is a naïve-realist approach whereas FRISCO is a constructivist approach, what can lead to different interpretations of Communication Analysis constructs. Communication Analysis is oriented towards the distinction of conceptions and representations and the BWW-model lacks this constructs, what can be a problem.

In any case, the evaluation has actually been clarifying because it has made explicit the alignment with the underlying ontology (the alignment was until now scattered throughout the definitions). More importantly, the evaluation has allowed us to improve the method in several ways. Two iterations of the evaluation were made. The above-mentioned are the results of the second iteration. The first iteration revealed that some concepts of the method were not properly defined (they were either ambiguous or not properly aligned with FRISCO 1.1) and that the metamodel could be redesigned to better fit the method concepts.

Communication Analysis concept redefinitions

The first iteration of the ontological evaluation revealed several meta-classes for which there was no underlying concept. It was the case of PROCESS and EVENT_VARIANT. The problem actually laid in the method ontology. The concept of business process was informally defined within Section 4.2.2, but a definition based on the previously-defined concepts was missing. Thus, we added a definition that combines the mechanistic and the constructivist view of business processes (see Def. 167 and the discussion in Section 4.6). This way, we kept the definition in line with the epistemological stance of FRISCO but we allow for a hard-systems modelling approach, which is convenient from a model-driven development point of view.

Section 4.2.2 also explained the purpose of event specialisation and the meaning of event variants. However, a rigorous definition was missing. Thus, we provided a definition for event variants (see Def. 174), based on the concept of choice state-transition structures. This way the concept is integrated within the FRISCO 1.1 ontology.

We also improved some definitions that were slightly ambiguous, highlighting the main concept in which they were based and qualifying it with a context. After these changes in the method ontology, we performed the second iteration of the ontological evaluation, the results of which are discussed above.

Redesign of the method metamodel

For instance, the metamodel initially did not explicitly include a meta-class for message structures, but it was modelled as a role of a structural relationship between COMMUNICATIVE_INTERACTION and COMPLEX_STRUCTURE (see Figure 128).

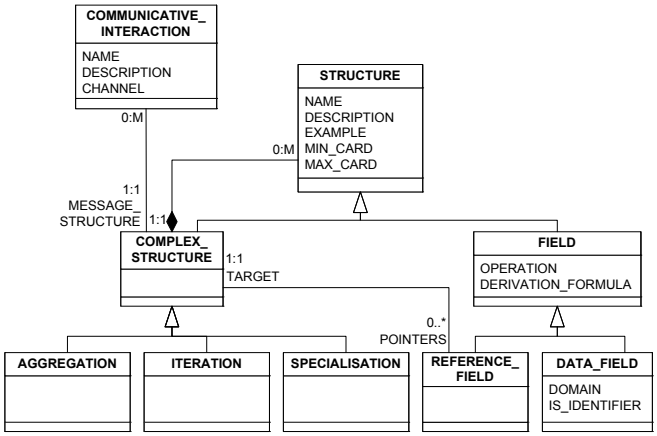


Figure 128. Implicit message structures in an earlier version of the metamodel

Although form a practical point of view, this metamodel supported the specification of message structures, the fact that the alignment of the Communication Analysis concept message structure with a meta-class was not clear, invited for a deeper analysis. We came up with an improved metamodel in which a meta-class Message_Structure was explicit. This way (i) the conceptual alignment is clearer, (ii) the metaclasses better map the BNF grammar rules of the language (the STRUCTURE of the earlier version is now SUBSTRUCTURE, better reflecting its nature; see Section 4.3.2.1) and (iii) the notion reusing complex substructures is simplified (in the earlier version, an integrity constraint was needed to prevent reusing a message structure as a complex substructure of another message structure).

The metamodel initially included a meta-class named FORMULA (see Figure 129); but the ontological evaluation revealed that it was neither aligned with a Communication Analysis concept, nor with a FRISCO 1.1 concept. It could simply be an incompleteness of the FRISCO ontology, which does not include concepts about algebras; but there was something suspicious with the meta-class.

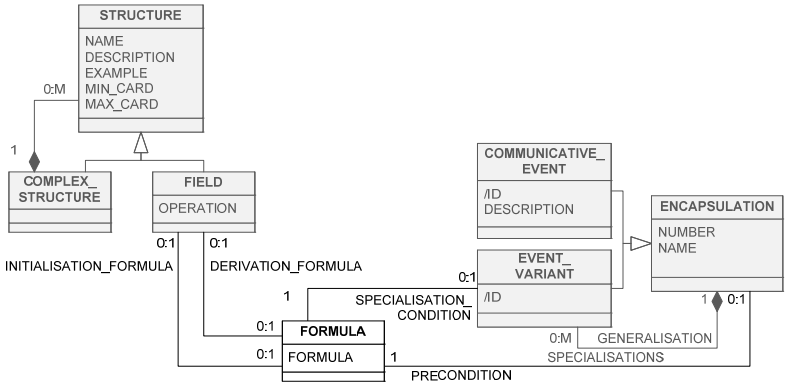


Figure 129. Meta-class FORMULA of an earlier version of the metamodel (ruled out later)

By analyzing the structural relationships in which FORMULA is involved, it soon became clear that there was an implicit specialisation: a formula is either a precondition, or a specialisation condition, or a derivation formula, or an initialisation formula, but not several of these at the same time. If the specialisation was made explicit (see Figure 130) then the integrity constraint was not needed, but the 1:1---1:1 cardinalities now raised doubts about the need of the meta-class.

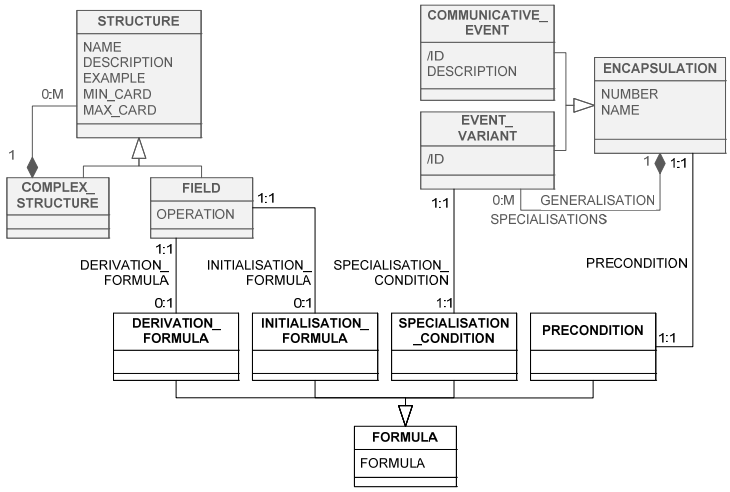


Figure 130. Specialised FORMULA (ruled out later)

We realised that that the metaclass was the result of factorising the formula in order to simplify the implementation of their management. However, this was a design decision that could be disregarded when creating the platform-independent metamodel. Thus, we ended up removing the metaclass Formula and representing each formula/condition as an attribute of their corresponding meta-class (see Figure 88); namely, ENCAPSULATION.precondition, EVENT_VARIANT.specialisat_cond, FIELD.derivation_formula and FIELD.initialisation_formula.

We also considered specialising FIELD into NEW_INFORMATION_FIELD and KNOWN_INFORMATION_FIELD; this would have allowed expressing the fact that the former can have an initialisation formula that increases the usability of the interface, whereas the latter necessarily have a derivation formula that specifies how the content of the field related to information previously known by the organisational system. However, this would have implied a double specialisation of the meta-class FIELD, which is something we intended to avoid for the same reason we avoided association classes (see argumentation in page 339). Also, instead of including both attributes, we kept only derivation_formula because the initialisation formula can be seen as a derivation formula whose result the user can later override.

Summing up, the ontological evaluation allowed us to improve both the Communication Analysis ontology and the method metamodel, as well as highlighting some issues that require further investigation.

6.3 *Lab demos (often wrongly referred to as case studies)*

According to Wieringa [2008] a *lab demo*, is the application of a method or technique by one or several of their authors on a realistic example in an artificial environment (e.g. from the comfort of your own laboratory). A lab demo is not a case study as defined by Runeson and Höst [2009] (which implies being conducted in real world settings); however, many authors wrongly refer to their lab demos as case studies. Whereas the results from a case study may serve to answer research questions and to perform analogical generalisation based on similarity of mechanisms in other projects, lab demos serve to demonstrate that the method or technique could, in principle, work in practice [Wieringa 2008].

Unlike a merely illustrative example (a small example to explain a technique, in order to allow the reader to understand it), a lab demo seeks to deal with realistic problems that put the method to the test. This way, it often provides valuable initial feedback to the method engineers (in the requirements engineering field, these are often the researchers themselves) about the strengths

and weaknesses. A lab demo can be used not only to provide a detailed example of how a method is applied, but also to identify possible improvements.

In the context of this thesis, we have created innumerable illustrative examples and we have conducted not one but several lab demos. We highlight the following, since these also cover the integration with the OO-Method (this aspect of the lab demos is discussed in Section 7.3). For each lab demo, a brief description of the case (the organisational system and its work practice) is offered, and some lessons learnt by performing the lab demo are discussed.

Table 46 shows some measures of the size of the lab demos, including their IFPUG function points (see [Abrahão, Poels et al. 2006; Giachetti, Marin et al. 2007] on how these are measured on OO-Method conceptual models).

Table 46. Size of the reference solutions to the lab demos

	Projects office	**Photography Agency Inc.**	**Master management system of TUO**	**SuperStationery Co.**
Communicative events	12	15	12[98]	11[99]
Classes	11	16	14	14
Attributes	56	74	94	55
Services	78	102	93	49
Structural relationships	9	19	18	9
IFPUG function points	355	537	N/A	253

6.3.1 Projects office

This case describes a projects office that carries out industrial electrical installations for their customer companies. The usual work practice is the following. Companies contact the projects office to request projects. Engineers draw up budgets (that need to be accepted by the company) and organise a project team to carry out the project. Low-budget and high-budget projects are treated differently (the latter need to have the budget approved by the projects office accountant and the installation is verified by the chief engineer.

[98] The case includes two CRUD subsystems that define basic data management, but each of these has been counted as one communicative event.

[99] The communicative event diagrams depict 16 communicative events, but only 11 of them are specified by means of event specification templates and are input of the conceptual model derivation process.

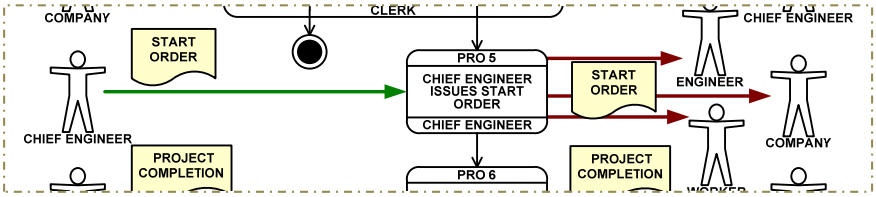


Figure 131. Fragment of a communicative event diagram that includes message symbols latter discarded

This was one of the first lab demos performed, it was based on teaching material created by Arturo González for the course "Conceptual Modelling of Information Systems" at Universitat Politècnica de València and later extended to increase the case complexity. It served to illustrate how the detailed specification of Communication Analysis was applied and to come up with some improvements to the method.

The first versions of the lab demo included some notation elements that were discarded in latter releases of the method specification. For instance, the message symbols above communicative interaction arrows (see Figure 131) were latter removed and the message name was indicated as a label to keep the diagrams simple. Also, the identification operator (id) in the message structure reference fields (see Table 47) was omitted since it was redundant with the fact that the domain of a reference field is a business object and not a basic domain; however, as a visual cue, a secondary notation guideline was included that recommends using a different colour for reference fields (e.g. dark red, although it can be customised by the analyst).

The material resulting from this lab demo have been later used as teaching material by Sergio España in the course "Model-driven Software Development" at University of Twente. A simplified version of the case is presented in [González, España et al. 2008a] to illustrate Communication Analysis, and in [Ruiz, España et al. 2010] to illustrate a model-driven diagramming tool that supports Communicative Event Diagrams.

Table 47. Message structures of communicative event *PRO 5* illustrating some changes in the technique notation

FIELD	OP	DOMAIN
START ORDER =		
< Project(id) +	i	Project
WORKERS =		
{ Worker(id) }e +	i	Worker
Commencement date	i	date
>		

FIELD	OP	DOMAIN
START ORDER =		
< Project +	i	Project
WORKERS =		
{ Worker } +	i	Worker
Commencement date	i	date
>		

6.3.2 Photography Agency Inc.

This case describes a photography agency that manages illustrated reports and distributes them to publishing houses. The scope of the case covers the management of photographers (e.g. application, selection, promotion, etc.) and publishing houses (e.g. subscription), as well as the management of regular reports (these are provided by photographers and become part of the agency catalogue, then publishing houses order them) and of exclusive reports (these are first requested by a publishing house and then assigned to a photographer); the delivery of both types of reports and the corresponding invoicing to publishing houses and payment to photographers are also within the scope of the case. Figure 132 shows the communicative event diagram of Photography Agency Inc.

Within this lab demo, some communicative events were refined into fine-grained business process models depicting the as-is work practice, using an ad-hoc notation (see Figure 133).

This lab demo is also based on teaching material created by Arturo González for the course "Conceptual Modelling of Information Systems" and by Óscar Pastor for the course "Software Technology for Web Environments", both at Universitat Politècnica de València. It served to put a second release of the detailed specification of Communication Analysis into test.

This case is used to illustrate Communication Analysis in [España, González et al. 2009]. The material resulting from this lab demo has later been used as teaching material by Sergio España in the course "Model-driven Software Development" (2009) at University of Twente and in course "Software Technology for Web Environments" (2010 and 2011) at Universitat Politècnica de València. Also, it is part of the instrumentation of the experiments described in Section 6.5 (results published in [España, Condori-Fernández et al. 2009] and [España, Condori-Fernández et al. 2010]) and in Section 7.5.

The case was part of the teaching material of the course "Model-driven System Development" (2009) at University of Twente, and was used as instrumentation for the experiment operated within the course.

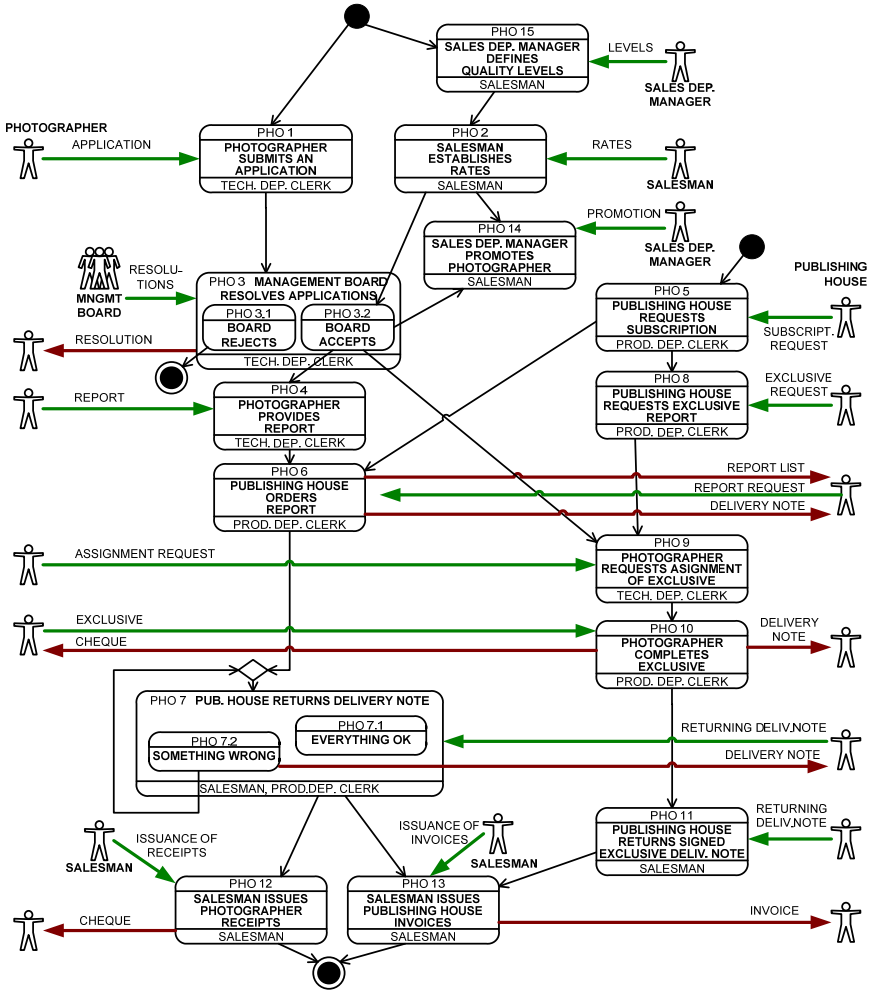


Figure 132. Business process model of Photography Agency Inc. using Communicative Event Diagram

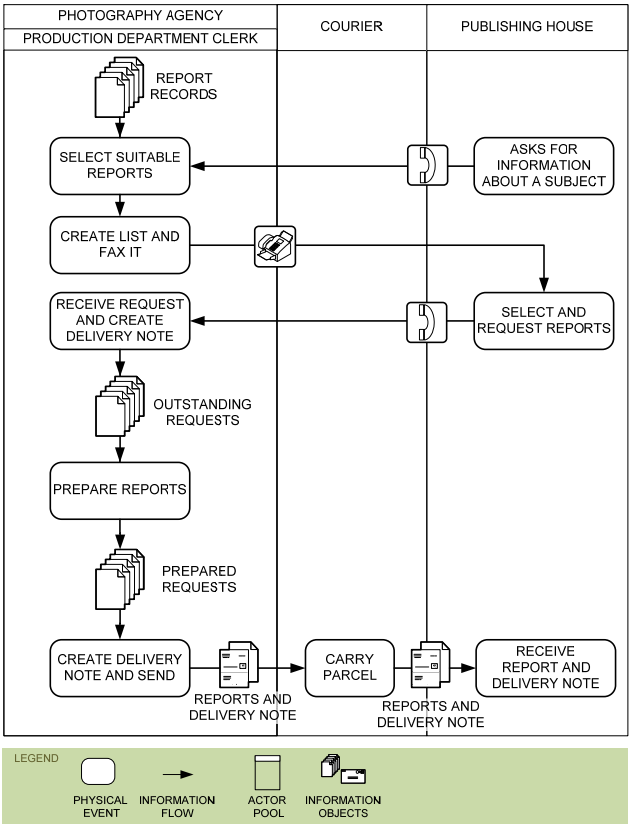


Figure 133. Refinement of communicative event *PHO 6*

This case is also part of the instrumentation of an ongoing experiment being carried out with researchers from the Information Systems research group of University of Twente; this experiment was operated within the bachelor course "Information Systems" (2010) at University of Twente. For this experiment, the requirements model was translated into a different notation, but keeping the structure and modularity prescribed by Communication Analysis. Instead of Communicative Event Diagrams, Activity Diagrams were used (see Figure 134). This experience demonstrated the feasibility of applying the guidelines and criteria of Communication Analysis with other notations.

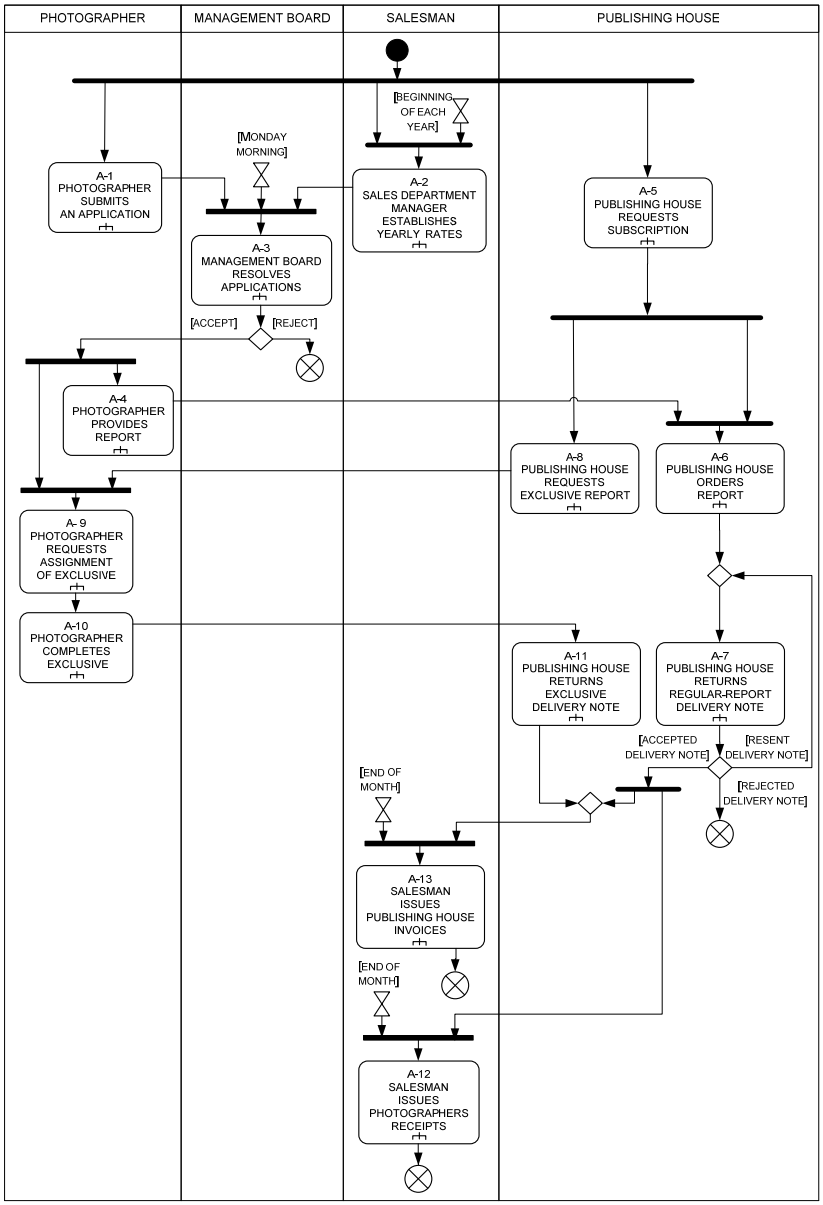


Figure 134. Business process model of Photography Agency Inc. using Activity Diagram

6.3.3 Master management system of TUO

Table 48. Message structure of communicative event *MAS 2*

FIELD	OP	DOMAIN	EXAMPLE VALUE
MASTER DOCUMENTATION =			
< Master edition +	i	Master edition	MISD XII edition
Start date +	i	date	15/10/2008
End date +	i	date	29/06/2009
Total hours +	i	number	300
Enrolment begin date +	i	date	02/09/2008
Enrolment end date +	i	date	10/09/2008
HT +	i	number	160
HP +	i	number	140
HS +	i	number	140
HI +	i	number	0
NT +	i	number	0
HD +	i	number	0
Lecture information +	i	text	Lectures are given Mondays and Wednesdays, ...
Max. students +	i	number	30
Surveys +	i	text	After each course. Permanent Education Centre...
Instalments +	i	[yes | no]	yes
Direct debit +	i	[yes | no]	yes
Insurance +	i	[PEC | other]	PEC
Notes +	i	text	
COURSES =			
{ Code +	i	text	Ren
Course name +	i	text	Requirements engineering
Type +	i	[B=basic | O=optional]	B
HT +	i	number	8
HP +	i	number	8
HS +	i	number	4
HI +	i	number	0
NT +	i	number	0
Start +	i	date	26-11-08
End +	i	date	10-12-08
Modular +	i	[y=yes | n=no]	y
Price +	i	money	240,00€
TEACHING =			
{ Lecturer +	i	Lecturer	26543311R, James L. Scanlon, DSE, Prof
HL +	i	number	16
HT	i	number	8
} TEACHING			
} COURSES			
>			

The Technical University of Orancia (TUO) is a small university that is starting to computerise their information systems in several small-sized projects. Now they want to develop a web-based information system to manage the specialisation degrees that the university departments propose. The case covers the proposal of new masters and editions of these maters (courses can change from one year to another), their evaluation by the TUO permanent education centre, as well as the enrolment of students in the master, diploma issuance, etc.

This lab demo is loosely based on bureaucracy procedures from Universitat Politècnica de València. The complexity of some of the business forms is much higher than in previous cases, what leads to larger message structures.

This case is part of the teaching material of the course "Model-driven System Development" (2009) at University of Twente, as well as part of the instrumentation the pilot experiment carried out within that course.

6.3.4 SuperStationery Co.

SuperStationery Co. is a company that provides stationery and office material to its clients. The company acts as a as intermediary: the company has a catalogue of products that are bought from suppliers and sold to clients. The organisational work practice is decomposed into seven business processes; namely, client management, product management, logistics, sales management, risk management, accounting and supplier management. However, only the sales management business process (and the communicative events that are precedents of this process) is actually specified in detail.

This is the most updated lab case to the date; its complete documentation is available as a technical report [España, González et al. 2011]. It has served to consolidate the Communication Analysis secondary notation guidelines, as well as the requirements structure. It has become part of the teaching material of the course "Software Technology for Web Environments" since 2010. It is part of the instrumentation of the experiment described in Section 2.4. Parts of this case appear in several articles related to Communication Analysis [Ruiz, España et al. 2010] [España, Ruiz et al. 2011] [González, España et al. 2011]. Lastly, most examples in this thesis are drawn from the SuperStationery Co. case.

6.4 An empirical framework for RE validation

In order to compare requirements engineering approaches, we adopt (and adapt) the Method Evaluation Model (MEM), which defines a theoretical model and associated measurement instruments for evaluating information system design methods [Moody 2003]. The MEM incorporates two aspects of method success;

namely, actual efficacy (whether the method improves performance of the task) and adoption in practice (whether the method is used in practice). Additionally, in order to better understand practitioners' reaction to the method, the MEM also includes variables related to perceived efficacy. In our framework, we focus on actual efficacy and perceived efficacy[100]. With regard to actual efficacy, the conceptual model quality framework by Lindland et al. is applied [Lindland, Sindre et al. 1994]. This framework distinguishes syntactic, semantic, and pragmatic quality. Since our intention is to apply the framework to requirements engineering method evaluation and comparison and not strictly to model assessment, we focus on semantic quality and pragmatic quality; we disregard syntactic quality because we do not find it is especially relevant to decide on the quality of a requirements method. We have extended the framework with four new metrics that are aimed at the assessment of model completeness and model modularity. With regard to perceived efficacy, a post-task survey is used.

Many authors have theorised and empirically validated conceptual model completeness (the degree to which a model specifies all the relevant statements of a domain) [Yadav, Bravoco et al. 1988; Lindland, Sindre et al. 1994]. However, the approach often consists of a reviewer rating completeness on a Likert scale (or similar), and the procedure for assigning the rating depends on a subjective judgement [Yadav, Bravoco et al. 1988; Moody, Sindre et al. 2002; Moody, Sindre et al. 2003]. We propose an approach based on the comparison of the reviewed model with a reference model that is built and agreed by an expert modelling committee. The metrics and the measuring procedure are precisely defined; for instance, the metric for completeness is not a Likert scale but a rather objective ratio that, in short, depends on the size complexity of the domain and the amount of information specified in the model.

Modularity, on its turn, is a much less investigated quality aspect, although it is an issue that has provoked debate in academic and industrial communities, particularly with regard to functional requirements such as use cases [Larman 1997; Kulak and Guiney 2000; Övergaard and Palmkvist 2004]. Kulak and Guiney, for instance, define use case granularity as the relative scope of individual use cases compared to the application's scope [Kulak and Guiney 2000]. In a more general sense, modularity is the design principle of having a complex system composed from smaller sub-systems that can be managed independently yet function together as a whole [Langlois 2002] and granularity measures the size of encapsulations or modules (it is a systemic notion). When modelling, the analyst relies on methodological guidelines in order to encapsulate concepts in modelling primitives. We refer to the criteria that allow determining granularity as unity criteria (see Section 4.3.1.2). It has been proved

[100] Measuring adoption in practice implies follow-up studies on the experimental subjects and this was not possible in our setting.

that modularity improves business process model understanding [Reijers and Mendling 2008]. Our framework proposes an evaluation of the quality of modularity that goes beyond previous proposals since sound unity criteria for business process modelling are taken as reference and precise metrics for modularity errors are defined.

In any case, for a requirements engineering method to be successful, it is not enough that the method is theoretically sounder, not even that it performs better in practice, but it also needs to be perceived by practitioners as more useful and easier to use [Yadav, Bravoco et al. 1988].

6.4.1 The Method Evaluation Model (MEM)

Following Moody's approach to validating information system design methods [Moody 2003], in deciding how to validate requirements specification techniques, one needs determining whether a method is successful or not. Moody argues that there are (at least) two dimensions of "success" that need to be considered in evaluating information system design methods:

- *Actual efficacy*, i.e. does the method improve performance?
- *Adoption in practice*, i.e. is the method used in practice?

Moody developed the Method Evaluation Model (MEM), a theoretical model that combines aspects of the Rescher's pragmatic method [Rescher 1977] and the Davis's Technology Acceptance Model [Davis, Bagozzi et al. 1989].

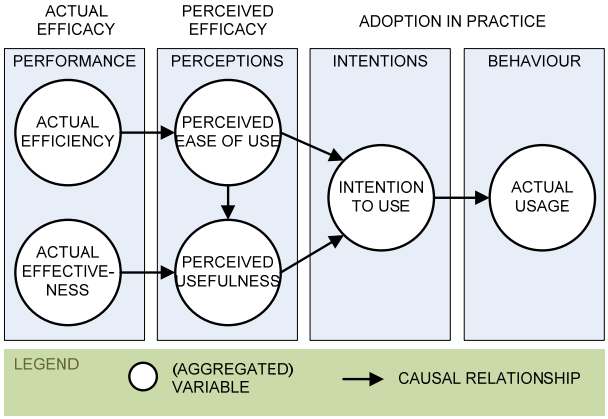


Figure 135. Method Evaluation Model (adapted from [Moody 2003])

As shown in Figure 135, MEM includes four groups containing six primary constructs:

- *Performance* refers to the degree of success that is achieved in a particular task when applying the method, as well as the usage of resources (e.g. the effectiveness and the efficiency of the analysts in creating a conceptual model that corresponds to the information system that an organisation needs).
  - *Efficiency* refers to the usage of resources (e.g. the time it takes the analysts to create the conceptual model).
  - *Effectiveness* refers to the success in the task and the quality of the result (e.g. the quality of the resulting conceptual models, when the analysts apply the method).
- *Perceptions* refers to how the users perceive the method. Perception-based variables are usually measured using questionnaires.
  - *Perceived ease of use* refers to how easy the users perceive the method to be.
  - *Perceived usefulness* refers to how useful the users perceive the method to be, in relation to the task being solved.
- *Intentions* refers to their plans for the future.
  - *Intention to use* refers to whether they intend to use the method in the future.
- *Behaviour* refers to the actions of the subjects in the future, with regards to the method[101].
  - *Actual usage* refers to whether the subjects use the method in later occasions or not (e.g. whether the analysts end up using the conceptual modelling method in industrial settings).

Moody's model is based on the theory that actual efficiency and actual effectiveness determine intentions to use a method only 'second-hand', via the perceived ease of use and the perceived usefulness. This is due to the fact that, in human behaviour, subjective reality is more important than objective reality. In our work, we are concerned with evaluating the requirements engineering methods with regards to the MEM, except for the actual usage, which is out of the scope.

[101] The *behaviour* of the subjects in the future (e.g. the actual usage of the method variants by the experimental subjects when they become industry practitioners) requires tracking them for several years and then assess whether they are actually using the method. We do not intend to do this.

6.4.2 Conceptual model quality frameworks

Lindland et al. [Lindland, Sindre et al. 1994] present a conceptual model quality framework that is founded on semiotics and linguistics. Three types of model quality are assumed:

- *Syntactic quality*. The degree to which the model adheres to the modelling language rules. Syntactic errors and deviations from the rules decrease syntactic quality.
- *Semantic quality*. The degree to which the model represents the domain. The more similar the model and the domain, the better the semantic quality.
- *Pragmatic quality*. The degree to which the model is correctly interpreted by its audience. The less misunderstanding, the better the pragmatic quality.

For each type of quality, absolute goals are defined and the means to achieve the goals are described. See Figure 136 for an overview of the framework.

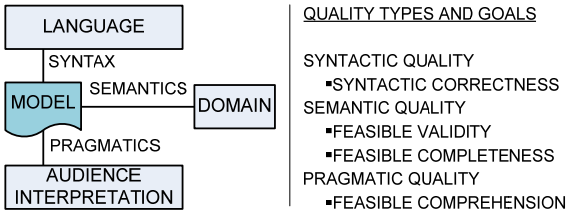


Figure 136. Conceptual model quality framework [Lindland, Sindre et al. 1994]

Other conceptual modelling quality frameworks have been proposed in the literature. Yadav et al. [Yadav, Bravoco et al. 1988] propose a framework that provides criteria to compare information systems RE methods in terms of the modelling process and the model itself; criteria are classified in four categories: syntactic, semantic, usability, and communicating ability. Davis et al. [Davis, Overmyer et al. 1993] explore the concept of RS quality and consider completeness, but acknowledge that the proposed metrics are "difficult to measure". Pohl [Pohl 1994] develops a framework for RE with three dimensions (specification, representation, and agreement) and defines three goals for a RS (formally represented, complete, and agreed). In [Krogstie, Sindre et al. 2006], the framework by Lindland et al. is extended by considering additional levels of the semiotic ladder [Falkenberg, Hesse et al. 1998]. In [Krogstie, Lindland et al. 1995], the framework by Lindland et al. is extended with notions on social construction theory from Pohl's framework. Moody et al. [Moody and Shanks 1994] present a quality framework for data models oriented towards practice. Schuette et al. [Schuette and Rotthowe 1998] present the Guidelines of Modelling, which is a

framework of principles that improve the quality of information models by reducing subjectivism in the information modelling process.

Some reviews of the state of the art and framework comparisons can also be found. In [Schuette 1999], the frameworks by Krogstie, Moody et al., and Schuette et al. are compared by means of their metamodels. In [Shanks and Darke 1997], several frameworks are compared and the frameworks by Lindland et al. and Moody et al are integrated. In [Moody 2005], an exhaustive review of related works is presented, and recommendations for research in conceptual model quality are given.

We have chosen the framework by Lindland et al. as a point of departure for our research for it has been acknowledged as a reference framework by many authors [Moody 2005; Krogstie, Sindre et al. 2006] and it has been empirically tested and used for evaluation purposes [Moody, Sindre et al. 2002; Moody, Sindre et al. 2003]. Moreover, there is empirical evidence that the framework is useful for evaluating requirements engineering methods [Abrahão, Insfran et al.].

6.4.3 Adopting and extending MEM for evaluating requirements engineering techniques

In the experiment described in Section 6.5, the MEM has been adopted as a framework for method comparison (see Figure 137). As recommended in [Moody 2003], actual efficacy variables have been further refined and adapted taking into account the particularities of the methods being compared and the task being evaluated, and a questionnaire has been used to measure perceived efficacy.

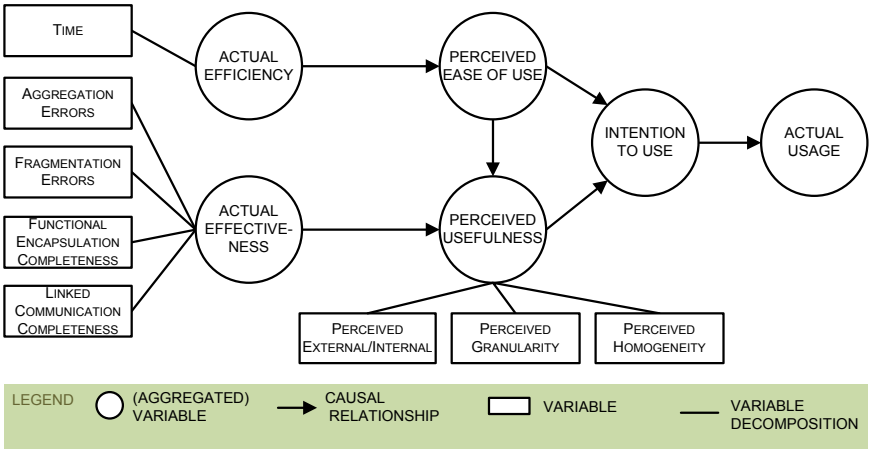


Figure 137. Refined MEM used in the experiment

6.4.3.1 Actual efficacy

Lindland et al. [Lindland, Sindre et al. 1994] define that a model (*M*) has achieved semantic completeness if it contains all the statements about the domain (*D*) that are correct and relevant. That is, $D \setminus M = \emptyset$. However, except for extremely simple and highly inter-subjectively agreed domains, total completeness cannot be achieved (due to resource restrictions). Hence, they relax the completeness goal by applying the notion of feasibility. Feasibility introduces a trade-off between the benefits and drawbacks for achieving a given model quality.

A model has achieved *feasible completeness* when there is no relevant statement about the domain, not yet included in the model, such that the additional benefit to the conceptual model from including the relevant statement exceeds the drawbacks of including it. That is, $D \setminus M = S \neq \emptyset$, where *S* is the set of correct and relevant statements not yet in the model.

However, from a constructivist stance (such as in [Krogstie, Sindre et al. 2006]), to determine whether there is a relevant statement of the domain not yet included in the model, one must first conceptualise the domain. This conceptual model of reference (*Mr*) needs not be written but at least it must exist in the reviewer's mind.

Also, Lindland et al. recommend adapting and refining the framework depending on the method being evaluated, as well as operationalising the framework by proposing metrics. For instance, they propose to decompose feasible completeness into feasible functional completeness, feasible non-functional completeness, etc. A germinal definition states that a model has achieved *feasible functional completeness* whenever all relevant functional requirements that are worth being specified have been included in the model (adapted from [Lindland, Sindre et al. 1994]). We further develop this idea by refining it, by explicitly considering the existence of a reference model, and by proposing metrics.

When Lindland et al. propose refining the framework they consider the existence of different types of statements about the domain, what leads to building models with multiple views. Each view can itself be considered a model. For instance, if the domain is considered to contain two types of statements (those about functionality -*FD*- and those about qualities and restrictions -*NFD*-), then the requirements model *M* will have two views: a functional requirements model *FM* and a non-functional requirements model *NFM*; that is, $D = FD \cup NFD$ and $M = FM \cup NFM$. Feasible functional completeness can now be formally defined as $FD \setminus FM = FS \neq \emptyset$, where *FS* is the set of functional requirements not yet in the model (but which are not worth the effort to be included).

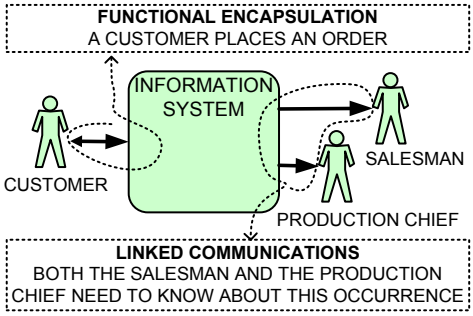


Figure 138. Abstraction of the primitives of information systems functional requirements models

Feasible functional completeness can be further refined if a specific type of domain is considered (e.g. information systems). Research about the essence of information systems has lead to the definition of sound conceptual frameworks. An information system (information system) is a socio-technical system that supports organisational communications [Lockemann and Mayr 1986]. The FRISCO report [Falkenberg, Hesse et al. 1998] laid the foundations of the area, upon which other researchers have built theories (we have also based our work on FRISCO, see Section 3.3). Based on this conception of information systems, two major abstract modelling primitives for functional requirements specifications are identified; namely functional encapsulations and linked communications.

- Wieringa defines function as a service provided by the information system o its environment; it is also considered to be an encapsulation of a useful behaviour of the system [12]. We therefore refer as *functional encapsulations* to information system functions, in order to highlight the importance of determining the boundaries of the encapsulation.
- We refer as *linked communication* to the message conveyance that is triggered by the occurrence of an event (the use or activation of a function) and by which the information system informs an actor of this occurrence.

This way, it is considered that a functional requirements model is composed of (at least) a set of functions (F) and a set of linked communications (LC). Then $FM = F \cup LC$. Figure 138 shows an example of these abstract primitives.

Furthermore, a constructivist approach takes us to define completeness with respect to a reference model. In industrial settings, a reference model may be impractical; customer reviews are essential [Davis, Overmyer et al. 1993]. In experimental settings, we propose that an expert modelling committee analyse a given domain and agree a model that strictly follows best practices in

modelling[102]. Some best practices depend on the modelling technique (e.g. correct use of use case *includes* and *extends* relations [Cockburn 2000]) whereas others are independent (e.g. requirements specifications should provide an external view of the system [Zave and Jackson 1997]).

The reference model is defined as $Mr = FMr \cup NFMr$, and the reference functional requirements model is defined as $FMr = Fr \cup LCr$. It is assumed that FMr contains all relevant functional requirements and linked communications. Measurable quality goals are defined in terms of the reference model (see Figure 139):

- *Functional encapsulations completeness with respect to (w.r.t.) a reference model*. All functional requirements specified in the reference model have been specified in the model. That is, $Fr \setminus F = \emptyset$. Note that the reference model substitutes the domain in the comparison.

This goal is related to the degree with which the reviewed model contains the functional encapsulations (i.e. use cases, communicative events) included in the reference model. A metric for this goal is the degree of functional encapsulations completeness w.r.t. a reference model, which is defined as $degFEC = |F|/|Fr|$.

- *Linked communications completeness w.r.t. a reference model*. All linked communications specified in the reference model have been specified in the model. That is, $LCr \setminus LC = \emptyset$.

This goal is related to the degree with which the reviewed model contains the linked communications (i.e. communications triggered by the occurrence of a use case or a communicative event) specified in the reference model. A metric for this goal is the degree of linked communications completeness w.r.t. a reference model, which is defined as $degLCC = |LC|/|LCr|$.

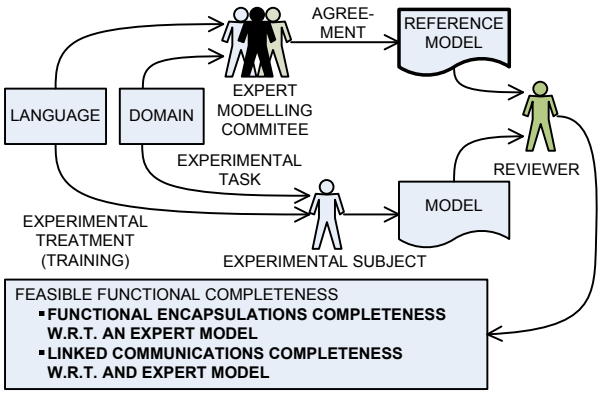


Figure 139. Model completeness evaluation with respect to a reference model

[102] This agreement may involve debate and several iterations.

Assuming that it is feasible to build the reference model, both goals are feasible quality goals.

With regard to modularity, we claim that it is a quality of requirements models that has not been sufficiently investigated. We refer as unity criteria to the norms that guide the identification of complex concepts (e.g. use cases) and the encapsulation of their components (i.e. the flow of actions a use case is composed of); therefore, unity criteria determine the granularity of the encapsulations and, thus, model modularity (see Section 3.4). Data models unity criteria are frequently based on the notion of identification; there is much consensus in this area. However, unity criteria for functional models (e.g. use case models) are far from being widely agreed. We argue that this lack of agreement leads to the proliferation of methodological guidelines that avoid the thorny issue of modularity or, in many cases, they define simplistic unity criteria (e.g. the *twenty use cases per system* rule of thumb [Jacobson 2004], the *one person, one sitting* test [Cockburn 2000]). Consequences are inconsistencies in modelling practice, heterogeneous model modularity, and the need for gurus to whom consult [Kroll 2004].

In Section 3.4 we unfold the notion of unity criteria and in Section 4.3.1.2 we propose unity criteria for business process modelling. Industrial practice has shown us that the criteria reduce subjectivity in the encapsulation of business processes.

A model is considered to have an *appropriate modularity* with respect to a given unity criteria when the encapsulations of the model are conform to the unity criteria. Taking a given set of unity criteria it is possible to identify modularity errors in a model (see Figure 140). If the reference model conforms to the unity criteria, it aids the review; but it is not strictly required.

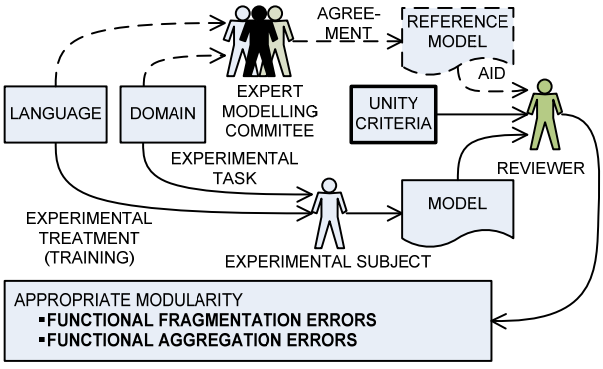


Figure 140. Model modularity evaluation with respect to unity criteria

For functional requirements models we propose identifying the following modularity errors:

- *Functional fragmentation error*. This error is the result of modelling two or more functional encapsulations for a part of the domain which, according to the given unity criteria, should have been modelled as only one encapsulation. For instance, two or more use cases of the reviewed model correspond to one use case of the reference model. It is measured as a variable named *errFra*.
- *Functional aggregation error*. This error is the result of modelling certain part of the domain as one functional encapsulation when, according to the unity criteria, the phenomena should be modelled using two or more encapsulations. For instance, two or more use cases of the reference model are modelled as only one use case in the reviewed model. It is measured as a variable named *errAgg*.

Therefore, a model can be said to achieve appropriate modularity whenever there are no modularity errors. That is, *errFra* = 0 and *errAgr* = 0.

6.4.3.2 Perceived efficacy

- *Perceived ease of use*: the degree to which a person believes that using a particular specification technique would be free of effort.
- *Perceived usefulness*: the degree to which a person believes that a particular specification technique will be effective in achieving its intended objectives. Three objectives were identified in our context to evaluate this perception:
    - Distinction between external and internal interaction. A good practice in requirements engineering is to focus the specification exclusively on external behaviours, what Davis refers to as system phenotype (as opposed to genotype, which corresponds to design) [Davis 2003b]. When specifying interactions, this distinction also applies, the requirements engineering method should aid the analysts in identifying external interactions and separating them from internal interactions. We intend to investigate the analysts' impression about the compared requirements specification techniques with regards to this issue.
    - Homogeneity of functional specification. In the context of this experiment, we refer by homogeneity to the degree of similarity of the models created by different analysts when confronted to the same problem statement. We acknowledge that there is space for creativity in requirements engineering; however, the fact that a modelling method enhances model homogeneity should not be viewed as a negative constraint to creativity. Creativity is applied when figuring out the problem statement, when analysing the domain (e.g. questioning assumptions and helping organisational actors think out of the box); also when creating a solution,

when designing the system. Here, homogeneity of specifications aims at facilitating communication of the models and facilitating their comprehensibility. "Process models created within a certain domain, might it be an organizational unit or a process model collection, should be modelled in a consistent and similar manner" [Smirnov, Weidlich et al. 2009], so requirements engineering methods should offer modelling support that increases the homogeneity of the modelling efforts. We intend to investigate the analysts' impression about whether the compared requirements specification techniques provide this kind of support.

- Appropriate modularity of the functional specification. As argued in Section 4.3.1.2, business process models (also functional models) should define different layers to enhance modularity, and within each layer, well-defined guidelines for encapsulating the system dynamics should be offered. We intend to investigate the analysts' impression about whether the compared requirements specification techniques facilitate creating models with an appropriate modularity.

Next section, we present an experimental study, which was designed and reported following the recommendations provided by Juristo et al. [Juristo and Moreno 2001] and Wohlin et al. [Wohlin, Runeson et al. 2000].

6.5 Controlled experiment: comparison of requirements specification methods

6.5.1 Previous empirical evaluations

Some empirical works assess the quality of functional requirements models. Special interest has been placed in evaluating use case specification guidelines. In regard to completeness, two approaches can be distinguished: assessing the completeness of a (single) use case description, rating the completeness of a use case model based in the reviewer's judgement, and measuring the size of the use case model.

Some works focus on a single use case and analyse its detailed textual description. For instance, an experiment by Achour et al. [Achour, Rolland et al. 1999] lays emphasis on the specification of the use case flow of actions. Cox and Phalp [Cox and Phalp 2000] replicate the previous experiment and extend the marking scheme with subjective metrics. Instead, as explained above, we are interested in assessing the whole functional requirements specification (i.e. a complete business process).

Other works focus on the whole use case model but rate completeness in terms of a value judgement. Yadav et al. [Yadav, Bravoco et al. 1988] propose assessing completeness by having an expert review committee rate each model on a 1 to 7 scale, based on judgement. In experiments by Moody et al. [Moody, Sindre et al. 2002; Moody, Sindre et al. 2003], quality ratings are also given on a 7 point Likert scale, from 1 (poor) to 7 (excellent). We advocate using metrics rather than ratings.

In [Anda, Sjøberg et al. 2001], the authors use a 0 to 3 scale to rate several properties of the model (mainly related to correctness). In regard to completeness, the paper states that "the number of identified actors and use cases [...] indicate quality of the guidelines—the higher number, the better quality". We believe that this statement is only sensible if just valid actors and use cases are counted and if use cases have an appropriate modularity, but the paper does not give a more detailed explanation.

Fortuna et al. [Fortuna, Werner et al. 2007] describe an experiment that considers functional coverage and granularity homogeneity in order to validate unity criteria for Use Cases; however, the procedure is not sufficiently described to learn lessons from it.

6.5.2 Overview of the compared methods

Use Cases

Use Cases is a requirements engineering method proposed by Jacobson [Jacobson 1987] and later revised by many authors [Cockburn 1997; OMG 2010b]. "A use case is a collection of possible sequences of interactions between the system under discussion and its external actors, related to a particular goal" [Cockburn 1997]. Use Cases have been used in many industrial projects [Dobing and Parsons 2006] but they have also generated strong debates on their usefulness [Simons 1999]. Use Cases can be modelled graphically by means of the Use Case Diagram. Actors represent users that interact with the system under development. With regard to a use case an actor can play either a primary actor or a secondary actor role. Extension, inclusion, and generalisation relations can be established among use cases. Figure 141 shows some of the modelling primitives of the Use Case Diagram technique (see [OMG 2010b] for more detail); the figure also shows the relation between the Use Case Diagram technique and the abstract primitives. *Functional encapsulations* correspond to use cases. It is by means of a use case that the functionality of the system is encapsulated; in other words, use cases are modules that represent functions.

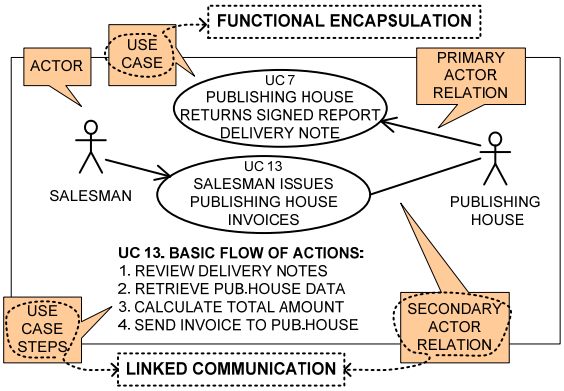


Figure 141. Use case diagram fragment

For instance, "UC 7 Publishing house returns signed report delivery note" encapsulates functionality related to the organizational work practice. *Linked communications* typically appear in the use case description (e.g. as a step in the flow of actions), but may as well appear as a relation between a use case and an actor. For instance, the fact that an invoice needs to be sent to a publishing house whenever the salesman issues monthly invoices is represented in two different ways: the use case "UC 13" has a secondary actor relation with "Publishing house" (which means that publishing houses are involved somehow in that use case) and step 4 in the "UC 13" flow of actions clarifies the type of communication.

Many works propose guidelines for use case modelling [Rolland and Ben Achour 1998; Constantine and Lockwood 1999]; guidelines by Cockburn [Cockburn 2000] were chosen for the experiment because he explicitly proposes unity criteria that are based on user goals. Following Cockburn's methodological guidelines, the requirements engineer needs to take the following steps [Cockburn 2000]:

1. Enumerate and define the actors involved.
2. Identify actor goals (see related guidelines below).
3. Make a list of use cases and a use case diagram, based on the list of goals defined in step 2.
4. Identify chances of reuse and update the diagram, making use of inclusion and extension relations.
5. For each use case, create a use case description template; specify the header fields (e.g. use case name, primary actor, preconditions) and give a brief textual description of the use case.
6. Define the basic flow of events and the alternative flows of actions.

Cockburn distinguishes several levels of goals to which use cases are related. The level of greatest interest is the user goal, i.e. the goal of the primary actor trying to get work done. In order to assess whether a goal is a user goal, Cockburn offers as guidelines two questions and a heuristic [Cockburn 2000] (we consider them to be unity criteria for encapsulation):

- "Can you go away happy after having done this?"
- "Does your job performance depend on how many of these you do today?"
- A user goal passes the "one person, one sitting (2-20 minutes)" test.

Communication Analysis

Figure 142 shows the some of the modelling primitives of the Communicative Event Diagram technique (see Section 4.3.1 for more detail); the figure also shows the relation between the notation and the abstract primitives.

Functional encapsulations correspond to communicative events. For instance, communicative event "PHO 7 Publishing house returns signed report delivery note" encapsulates part of the functionality of the system. *Linked communications* correspond to outgoing communicative interactions. Unlike the Use Case Diagram technique, the Communicative Event Diagram has a specific modelling primitive devoted to linked communications. For instance, the outgoing arrow from "PHO 13" to the actor "Publishing house" indicates that, whenever the salesman issues an invoice, the invoice has to be sent to the publishing house (it represents a message conveyance).

Following Communication Analysis guidelines, the requirements engineer needs to take the following steps:

1. Define the purpose of the organisation, identifying the main business objects and the main functions the users want to apply to each business object.
2. For each business object, identify the initial communicative event that creates the object.
3. Assign an identifier and give the communicative event an appropriate name.
4. Define the encapsulation of the communicative event (see unity criteria below).
5. Identify other communicative events that affect the business object under analysis. For each new communicative event repeat steps 3 and 4.
6. Identify outgoing communicative interactions (i.e. output messages) that users need to carry out their work.

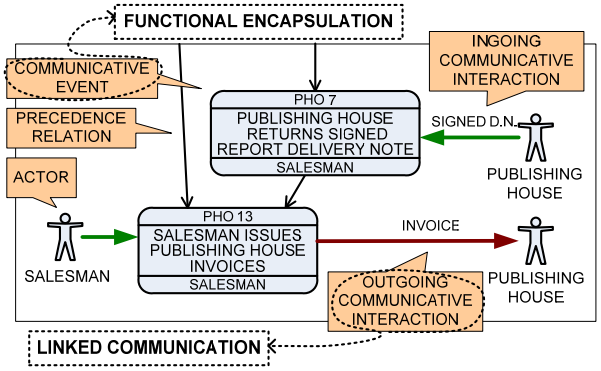


Figure 142. Communicative Event Diagram fragment

Communication Analysis offers the following unity criteria, which serve as guidelines for identifying (encapsulating) and modelling communicative events (see Section 4.3.1.2):

- *Trigger unity*. The event occurs as a response to an external interaction and, therefore, some actor triggers it.
- *Communication unity*. Each and every event involves providing new meaningful information. This input message needs to be specified.
- *Reaction unity*. The event is a composition of synchronous activities. Events are asynchronous among each other.

The modelling notation of both techniques is quite similar. However, their modelling primitives have distinct underlying concepts and, therefore, the unity criteria are different.

6.5.3 Experimental planning

The goal of our experiment, according to the goal/question/metric template [Basili and Rombach 1988], is to:

- **analyse** functional requirements models
- **for the purpose of** carrying out a comparative evaluation of RS methods
- **with respect to** their actual effectiveness and perceived efficacy
- **from the viewpoint of** the information systems researcher
- **in the context of** computer science master students acting as surrogates for analysts.

With respect to the actual effectiveness of the requirements specification methods, the experiment addresses the following research questions[103]:

RQ1. Will the subjects applying Communication Analysis produce functional requirements models with higher degree of completeness than the subjects applying Use Cases?

RQ2. Will the subjects applying Communication Analysis produce functional requirements models with a more appropriate modularity than the subjects applying Use Cases?

With respect to the perceived efficacy of the requirements specification methods, the experiment addresses the following research questions:

RQ3. Will Communication Analysis be perceived as easier to use than Use Cases?

RQ4. Will Communication Analysis be perceived as more useful than Use Cases?

6.5.3.1 Experimental context

The selected subjects were 36 computer science master students enrolled in the 2007-2008 "Conceptual Modelling of Information Systems" course at Universitat Politècnica de València, Spain. Participation was anonymous (aliases were used instead of names) and students were ensured that performance would not influence academic marks.

Previous to this course, students have taken several courses on programming, design and analysis (a bottom-up pedagogical approach is followed), so they have knowledge on data structures and algorithms, organisations and information systems, structured analysis and design, object-oriented analysis, database design and technologies, and requirements engineering principles (see the Faculty of Computer Science website for more details on the degree http://www.fiv.upv.es/default_i.htm).

6.5.3.2 Experimental design

As we want to compare two treatments (that is, both requirements engineering methods) against each other; a *paired comparison design* [Wohlin, Runeson et al. 2000] was planned: each subject uses both treatments on the same object; to minimise the effect of the order in which subjects apply the treatments, both orders need to be considered. As Table 49 shows, the experiment was carried out in two groups; each subject applied each of both methods (treatment) to specify

[103] Do not confuse these research questions with those formulated in Section 1.2. The whole experiment is related to the original question *RQ4*. How can Communication Analysis be validated as a requirements engineering method?

the requirements of a photography agency information system (the experimental object). This was enacted in two rounds.

Table 49. Paired comparison design

Subjects	Treatment (A) Communication Analysis	(B) Use Cases
Group 1	1st round	2nd round
Group 2	2nd round	1st round

The experimental object is a four pages textual description of the needs for an information system that supports the work practice of a photography agency (it also includes some organizational forms). This enterprise acts as an intermediary between photographers that provide illustrated reports and publishing houses that request and buy those reports.

Using two different experimental objects (two information system descriptions) would have avoided the fact that subjects already know the problem domain when they face the second round. However, we chose to use only one experimental object due to timing constraints (each round lasted several weeks) and also to avoid tiredness effects in the subjects that would affect their performance and their perceptions of the methods.

It should be noted that, while similar experiments have used as experimental object a single use case [Achour-Salinesi, Opdahl et al. 2002] [Anda, Sjøberg et al. 2001], our intention was to assess the guidelines of the respective methods for dealing with complete information systems, so we needed a bigger experimental object. The reference model (built by a committee of expert modellers) has 13 use cases; its corresponding software application (built by a software development company) has 537 IFPUG function points.

6.5.3.3 Variables

We identified three types of variables [Juristo and Moreno 2001]:

Response variables

In our study, functional completeness (functional encapsulation completeness, linked communication completeness), modularity (aggregation errors, fragmentation errors), perceived ease of use (PEOU), and perceived usefulness (PU) were identified as outcomes of the experiment (a.k.a. dependent variables). Section 6.4.3 defines the metrics: *degFEC*, *degLCC*, *errFra*, and *errAgg* for the first two response variables. The latter two variables were measured using a

questionnaire with several items intended to gather users' perceptions by means of a rating in the form of a 5-point Likert scale: *ratPEOU* and *ratPU*.

Factors

The requirements engineering method was identified as a variable that could affect the response variables (a.k.a. independent variable). Two treatments were considered: (1) Use Cases (mainly Use Case diagram and textual use case descriptions) and (2) Communication Analysis (mainly Communicative event Diagram). Another factor is the group to which the subject belongs, since each group applied the methods in a different order.

Parameters

Variables that we do not want to influence the experimental results have been fixed: application domain, complexity of the information system (problem statement) and previous requirements engineering experience.

6.5.3.4 Hypotheses

The hypotheses formulated from the research questions defined above are the following (a suffix is added to the variable name to indicate the technique: *CA* stands for Communication Analysis, *UC* stands for Use Cases):

Hypothesis 1 (*RQ1*)

H_{10} (null hypothesis): $degFEC\_CA = degFEC\_UC$

Use Cases and Communication Analysis allow obtaining requirements models with same degree of functional encapsulations completeness.

H_{11} (alternative hypothesis): $degFEC\_CA > degFEC\_UC$

Communication Analysis allows obtaining requirements models with greater degree of functional encapsulations completeness than Use Cases.

Hypothesis 2 (*RQ1*)

H_{20} (null hypothesis): $degLCC\_CA = degLCC\_UC$

Use Cases and Communication Analysis allow obtaining requirements models with same degree of linked communications completeness.

H_{21} (alternative hypothesis): $degLCC\_CA > degLCC\_UC$

Communication Analysis allows obtaining requirements models with greater degree of linked communications completeness than Use Cases.

Hypothesis 3 (*RQ2*)

H_{30} (null hypothesis): $errFra\_CA = errFra\_UC$

Use Cases and Communication Analysis allow obtaining requirements models with same number of functional fragmentation errors.

H_{31} (alternative hypothesis): *errFra_CA* < *errFra_UC*

Communication Analysis allows obtaining requirements models with less functional fragmentation errors than Use Cases.

Hypothesis 4 (*RQ2*)

H_{40} (null hypothesis): *errAgg_CA* = *errAgg_UC*

Use Cases and Communication Analysis allow obtaining requirements models with same number of functional aggregation errors.

H_{41} (alternative hypothesis): *errAgg_CA* < *errAgg_UC*

Communication Analysis allows obtaining requirements models with less functional aggregation errors than Use Cases.

Hypothesis 5 (*RQ3*)

H_{50} (null hypothesis): *ratPEOU_CA* = *ratPEOU_UC*

Use Cases and Communication Analysis are equally perceived as easy to use.

H_{51} (alternative hypothesis): *ratPEOU_CA* > *ratPEOU_UC*

Communication Analysis is perceived as easier to use than Use Cases.

Hypothesis 6 (*RQ4*)

H_{60} (null hypothesis): *ratPU_CA* = *ratPU_UC*

Use Cases and Communication Analysis are equally perceived as useful.

H_{61} (alternative hypothesis): *ratPU_CA* > *ratPU_UC*

Communication Analysis is perceived as more useful than Use Cases.

6.5.3.5 Instrumentation

The instruments used in this experiment include the demographic questionnaire, the experimental object, training materials, and the post-task survey.

The demographic questionnaire consisted on 26 questions (5-point Likert scale) aimed at assessing the subjects level of knowledge on several information system analysis and design methods; their actual experience with those methods; their appreciation on how important those methods are in industrial projects; whether, to the best of their knowledge, there exist enough methodological support and guidelines for applying those methods; the size complexity of software they have programmed before; and the degree of modularity of the software they develop.

The experimental object is a problem statement that describes in natural language the structure and business processes of a photography agency; it includes scanned samples of business forms.

The training materials are the following: a set of instructional slides on Use Cases guidelines based on the work of Cockburn [Cockburn 2000], and a set of instructional slides on Communication Analysis guidelines as fully explained in

Section 4.3.1 and summarised in Section 6.5.2. Small exercises were also proposed and solved by the students during the classes.

The post-task survey includes 13 closed questions (5-point Likert scale) that were identified for measuring response variables. Perceived ease of use was measured using 6 items in the survey (Questions Q1, Q2, Q4, Q6, Q10, and Q12); perceived usefulness was measured using 7 items in the survey (Questions Q3, Q5, Q7, Q8, Q9, Q13, and Q14).

6.5.3.6 Experimental procedure

The experimental procedure is depicted in Figure 143 and explained next.

Demographic questionnaire

With the purpose of identifying the background and experience using different specification techniques, such as Data Flow Diagram, Use Cases Diagram, Activity Diagram, Entity Relationship Diagram, Class Diagram, and Workflow Diagram; a demographic questionnaire was applied. 62.5% of the students answered to have a good knowledge about the syntaxes of the Entity Relationship Diagram (score superior to 3 points using a 5-Likert scale). Experience using this technique was also good since 45% reported having to solve complex case studies. However, 83% of the students had little knowledge about the syntaxes of Use Case Diagram (score inferior to 3 points using a 5-likert scale). Similar results were obtained for other specification techniques. Therefore, prior to the experimental task, the subjects were trained to use adequately the respective guidelines of Use Cases and Communication Analysis.

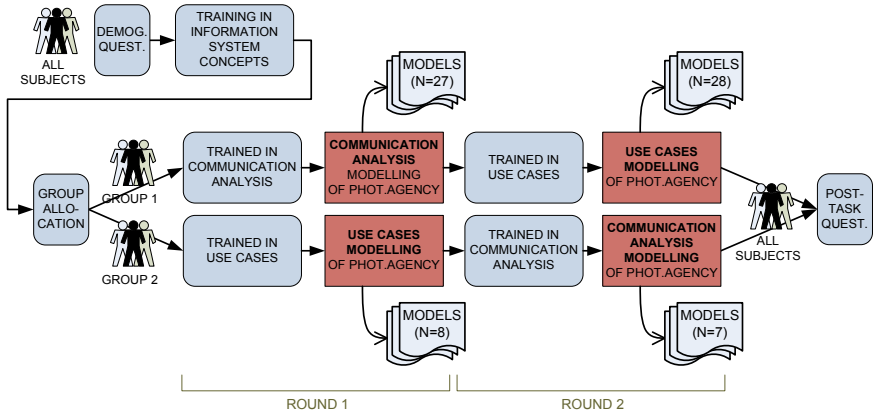


Figure 143. Experimental procedure

Allocation of teams to treatment groups

The groups were formed according to student's availability. A schedule of sessions for the experiment was made available and the subjects decided which group they were able to attend. At the time of choosing the group, the subjects knew very little details of the experiment, and did not know which methods were being tested and in which order. Their decision was, in principle, based on their availability. All in all, this was a bad move that led to unbalanced groups (as discussed later) but there is no evidence that the allocation induced any bias.

Treatment: training in a requirements modelling method

The experiment involved training the subjects on two requirements engineering methods. The time used for training in each requirements engineering method was 8 hours distributed over 4 days.

During the first round, Group 1 was trained in Communication Analysis, and Group 2 was trained on Use Cases. Then the subjects received a natural language problem statement describing the structure and processes of a photography agency. They applied the requirements specification method to specify the needed information system. The second round was analogous but now Group 1 was trained on Use Cases, and Group 2 was trained on Communication Analysis. Figure 144 shows two of the models collected during round 1.

Experimental task: modelling a case

In each round, after the training the subjects had to individually model the Photography Agency case. They were handed the textual description and given some time for reading it carefully. Both groups received the same textual description. The total time to complete the modelling task (including the time to read the description) was 2 hours. The subjects created a model in paper and handed it back to the researchers. The task instructions requested the models to contain the following information:

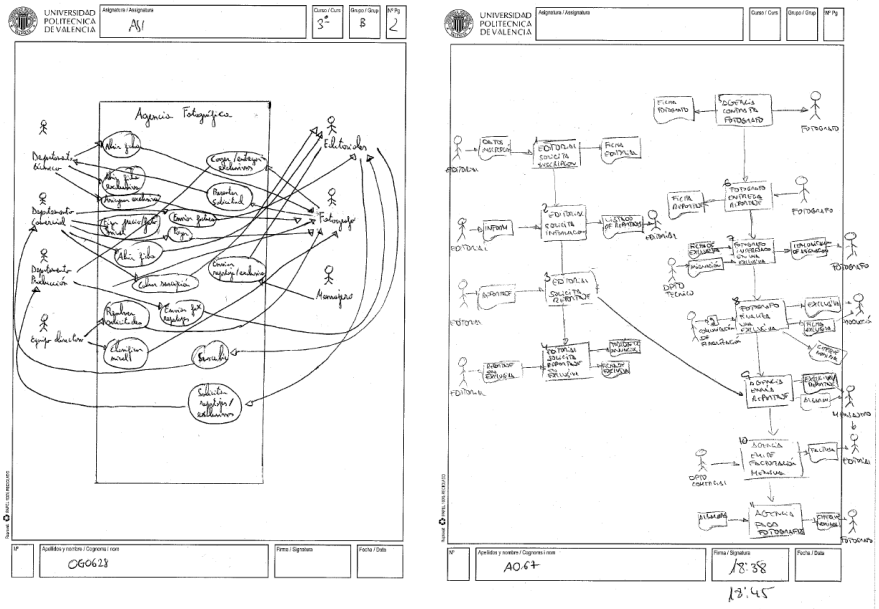
- Use Cases models should include (1) a list of organisational roles and their goals, (2) a use case diagram depicting use cases, primary and secondary actors, and (optionally) packages representing sub-systems, (3) for each use case, a use case template containing the actors involved, the type of operation (create, read, update, modify, or any combination of these), the basic flow of events and (if any) the alternative flows of events. Figure 144.a shows an example of a use case diagram.
- Communication Analysis models should include (1) a communicative event diagram depicting communicative events, the precedence relations among them, primary and receiver actors, ingoing communicative interactions, and (optionally) outgoing communicative interactions, (2) for each communicative event, an event description template containing the actors involved, the event

precondition, the message structure, and a description of the organisational reaction. Figure 144.b shows an example of a communicative event diagram.

Not all subjects produced complete models, but in all cases the use case diagram and the communicative event diagram was included in the model.

Post-task questionnaire

In order to capture the subjects' first impressions after using the respective methods to specify the Photography Agency system, they responded a questionnaire. This post-task questionnaire was handed out printed in paper; the researchers later transcribed the responses and the collected data was used to evaluate the perception-based variables.



a) Use-case diagram **b)** Communicative event diagram

Figure 144. Two models of the Photography Agency case by two different subjects

6.5.4 Results analysis and interpretation

6.5.4.1 Actual effectiveness of the requirements specification methods

An expert reviewer used the reference model and the unity criteria to aid him in the correction of the requirements models. The measures obtained by the expert reviewer were analyzed (see Table 50).

The degree of functional encapsulations completeness w.r.t. a reference model (*degFEC*) is 81% for Communication Analysis and 75.64% for Use Cases. A bigger difference appears in the degree of linked communications completeness (*degLCC*), with up to 75% for Communication Analysis and only 50.46% for Use Cases. With regard to modularity errors, both error measures indicate that subjects applying Communication Analysis perform better than subjects applying Use Cases.

Table 50. Descriptive statistics

Measure		**N**	**Mean**	**Std. dev.**
Degree of functional encapsulations completeness	*degFEC_CA*	34	0.8100	0.11710
	degFEC_UC	36	0.7564	0.11557
Degree of linked communications completeness	*degLCC_CA*	34	0.7500	0.22937
	degLCC_UC	36	0.5046	0.15164
Functional fragmentation errors	*errFra_CA*	34	1.8824	0.97746
	errFra_UC	36	2.1944	1.75368
Functional aggregation errors	*errAgg_CA*	34	0.5588	0.78591
	errAgg_UC	36	1.2222	0.68080

6.5.4.1.1 Functional completeness w.r.t. a reference model

By applying the Kolmogorov–Smirnov test, we noted that both *degFEC* and *degLCC* measures had a normal distribution ($p > 0.5$); the paired sample test was applied to verify the null hypotheses H_{10} and H_{20}.

Table 51. Paired samples test for completeness measures (*degFEC* and *degLCC*)

	95% confidence interval of the difference			**Sig. (2-tailed)**
	Lower	**Upper**	**t**	
degFEC_CA – degFEC_UC	–0.00178	0.10133	1.96	0.058
degLC_CA – degLC_UC	0.15657	0.33362	5.63	0.000

As seen in Table 51, there is a medium significance difference (p = 0.05) between the Use Cases and the Communication Analysis techniques, with respect to the degree of functional encapsulations completeness (*degFEC*). Besides, there is a very high significance difference (p = 0.000) with respect to the degree of linked communications completeness w.r.t. a reference model (*degLCC*). Therefore, H10 and H_{20} are refuted with a 95% confidence and the alternative hypotheses H_{11} and H_{21} are corroborated. This means that Communication Analysis allows obtaining RS with greater degree of functional encapsulations and linked communications completeness than Use Cases.

This outcome can be explained by the fact that Communication Analysis methodological guidelines for the identification and specification of system functions follow a more systematic procedure than Use Cases guidelines. Also, Communication Analysis approaches functional requirements specification from a business process perspective; this way, temporal precedence relations of the specified processes facilitate the discovery of missing communicative events, thus contributing to higher completeness. Moreover, the Communicative Event Diagram technique devotes a modelling primitive to linked communications (outgoing communicative interactions) so they are explicitly specified.

6.5.4.1.2 Appropriate modularity

The Kolmogorov–Smirnov test was applied to normality test of both *errFra* and *errAgg* measures. As we note that only *errFra* measure had a normal distribution (p > 0.5), Paired Sample Test was also applied to verify the null hypothesis H_{30}. We note in Table 52 that, with respect to fragmentation errors (*errFra*), there is not a significant difference (p = 0.352) between both methods.

Table 52. Paired samples test for fragmentation error measures (*errFra*)

	95% confidence interval of the difference			**Sig. (2-tailed)**
	Lower	**Upper**	**t**	
errFra_CA − errFra_UC	−0.92842	0.34018	−0.94	0.352

Therefore, hypothesis H_{31} was not corroborated. By using the Wilcoxon signed-rank non-parametric test for verifying null hypothesis H_{40}, we observe in Table 53 that 20 out of 34 subjects made a greater number of functional aggregation errors when applying Use Cases than when applying Communication Analysis. This statistical differences presented a high significance level (see Table 54, p = 0.002). Therefore, hypothesis H_{41} was corroborated with 95% confidence.

Table 53. Wilcoxon signed-rank test for fragmentation error measures (*errAgg*)

Items	Ranks	N	Mean rank	Sum of ranks
errAgg_CA− errAgg_UC				
[a] *errAgg_CA < errAgg_UC*	Negative ranks	20[a]	13.63	272.50
[b] *errAgg_CA > errAgg_UC*	Positive ranks	5[b]	10.50	52.50
[c] *errAgg_CA = errAgg_UC*	Ties	9[c]		
	Total	34		

Applying Communication Analysis leads to functional requirements specifications with a more appropriate modularity than applying Use Cases (H_{41} was corroborated). This outcome may be explained by the fact that Communication Analysis methodological guidelines for Communicative Event Diagrams are based on more objective and prescriptive criteria that those of Use Case Diagrams.

Table 54. Test statistic-significance

	errAgg_CA − errAgg_UC
Z	−3.062
Asymp. Sig. (2-tailed)	0.002

6.5.4.2 Perceived efficacy of the requirements specification methods

6.5.4.2.1 Perceived ease of use

First, the scores of each subject were averaged over the six questions that are relevant for determining the perceived ease of use (Q1, Q2, Q4, Q6, Q10, and Q12). Descriptive statistics were then calculated for both methods (see Table 55).

Table 55. Descriptive statistics for the perceived ease of use (*PEOU*)

Statistics	**Communication Analysis (*PEOU_CA*)**	**Use Cases (*PEOU_UC*)**
N	22	25
Average	3.04	2.98
Standard dev.	0.69	0.72
Minimum	1.67	1.67
Maximum	4.17	4.33

Note that the averages obtained are close to the value 3 (middle score on the Likert scale). Values ranged from a low of 1.67 to a maximum of 4.17 (A) and 4.33 (B). Furthermore, standard deviations of 0.69 (A) and 0.72 (B) were obtained, implying that the averages obtained for both techniques are representative.

In order to verify the null hypothesis H_{50}, the Wilcoxon signed-rank non-parametric test was applied, by comparing the averages of two samples of the respective techniques to determine whether there are differences between them.

In Table 56, we observe that 13 out of 19 subjects perceived Use Cases as easier to use than Communication Analysis. However, we have not detected a significant level (p = 0.559). A possible interpretation for this outcome is that Use Cases guidelines are expressed in more informal terms than Communication Analysis guidelines; this fact makes them more understandable at first glance (that is, they are more intuitive).

Table 56. Wilcoxon signed-rank test for ease of use (*PEOU*)

Items	**Ranks**	**N**	**Mean rank**	**Sum of ranks**
PEOU_UC – PEOU_CA				
[a] *PEOU_UC < PEOU_CA*	Negative rank	6[a]	13.42	80.50
[b] *PEOU_UC > PEOU_CA*	Positive rank	13[b]	8.42	109.50
[c] *PEOU_UC = PEOU_CA*	Ties	0[c]		
	Total	19		

6.5.4.2.2 Perceived usefulness

We averaged the scores assigned by each subject over the seven relevant questions for determining perceived usefulness (Q3, Q5, Q7, Q8, Q9, Q13, and Q14). Descriptive statistics were then calculated for both RE methods (see Table 57). In order to determine whether significant differences exist between both methods with respect to perceived usefulness, null hypothesis, H_{60}, was verified by using the Wilcoxon signed-rank test.

Table 57. Descriptive statistics for perceived usefulness (*PU*)

Statistics	**Communication Analysis (*PU_CA*)**	**Use Cases (*PU_UC*)**
N	22	25
Average	3.48	3.39
Standard dev.	0.39	0.55
Minimum	2.71	1.86
Maximum	4.17	4.80

As we can see in Table 58, 10 out of 19 subjects perceived Communication Analysis as more useful than Use Cases. This difference was statistically significant at medium level (p = 0.024). A possible interpretation for this outcome is the fact that the Communicative Event Diagram allows specifying more analytical information (e.g. communicative interactions and precedence relations, which are external aspects of the information system) than Use Case Diagrams (e.g. «include» relations are typically used for decomposition and reuse purposes, which are design-time decisions [Simons 1999] and Jacobson even discourages their use [Jacobson 1987]). Also, Communication Analysis guidelines, although perceived as less easy to understand, are more prescriptive and, therefore, may lead to more homogeneous specifications. Moreover, the unity criteria for communicative events are based on systems theory and communication theory in order to improve the adequacy of event modularity.

Table 58. Wilcoxon signed-rank test for perceived usefulness (*PU*)

Items	**Ranks**	**N**	**Mean rank**	**Sum of ranks**
PU_UC – PU_CA				
[a] *PU_UC < PU_CA*	Negative rank	10[a]	9.05	90.50
[b] *PU_UC > PU_CA*	Positive rank	6[b]	7.58	45.50
[c] *PU_UC = PU_CA*	Ties	3[c]		
	Total	19		

To clarify the previous point, we analyzed separately each one of the questions Q8, Q9 and Q13, which respectively relate to: (1) Distinction between external and internal interaction; (2) Homogeneity of functional specification; (3) Adequate level of modularity of functional specification.

In Table 59, we note that 9 out of 18 subjects perceived that Communication Analysis facilitates specifying functional requirements with a more appropriate level of modularity than the Use Cases. Only 3 out of 18 subjects perceived the opposite. This statistically difference presented a medium significance level (p = 0.04).

The number of ties in for the other two questions does not allow drawing any conclusions about them.

Table 59. The Wilcoxon signed-rank test for items Q8, Q9, and Q13 related to perceived usefulness

Items	Ranks	N	Mean rank
Q8_UC − Q8_CA			
[a] *Q8_UC < Q8_CA*	Negative rank	5[a]	4.50
[b] *Q8_UC > Q8_CA*	Positive rank	4[b]	5.63
[c] *Q8_UC = Q8_CA*	Ties	9[c]	
	Total	18	
Q9_UC − Q9_CA			
[d] *Q9_UC < Q9_CA*	Negative rank	4[d]	4.00
[e] *Q9_UC > Q9_CA*	Positive rank	2[e]	2.50
[f] *Q9_UC = Q9_CA*	Ties	11[f]	
	Total	17	
Q13_UC − Q13_CA			
[g] *Q13_UC < Q13_CA*	Negative rank	9[g]	7.00
[h] *Q13_UC > Q13_CA*	Positive rank	3[h]	5.00
[i] *Q13_UC = Q13_CA*	Ties	6[i]	
	Total	18	

6.5.5 Validity evaluation

This section discusses issues with the potential to threaten the validity of the experiment [Wohlin, Runeson et al. 2000].

6.5.5.1 Conclusion validity

We verified that the subjects had a homogeneous background by means of a questionnaire, so there is no threat due to random heterogeneity of subjects, which could give rise to greater variability in the measures. As a trade-off, homogeneity limits external validity.

The proposed measures related to actual efficacy have been theoretically reasoned and intend to be quite objective. However, an empirical testing of the metrics is advisable, in order to ensure the reliability of measures; this is planed as future work. Also, we plan to have the subjects' models reviewed by more expert reviewers and to perform agreement rounds; this will allow assessing the level of objectiveness of the metrics and the measuring procedure (inter-reviewer agreement).

With respect to the reliability of perception-based measures, we conducted a reliability analysis on the survey using the Chronbach alpha technique. The generic value obtained was 0.72, indicating that the items on the survey are

adequately reliable. However, according to Garson [Garson 2008], this score could be improved with a cut-off of 0.80 for a "good" scale. In the social psychological literature there is evidence that attitude scales, as well as skills scales, are reliable [Shaw and Wright 1967].

6.5.5.2 Internal validity

Instrumentation is the effect caused by the instruments used in the experiment; in particular, the fact that paper form surveys are error-prone. To minimise this threat, the transcription of paper forms into spreadsheets and statistical analysis tools was double-checked by two experimenters. However, we could not avoid subjects making errors which reduced the number of valid observations (e.g. identification of the evaluated method, identification of subject, unanswered questions). Therefore, in the future, we plan to use software-based surveys that prevent errors while subjects fill out relevant data.

There is a risk related to the allocation of subjects to groups. Letting students decide which group to join according to their availability was a mistake; it led to imbalanced groups and, even though we perceived no evidence that the resulting groups had different characteristics (i.e. motivation), this cannot always be assumed. A lesson learned is that subjects should be allocated randomly. We acknowledge a threat of maturation; that is, during the second round, the subjects already know the photography agency problem statement. We intended to also make a comparison of the results of the first round, but this was not possible due to the imbalance between both groups.

6.5.5.3 Construct validity

Two of the researchers involved in the experiment are authors of Communication Analysis who have expectancies about this method performing better. In order to reduce the threat of bias, two experimenters without expectancies have been involved.

A reference model of low quality is a threat. However, the authors have been using the Photography Agency case for research and education for more than 10 years, and its conceptual model is highly agreed by now. A three-person expert modelling committee made the final adjustments.

The experiment includes a single information system description so it may under-represent the construct of all information systems. Since the subjects were trained in both methods, the results of the second round may be affected by their previous knowledge. Again, the imbalance between groups did not allow a comparison of the first round.

Table 60. Inter-item correlation analysis of the perception-based variables

	Perceived ease of use (*PEOU*)						**Perceived usefulness (*PU*)**							**Validity**		
	Q1	***Q2***	***Q4***	***Q6***	***Q10***	***Q12***	***Q3***	***Q5***	***Q7***	***Q8***	***Q9***	***Q13***	***Q14***	**CV**	**DV**	
Q1	1,00	0,27	0,61	0,35	0,80	0,78	0,45	0,24	0,14	0,40	0,19	0,06	0,04	0,63	0,22	**Yes**
Q2	0,27	1,00	0,23	0,33	0,30	0,38	0,26	0,24	0,50	0,34	0,04	0,09	0,07	0,42	0,22	**Yes**
Q4	0,61	0,23	1,00	0,46	0,72	0,65	0,32	0,45	0,42	0,38	0,28	0,11	0,18	0,61	0,31	**Yes**
Q6	0,35	0,33	0,46	1,00	0,51	0,48	0,39	0,66	0,31	0,27	0,18	0,36	0,70	0,52	0,41	**Yes**
Q10	0,80	0,30	0,72	0,51	1,00	0,76	0,37	0,38	0,20	0,63	0,20	0,18	0,11	0,68	0,29	**Yes**
Q12	0,78	0,38	0,65	0,48	0,76	1,00	0,41	0,20	0,14	0,38	0,04	-0,02	0,14	0,67	0,18	**Yes**
Q3	0,45	0,26	0,32	0,39	0,37	0,41	1,00	0,31	0,26	0,30	-0,08	0,32	0,49	0,37	0,37	**Yes**
Q5	0,24	0,24	0,45	0,66	0,38	0,20	0,31	1,00	0,44	0,49	0,32	0,35	0,51	0,49	0,36	**Yes**
Q7	0,14	0,50	0,42	0,31	0,20	0,14	0,26	0,44	1,00	0,22	0,36	0,50	0,26	0,44	0,29	**Yes**
Q8	0,40	0,34	0,38	0,27	0,63	0,38	0,30	0,49	0,22	1,00	0,19	0,48	0,07	0,39	0,40	**Yes**
Q9	0,19	0,04	0,28	0,18	0,20	0,04	-0,08	0,32	0,36	0,19	1,00	0,29	0,15	0,32	0,15	**Yes**
Q13	0,06	0,09	0,11	0,36	0,18	-0,02	0,32	0,35	0,50	0,48	0,29	1,00	0,47	0,48	0,13	**Yes**
Q14	0,04	0,07	0,18	0,70	0,11	0,14	0,49	0,51	0,26	0,07	0,15	0,47	1,00	0,42	0,21	**Yes**

In order to demonstrate that we have evidence for construct validity, an inter-item correlation analysis of the perception-based variables (*PEOU*, *PU*) was applied. To do so we used two criteria: Convergent Validity (CV), which refers to the convergence among different indicators used to measure a particular construct, and Discriminant Validity (DV), which refers to the divergence of indicators used to measure different constructs. Average DV should be lower than the average CV. The results of the validity analysis for each construct show that the CV value was higher than the DV value for all PEOU and PU items (see Table 60).

6.5.5.4 External validity

With respect to the use of students as experimental subjects, several works suggest that, to a great extent, the results can be generalised to industry practitioners [Runeson 2003]. In any case, we are aware that more experiments with a larger number of subjects are necessary.

We thoughtfully selected a representative problem statement. However, more empirical studies with other requirements specifications are necessary.

6.5.6 Discussion

A laboratory experiment has been carried out to compare two requirements engineering methods; namely Use Cases [Cockburn 2000] and Communication Analysis. Functional requirements specifications have been quantitatively

evaluated with respect to the proposed metrics and qualitatively evaluated with respect to the MEM perception variables.

The following hypotheses were verified: Communication Analysis allows obtaining requirements specifications with greater degree of functional encapsulation completeness and greater degree of linked communication completeness than Use Cases; also, Communication Analysis allows obtaining RS with less functional aggregation errors than Use Cases. With regards to functional fragmentation errors, the difference, although favourable for Communication Analysis, was not statistically significant. These outcomes can be due to the fact that Communication Analysis offers prescriptive methodological guidance to modularisation. We believe that use case-based methods would benefit from taking into account the unity criteria proposed by Communication Analysis (see Section 4.3.1.2).

In addition, our findings were also that Communication Analysis was perceived as more useful than Use Cases; the subjects perceived that it facilitates determining the appropriate level of modularity of functional specifications. Other characteristics of usefulness, namely homogeneity of functional specifications and the distinction between external and internal interactions were not empirically corroborated due to the low level of significance obtained. We also noted that Use Cases were perceived as easier to use than Communication Analysis (13 out of 19 subjects); however, this difference would have to be confirmed with an improved level of significance.

Empirical evaluations such as this experiment allow comparing methods and highlighting their strengths and weaknesses. However, theoretical evaluations allow understanding better why these differences arise. This is planned for future research.

We acknowledge that a higher number of subjects is needed to reconfirm these initial results. In addition, we plan to take into account experience in, or knowledge of, the use of requirements engineering methods as a relevant factor in further experiments. The experience collected in this first study will facilitate us to address several of the identified threats. We want to analyze in depth the "whys" of the obtained results. To do this, we plan carry to out an evaluation of actual effectiveness with respect to the other quality attributes and to analyze the causality relations between actual efficacy and perceived efficacy.

6.6 Summary

In Chapter 4, Communication Analysis has been specified in detail, including de definition of concepts based on an information systems ontology (i.e. the conceptual framework presented in Section 3.3 built upon FRISCO [Falkenberg, Hesse et al. 1998]) and the design of a platform independent metamodel. This specification has paved the way for the integration of Communication Analysis into a model-driven software development framework; namely, the OO-Method (see Chapter 5). In this chapter we present several efforts to validate the detailed specification of Communication Analysis, both from the theoretical and the empirical perspectives. As a theoretical validation, an ontological analysis of Communication Analysis has been performed (see Section 6.2.2). Since the method has been built up on top of a rigorous conceptual framework, its ontological analysis has involved mapping the constructs of the method with the constructs of the conceptual framework. Also, the constructs of the method have been mapped to metaclasses of the metamodel. This analysis has been performed twice, allowing to iteratively improve both the conceptual foundations of the method as well as the metamodel (see the discussion in Section 6.2.3). The first iteration revealed that some issues concerning the alignment between constructs of Communication Analysis and constructs of FRISCO. It also showed up that some aspects of the metamodel were capable of improvement. After some rework, the second ontological analysis was done, leading to clearer alignment.

Empirical evaluation of requirements engineering methods is a strong need in the area of requirements engineering. By means of several lab demos have demonstrated the feasibility of the approach and have provided useful feedback to improve methodological guidelines and notations (see Section 6.3). However, in order to investigate deeper in the methodological quality of Communication Analysis, controlled experiments are a more appropriate evaluation technique. Since experimentation needs to be properly founded in theoretical constructs, we have adopted the empirical framework Method Evaluation Model (MEM) proposed by Moody [2003]. We have integrated the MEM with the conceptual modelling quality framework by Lindland et al. [1994], which, in turn, we have extended by refining quality goals and proposing metrics that allow their operationalisation (see Section 6.4). The focus is put on actual efficacy and four new variables related to semantic completeness and appropriate modularity are defined. In fact, our extension to the MEM constitutes a contribution *per se*:

- The concepts of completeness and granularity are discussed from a semantic point of view and we have grounded them in sound theory. The concepts are then operationalised by proposing metrics for their quantification. Some metrics extend previous approaches; other metrics adopt a novel approach that, according to the results, is promising.

- With regard to semantic completeness, most of the previous works propose a rating based on judgement (e.g. using a Likert scale). We propose measuring the degree of functional encapsulations completeness and the degree of linked communications completeness with respect to a reference model (which is agreed by an expert modelling committee).
- Also, we propose assessing whether a requirements model has an appropriate modularity with respect to a given set of unity criteria (in our case, we selected the criteria defined in Section 4.3.1.2); this allows determining the number of functional fragmentation errors and functional aggregation errors.

We acknowledge that the proposed extension of the MEM is more practical in experimental settings than in industrial practice (where a reference model of the domain is very unlikely to be available). In any case, the proposed empirical framework has proved to be a useful tool for research in requirement engineering and it provides a valuable strategy to feedback method designers.

We have applied the above-mentioned empirical framework in a controlled experiment that has demonstrated several benefits of using Communication Analysis over using Use Cases (see Section 6.5). Although we focus on two particular methods, the proposed strategy is general enough to fit the evaluation of other methods. Evidence shows that Communication Analysis leads to models that are more complete and have less modularity errors. With respect to perceived efficacy, a questionnaire has been used. Although Use Cases is perceived to be easier to use, Communication Analysis is perceived to be more useful. These outcomes are discussed and related to the features of the methods.

We have also carried out an additional experiment that compares two different approaches to business process modularity with regards to the effect they have in the comprehensibility of requirements specifications and their suitability for requirements validation. It was designed in collaboration with Klaas Sikkel, Maya Daneva and Nelly Condori and it was conducted in University of Twente in 2010. It involved around 200 bachelor students acting as experimental subjects. However, we are still analysing the resulting data.

Chapter 7
Validation of the conceptual model derivation technique

"A theory is something nobody believes, except the person who made it.
An experiment is something everybody believes, except the person who made it."

Albert Eintein

7.1 Motivation

The same way that in Chapter 6 we have validated Communication Analysis, we also need to validate its integration with the OO-Method. Communication Analysis may well serve the purpose of discovering, specifying, analysing and communicating information system requirements. However, does the set of derivation techniques proposed in Chapter 5 serve the purpose of deriving an object oriented conceptual model that (i) represents the intended computerised information system, (ii) has a good quality and (iii) is complete enough to allow for automatic code generation? This chapter provides an answer to this question. As discussed in Section 6.1, many evaluation techniques are available. Again, we perform a theoretical validation and an empirical validation. The theoretical validation intends to ascertain whether the constructs of Communication Analysis and the OO-Method have been properly aligned. We have chosen two empirical validation techniques to evaluate the derivation technique: lab demos and a controlled experiment; the intention is to investigate whether the technique is feasible, effective and efficient.

The remaining of the chapter is structured as follows. Section 7.2 presents the theoretical evaluation of the derivation technique. For this purpose, an ontological analysis of the OO-Method is performed, as well as an analysis of its integration with Communication Analysis. Section 7.3 presents some lab demos

in which we have put the derivation technique to test. This experience has provided valuable information about the capabilities of our conceptual model derivation technique; it revealed features capable of improvement. We used the gathered feedback to extend and tune up the manual derivation rules. We present some lessons learned and we enumerate some issues that still need further investigation. Section 7.4 describes a controlled experiment that intends to assess the impact of including Communication Analysis within the current OO-Method development process. Section 7.5 describes a controlled experiment by which we have compared the performance of subjects applying a text-based derivation of conceptual models (a practice that intends to mimic what OO-Method how practitioners actually act in real development projects) with the performance of subjects applying our communication-based derivation technique (see Chapter 5). Among other conclusions, the results show that subjects applying the communication-based derivation technique produce models of higher quality that those applying the text-based derivation technique. Section 7.6 presents a summary of the chapter.

7.2 *Theoretical validation*

As part of the integration of Communication Analysis and the OO-Method, the conceptual model derivation rules defined in Section 5.5 relate constructs of Communication Analysis and the OO-Method. By means of ontological evaluations, we intend to ascertain whether the constructs of both methods have been properly aligned.

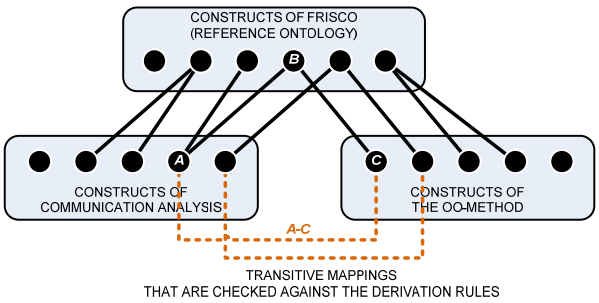


Figure 145. Investigating transitivity of ontological mappings

The procedure is as follows. We first perform the ontological evaluation of the OO-Method with respect to FRISCO 1.1; the ontological evaluation of Communication Analysis has been presented in Section 6.2.2). Then we investigate the mappings that result from both ontological evaluations; whenever

a construct of Communication Analysis (*A*, in Figure 145) is mapped to the same ontological construct (*B*) as a construct of the OO-Method (*C*), then a transitive mapping is established between both method constructs (*A*-*C*). For each transitive mapping, we investigate whether there is be a derivation rule that takes as input elements of type *A* and derives elements of type *C*. In an ideal setting, for each transitive mapping there would be one derivation rule and vice versa.

7.2.1 Ontological evaluation of the OO-Method

Table 61 presents the mapping among concepts of the OO-Method (they are briefly described in Section 5.3, further information can be found in [Pastor and Molina 2007]) and concepts of FRISCO 1.1 (i.e. the conceptual framework presented in Section 3.3). The analysis covers the Object Model, part of the Dynamic Model and the Functional Model. Both the Presentation Model and the Object Interaction Diagram (part of the Dynamic Model) are left out of the scope of the analysis, because their derivation is not targeted in this thesis.

For the sake of brevity, we refer to some concepts of FRISCO 1.1 whose definition is complex with the name of its corresponding OO-Method concept between apostrophes. For instance, we refer as "object" to a predicated thing which is not a predicator, a relationship or a state.

Table 61. Mapping among constructs of the OO-Method and the reference ontology

OO-Method construct	**FRISCO 1.1 construct**
Object Model	
object	a predicated thing which is not a predicator, a relationship or a state
class	a type of "object"
attribute	predicator of "objects"
identification function	predicator indicating that a set of predicators of "objects" (i.e. as et of attributes) act as a label
derivation formula	rule which specifies how to calculate the value of an attribute
service	action that applies to a type of "object"
inbound argument	a piece of data, part of a message, being an input actand of an action
outbound argument	a piece of data, part of a message, being an output actand of an action
data-valued argument	a piece of data (with a simple domain), part of a message
object-valued argument	a piece of data (with a type of "object" as domain), part of a message

OO-Method construct	**FRISCO 1.1 construct**
atomic service (a.k.a. event)	action that applies to a type of "object", not being a composite action
creation service	an action that applies to a type of "object" and has an "object" of such type in its post-state that is not in its pre-state
destruction service	an action that applies to a type of "object" and has an "object" of such type in its pre-state that is not in its post-state
edit service	an action that applies to a type of "object" and that has the same "object" of such type both in its pre-state and its post-state, whereby both states differ in at least one predicator that involves the "object"
shared service	an action that affects two "objects" by creating, destroying or changing relationships between both "objects"
insertion shared service	an action having the same pair of "objects" both in its pre-state and in its post-state, and having a relationship between both "objects" that is an element of the action post-state but not of its pre-state
deletion shared service	an action having the same pair of "objects" both in its pre-state and in its post-state, and having a relationship between both "objects" that is an element of the action pre-state but not of its post-state
change shared service	a composite action combining the two actions above (i.e. insertion and deletion shared service) in a state-transition structure of type sequence, where only one of the "objects" is involved in both relationships
transaction	composite action that applies to a type of "object", whereby if any rule is broken during the action occurrence, then the post-state is equal to the pre-state
transaction formula	state-transition structure
operation	composite action that applies to a type of "object", whereby even if some rule is broken during the action occurrence, the post-state can be different to the pre-state
precondition	rule that determines permissible action occurrences; it defines a condition that has to be fulfilled in by the pre-state of a specific action of a given type of "object"
integrity constraint	rule that determines permissible action occurrences; it defines a condition that has to be fulfilled by all the pre-states and post-states of any action of a given type of "object"
agent relationship	relationship between a type of actor and an action (i.e. actor-of)
agent	type of goal-pursuing actor

OO-Method construct	**FRISCO 1.1 construct**
structural relationship (association, aggregation and composition)	type of relationship between types of "objects"
cardinality	rule that determines the permissible relationships between "objects"
temporality	rule that determines the whether a relationship between "objects" can be destroyed without destroying any of the related "objects" as well
identification dependency	predicator of a type of relationship between types of "objects"
specialisation relationship	special binary relationship between a type of "objects", characterised by the special predicator called "is-parent", and another type of "objects", characterised by the special predicator called "is-child". It denotes that every "object" being member of the population of the type that "is-child" is also a member of the population of the type that "is-parent", but that the inverse does not necessarily hold
carrier service	action whereby an "object" being member of the population of "parent" becomes also a member of the population of "child"
freeing service	action whereby an "object" being member of the population of "parent" and of "child" stops being a member of the population of "child"
cluster	sub-system of the subject system, composed of several types of "objects"
Dynamic Model	
state	type of states
pre-creation state	pre-state of an action of type "creation service"
simple state	state being either the pre-state or the post-state of a service and not being a "pre-creation state" or a "destruction state"
destruction state	post-state of an action of type "destruction service"
transition	rule determining the permissible action occurrences (it restricts the permissible actors, pre-state and post-state)
control condition	rule determining the permissible action occurrences (it restricts the permissible pre-state and/or post-state)
Functional Model	
evaluation	rule determining the post-state of an action (specifically, of an "edit service" or a "shared service")
transaction formula	rule determining the post-state of an action (specifically, of a "transaction")

This alignment has the purpose of facilitating the evaluation of the derivation technique. It is out of the scope of the thesis to evaluate the OO-Method by itself. In any case, the following issues are worth clarification.

Although both preconditions and integrity constraints are rules that determine permissible action occurrences and even some works compare their expressiveness [Lawley, Topor et al. 1993; Mammar, Gervais et al. 2006], both modelling primitives have different scopes: one service and all services of a given class, respectively.

Also, there is a similarity between the constructs transition and control condition; however, in practice, they are complementary. Transitions (without a control condition) define from which pre-state to which post-state an object can shift, and which actors can trigger the action associated to the transition. Then a control condition further specifies some predicates that need to hold either in the pre-state or in the post-state. All in all, the control condition is part of the specification of a transition.

In any case, we acknowledge that the expressiveness of preconditions, integrity constraints, transitions and control conditions overlaps at some points (see a discussion in section 5.4.2.1).

7.2.2 Evaluation of the integration of Communication Analysis and the OO-Method

In Table 62 we have intended to include transitive relationships among Communication Analysis constructs and OO-Method constructs, based on their relationship with FRISCO constructs.

Some Communication Analysis constructs have been omitted because they correspond to instances; namely, business object representation (a representation of an instance of a class) communicative event occurrence (an instance of communicative event), organisational actor, primary actor (an instance of primary role), receiver (an instance of receiver role), actor support actor (an instance of support role). The only exception is business object, which is aligned to object (an instance of an OO-Method class). Despite the name, interface actor is actually a role.

Also, some constructs simply do not fit the intended investigation. For instance, communicative event unity criteria are methodological guidelines and cannot be mapped to constructs FRISCO or of the OO-Method.

Table 62 shows the transitive mappings; to understand how they have been defined, we now explain their rationale.

Table 62. Transitive mappings among method constructs

Communication Analysis construct		pivotal FRISCO 1.1 construct	OO-Method construct
L1	organisational unit	sub-system	cluster
L2	business process	sub-system	cluster
	business object	predicated thing	object
	business object class	a type of predicated thing	class
L3	business object class property	predicator of a type of predicated thing	attribute

According to Table 43, a business object is a predicated thing which is not a predicator, a relationship or a state, and it is part of the domain model. According to Table 61, an object is a predicated thing which is not a predicator, a relationship or a state. Since both method constructs are mapped to the FRISCO construct predicated thing, then a transitive mapping is defined between business object and object.

A business object class is a type of "business object". According to Table 61, a class is a type of "object". Again, a transitive mapping is defined between, a transitive mapping is defined between business object class and class.

A business object class property is a predicator of "business objects". According to Table 61, an attribute is a predicator of "objects". Again, a transitive mapping is defined between business object class property and attribute.

A business process is a sub-system of the subject system, often focused on actions and actors. Also, an organisational unit is a sub-system of the organisational system. According to Table 61, a cluster is a sub-system of the subject system, composed of several types of "objects". Now, two transitive mappings are defined, one between business process and cluster and one between organisational unit and cluster.

The first impression when attempting to find the transitive mappings is that the methods have different abstraction levels; most of the Communication Analysis constructs considered in the ontological evaluation are ascribed to layers L1, L2 and L3, whereas the constructs of the OO-Method belong to layers L3 and L4. The overlapping between both methods occurs mainly at the L4 layer.

For instance, Figure 146 shows some concepts of Communication Analysis and relations between them (in black and regular font), as well as concepts of the OO-Method and relations between them (in blue and in italics). The Communication Analysis part should be read as follows. A communicative event has a message structure (the one that corresponds to its ingoing communicative interaction) that is composed of aggregation substructures, which in turn are composed of message fields.

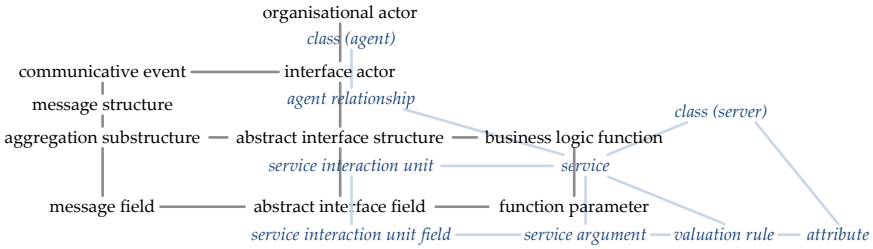


Figure 146. Related and overlapping concepts

For each aggregation substructure, one or several abstract interface structures are defined, each of them having one or several abstract interface fields that are related to the message fields. The organisational actor playing the interface actor within the communicative event is in charge of editing the message; that is, the interface actor fills the abstract interface structures. For each of those abstract interface structures, a business logic function is in charge of processing the data, which the function receives as function parameters. In turn, the OO-Method part should be read as follows. A service belongs to a specific class (referred to as server class) and has service arguments. A service can be triggered by those users for which an agent relationship exists (the user logged in an application is represented by a class, referred to as agent class). To enter the data that a service needs as input, service interaction units (a type of an interface pattern) are defined. A service interaction unit corresponds to one service and has one or several service interaction unit fields, often one per each service argument. Within each service, valuation rules define how the data in the service arguments is used to set the value of class attributes.

Figure 146 indicates (by proximity) that six constructs of both methods overlap. For instance, a Communication Analysis abstract interface structure (see page 145) is equivalent to an OO-Method service interaction unit (see [Pastor and Molina 2007]). Since interface modelling has been left out of the scope of this thesis, some Communication Analysis constructs have not been rigorously defined and operationalised in the metamodel. Similarly, the conceptual model derivation technique has omitted the derivation of the OO-Method Presentation Model. Consequently, the derivation technique considers does not consider all the concepts of both methods, but some of them as shown in Figure 147, bridging the gaps by means of derivation rules (note that some derivation rules that concern properties of those concepts or other concepts are not represented in the figure).

Our impression after attempting the theoretical validation of the method integration is that it would have been easier provided that all the method constructs had been considered in this thesis.

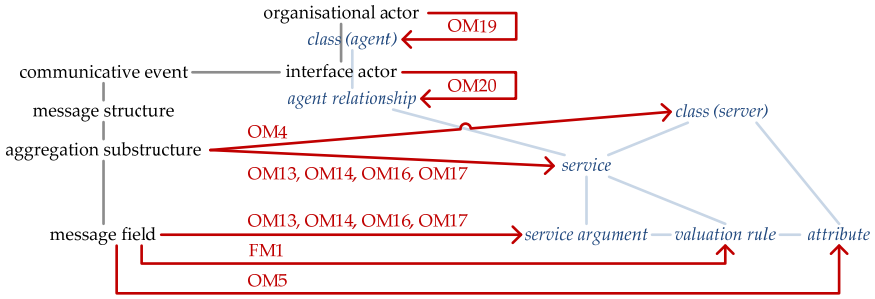


Figure 147. Related concepts and derivation rules

7.3 *Lab demos*

We have performed several lab demos in which we have taken a Communication Analysis requirements model as input for the derivation and, applying the derivation rules manually, we have derived the OO-Method conceptual models. The lab demos have been introduced in Section 6.3, as well as the interesting feedback and conclusions they provided us with regards to Communication Analysis. We now focus on the feedback the lab demos have provided with regards to the derivation technique. In some cases, the experience led us to extend the derivation technique with new derivation rules, in other cases, we rewrote some rules to improve their understanding or include some aspect that was not considered. However, we also acknowledge that our derivation rules do not cover all the possible cases; we are fully aware of some of its limitations.

Moreover, in this investigation, we received the collaboration and support of CARE Technologies. Experts in the Integranova technology created their own version of the conceptual models for the cases Projects office, Photography Agency and Master management system of TUO, taking as input a textual requirements specification of the cases. This allowed for a detailed comparison of the OO-Method conceptual models resulting from the derivation according to our guidelines and according to their own expert heuristics. The results of the comparison were very valuable. By this means we could identify some derivation rules that were not producing the outcome an expert modeller would expect.

In the following, we comment on some of the important breakthroughs related to each lab demo.

7.3.1 Photography Agency Inc.

The derivation of the conceptual model related to this case has been performed twice, before and after carrying out the Projects Office lab demo. During the first experience, we explored the derivation of the information system memory using the Entity-Relationship Diagram (ERD) modelling technique. The reason is that, both the ERD and relational models (graphical views of relational database schemas, such as those supported by CA ERwin Data Modeler) were being used in industrial projects with Anecoop S.Coop. We intended to first envision the derivation heuristics and patterns, before moving to the object orientation.

We started conceiving the data model derivation as an incremental integration of data-model views. Figure 148 shows the entity-relationship diagram of the Photography Agency case; we have zoomed the ERD view of communicative event *PHO 4 Photographer provides report*.

During the second experience, we already undertook class diagram derivation. Also, we investigated inheritance networks as a support for system dynamics (see Figure 149), as discussed in Section 5.4.2.1. However, since their expressiveness has limitations that make this approach ineffective when complex dynamics are involved, we later ruled this option out.

With regards to transactions, by comparing the researchers' conceptual model with the expert practitioner's model, we identified that it was common practice to encapsulate atomic services into a transaction (see Figure 150). We started conceiving what later has become derivation rule **FM2** (definition of a transaction formula for complex reactions).

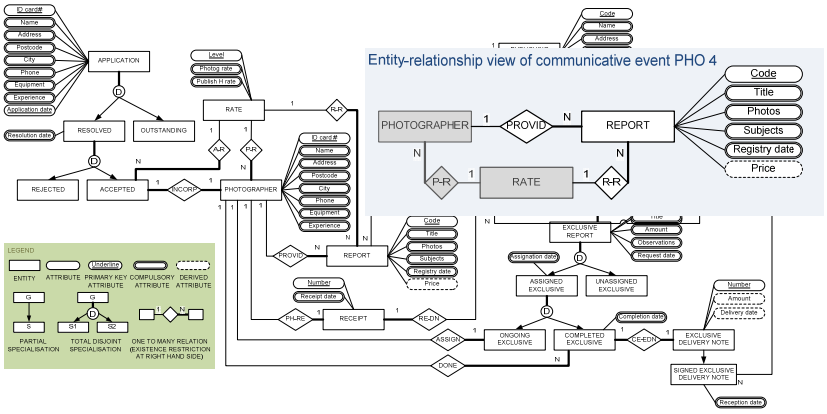


Figure 148. Entity relationship diagram derived from the Photography Agency requirements model (view of *PHO 4* is zoomed)

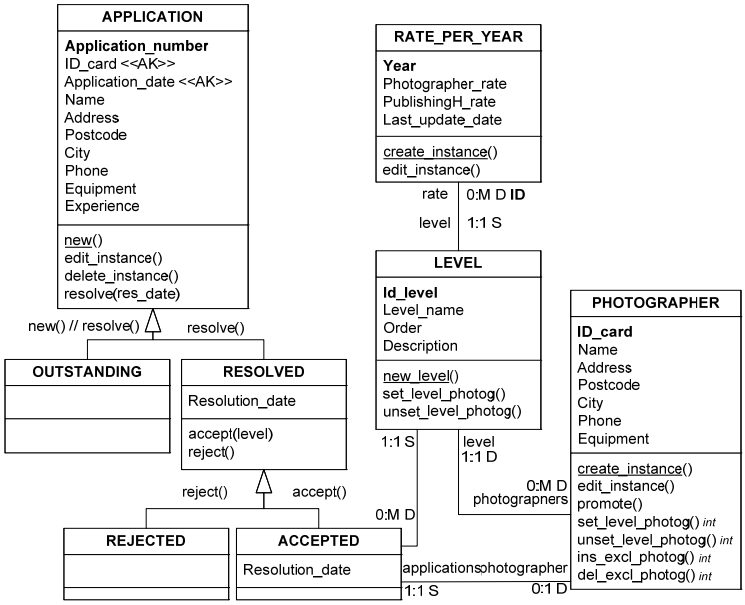


Figure 149. Part of the class diagram or the Photography Agency (integrated views of events *PHO 1* to *PHO 3*)

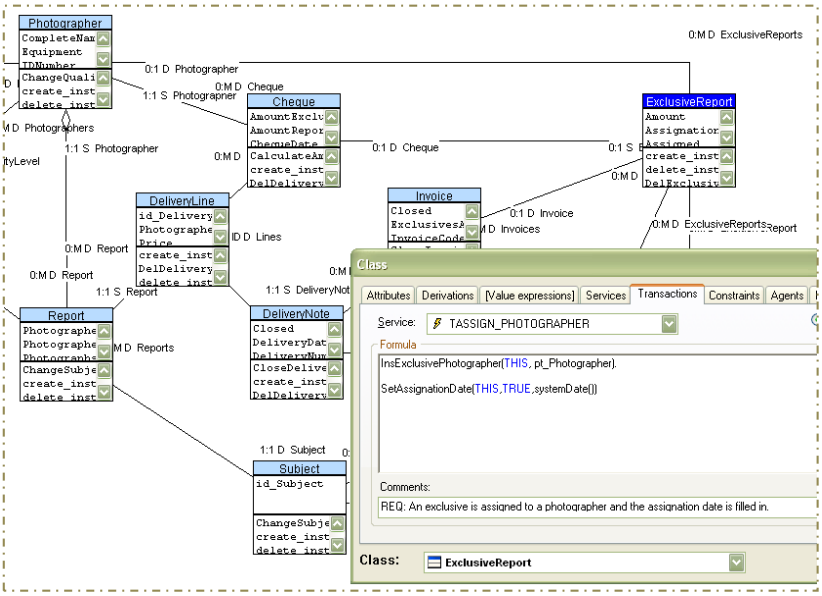


Figure 150. Part of the class diagram or the Photography Agency (integrated views of events *PHO 1* to *PHO 3*)

7.3.2 Projects office

At the moment of the derivation of the OO-Method conceptual model for this case, the researcher that performed the lab demo (the author of this thesis) was still missing some expertise in the OO-Method and, in particular, in the Integranova technology. The first versions of the conceptual model were made using a general purpose diagramming tool; namely, Microsoft Visio. Also, at that time, the derivation technique was immature and only consisted in a set of heuristics guidelines without much detail (many properties of the modelling primitives were not considered). The conceptual model produced by the researcher was compared with the one produced by an expert practitioner. In this comparison, two expert practitioners aided the research team in identifying and qualifying the differences. The differences were classified in four categories:

- *Flaws*. The conceptual model produced a model included several flaws that were related to the inexperience of the researcher (not directly related to the derivation guidelines), were amended. They were primarily syntactic errors. For instance, when the control conditions are shown graphically in a state-transition diagram, their syntax is service_name when control_condition and not [control_condition] service_name.
- *Style*. Some differences were a matter of modelling style; that is, modelling decisions that did not notably affect the quality of the model itself. When the rationale of a particular practitioner decision was convincing (e.g. it was the result of a good practice adopted after years of experience, or it led to a better software performance), they were used as feedback to improve the derivation technique. Otherwise, they were disregarded. See below an example concerning the modelling of state-transition diagrams for complex classes of business objects (i.e. those that are related to several classes of the Object Model).
- *Validity errors*. Other differences revealed that the guidelines derived some patterns of conceptual modelling elements that were invalid, in the sense that they did not appropriately support the requirements they were intended to. These conclusions were also used as feedback to improve the derivation technique. For instance, the derivation of shared services (i.e. services related to structural relationships) was not producing valid elements.
- *Incompleteness*. Given the immaturity of the derivation technique at that time, many differences were due to the incompleteness of the researcher's conceptual model. The most important missing elements were added to a *to-do* list and prioritised according to their implications they had in relation to successful automatic code generation. For instance, agent classes were not being derived yet; a model without agent classes and agent relationships will produce a software application that does not allow any interaction and is,

therefore, useless. Also, in Integranova, whenever a transition that is associated to a transaction, other transitions associated to the services included in the transaction formula must be included in the state-transition diagram; otherwise, the generated code will not allow a successful execution of the transaction. Although both severely affect the generated software, the first issue was assigned a higher priority due to the fact that it implied covering other concepts of Communication Analysis that were yet disregarded; namely, the interface actors.

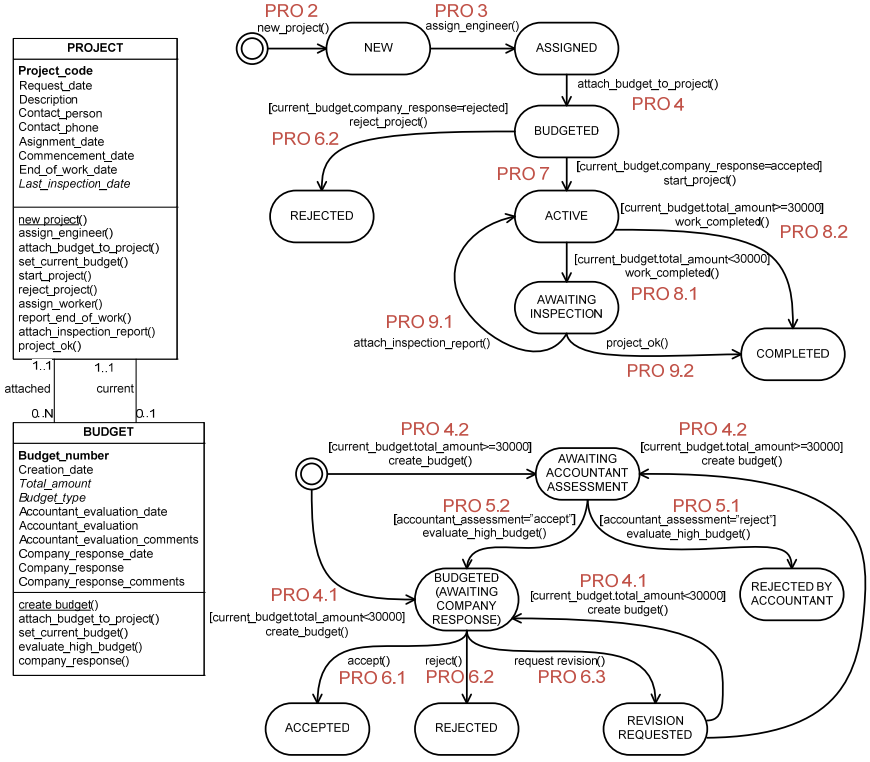


Figure 151. The researcher defined two separate state-transition diagrams for classes PROJECT and BUDGET

We present some details of the differences encountered in the Dynamic Model of the Projects Office case. The reader should take into account that, whereas the researcher's model (see Figure 151) was modelled using Microsoft Visio, the practitioner's model (see Figure 152) entirely modelled using Integranova Modeler. However, in both figures the state-transition diagrams are represented

using Visio to facilitate their comparison. For the same reason, we have added the identifiers of the communicative events related to the transitions.

Note that the researcher defined two separate state-transition diagrams to constrain the dynamics of the business object project (which includes an associated budget). The one at the top of the figure corresponds to class PROJECT and the one at the bottom corresponds to BUDGET. In contrast, the practitioner defined a single state-transition diagram associated to class PROJECT. In principle, as long as the transitions and their corresponding services (more specifically, their valuation rules and transaction formulas) are appropriately defined, both approaches are valid.

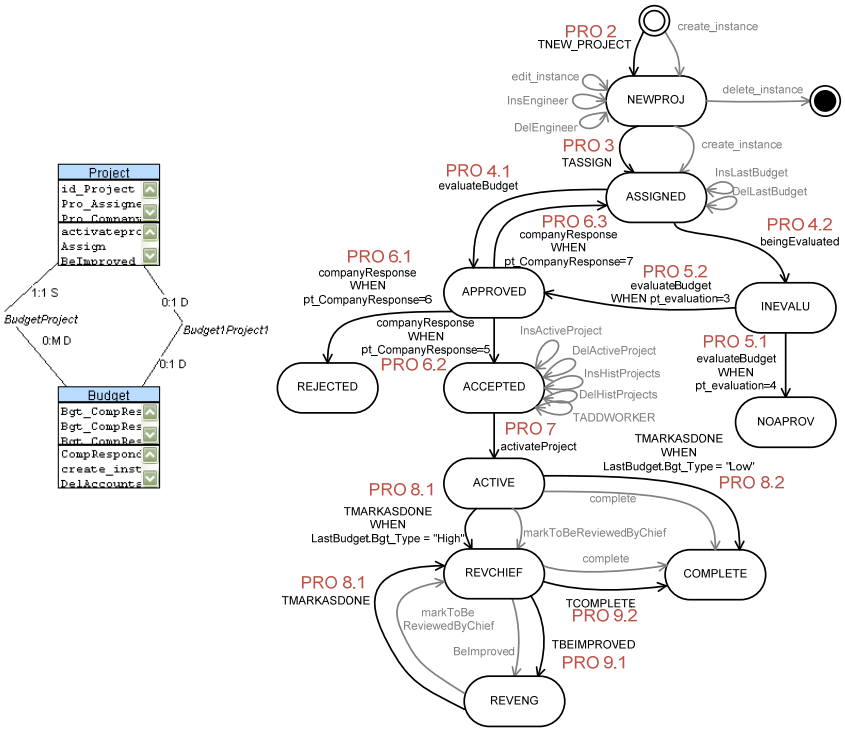


Figure 152. The practitioner defined a single state-transition diagram for class PROJECT

7.3.3 Master management system of TUO

The complexity of this lab demo made us realise that our derivation technique does not yet cover all the needs of real software development projects. The requirements model of the TUO Master case is an adaptation of a real business process in Universitat Politècnica de València. Although it simplifies some parts of the process and departs from reality in others, there are complexity factors related to message editing and to the reaction of the information system to communicative events that our derivation technique does not support yet. Even the expert industrial practitioner decided not to complete the conceptual model (the effort to complete it exceeded the amount of time she was allowed to devote to our research collaboration; it was not due to a lack of modelling competence).

In any case, the lab demo again shed some light to the state of our derivation technique. As commented above, we started defining rules for the derivation of agent classes and agent relationships (see Table 63 and Table 64, respectively).

Table 63. Agent classes of the TUO Master case

Agents
MASTER_DIRECTOR
MASTER_SECRETARY
MASTER_COORDINATOR
MASTER_ASSISTANT
PEC_DEPUTY_HEAD
SED_SECRETARY_OF_EDUCATION
TUO_LECTURER
UNIVERSITY_MEMBER

Table 64. Agent relationships for some of the class services of the TUO Master

Server class	Service	Agent classes
MASTER	new_master	SED_SECRETARY_OF_EDUCATION
EDITION	new_edition	SED_SECRETARY_OF_EDUCATION
	prepare_documentation	MASTER_DIRECTOR MASTER_ASSISTANT
	director_sign	MASTER_DIRECTOR
	SED_sign	SED_SECRETARY_OF_EDUCATION
	present_at_PEC	MASTER_ASSISTANT MASTER_SECRETARY
	PEC_resolve	PEC_DEPUTY_HEAD
	submit_expert_report	MASTER_DIRECTOR
	PEC_resolve_2	PEC_DEPUTY_HEAD
	issue_diplomas	PEC_DEPUTY_HEAD
TUO_TEACHING	accept	TUO_LECTURER
ENROLMENT	new_enrolment	UNIVERSITY_MEMBER
	increment_absences	MASTER_ASSISTANT MASTER_COORDINATOR

This lab demo also allowed us to consolidate previously improved derivation rules. However, it is the SuperStationery Co. lab demo that allowed us making a big leap forward.

7.3.4 SuperStationery Co.

The results of this lab demo are published as a technical report [España, González et al. 2011]. It has also been used as a running example to illustrate the application of the derivation rules in Section 5.5. Several iterations have been performed while tuning up the derivation rules. In general, most modelling decisions could be reasoned from (and traced back to) the requirements model; however, other decisions needed to be made during the construction of the conceptual model. We have found several reasons for this:

- In some cases, the requirements model is incomplete and, although it could, it does not provide all the necessary information to ground each and every conceptual modelling decision (e.g. a missing structural constraint). One possible solution is to prescribe that the requirements model must be complete before applying the derivation technique. However, it is not always feasible to complete the requirements model. In fact, there is a trade-off between fulfilling requirements model completeness and the available resources (e.g. time, money). The time to terminate a modelling activity is thus not when the model is perfect (which will never happen) but when it has reached a state where further modelling is less beneficial than applying the model in its current state [Lindland, Sindre et al. 1994]. Thus, the derivation technique should still be possible to be applied in the face of incompleteness (or even invalidity).
- In other cases, the information that is needed to make a given conceptual modelling decision is not provided in the requirements model because it refers to a design aspect (e.g. an initialisation formula, derived information related to business objects such as total amounts). There are several options: (i) either to include this information in the model, being aware of its design nature, (ii) to provide conceptual modelling patterns that the analyst can choose between, (iii) and to let the analyst design his/her own solution. A in-depth analysis of these situations will help to improve the derivation technique. In any case, some (design-related) marks have already been defined in order to account for some issues such as deciding whether an aggregation structure is providing information about a new business object or extending the information that the organisation has about the object (see rule **OM2**), or designating which pieces of data the organisation wants to use to identify a class of business objects (see these marks depicted in Figure 98)

- In other cases, the derivation technique lacked the proper guidelines to tackle the situation. We progressively improved the derivation rules to account for some of these issues. For instance, during the initial iterations, the property by which a structural relationship is designated as static/dynamic was not addressed by the derivation technique. The result was that the analysts have to decide by themselves. This is not a problem *per se*. In later versions of the derivation rules, we have included recommendations that can be used as a default decision (see rules **OM11** and **OM12**), However, we consider that it is worth to investigate whether this conceptual modelling decision can be better grounded in the information specified in the requirements model.

In the latest iteration the most updated version of the derivation rules have been applied. The results have been encouraging. However, they have also allowed the researchers to realise that there is still space for improvement. In the following, we highlight some issues that require further investigation.

Not all possible updates of the business objects are considered

Aside from those updates that fulfil the unity criteria [González, España et al. 2009] and therefore deserve the status of communicative event (e.g. the registry of a client; see *CLIE 1*) there may exist some changes in the state of business objects that are nonetheless important for the information system (e.g. updating the telephone number of a client, deleting a client record). These state changes mainly correspond to basic *update* and *delete* operations (i.e. the U and D operations of the typical CRUD acronym). From the point of view of Communication Analysis, these events do not entail much analytical complexity and it is advised not to include them in the communicative event diagram. For the sake of simplicity, we opt for modelling only *create* operations and those *update* and *delete* operations that are really meaningful for the organisation and affect business process understanding. This guideline has proved to be valuable in practice when complex organisational work practice if tackled.

In any case, although the above-mentioned events are of minor importance in analysis time, they constitute operations that must be supported by the information system. In real development projects that apply Communication Analysis, the information system is designed to include these operations even though they are not included in a communicative event diagram. They are often specified as textual requirements[104].

Therefore, we face the challenge of formalising the specification of basic CRUD operations in a way that they do not burden the analyst with additional workload (it needs to be an agile modelling practice, supported by the requirements

[104] Especially when the software implementation is outsourced, every requirement needs to be explicitly stated.

structure in a way that these operations are easy and fast to be modelled). Moreover, the specification of CRUD operations needs to be systematically processed during the conceptual model derivation. In short, some primitives need to be added to the Requirements Model and the derivation technique needs to be extended in order to take these new primitives into account. We plan to build upon a previous work that provides a viable notation for CRUD operations [España 2005] to improve Communication Analysis and the derivation technique.

Service derivation can be further optimised

In some situations, the transaction that groups two atomic services that correspond to the reaction of the same communicative event could be avoided by merging the two services into a single one. It is the case of communicative events that update the state of an existing business object by linking it to another business object (e.g. when a client order is assigned to a supplier, the assignment date is recorded and the order form is linked to a supplier record; see rule **OM17** and its illustrative examples). When processing such communicative events during conceptual model derivation, new attributes are added to an existing class (e.g. assignment_date is added to class CLIENTORDER), an atomic service to set the value of the new attributes is added to the class as well (e.g. set_assignment_date), a structural relationship between the affected classes is added to the class diagram (e.g. clientorder_supplier), and the shared services that correspond to the relationship are added to both classes (ins_supplier and del_supplier). Also, in order to ensure the atomic execution of both the service that sets the value of attributes and the service that sets the link between instances (e.g. set_assignment_date and ins_supplier, respectively), a transaction is added to the main class (e.g. SALE2_ASSIGN_SUPPLIER is added to CLIENTORDER).

However, in some cases it would be possible to avoid the creation of a transaction by taking advantage of the insertion shared service and using it to also set the value of the new attributes. This implies adding valuation rules to the shared service. However, this is only possible in some cases and, thus, we have postponed this optimisation until further researched is conducted.

Not all complex business objects need an end of editing service

An *end of editing* service is an atomic service that is added to a class in order to indicate that all the information that constitutes the message related to a communicative event has been already entered and that the information system can now react to the communicative event. For instance, such a service is needed for the client order so as to indicate when the information about the order (including the destinations and the order lines) has been completely entered; until that moment, the order is still being edited and the information system cannot yet react (e.g. the order should not be communicated to the Sales Manager or else he would get confused by the incomplete information).

On the contrary, the client record does not need an end of editing service because, although it is a complex business object, it does not require further information system reaction aside from having the information recorded. Adding such a service increases the workload of the salesman because it complicates the interaction with the interface of the information system: it requires at least one additional, unnecessary mouse click and users dislike this.

So we face the challenge of investigating which type of business objects require an end of editing service and which do not. We may come up with some derivation guidelines that allow making a modelling decision using heuristics such as "if the reaction of the information system to a communicative event that creates a new complex business object includes a linked communicative interaction to another actor then an end of editing service is needed to explicitly trigger the information system reaction."[105] In other cases, the users or the analyst should make the decision and, in order to account for this decision in the requirements model, a new type of requirement should be added to the requirements model. This type of requirement could be of the form "Explicitly confirm end of editing"; it would only apply to a communicative event that involves creating a new complex business object (or, more specifically, to an aggregation substructure that represents a complex business object). Its value is simply a Boolean value (a Yes/No or a checkbox). Also, the corresponding derivation rules should be extended in order to process this new requirement.

The new type of requirement specifies whether an end of editing service needs to be derived or not. Note that this requirement belongs to level *L4. Usage environment* requirements level of Communication Analysis (see Section 4.2). Therefore, it specifies an aspect of design and should be included as a mark, similarly to the others that have already been defined (see these marks depicted in Figure 98).

Attribute default values can be derived from the Requirements Model

Message structures can be used in analysis time and in design time. Depending on the stage, different properties of the message fields can be defined. This way, when switching from information system analysis to design, the message structures can be extended in order to include initialisation formulas (see Section 4.3), as shown in Table 65.

[105] This guideline is just a draft and it needs a further investigation.

Table 65. A message structure that includes the Initialisation property (it corresponds to *SALE 1*)

FIELD	OP	DOMAIN	INITIALISATION	EXAMPLE VALUE
ORDER =				
< Order number +	g	number		10352
Request date +	i	date	today()	31-08-2009
Payment type +	i	text		Cash
Client +	i	Client		56746163-R, John Papiro Jr.
DESTINATIONS =				
{ DESTINATION =				
< Address +	i			
	i	Client address text		Blvd. Blue mountain, 35-14A, 2363 Toontown Brayden Hitchcock
Person in charge +				
LINES =				
{ LINE =				
< Product +	i	Product	Product.Price	ST39455, Rounded scissors (cebra) box-100
Price +	i			
Quantity >	i	money number		25,40 € 35
}				
>				
}				
>				

As explained in Section 5.5.1, data fields lead to the derivation of class attributes and reference fields lead to the derivation of structural relationships. For data fields, this property can be processed in order to determine the formula that defines the default value for the attributes. For reference fields, this property can be processed in order to define a transaction that initialises the link.

The type of structural relationship is not derived

We refer as structural relationships to semantic relationships between classes as defined by Pastor and Molina [2007]. Although the OO-Method defines several types of structural relationships (namely, association, aggregation and composition), only associations are derived for the moment.

Actually, this is not a serious issue because the type of structural relationship actually has no implication in the automatic code generation (it does not change the software behaviour). However, we could take advantage of their different notation and semantic meaning in order to improve class diagram understanding.

Better guidance could be offered for the derivation of the attribute type

A class attribute can be of the following types:

- *Constant*. Its value cannot be changed after the initialisation
- *Variable*. Its value can be changed after the initialisation, provided that the appropriate service is defined.
- *Derived*. Its value depends on the value of other attributes and, therefore, it is determined by a derivation formula (it is not entered by the user).

We provide some heuristics and default values but we face the challenge of investigating whether the requirements model can provide enough information to make a better-grounded recommendation on whether an attribute should be constant or variable. For instance, the fact that two data fields that appear in two different communicative events actually refer to the same piece of information of a business object (and, therefore, to the same class attribute) is a sign that the attribute should be variable (it may be necessary to update the value of that attribute after its initialisation).

With regards to derived attributes, there is a trade-off to consider because defining derivation formulas in analysis time increases the workload of the analyst. Furthermore, it is an aspect of design very well supported by the OO-Method and the Integranova Modeler.

Domain ontologies could give support for a more accurate derivation

This research challenge is very wide. Just as an example, the determination of the data type of class attributes (including its size, in the case of the String type) could be supported by a domain ontology. For instance, if an attribute refers to the name of fish species, then an ontology of the fishing industry could provide guidance for determining the proper size of this attribute.

Support for other types of reaction to communicative events

For each communicative event the analyst analyses the event specification template in order check what is the expected reaction of the information system. At least two sections of the specification template have to be analysed: communication requirements (especially the communication structure), and reaction requirements.

We provided derivation rules to analyse communication structures, in order to derive the services that will support the introduction of the data related to a communicative event. With regard to reaction requirements, it may happen that the reaction of the information system to the event involves other kind of actions, such as:

- Communicating the occurrence of the event to a receiver actor. In case the communication can be done asynchronously (because there is neither urgency

nor risk), then it is possible in Integranova to support it by creating the appropriate listings and business indicators in the interface. If the communication needs to be synchronous (e.g. email, SMS, alarm, pop-up window), then it should be implemented as a user function (the code is later introduced manually by a programmer).

- Calculations and information processing. If possible, the analyst should define the appropriate transaction. In case complex algorithms are required, then user functions and manual changes are necessary.

However, these actions are not supported by the derivation technique in its present state. They require further investigation.

Our derivation guidelines do not provide support for specialisation substructures

The inclusion of specialisation substructure in a message structure implies that the message can have different structures. This imposes some constraints on the information system. In case the system is computerised, then these constraints need to be automated. At the moment, we do provide support for specialised communicative events but we do not provide support for message specialisation. Thus, specialisation substructures should be avoided if applying the derivation technique is intended.

This issue could be solved in the future. There are plenty of mechanisms in object-oriented methods (and in the OO-Method in particular) to support the necessary constraints (e.g. integrity constraints, preconditions, defining different services depending on the event variant, enforcing the specialisation constraints in the interface by means of the dependency pattern of the Presentation Model, etc.)

7.3.5 Concluding remarks

During the first iterations of the lab demo, other issues appeared, but they were solved by improving the existing derivation rules or by adding new rules. As an example, an important already-solved issue that is worth mentioning is the fact that there was no guidance for deriving the Functional Model. The OO-Method Functional Model offers a very expressive pseudocode to express the reaction of the information system. We saw the need of improving the derivation technique to guide the modelling of valuation rules (rules that specify the behaviour of atomic services) and transaction formulas (specifications of the behaviour of transactions). The addition of rules **FM1** and **FM2** solved this issue, at least for some types of information system reactions.

The above-mentioned unsolved issues do not attempt to provide a comprehensive list of the limitations of our approach, but their enumeration

serves the purpose of acknowledging those that have become more evident during the lab demos. They also constitute a *to-do* list for future work.

Anyhow, it should be highlighted that we are convinced that the benefits of applying our derivation technique outweigh the disadvantages. After all, if no derivation guidelines are applied to the construction of the OO-Method conceptual model, the analyst faces the above-mentioned issues and many more that we have actually solved successfully. To prove this argument, we have conducted an experiment that compares the quality of OO-Method conceptual models when our derivation technique is applied and when analysts ground their decisions purely on the domain knowledge expressed in textual form.

7.4 Controlled experiment: comparison of two information systems analysis method variants

According to inquiries conducted by the researchers[106], the current practice of industrial practitioners that apply the OO-Method includes eliciting requirements via interviews and specifying them in an unstructured textual form (see Figure 153). As stated by a project manager, "there is one requirements engineering method per analyst"; he also stated that most analysts consider requirements modelling a burden, since the documentation is only used as a contract with the client. Since the advent of requirements engineering, the attitude of many software development companies towards this practice is: if it is not written in the textual requirements specification, it was not included in the price (this statement was not formulated by the interviewees, but reflects a widespread standpoint among software development organisations). Defining the scope the project is indeed an appropriate use of such document. The problem that many requirements engineering researchers and trainers face is convincing industrial practitioners of also using requirements engineering methods as tools to discover, analyse and communicate the problem domain, not only as a tool to create legally binding documents.

[106] The researchers have access to the staff of the company that develops the Integranova technology and to industrial practitioners that use the OO-Method in real (all-sized) software projects. In some meetings held in 2009, the researchers carried out some informal, unstructured interviews to elicit information regarding the requirements engineering practice of the analysts. We acknowledge that a more systematic ethnographical research would allow discovering the actual practice with more accuracy and reliability.

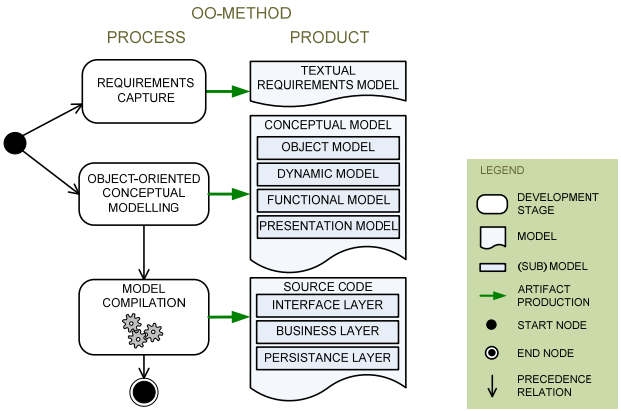


Figure 153. Overview of the OO-Method as used in current industrial practice

In practice, the OO-Method analysts create the requirements specification and the conceptual model concurrently. This information has been corroborated by at least two interviewed practitioners.

We now describe the design of an experiment intended to compare (i) the outcome of applying the current text-based requirements engineering and conceptual modelling practice, and (ii) the outcome of applying Communication Analysis requirements engineering (as described in Chapter 4) and the conceptual model derivation technique (as described in Chapter 5). We also report on the operation and results of a pilot experiment based on the mentioned design. This way, we investigate the benefits (if any) that the OO-Method analysts would obtain from including Communication Analysis requirements engineering in their work practice.

A technical report [España, Condori et al. 2011] describes in full detail the experimental planning (additional variables, instrumentation, etc.), as well as the experiment operation and lessons learned.

7.4.1 Experimental planning

The goal of the research, summarised according to the Goal/Question/Metric template [Basili and Rombach 1988], is to:

- **analyse** the resulting models of two model-based information systems analysis method variants; namely, the OO-Method along with its current requirements engineering practice (OOM) and the integration of Communication Analysis and the OO-Method (CA+OOM),
- **for the purpose of** carrying out a comparative evaluation
- **with respect to** performance of the subject and acceptance of the method;

- **from the viewpoint of** the information systems researcher
- **in the context of** bachelor students acting as surrogates for analysts.

Purpose

This research intends to assess to which extent the OO-Method benefits from its integration with Communication Analysis.

Object of study

To do this assessment, the researchers compare the outcomes of applying the two method variants (OO-Method and the integration of Communication Analysis and the OO-Method). Thus, the objects of study are the conceptual models (see Figure 154).

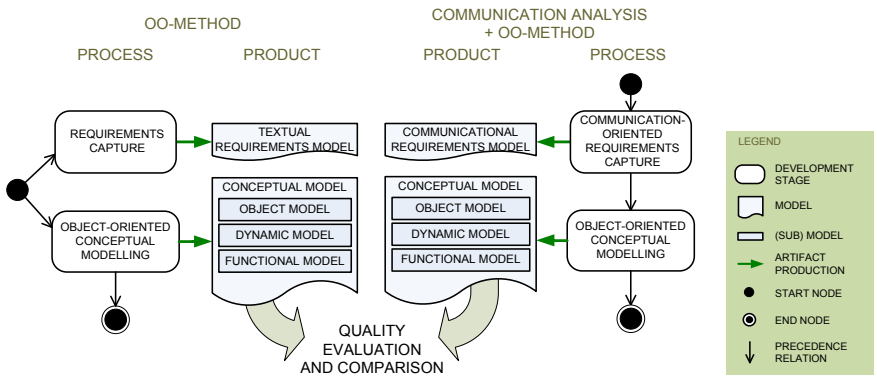


Figure 154. Overview of the comparison

The OO-Method conceptual model is composed of four views; namely the Object Model, the Dynamic Model, the Functional Model and the Presentation Model. The Presentation Model falls out of the scope of this research. The requirements models are neither compared.

Quality focus

As a general framework for method comparison, we adopt the Method Evaluation Model (MEM) [Moody 2003] (see Section 6.4.1). It defines a set of (aggregated) variables and causal relationships among them.

Perspective

The research is carried out from the point of view of the researchers. The researchers would like to know whether there is a significant difference between both method variants, in terms of the quality of the produced conceptual models. Based on previous experiences with both the OO-Method and Communication

Analysis, the researchers expect that applying a communication-based requirements engineering method will improve some aspects of the conceptual model quality. From the point of view of the method designers, empirical evidences can be helpful for improving the methods.

Context

The expected subjects are 30 bachelor students of informatics and of business informatics (INF and BIT bachelor programs) at Twente University (Universiteit Twente, The Netherlands). They are studying to obtain their bachelor and they have finished their first year but may be in their second or third year.

The experiment is planned as part of a course that counts as Independent Project (*Vrij Project*), which means that it counts as 5 credits in the students curriculum. Thus, the selection of subjects from the population of all students is not the result of probability sampling but a convenience sampling; subjects freely choose to enrol in the course (and, therefore, to participate in the experiment).

Research questions

This research is designed as an experiment. The experiment addresses the following research questions[107]:

RQ1. Will the method variant influence the performance of experimental subjects?

RQ2. Will both method variants have different acceptance among experimental subjects?

RQ3. Will the modelling competence of experimental subjects influence their performance?

RQ4. Will the modelling competence of experimental subjects influence their method variant acceptance?

RQ5. Will there be a significant interaction between the modelling competence of experimental subjects and the method variant with regard to their performance?

RQ6. Will there be a significant interaction between the modelling competence of experimental subjects and the method variant with regard to their acceptance of the method?

RQ7. Will the MEM causal relationships appear in our experiment?

[107] Do not confuse these research questions with those formulated in Section 1.2. The whole experiment is related to the original question *RQ5.* How can the integration of Communication Analysis into the OO-Method framework be validated?

7.4.1.1 Experimental design

Each experimental subject solves three problems individually (problems 1, 2 and 3 are the experimental objects). All subjects solve the three problems in the same order. Subjects are randomly[108] split into two groups (A and B). The method variant with which a subject solves each problem depends on the group they have been assigned (see Table 66). The result is a blocked subject-object study (according to [Wohlin, Runeson et al. 2000]).

Table 66. Blocked subject-object study

	Problem 1	**Problem 2**	**Problem 3**
Group 1	OO-Method	OO-Method	Communication Analysis + OO-Method
Group 2	OO-Method	Communication Analysis + OO-Method	Communication Analysis + OO-Method

7.4.1.2 Variables

We identify three types of variables [Juristo and Moreno 2001]: response variables (a.k.a. dependent variables), factors (a.k.a. independent variables) and parameters.

Response variables

Table 67 summarises the response variables considered in the experiment. Then, the variables are explained in detail.

The following mnemonic is used to refer to the instrumentation that supports the measurement: RAT stands for rating on a Likert scale, QUE stands for list of questions, and STA stands for list of statements. Also, note that # is an abbreviation for number.

[108] The randomization procedure is designed in a way that balance among groups is ensured.

Table 67. Response variables considered in the experiment

Group of variables			Variable	Metric
Performance	Efficiency		Time to finish task	*Time* = # of hours spent on the creation of the conceptual model, reported by the user
	Effectiveness	Semantic quality	Feasible validity	*Validity_STA* = 1 – (# of statements that are incorrectly specified in the model / # statements)
			Feasible completeness	*Completeness_STA* = # of statements that are correctly specified in the model / # statements
		Pragmatic quality	Feasible comprehension	*Comprehension_STA* = mean of Likert scale values (only considering correctly and incorrectly specified statements, not those statements not specified in the model)
Acceptance	Perception-based		Perceived ease of use	*PEOU_RAT* = mean of the Likert scale values of questionnaire items related to Perceived ease of use
			Perceived usefulness	*PU_RAT* = mean of the Likert scale values of questionnaire items related to Perceived usefulness
	Intention-based		Intention to use	*ITU_RAT* = mean of the Likert scale values of questionnaire items related to Intention to use

Initially, we planned for three alternative conceptual model evaluation techniques that led to distinct sets of variables. This way, a rating by an expert evaluator on a 7-point Likert scale (*Completeness_RAT*), the degree of predefined questions about the domain answered correctly by the model (*Completeness_QUE* = # of questions answered correctly / # questions), and the degree of predefined statements about the domain that are correctly specified in the model (*Completeness_STA*) are three ways of measuring feasible semantic completeness. In the end, due to resource constraints, only the statement-based evaluation was used. Also, syntactic correctness was disregarded.

Factors

The following factors are identified:

- Method variant. The variant of the systems analysis method applied by the subject to engineer requirements and develop a conceptual model is the main factor in which we are interested in this research. Two treatments are considered: (a) the method that integrates Communication Analysis and OO-Method (CA+OOM) and (b) the OO-Method (OOM). The variable is named *Method_variant*.
- Modelling competence. The competence of the experimental subject in applying the OO-Method. This factor is a continuum ranging from 0 to 10 and it is valued by means of a knowledge level assessment (a test) and a modelling task. The variable is named *Competence_OOM*. A value from 5 to 7,5 means that the subject has an average competence level; a value greater that 7,5 means that the subject has a high competence level.

Parameters

Variables that we do not want to influence the experimental results have been fixed:

- Problem domain: information systems analysis.
- Problem-size complexity: three problem statements of a similar size are chosen.
- Background of the experimental subjects: students with a similar background are chosen; however, their homogeneity is verified by means of an intake knowledge level assessment.

7.4.1.3 Hypotheses

In the following, we specify research questions as precise hypotheses; we also acknowledge our *a priori* expectations.

RQ1. Will the method variant influence the performance of experimental subjects?

H1.1: There is a significant difference between the values of *Time* for both method variants.

We expect it takes the subjects less time to build the conceptual model when applying OOM than when applying CA+OOM.

H1.2: There is a significant difference between the values of *Validity_STA* for both method variants.

We expect that conceptual models built by applying CA+OOM are significantly more valid than conceptual models built by applying OOM.

H1.3: There is a significant difference between the values of *Completeness_STA* for both method variants.

We expect that conceptual models built by applying CA+OOM are significantly more complete than conceptual models built by applying OOM.

H1.4: There is a significant difference between the values of *Comprehension_STA* for both method variants.

We expect that conceptual models built by applying CA+OOM are significantly more comprehensible than conceptual models built by applying OOM.

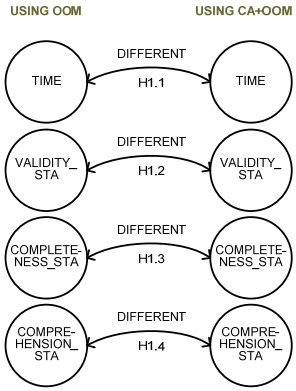


Figure 155. Graphical representation of hypotheses related to RQ1

RQ2. Will both method variants have different acceptance among experimental subjects?

H2.1: There is a significant difference between the values of *PEOU_RAT* for both method variants.

We expect that subjects significantly perceive OOM as more easy to use than CA+OOM.

H2.2: There is a significant difference between the values of *PU_RAT* for both method variants.

We expect that subjects significantly perceive CA+OOM as more useful than OOM.

H2.3: There is a significant difference between the values of *ITU_RAT* for both method variants.

We do not expect a significant difference between the intentions of the subjects with regards to using any of the method variants in the future.

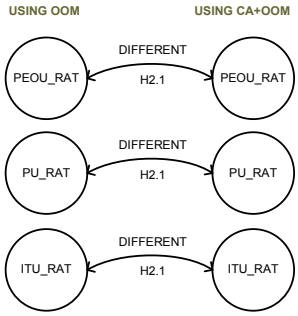


Figure 156. Graphical representation of hypotheses related to RQ2

RQ3. Will the modelling competence of experimental subjects influence their performance?

H3.1: *Competence_OOM* correlates with *Time*.

We expect that the more modelling competence in the OO-Method a subject has, the less time it takes them to build the conceptual model.

H3.2: *Competence_OOM* correlates with *Validity_STA*.

We expect that the more modelling competence in the OO-Method a subject has, the more valid their model is.

H3.3: *Competence_OOM* correlates with *Completeness_STA*.

We expect that the more modelling competence in the OO-Method a subject has, the more complete their model is.

H3.4: *Competence_OOM* correlates with *Comprehension_STA*.

We expect that the more modelling competence in the OO-Method a subject has, the more comprehensive their model is.

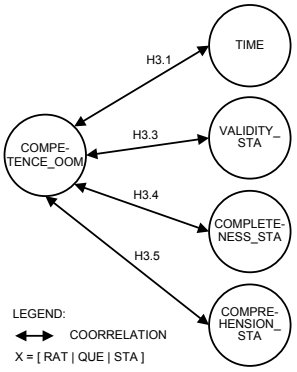


Figure 157. Graphical representation of hypotheses related to RQ3

RQ4. Will the modelling competence of experimental subjects influence their method variant acceptance?

H4.1: *Competence_OOM* correlates with *PEOU_RAT*.

We expect that the more modelling competence in the OO-Method a subject has, the more they perceive the method variant to be easy to use.

H4.2: *Competence_OOM* correlates with *PU_RAT*.

We expect that the more modelling competence in the OO-Method a subject has, the more they perceive the method variant to be useful.

H4.3: *Competence_OOM* correlates with *ITU_RAT*.

We expect that the more modelling competence in the OO-Method a subject has, the more they intend to use the method variant in the future.

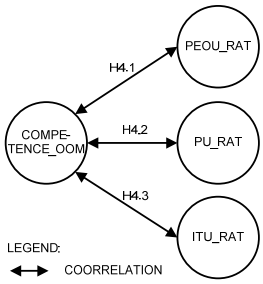


Figure 158. Graphical representation of hypotheses related to RQ4

RQ5. Will there be a significant interaction between the modelling competence of experimental subjects and the method variant with regard to their performance?

H5.1: The difference between the values of *Time* for experimental subjects with high modelling competence that used *Method_variant* OOM and for experimental subjects with high modelling competence that used *Method_variant* CA+OOM is significantly different to the difference between the values of *Time* for experimental subjects with low modelling competence that used *Method_variant* = OOM and for experimental subjects with low modelling competence that used *Method_variant* = CA+OOM.

We do not expect a significant interaction between the OO-Method modelling competence and the method variant with respect to the time it takes to build the conceptual model.

H5.2: Idem for *Validity_STA*.

We expect that the effect of CA is more pronounced for subjects with bad modelling competence than for subjects with good modelling competence.

H5.3: Idem for *Completeness_STA*.

We expect that the effect of CA is more pronounced for subjects with bad modelling competence than for subjects with good modelling competence.

H5.4: Idem for *Comprehension_STA*.

We expect that the effect of CA is more pronounced for subjects with bad modelling competence than for subjects with good modelling competence.

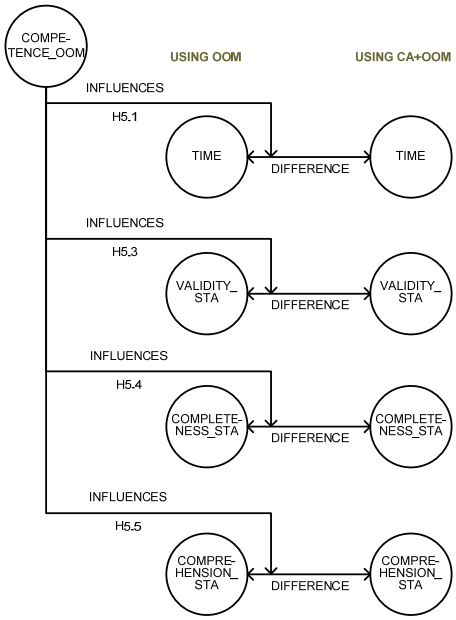


Figure 159. Graphical representation of hypotheses related to RQ5

RQ6. Will there be a significant interaction between the modelling competence of experimental subjects and the method variant with regard to their acceptance of the method?

H6.1: The difference between the values of *PEOU_RAT* for experimental subjects with high modelling competence that used *Method_variant* = OOM and for experimental subjects with high modelling competence that used *Method_variant* = CA+OOM is significantly different to the difference between the values of *PEOU_RAT* for experimental subjects with low modelling competence that used *Method_variant* = OOM and for experimental subjects with low modelling competence that used *Method_variant* = CA+OOM. We do not expect a significant interaction.

H6.2: Idem for *PU_RAT*. We do not expect a significant interaction.

H6.3: Idem for *ITU_RAT*. We do not expect a significant interaction.

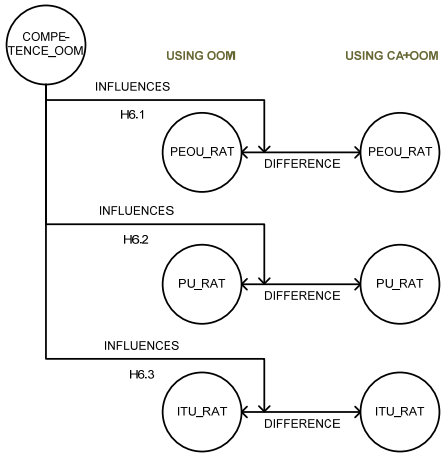


Figure 160. Graphical representation of hypotheses related to RQ6

RQ7. Will the MEM causal relationships appear in our experiment?

H7.1: *TIME* correlates with *PEOU_RAT*.

H7.3: *Validity_STA* correlates with *PU_RAT*.

H7.4: *Completeness_STA* correlates with *PU_RAT*.

H7.5: *Comprehension_STA* correlates with *PU_RAT*.

H7.6: *PEOU_RAT* correlates with *PU_RAT*.

H7.7: *PEOU_RAT* correlates with *ITU_RAT*.

H7.8: *PU_RAT* correlates with *ITU_RAT*.

We expect all this correlations to appear, although some could be non significant.

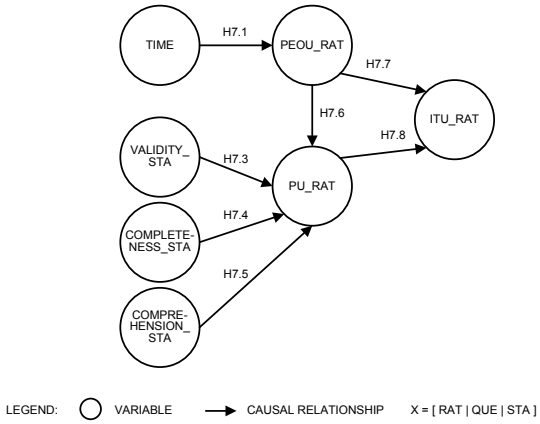


Figure 161. Graphical representation of hypotheses related to RQ7

7.4.1.4 Experimental procedure and instrumentation

The procedure to train the subjects in both method variants and have them solve the problems is depicted in Figure 162.

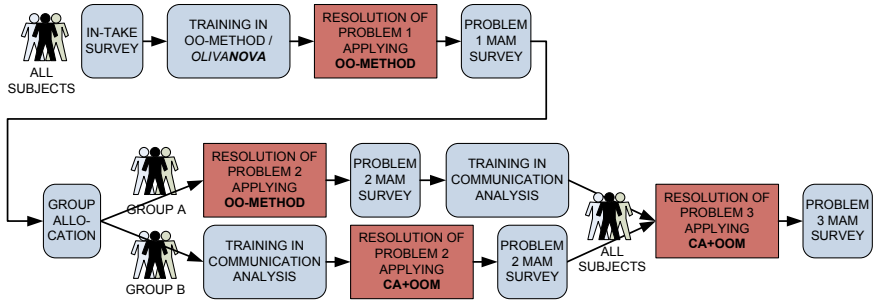


Figure 162. Overview of the experimental procedure

In-take survey

During a kick-off meeting, the subjects are informed about the nature of the research, they are given an overview of the experiment, and they are explained their expected dedication and workload.

In this meeting, the subjects fill out an in-take survey that aims to assess the background and experience of the subjects on several data-oriented and process-oriented specification techniques, such as Data Flow Diagram, Use Cases Diagram, Activity Diagram, Entity Relationship Diagram, Relational (database) Model, Class Diagram, Workflow Diagram, State-Transition Diagram, and the Business Process Modelling Notation. We assessed (i) the subjects' level of knowledge of the techniques (5-point Likert scale), (ii) their actual experience with those techniques (5-point Likert scale), and (iii) whether they would use them in future industrial projects (7-point Likert scale). The survey also includes two small-sized modelling exercises in order to assess the subjects' modelling competence before they attend the training seminars; one of the exercises is data-oriented and the other one is process-oriented. All surveys have been supported online by SurveyGizmo.

Training in OO-Method/Integranova

The subjects have received 14 hours of training on the OO-Method. An expert trainer from CARE Technologies has been involved in the course and the official training material (slides and exercises) has been used. The subjects have been trained in the modelling techniques (Object Model, Functional Model and Dynamic Model) and in using the Integranova technology (especially the Modeler). During the course, a representative information system conceptual

model is developed. The subjects can use both the course material and the representative example as references during later experimental tasks.

After the OO-Method training, the knowledge level of the subjects is assessed. The OO-Method modelling competence assessment consists of a test with 30 multiple-choice questions and a small-sized conceptual modelling exercise. The aim is to assess the degree of achievement of Bloom's educational objectives [Bloom, Engelhart et al. 1956] (namely; knowledge, comprehension, application, analysis and synthesis). This assessment allows giving value to the variable *Competence_OOM*. We determined that the value of *Competence_OOM* above which a subject is considered to have a high modelling competence in the OO-Method is 7,5 over 10.

Experimental task: resolution of Problems 1, 2 and 3

The subjects are given the task instructions, which is a guide describing the experimental task and the different protocols to be followed: how to elicit requirements, how and when to deliver a snapshot, the schedule of follow-up meetings, the expected outcomes of the experimental task, the tools they are expected to use (e.g. a diagramming tool in order to enhance the comprehensibility of models), etc.

Subjects are first given an initial problem statement that briefly describes the problem (an organisation and its work practice) and does not include each and every requirement of the information system. Then subject has to inquire about missing information. In order to elicit requirements, the subjects place questions to the domain expert (in our case, the researcher) via a helpdesk. Our intention is to offer an unbiased source of requirements and a repeatable (replicable) elicitation procedure.

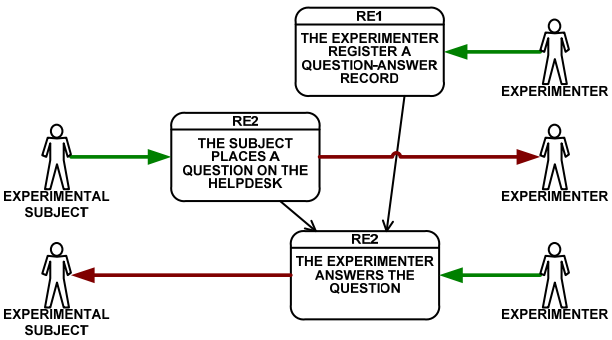


Figure 163. Communicative event diagram of the helpdesk-based elicitation

Whenever a subject needs to know something about the problem at hand, they can place a request containing questions, via the helpdesk (see Figure 163). The experimenter looks up the question in a question-answer catalogue and sends the answer to the subject. In case the question is not found in the question-answer catalogue, the experimenter registers the question and the corresponding answer in the catalogue; this way, if another subject asks the same question later, the same answer can be given. A request can contain several questions (see Figure 164); in such case, the experimenter splits the request into questions, looks up their answers, and combines them to create the response to the request.

Problem 1 is based on the Projects office lab demo (see Section 6.3.1), Problem 2 is based on the Photography Agency Inc. lab demo (see Section 6.3.2), Problem 3 is based on the Master management system of TUO lab demo (see Section 6.3.3).

As the subjects discover information about the domain, they create a requirements model and the OO-Method conceptual model (the requirements model is optional in case they have not been trained in Communication Analysis yet). When the time allotted for the experimental task ends, the subjects hand in their models (see below) and they answer a MAM survey, in which they express their perceptions and attitudes towards the method.

In case they are applying the OO-Method, then they have to build the OO-Method Conceptual Model; in case they are applying Communication Analysis and the OO-Method, then they also have to build the Communication Analysis requirements model.

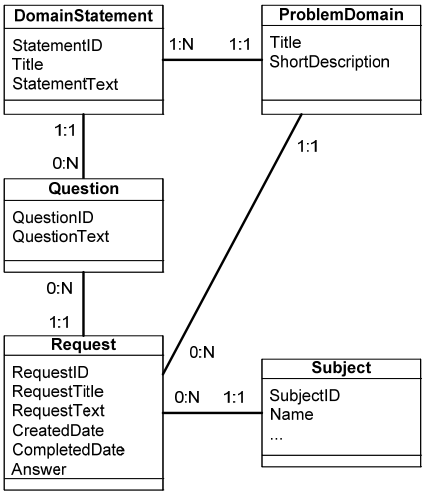


Figure 164. Conceptual model that underlies helpdesk-based elicitation

After each experimental task, the doubts that the subjects had while applying the method are discussed in 2-hour seminars before the subsequent task.

Allocation of subjects to treatment groups

After Problem 1, the subjects are split into two groups. Group A will next solve Problem 2 applying the OO-Method, while Group B receives training in Communication Analysis before solving Problem 2. Subjects are allocated to groups randomly but the following restrictions are imposed:

- Blocking. Subjects that have a high knowledge of OO-Method (if any) are evenly distributed throughout both groups.
- Balance. Both groups should have (approx.) the same amount of subjects.

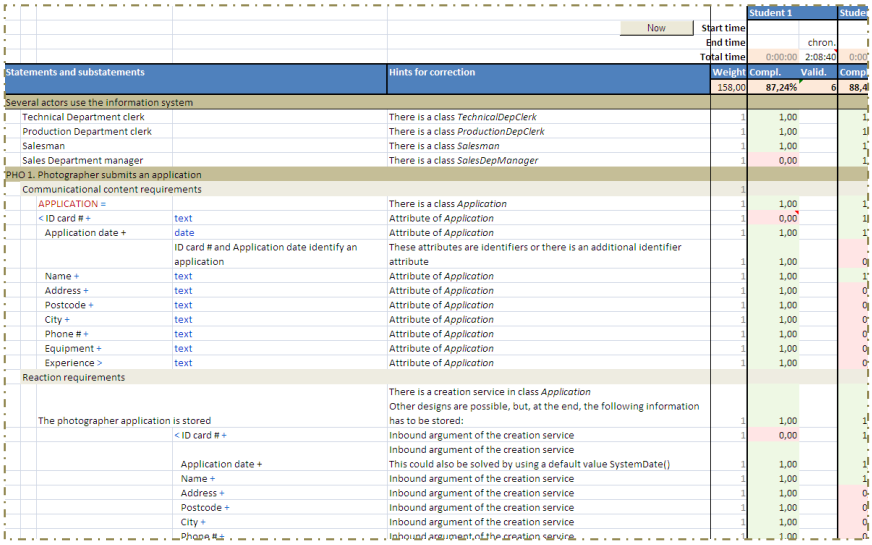
Training on Communication Analysis

The subjects have received 16 hours of training on Communication Analysis. The trainer has been a researcher who is expert in the method and has applied it in industrial settings; training material (slides and exercises) in English was created based on available teaching material in Spanish from the course "Conceptual modelling of information systems" taught in Universitat Politècnica de Valencia (Spain). The subjects of Group A and Group B have been trained separately. During the course, a representative example of an information system requirements model is developed; this example is based on the SuperStationery Co. lab demo (see Section 6.3.4). The subjects can use both the course material and the representative example as references during later experimental tasks.

After the Communication Analysis training, the knowledge level of the subjects is assessed by means of a test with 26 multiple-choice questions. The test was not appropriate for measuring the modelling competence because it only covered two of the six educational objectives [Bloom, Engelhart et al. 1956] (namely; knowledge and comprehension); also, the test had not been pre-tested and it happened to be too focused on terminology. However, all subjects passed the test by scoring 13 or above. Also, it allowed identifying some misconceptions that were clarified before the experimental tasks.

Problem MAM surveys

In order to capture the subjects' first impressions after using the respective methods to develop the models of the information systems corresponding to problems 1, 2 and 3, they responded a questionnaire. The MAM survey is similar to the one used in [Moody 2003]; it has only been slightly adapted to suit the methods evaluated in this experiment. All the questions are rated by the experimental subject on a 7-point Likert scale. The collected data was used to evaluate the perception-based variables.



				Student 1	Stude
		Now	Start time		
			End time	chron.	
			Total time	0:00:00 2:08:40	0:00
Statements and substatements		Hints for correction	Weight	Compl. Valid.	Compl
			158,00	87,24% 6	88,4
Several actors use the information system					
Technical Department clerk		There is a class *TechnicalDepClerk*	1	1,00	1
Production Department clerk		There is a class *ProductionDepClerk*	1	1,00	1
Salesman		There is a class *Salesman*	1	1,00	1
Sales Department manager		There is a class *SalesDepManager*	1	0,00	1
PHO 1. Photographer submits an application					
Communicational content requirements			1		
APPLICATION =		There is a class *Application*	1	1,00	1
< ID card # +	text	Attribute of *Application*	1	0,00	1
Application date +	date	Attribute of *Application*	1	1,00	1
	ID card # and Application date identify an application	These attributes are identifiers or there is an additional identifier attribute	1	1,00	0
Name +	text	Attribute of *Application*	1	1,00	1
Address +	text	Attribute of *Application*	1	1,00	0
Postcode +	text	Attribute of *Application*	1	1,00	0
City +	text	Attribute of *Application*	1	1,00	0
Phone # +	text	Attribute of *Application*	1	1,00	0
Equipment +	text	Attribute of *Application*	1	1,00	0
Experience >	text	Attribute of *Application*	1	1,00	0
Reaction requirements					
		There is a creation service in class *Application* Other designs are possible, but, at the end, the following information			
The photographer application is stored		has to be stored:	1	1,00	1
	< ID card # +	Inbound argument of the creation service	1	0,00	1
		Inbound argument of the creation service			
	Application date +	This could also be solved by using a default value SystemDate()	1	1,00	1
	Name +	Inbound argument of the creation service	1	1,00	1
	Address +	Inbound argument of the creation service	1	1,00	0
	Postcode +	Inbound argument of the creation service	1	1,00	0
	City +	Inbound argument of the creation service	1	1,00	0
	Phone # +	Inbound argument of the creation service	1	1,00	0

Figure 165. Fragment of the correction template for Problem 2

Assessment of the quality of the models

A researcher evaluated the models using a correction template. The template consists of a tree of statements and substatements about the problem domain. The evaluator has to inspect the conceptual model so as to assess whether the substatements are included in the model or not. Each substatement has hints for the evaluator (in order to facilitate the conceptual model inspection). The correction template is derived from the requirements specification of the information system. Figure 165 shows a partial view of the correction template for Problem 2. The template gives value to the variables *Completeness_STA* and *Validity_STA* for each subject.

7.4.2 Results analysis and interpretation

The course was advertised by several means (the campus newspaper, posters, emails, etc..; see Figure 166 and Figure 167); however, the course was planned in a late evening schedule that is not convenient for students, so we finally had 10 students enrolled. Four sessions later the number dropped to 5 students. Known reasons for dropouts are: lack of interest in the model-driven development paradigm (some students declare themselves born programmers), problems with the schedule (incompatibility with other activities; e.g. sports), and lack of interest in the information systems domain (e.g. one student preferred video game development).

We involved CARE Technologies in the experiment. A professional trainer was invited to train the students in the OO-Method and the Integranova technology.

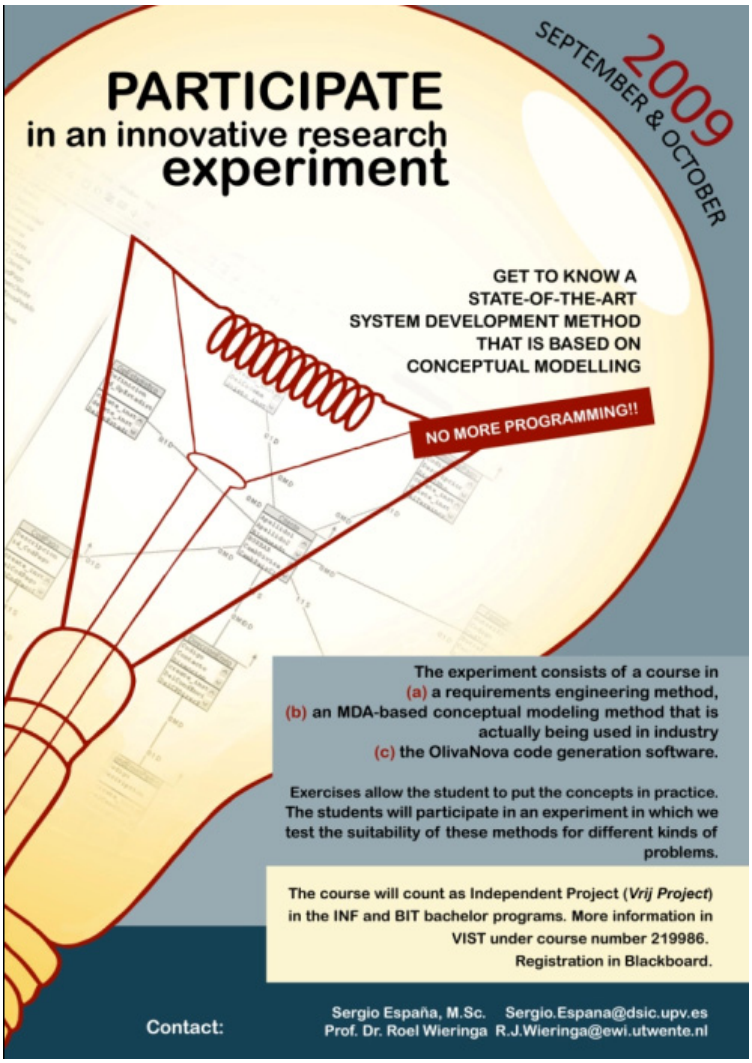


Figure 166. Course advertisement poster


weekblad van de Universiteit Twente
nummer 21 Donderdag 25 juni 2009
UT NIEUWS
www.utnieuws.utwente.nl
2
10
19
Samen aan de slag
Verzamelaar van zwarte dannen
NEW COURSE MODEL-DRIVEN SYSTEM DEVELOPMENT (219986)
INF en BIT B2/B3. In the first quarter of 2009-2010 a course on Model-driven system development will be given that can be counted as Independent Project of 5 EC in the INF program (219985) and in the BIT program (238100). Students will learn a state of the art business process modeling method and conceptual modeling method and learn how to automatically generate production quality code from these models for various commercial platforms, using the OlivaNova model compiler. More information: VIST. Registration in Blackboard. Contact persons: Roel Wieringa (roelw@cs.utwente.nl) and Sergio España (sergio.espana@dsic.upv.es).

Figure 167. Composition that shows the front page and course advertisement

7.4.2.1 The in-take survey

The in-take assessment was carried out by 9 subjects (first nine data-rows; we only identify those that did not drop out). As mentioned above, the in-take survey had three groups of questions. Table 68 shows the mean values of these groups of questions for each subject.

Table 68. Results of the in-take survey

Subject	Knowledge of the syntax of analysis techniques	Experience in applying analysis techniques	Intention to use analysis techniques
3	3,0	3,0	5,0
4	4,0	4,0	6,4
	2,0	1,9	4,0
	1,3	1,4	3,6
	3,5	3,1	4,6
1	2,5	2,3	5,4
2	2,4	2,6	4,4
	3,4	3,5	5,3
	2,6	2,4	3,8
Reference	4,3	3,4	5,0

Interestingly, Subject 1 rated his own experience low in some of the techniques but had actually a high modelling competence. In contrast, Subject 4 rated his knowledge and experience quite high in comparison to his later performance

during the course. These results warn against taking user perceptions too seriously, since they are very subjective and seem to depend on factors that are variable upon subjects (e.g. self-confidence, pride, humility).

Just as a reference, the last data-row corresponds to an lecturer from University of Twente (he was not an experimental subject), with actual high knowledge of many of the syntaxes, a demonstrated industrial experience in many of the techniques, and the conviction that system analysis techniques are of great value.

7.4.2.2 Training on OO-Method / Integranova

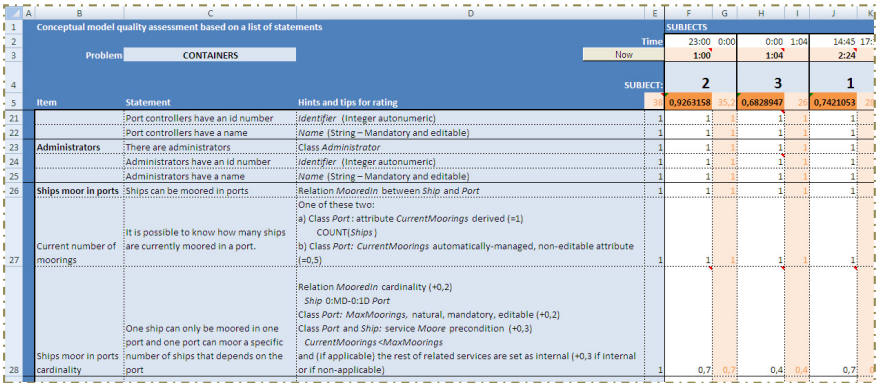
The training sessions were successful; the subjects were able to learn the concepts and discuss them with the trainer. They managed to use the Integranova Modeler to carry out some exercises. In any case, there is some space for improvement, in case the experience is repeated in the future.

A brief introduction to information systems (depending on the previous knowledge of the subjects) would be convenient before entering the Integranova training, which is very tool-oriented. Full training needs more time, especially for the exercises (at the end of each session, students have not finished the exercise). Also, at the end of the last Integranova training session, we briefly explained the modelling task and its instructions, including how the helpdesk-based elicitation worked. However, it would be convenient to have them place a question or two in the helpdesk to avoid later troubles.

With regards to the Integranova modelling competence assessment, we allotted 0:30 for the knowledge test (one minute per question) and 1:45 for the (simplified) Containers modelling exercise. Some subjects are not able to finish the exercise on time. Subjects could not consult the course material during the test but could indeed do it during the exercise.

Model quality assessment was done using a template based on statements about the domain (see Figure 168). It is cumbersome (a mean time of 1:20, one hour and twenty minutes, per model) but models can be assessed this way (e.g. subject 2 has a degree of completeness of 92,6%).

An open issue is how to assign weights to statements and sub-statements, since now all statements have the same weight in the degree of completeness. A possibility is having two meta-reviewers make an initial assignment of weights (and also they judge the template) and then compare their weights, discuss their rationale and agree the final weights.



	A	B	C	D	E	F	G	H	I	J	K
1		Conceptual model quality assessment based on a list of statements				SUBJECTS					
2					Time	23:00	0:00	0:00	1:04	14:45	17:
3		Problem	CONTAINERS		Now	1:00		1:04		2:24	
4					SUBJECT:	2		3		1	
5		Item	Statement	Hints and tips for rating		0,9263158		0,6828947		0,7421053	
21			Port controllers have an id number	*Identifier* (Integer autonumeric)	1	1		1		1	
22			Port controllers have a name	*Name* (String – Mandatory and editable)	1	1		1		1	
23		**Administrators**	There are administrators	Class *Administrator*	1	1		1		1	
24			Administrators have an id number	*Identifier* (Integer autonumeric)	1	1		1		1	
25			Administrators have a name	*Name* (String – Mandatory and editable)	1	1		1		1	
26		**Ships moor in ports**	Ships can be moored in ports	Relation *MooredIn* between *Ship* and *Port*	1	1		1		1	
27		Current number of moorings	It is possible to know how many ships are currently moored in a port.	One of these two: a) Class *Port:* attribute *CurrentMoorings* derived (=1) COUNT(*Ships*) b) Class *Port: CurrentMoorings* automatically-managed, non-editable attribute (=0,5)	1	1		1		1	
28		Ships moor in ports cardinality	One ship can only be moored in one port and one port can moor a specific number of ships that depends on the port	Relation *MooredIn* cardinality (+0,2) *Ship* 0:MD-0:1D *Port* Class *Port: MaxMoorings*, natural, mandatory, editable (+0,2) Class *Port* and *Ship:* service *Moore* precondition (+0,3) *CurrentMoorings<MaxMoorings* and (if applicable) the rest of related services are set as internal (+0,3 if internal or if non-applicable)	1	0,7		0,4		0,7	

Figure 168. Fragment of the correction template for the Containers case

7.4.2.3 Training on Communication Analysis

The training sessions were successful; the subjects participated actively and improved their requirements engineering abilities. In the following, we enumerate some observations.

With regards to the Communication Analysis modelling competence assessment test, it was not appropriate for measuring the modelling competence because it only covered two of the six educational objectives [Bloom, Engelhart et al. 1956] (namely; knowledge and comprehension); also, the test had not been pre-tested and it happened to be too focused on terminology. The subjects have trouble with all the new terminology, which does not mean that they cannot apply the techniques properly if they have the method documentation available. In any case, all subjects passed the test by scoring 13 or above (over 24 questions). Also, the test allowed identifying some misconceptions that were clarified before the experimental tasks.

For future experiments, it would also be interesting to have a small case to be modelled with Communication Analysis, as part of the modelling competence assessment (as in the OO-Method modelling competence assessment).

7.4.2.4 Allocation of subjects to groups

Figure 169 shows the procedure we followed to allocate the subjects to the treatment groups. We now comment some issues.

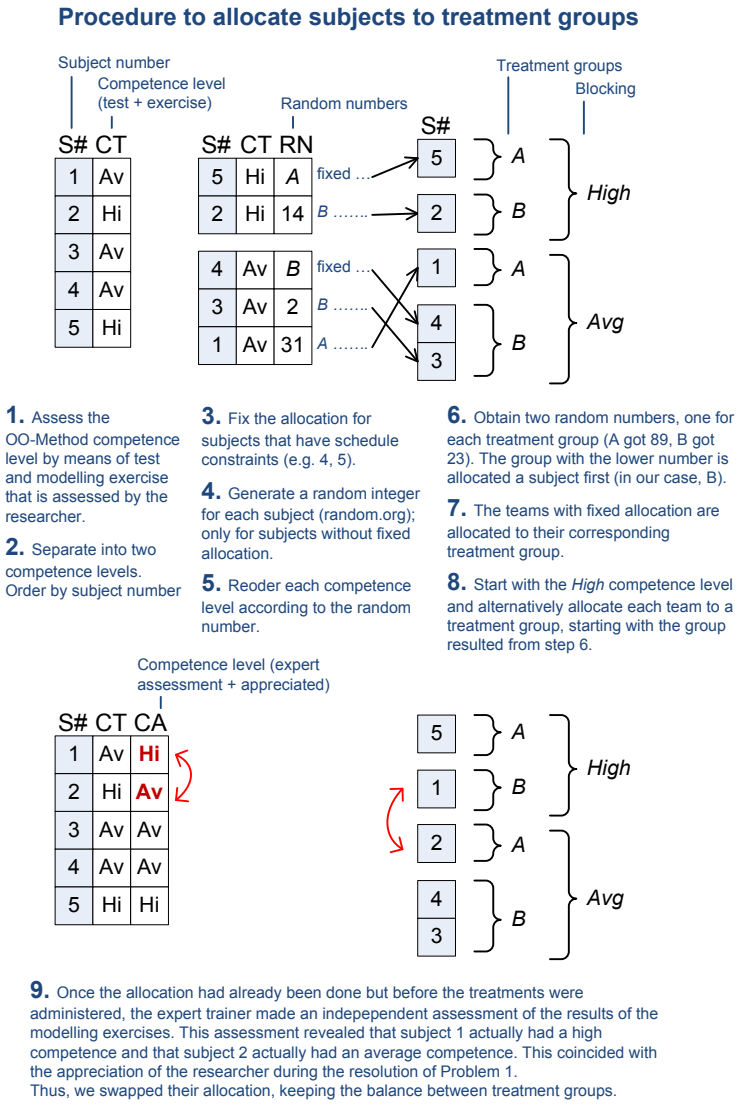


Figure 169. Allocation procedure

For random number generation we used a simple "True Random Number Generator" service available at RANDOM.ORG[109] (we executed the service 7 times, with the parameters set at Min=1 and Max=100). However, by using this

[109] http://www.random.org

service, it is possible to get duplicate numbers. In order to avoid duplicates, the "Random Sequence Generator" service is recommended.

The modelling competence assessment did not take into account how well they managed to elicit requirements; it just focused on how well they managed the tool and how well they translated a detailed object-oriented textual requirements specification into an Integranova conceptual model. Therefore it was not a good estimator of how well they performed during the first conceptual modelling task (Problem 1). In fact, a student that was initially considered to have a high modelling competence ended up not performing too well (subject 2). And a student that was initially considered to have a medium-high modelling competence ended up being quite an expert (subject 1). In fact, his smart requirements elicitation questions led me to ask him if he had previous experience in real projects; he told me he has his own software development company.

Moreover, the industrial trainer inspected the conceptual models of the knowledge assessment and confirmed the above-mentioned appreciations. This definitely means that the correction template is not very good (or that my interpretations while applying the template where wrong). The initial random allocation of students to groups had been based in inaccurate data. Thus, in view of the new evidences, we tweaked the allocation by swapping the subjects.

7.4.2.5 Problem resolution

Supporting the question-answer catalogue

Since the helpdesk system does not fully suit our needs concerning the experiment, we had to store the catalogues of questions in Microsoft Word documents. The catalogues were embedded into the Communication Analysis requirements models of the problems at hand (see Figure 170).

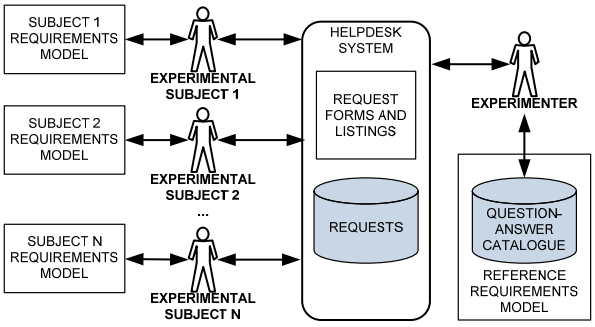
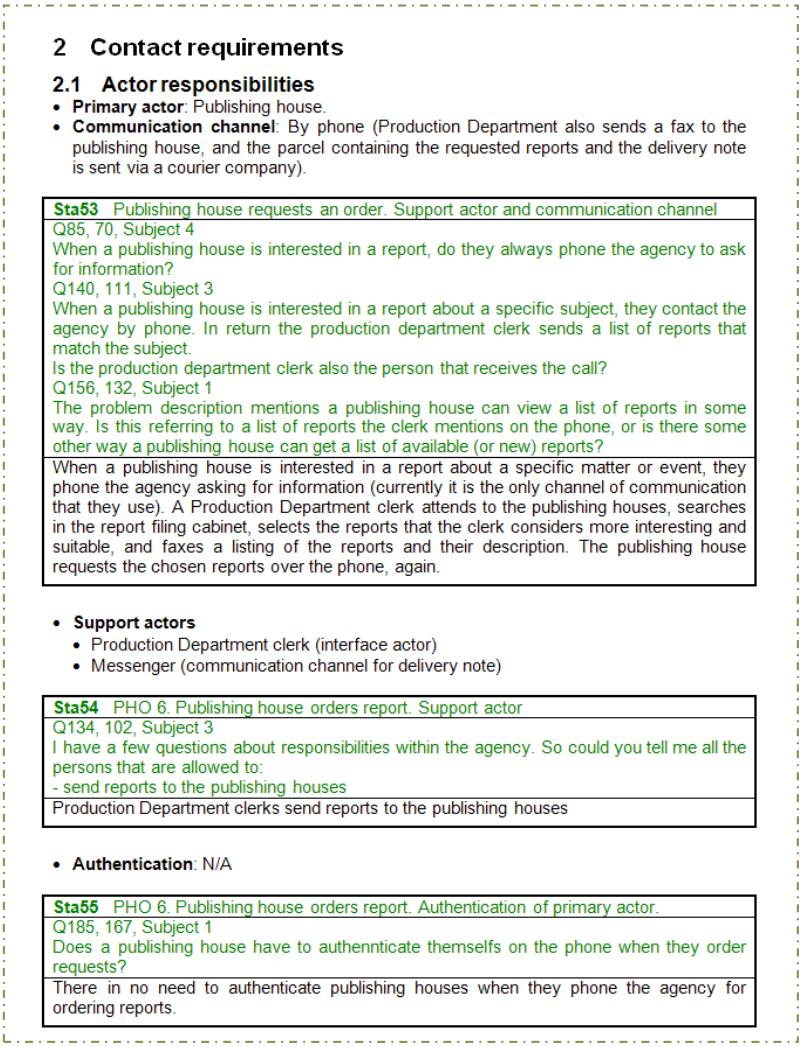


Figure 170. The support for requirements elicitation

Statements and questions were inserted as tables within the requirements model, as close as possible to the section related to the statement (see Figure 171). This way, the document structure provided an index to the catalogue, thus facilitating search, retrieval and insertion of questions.



2 Contact requirements

2.1 Actor responsibilities

- **Primary actor**: Publishing house.
- **Communication channel**: By phone (Production Department also sends a fax to the publishing house, and the parcel containing the requested reports and the delivery note is sent via a courier company).

Sta53 Publishing house requests an order. Support actor and communication channel
Q85, 70, Subject 4 When a publishing house is interested in a report, do they always phone the agency to ask for information? Q140, 111, Subject 3 When a publishing house is interested in a report about a specific subject, they contact the agency by phone. In return the production department clerk sends a list of reports that match the subject. Is the production department clerk also the person that receives the call? Q156, 132, Subject 1 The problem description mentions a publishing house can view a list of reports in some way. Is this referring to a list of reports the clerk mentions on the phone, or is there some other way a publishing house can get a list of available (or new) reports?
When a publishing house is interested in a report about a specific matter or event, they phone the agency asking for information (currently it is the only channel of communication that they use). A Production Department clerk attends to the publishing houses, searches in the report filing cabinet, selects the reports that the clerk considers more interesting and suitable, and faxes a listing of the reports and their description. The publishing house requests the chosen reports over the phone, again.

- **Support actors**
  - Production Department clerk (interface actor)
  - Messenger (communication channel for delivery note)

Sta54 PHO 6. Publishing house orders report. Support actor
Q134, 102, Subject 3 I have a few questions about responsibilities within the agency. So could you tell me all the persons that are allowed to: - send reports to the publishing houses
Production Department clerks send reports to the publishing houses

- **Authentication**: N/A

Sta55 PHO 6. Publishing house orders report. Authentication of primary actor.
Q185, 167, Subject 1 Does a publishing house have to authennticate themselfs on the phone when they order requests?
There in no need to authenticate publishing houses when they phone the agency for ordering reports.

Figure 171. Fragment of the requirements model of Problem 2 with the embedded catalogue of questions

The template that we used is the following.

StatementID Title
(All questions asking about the statement are included here.) QuestionID[110], RequestID, SubjectID QuestionText
StatementText

Analysing helpdesk-based elicitation from the subject perspective

With regards to the helpdesk-based elicitation, there are big differences in the performance of different subjects. E.g. Subject 1 placed very accurate questions, whereas others such as Subject 4 have more trouble asking proper requirements elicitation questions. For instance, in view of questions such as "What are the rights of each of the members of the Office Staff, that are not described in the problem description?" we answered as a domain expert who is not aware of the Integranova terminology: "What do you mean by rights?".

During Problem 1, Subject 1 had an outstanding performance in comparison to the other subjects (see Table 69). As discussed above, he had some industrial experience in requirements engineering. He placed 26 requests in the helpdesk, which were split into 39 questions. Subjects 3 and 4 only placed 11 questions and 7 questions, respectively. The completeness of the resulting conceptual models shows some correspondence with their elicitation performance. Subject 1 has a degree of completeness of 73,24%, whereas Subjects 3 and 4 had 48,68% and 57,79%, respectively.

Table 69. Requests, questions and model completeness

	Subject 1			**Subject 3**			**Subject 4**		
	R	**Q**	**C**	**R**	**Q**	**C**	**R**	**Q**	**C**
Problem 1	26	39	73,24%	8	11	48,68%	6	7	57,79%
Problem 2	26	31	88,45%	31	50	87,24%	47	57	69,30%
Problem 3	22	52	72,57%	36	66	95,50%	50	80	65,43%
Total	*74*	*122*		*75*	*127*		*103*	*144*	

After Problem 1, the three subjects were trained in Communication Analysis and this resulted in the following figures. During Problem 2, Subject 1 kept performing well (31 questions and a degree of completeness of 88,45%); he formulated accurate questions that unveiled many requirements, including exceptional organisational behaviour. Subjects 3 and 4 improved their performance; they placed 50 and 57 questions, respectively. The degrees of

[110] The question identifier was actually omitted.

completeness of their models was 87,24% and 69,30%, respectively. The difference with respect to Subject 1 was the accuracy of their questions; in any case, the improvement was noticeable.

By the moment Problem 3 started, the subjects were starting to be fatigued by the amount of work the course involved (as they reported in later interviews). The performance of Subject 1 dropped due to tiredness (52 questions and a degree of completeness of 72,57%). Subject 3 performed very well (66 questions and a degree of completeness of 95,50%) and Subject 4 kept a reasonable performance (80 questions and a degree of completeness of 65,43%).

We did not arrange a schedule for the elicitation; that is, we did not fix a time frame when we would be available answering questions. Therefore, the subjects placed requests along the day, at any moment they had available. The experimenter had to be aware of new requests being placed and act as quickly as possible, especially when the subject seemed to be online. However, sometimes the subjects placed requests and it was not possible to answer them promptly. The result is a variability in the response time. The histogram in Figure 172 shows the histogram of the response time; that is, the time elapsed between the subjects placed a request and they were sent the response.

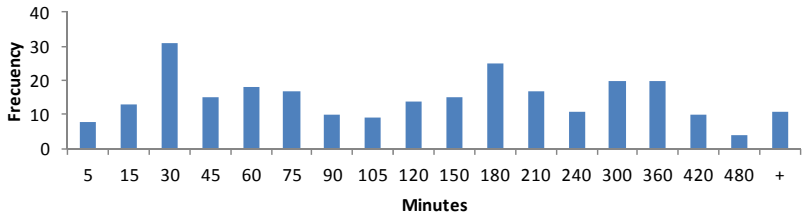


Figure 172. Histogram of the response time

During the several interviews carried on during the course, the subjects expressed a positive attitude towards the helpdesk-based elicitation. They found the setup quite realistic and considered the helpdesk-based interaction as a good surrogate for the elicitation interviews. According to their previous experience in other courses, textual specifications were not regarded as realistic and role plays often restricted to one interview without later follow-ups. They even appreciated the quick response we gave.

Analysing helpdesk-based elicitation from the experimenter perspective

The helpdesk proved to be very useful for simulating a realistic elicitation process. However, we had some problems (e.g. it did not notify when a request was placed, it malfunctions from time to time when copy-pasting from Microsoft

Word). All in all, a tailor-made tool to support requirements elicitation or a proper adaptation of the helpdesk would be a great tool for experimentation.

The experimenters prepared a question-answer catalogue in advance, before the elicitation began. However, many questions were not anticipated and, whenever a subject placed a question that was not in the catalogue, an extra effort was needed to record the question, classify it within the catalogue (i.e. place it in the Word document of the requirements model, in the appropriate section), decide the answer and send it to the subject. Of course, when other subjects asked that same question later on, the experimenter only had to retrieve the answer and send it to the subject; the effort was lower. To analyse the effort involved by the help-desk based elicitation, we recorded the amount of time we spent in processing requests.

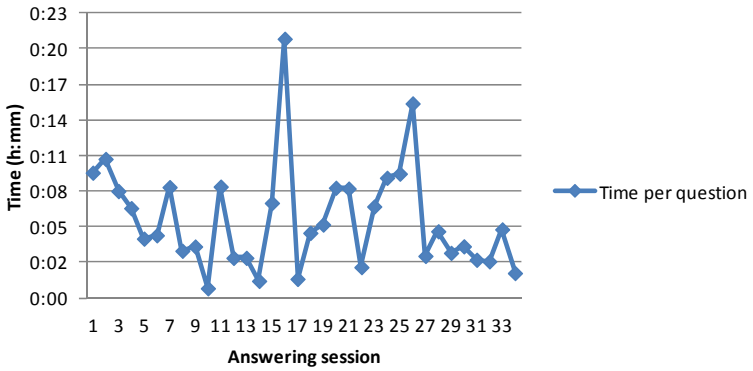


Figure 173. Mean time per question, within each session, for Problem 3

This data allows analysing the effort graphically. Figure 173 shows the means time per question, within each answering session. We have chosen Problem 3 because the experimenter had already got the hang on the helpdesk-based elicitation process. For each session, the time spent answering questions (*Time*) is divided by the number of questions (*Questions*) effectively answered within that session. This results in a mean time that differs from session to session, depending on the difficulty of the request analysis and whether the questions that the request contains are already stored in the question-answer catalogue or not. For instance, during the first sessions, after an initial investment in order to prepare the answers to the questions formulated by the leading student, the mean time dropped because the other subjects started asking similar, foreseen, questions. Later on, during session 15, a subject placed a single, difficult, question that took 20 minutes to process (i.e. to focus in the answering task, analyse the request, answer it, update the catalogue, and respond via the helpdesk).

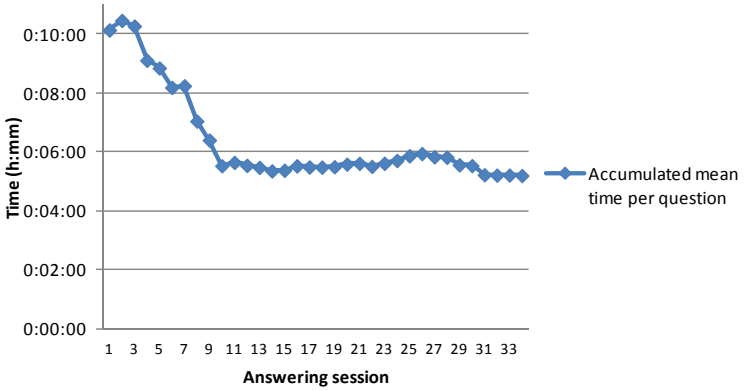


Figure 174. Accumulated mean time per question, throughout sessions, for Problem 3

Figure 174 shows the accumulated mean time per question. For each session, the accumulated time (the sum of *Time* from the first to the current session) is divided by the accumulated amount of answered questions (the sum of *Questions* from the first to the current session). The figure shows that, the mean time per question starts high (around 10 minutes per question) and then decreases drastically until it stabilises; the final mean time per question is 5:12 (five minutes and twelve seconds). For problems 1 and 2 the mean time is 6:49 and 5:50, respectively; this indicates that, as the experimenter had more experience with the elicitation process, he became more efficient.

During Problem 3, the three subjects placed an amount of 198 questions (in 108 requests) via the helpdesk, which implies an average 66 questions per subject. If the group had 15 subjects (a convenient number for statistical purposes) then we can assume that they would have placed around 990 questions. Since each question took, on average, 5:12 minutes to get answered, then the experimenters would need to devote 5155 minutes (around 86 hours) attending the helpdesk. This implies an average of around 12 hours per day for a whole 7-day week. Unless more several experimenters are involved or a more automated support is offered, the helpdesk-based elicitation is not viable for bigger groups.

Some problems are related to researcher availability. We need a mechanism to receive a notification when a question is placed by a subject; otherwise the domain expert (in our case, the experimenter) needs to be looking at the helpdesk and refreshing the list of requests permanently. It could also be a good idea to agree time frames when the domain expert will be available.

A tailor-made tool to support requirements elicitation or, at least, an adaptation or the helpdesk would improve the efficiency of the experimenters. For instance, an automated support for splitting requests into questions, looking them up in

the catalogue of statements and building the answer, if properly integrated with the helpdesk system, would facilitate the work of the experimenters. It should be possible to map questions to parts of the reference requirements specification easily.

The involvement of the subjects in the experimental tasks

After Problems 2 and 3, we gathered data about the dedication of the subjects to the experimental task.

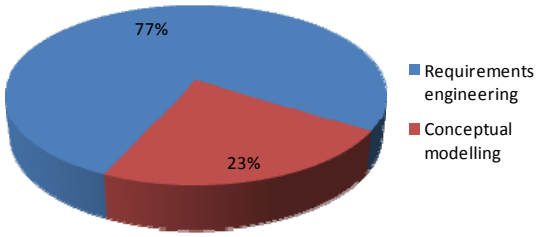


Figure 175. Distribution of time between requirements engineering and conceptual modelling

The average amount of time spent by the subjects per experimental task is 16:20 (hh:mm). With respect to the distribution of time between requirements engineering and conceptual modelling, on average the subjects devoted 77% of their time to engineer requirements and 23% of their time to derive and specify conceptual model (see Figure 175).

Conceptual model quality assessment

The conceptual models have been evaluated by an experimenter, using the list of statements template (see Figure 165). The degree of completeness has been thoroughly assessed by checking whether each substatement has been properly included in the model. However, the validity of the model has not been checked systematically; the protocol was that whenever a validity error was encountered, it was annotated in the template. This way, although an approximation to the validity of the model is obtained, it is likely that many validity errors remain unnoticed in the model after the assessment. The template has proved to be valuable to assess the model completeness, but the process to do so is very cumbersome due to the size of the models. Table 70 shows the time spent by the experimenter on evaluating the models, as well as the order in which the models have been evaluated and their degree of completeness (columns *O*, *Time* and *C*, respectively).

	Subject 1			Subject 3			Subject 4		
	O	Time	C	O	Time	C	O	Time	C
Problem 1	2	2:02	73,24%	3	1:53	48,68%	1	1:20	57,79%
Problem 2	2	1:14	88,45%	1	2:08	87,24%	3	0:53	69,30%
Problem 3	1	3:00	72,57%	2	2:52	95,50%	3	0:57	65,43%

Table 70. Time spent by the experimenter on evaluating the models (*O* order, *Time* h:mm, *C* completeness)

Our experience was that, the first model assessed within each problem implied a great effort. The reason is that the domain (i.e. the problem statement) has to be loaded in memory and certain decisions on whether a given domain statement can he modelled in one or several ways need to be made. Later model assessments benefit from this effort and take less time. However, it was also our impression that the more incomplete the model is, the shorter the assessment of its quality. This is straightforward because the moment the reviewer realises that a given part of the domain has not been discovered by the subject during requirements elicitation, the assessment of all its associated statements is quite fast (e.g. if a given class service has not been created, then none of its arguments have been created either). The data seems to support these impressions. The order *O* seems to correlate positively with *Time*, and completeness *C* seems to correlate negatively with *Time* (more data points would be necessary to verify these hypotheses, though).

The quality of the resulting models

The students learnt how to create requirements models as well as conceptual models, and they were able to analyse and criticise them. The completeness of their models was not good enough to automatically generate a software application that would fully satisfy the customers. However, given their previous inexperience with the methods and the time constraints, the result is very satisfactory. Figure 177 shows part of the requirements model of Problem 2, handed in by a subject belonging to group B. Figure 178 shows the Object Model handed in by that same subject.

The evolution of the three subjects is quite different (see Figure 176). Interestingly, we consider them to be three representatives of different phenomena that can, in principle, occur in an experiment such as ours. Although there is not enough data points (and no control group) to obtain significant statistical conclusions, our impressions are that the completeness of the conceptual models is related to the amount of questions that the subjects place.

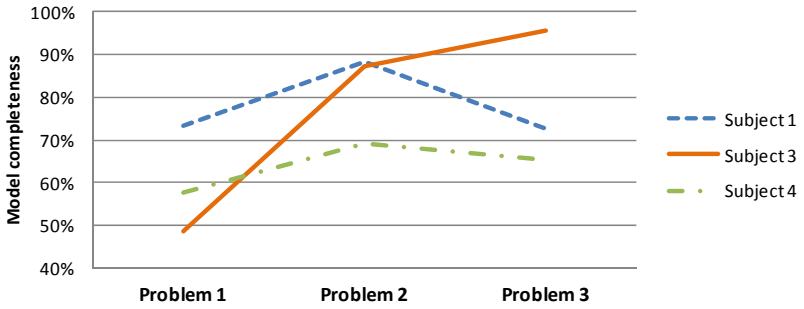
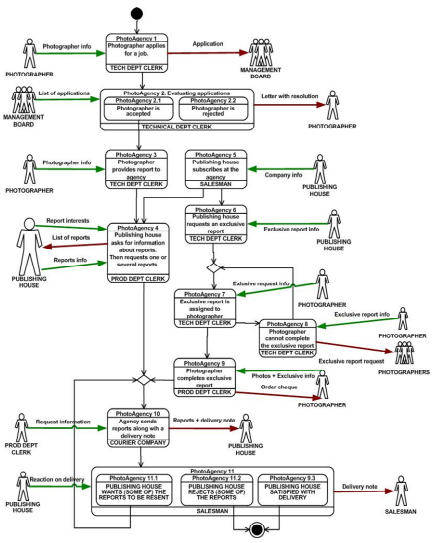
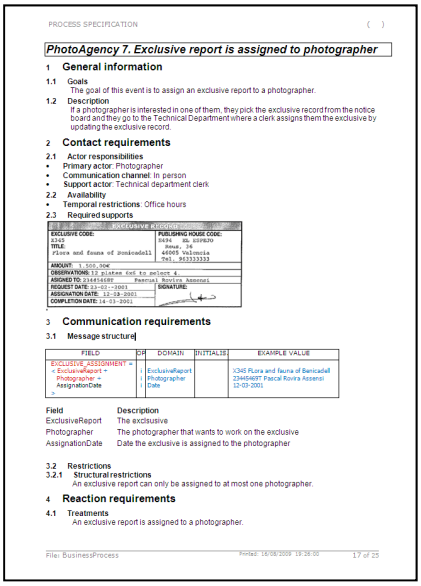


Figure 176. Evolution of model completeness throughout the experimental tasks



b) Communicative event diagram



a) Event specification template

Figure 177. Part of the requirements model of Problem 2, by a subject of group B

This relation can be seen clearly in Subject 3, who has shown a continuous increment in performance. He did not have previous experience in requirements engineering. Then, the more he learnt Communication Analysis, the better he elicited requirements and, therefore, the more complete his models were.

Subject 1 was already proficient in requirements engineering. Nevertheless, as in the case of Subject 3, learning Communication Analysis helped him to improve even further his performance. His performance decrease during the last experimental task was due to fatigue; he simply quit the exercise because the

reward (getting good marks) was not worth the effort (excerpt from an email: "I am aware of the deadline, but I've seen enough of Olivanova for the day").

Subject 4 did not have previous requirements engineering experience. He increased his performance after learning Communication Analysis and stabilised around 60-70% completeness. Although he made a greater effort in the last experimental task, his elicitation practice was not efficient enough. This, along with his difficulties with the conceptual modelling language, did not allow him to complete his conceptual model properly.

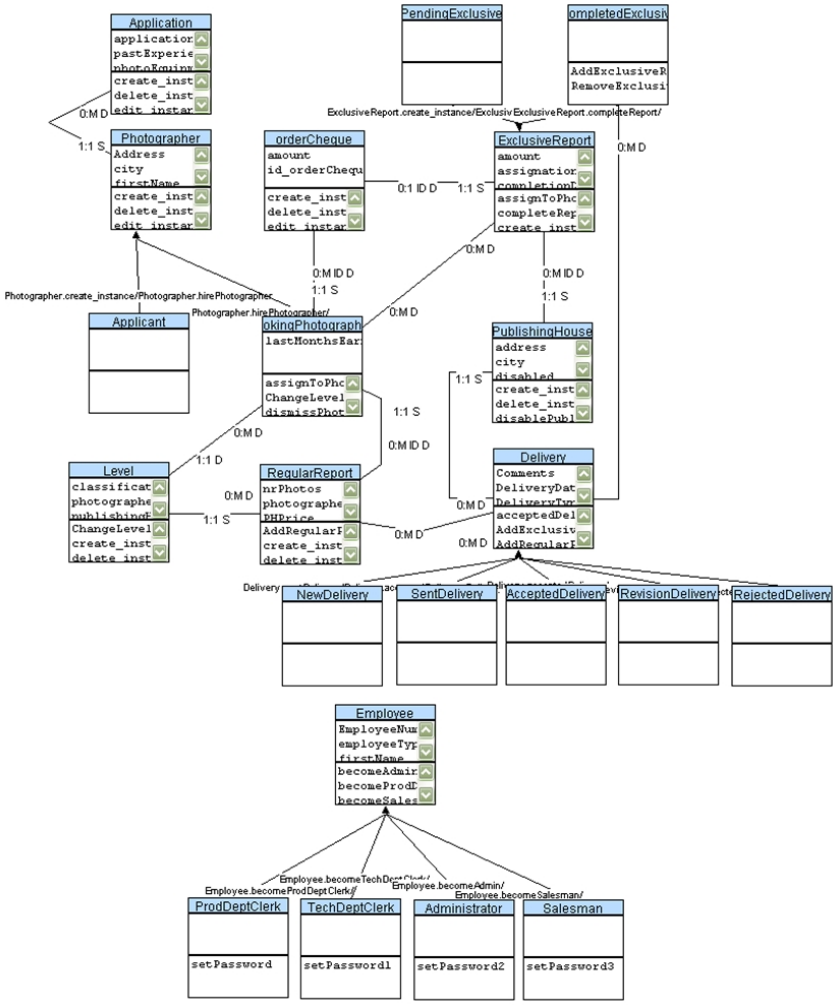


Figure 178. The Object Model of Problem 2, by the same subject in Figure 177

7.4.3 Validity evaluation

In the following, we elaborate on some threats to the validity of the experiment, as discussed in [Wohlin, Runeson et al. 2000]. Also, we point out some improvements on the experimental design that would benefit the validity of the results.

7.4.3.1 Conclusion validity

It concerns issues that affect the ability to draw the correct conclusion about relations between the treatment and the outcome of the experiment.

A *random heterogeneity of subjects* is a risk since the variation due to individual differences can be larger than due to the treatment. To minimise this threat we have selected bachelor students, who are likely to have a similar knowledge and background. Additionally, the in-take assessment is intended to verify the homogeneity of the subjects and identify special subjects. As a trade-off, homogeneity reduces the generalisability of the conclusions (external validity).

Given the loose control that the experimenters have when the subjects are carrying out the experimental tasks (the tasks are part of their homework), *random irrelevancies* in their environment (e.g. noise, distractions, team-work) may disturb the results. However, given the length of the tasks it is the only viable experimental setting.

The *reliability of treatment implementation* depends on the similarity of the two Communication Analysis trainings. The content of the course has been established a priori; however, the students of both groups can formulate different questions or ask for further explanations on different aspects of requirements engineering. This is an unavoidable issue that, in any case, is not expected to influence significantly the results. Moreover, the Communication Analysis competence assessment (if improved, see Section 7.4.2.3) can be used to verify that both treatment groups end up with a similar competence of Communication Analysis after their respective training courses.

There is a threat related to the *reliability of measures*. Although the conceptual model quality framework is soundly founded in theory, our implementation of the correction template has not been tested. As part of our future work, we plan to check whether several experimenters assessing the quality of a conceptual model with the correction template based on the list of statements (see Figure 165) obtain the same results (i.e. the same degree of completeness and the same number of validity errors). Moreover, having several reviewers (instead of only one) and applying an inter-reviewer agreement protocol will increase the reliability of the conceptual model quality measures.

Also, with respect to the reliability of perception-based measures, a reliability analysis should be carried on the results of the MAM surveys using the Chronbach alpha technique. In the social psychological literature there is evidence that attitude scales, as well as skills scales, are reliable [Shaw and Wright 1967].

If we intended to draw statistical conclusions from the data gathered during the experiment operation we would face the problem of a low statistical power or that results are not significant. However, we are simply considering it as a pilot experiment that allows us to validate the experimental design and the instrumentation. In the future, if the experiment is to be operated again, a sufficient number of subjects is needed.

7.4.3.2 Internal validity

It is related to issues that may interfere with the treatment-outcome relationship, acting as confounding factors.

The experimental task consisting on the resolution of Problem 2 is carried out by treatment groups A and B at different moments in time (see Figure 162). This threat, referred to as *history*, is related to the fact that the circumstances may not be the same in both occasions (e.g. the students may have different levels of stress in different weeks, depending on other courses). We cannot avoid this threat but we can measure their interaction via the helpdesk, ask them if we observe anything unusual, and react according to their involvement (e.g. send remainders by email, extend deadline in case they have an exam, etc.).

There are two threats related to *maturation*. Firstly, throughout the experiment, the subjects carry out several experimental tasks (Problems 1, 2 and 3); they gradually increase their modelling competence in the OO-Method. This is expected to positively affect the quality of their models; this way, maturation due to several applications of the OO-Method could hinder measuring the effect of applying Communication Analysis to engineer the requirements. To minimise the above-mentioned threats of history and maturation, the experiment design considers a staggered application of the treatment (i.e. Group A is trained in Communication Analysis before Problem 2, whereas Group B is trained before Problem 3; see Figure 162). This is often done in agricultural long-term experiments [McRae and Ryan 1996].

Secondly, since the experimental tasks and the overall experiment is long, the subjects can get bored or tired. This has indeed happened during the pilot experiment (Subject 1 acknowledged ending the task before solving Problem 3 completely, see page 438). This cannot be avoided, but it can be minimised by encouraging the subjects (motivating participation) and rewarding them properly (their grades depend on the quality of their models); we can also interview them

as in the pilot experiment, to find out whether they were affected by tiredness, so as to take it into account when interpreting the results.

The trade-off of our design is that the subjects in Group B have to carry out two experimental tasks consecutively. This increases the risk referred to as *resentful demoralisation*, by which a subject receiving the less desirable treatment may give up or not perform as good as it generally does. The subjects in Group B are tired after Problem 1 this may influence their performance during Problem 2. In fact, this could have been the reason behind the dropout of Subject 2 after Problem 1 (this could not be ascertained). In future replications, it is recommended to interact with the subjects of Group B before proceeding with Problem 2, in order to keep them engaged in the course-experiment.

With regards to the experiment design, it could be interesting to actually take into account the prior modelling competence of the subjects and their modelling competence in Communication Analysis. We have measured them (by means of the in-take assessment and the Communication Analysis knowledge test) but we still have not defined them as variables; also, some hypotheses related to these variables should be defined and later verified. This way, we will avoid the risk of having a hidden or *confounding factor*.

The design of the *instrumentation* affects the experiment outcome. We have learnt from previous experiments (see Section 6.5.5.2) that paper-based surveys are error-prone. In this experiment, we have used web-based surveys to prevent blank data and transcription errors. The experience has been satisfactory in this sense.

The pilot experiment has revealed that another instrument that is capable of improvement is the Communication Analysis competence assessment. In order to effectively measure subjects competence, the following elements should be included:

- More questions in the test so as to address aspects of the method not being assessed: (i) subjects should be able to interpret a message structure; (ii) they should be able to apply Communication Analysis guidelines for model modularity.
- An elicitation exercise, so as to assess their competence in formulating questions that ask for information about an incomplete case description. The corresponding correction template/protocol is also needed.
- A modelling exercise so as to assess the requirements modelling competence of the subjects. The corresponding correction template/protocol is also needed.

There have been many student dropouts during the course-experiment; this is a threat known as *mortality*. However, we have gathered qualitative information

that explains the reasons behind the dropouts (see Section 7.4.2); in view of the evidences, the experiment results are not compromised.

7.4.3.3 Construct validity

It is concerned with the relationship between theory and observation; that is, how well the treatment reflects the construct of the cause and how well the outcome reflects the construct of the effect.

We minimised the threat of *mono-operation bias* by allocating several subjects to each treatment group, and by using several problem descriptions for the experimental tasks. However, even if we are using three different cases (a projects office, a photography agency and a university master management system), the wide range of information systems is necessarily under-represented. The cases intend to be realistic and are definitively not toy examples, but their size and difficulty is still small in comparison with industrial projects.

There is a threat of *mono-method bias* with regards to the measurement of model quality, since only one way of measuring the quality of the models has been applied. We had envisioned two other evaluation templates for measuring conceptual model quality, one based on Likert scales and one based on a list of questions (see [España, Condori et al. 2011]), but none of them has been used due to lack of time and resources. This should be done in future replications. Also, it will allow comparing the three evaluation approaches, which an is interesting research. An experiment could be set to do this: having several evaluators using one or several of the evaluation instruments. In such case, the procedure for distributing the models among the available reviewers needs to be defined a priori. For instance, each reviewer receives N models that they have to review with one quality assessment technique and N models that they have to review with another technique. They give back the results and they fill in two surveys, each one evaluating a technique (their perception about the quality assessment technique). This way we make quality assessment an experiment in itself.

With regards to the *interaction of testing and treatment*, the facts that the subjects know that the experimenters assess their conceptual models and that the outcome of this assessment influences their grades is expected to influence their performance (they will presumably be motivated to produce models of high quality), but we argue that this is similar to real model-driven software development settings, where the pressure comes from the clients and from the project managers.

The *experimenters expectancies* can bias the results of an experiment. Several of the experimenters are method engineers involved in the design of Communication Analysis and indeed expect to prove the benefits of

Communication Analysis. To minimise this risk, other experimenters without expectancies have been involved in this research.

We have identified three threats related to the *inadequate preoperational explication of constructs*. Firstly, with regards to the problem correction template based on the list of statements, although Lindland, Sindre and Sølvberg [1994] formalised what they meant by semantic completeness of a conceptual model, it is not clear whether all statements of a domain actually matter the same. In our template, all the (sub)statements are assigned the same weight (they contribute the same to completeness) but it is arguable whether including the registry date of an illustrated report is as important as calculating its price correctly. An open issue is how to assign weights to statements and sub-statements. A possibility is having two meta-reviewers make an initial assignment of weights (and they should also judge the template itself) and then compare their initial weights, discuss their rationale and agree the final weights.

Secondly, the short problem description (the initial description of the system given along with the task instructions at the beginning of each experimental task) is expected to influence the performance of the subjects. Depending on how much information it contains, the subjects may discover or overlook a hidden part of the system. For instance, how many business forms should the short problem description include? In the pilot experiment, we only provided some forms; perhaps one of each kind should be included. Some subjects have asked for more forms in a general way and others have asked if the organisation had a given type of forms, but then they didn't ask the same for other types of forms. In view of the questions placed via the helpdesk during Problem 2 and the resulting models, it seems that some subjects thought that the exclusive report record was the same as the regular report record. The same applies for the exclusive report delivery note and the regular report delivery note. These types of business forms are, in fact, different but the short problem description only included those of the regular report. All in all, there is a lack of theory that addresses how this issue affects the requirements elicitation and the conceptual modelling tasks.

Thirdly, the Problems 1, 2 and 3 are complex and big in comparison to other experimental tasks explained in the literature. Some subjects expressed that they did not have time to finish the assignment or that they were not willing to spend more time on them. For instance, Subject 4 expressed that he would have needed from 8 to 12 hours more to completely solve Problem 2 (half of this hours for requirements engineering and half for conceptual modelling). Subject 3 felt confident about his conceptual model for Problem 2 but reported that his requirements model was incomplete because he felt it was not worth spending more time on it (it was not being graded). However, a complex problem is needed in order to demand a strong effort from the subject and advanced features from the requirements engineering method. Otherwise the subjects would perform similarly well with any requirements engineering method, just applying

common sense and standard elicitation practices (e.g. placing simplistic questions). Our impression is that there is a lack of theory about the effect of the complexity of the problems in the observability of the benefits of requirements engineering method; that is, the interaction of problem complexity and treatment. Note that complexity does not only refer to size (e.g. the amount of business objects that an information system needs to manage or the amount of business processes that are included in problem scope) but also on other factors (e.g. the fact that there exist business process specialisations and exceptional behaviours in the company).

7.4.3.4 External validity

It is related to the extent to which the conclusions can be generalised outside the scope of the experiment.

We are aware that the experiment would benefit by using real practitioners as experimental subjects. However, this may not be possible given the long involvement that it is required. In any case, Runeson [2003] suggested that the results obtained by using students as experimental subjects can be, to a great extent, generalised to industry practitioners. In any case, the pilot experiment did not allow to verify the hypotheses and the experiment would need to be operated again experiments with a larger number of subjects.

The complexity of the problems used for the experimental tasks is not fully comparable to real industrial problems. Nonetheless, they were thoughtfully selected because they balance complexity and feasibility (they can be tackled by students) with a limited availability.

7.4.3.5 Ethical concerns

Since this experiment is embedded in an academic course, some ethical concerns arise. The expected dedication of the students to the course/experiment amounts to 40 class hours and 88 homework hours, which conforms to the regulations for optional courses in University of Twente. Also, the group to which subjects are allocated is expected to influence their performance and, therefore, their grades will be influenced too. This has to be checked and taken care of, in case there is a significant difference in grades between subjects of both groups.

Lastly, the students are aware of the fact that they are experimental subjects. It is important that they do not feel as guinea pigs, but as students. Giving them some feedback on the experiment design, the operation and the expectations of the researchers can help improving their overall perception of the experience.

7.4.4 Discussion

We have designed an experiment that has the capability of assessing the impact that the application of Communication Analysis during requirements engineering has on the quality of OO-Method conceptual models (see the planning in Section 7.4.1 and the validity evaluation in Section 7.4.3). The experiment has been operated in University of Twente from September to November 2009 (see the report in Section 6.5.4). Due to the small amount of subjects, no statistical conclusions can be drawn, but the pilot experiment has proved the feasibility of the experimental design. We have collected and reported much information regarding the operation of the experiment.

We can conclude that the experiment is promising and it has the potential, not only to verify the hypothesis, but also to enlighten about many mechanisms that underlie requirements engineering and conceptual modelling.

In the future, we intend to repeat the operation of this experiment whenever the resources are available. We are of course open to external collaboration and we will provide support to replications. However, some improvements on the design and the instrumentation are recommended (see Section 7.4.2 and 7.4.3).

We have realised, however, that the experiment design is very ambitious in the sense that it covers several tasks carried out by the analysts (properly speaking, by the experimental subjects acting as surrogates for analysts); namely, requirements elicitation, requirements modelling, conceptual model derivation (the later actually includes deriving an initial conceptual model by applying transformation rules to the requirements model and later completing it by adding new elements to it).

There problem with this is that we may be able to verify the hypothesis that applying Communication Analysis to engineer requirements has a positive effects on the quality of the OO-Method conceptual models, but we may be unable to ascertain which part of Communication Analysis (e.g. the elicitation guidelines, the modelling language, the conceptual model derivation rules) are responsible for this effect. From the qualitative point of view we can indeed draw some conclusions but the experiment does not allow this kind of analysis from the quantitative (i.e. statistical) point of view.

This led us to design an experiment with a narrower scope, restricting itself to conceptual model derivation. This which was operated in Valencia in 2010 and replicated again in 2011. It is described in Section 7.5. However, we still believe that the wider experiment described in this section is worth the effort, due to the rich analyses that it will allow.

7.5 *Controlled experiment: comparison of conceptual model derivation techniques*

Current practice of industrial practitioners that apply the OO-Method does not include model-driven requirements engineering; that is, the analysts indeed elicit requirements and specify them in an unstructured textual form (see Figure 153), but not with the engineering goal of later deriving the conceptual model from the requirements model (their intention is strictly using the requirements model as a binding contract with the client). In the following, we describe the design, operation and results of an experiment intended to compare (i) the outcome of creating a conceptual model based on the knowledge contained in a textual requirements specification, and (ii) the outcome of applying the conceptual model derivation technique described in Chapter 5 to a Communication Analysis requirements model. This way, we investigate the benefits (if any) that the OO-Method analysts would obtain from systematically deriving the conceptual models from Communication Analysis requirements models. This experiment is based on the one described in Section 7.4, but restricts its scope to the creation of conceptual models.

Fortuna, Werner and Borges [2008] have carried out an experiment with a similar goal but different design and less experimental subjects, in which they compare the class diagrams derived from a Use Case model with those derived from an InfoCases model (an extension of Use Cases).

7.5.1 Experimental planning

The goal of our experiment, according to the goal/question/metric template [Basili and Rombach 1988], is to:

- **analyse** he guidelines for deriving object-oriented conceptual models from requirements specifications
- **for the purpose of** carrying out a comparative evaluation
- **with respect to** the effectiveness of the guidelines
- from the viewpoint of the method designer
- **in the context of** master students in Computer Science of the Universitat Politècnica de València acting as surrogates for analysts.

With respect to the actual effectiveness of the derivation techniques, the experiment addresses the following research questions. As discussed later, we take into account two different blocks of teams of analysts depending on their level of OO-Method modelling competence: high-competence teams and average-competence teams.

RQ1. Within each competence block, do the teams applying communication-based derivation produce conceptual models with higher semantic completeness than the teams applying text-based derivation?

RQ2. Within each competence block, do the teams applying communication-based derivation produce conceptual models with a higher semantic validity than the subjects applying text-based derivation?

7.5.1.1 Experimental context

The Master in Software Engineering, Formal Methods and Information Systems (Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información) of the Universitat Politècnica de València (Spain) offers a course named "Software Technology for Web Environments" ("Tecnología software para ambientes web", TAW). This course is offered to 4th and 5th year master students. In this course, the students learn to create object-oriented conceptual models for information system development. Although the practical orientation of the course is aimed at web environments, the actual methods that are taught in the course are generic for any other platform (e.g. desktop applications). Actually, the OO-Method is used as a paradigm of a method that complies with the Model-Driven Architecture. The students are trained in OO-Method and Integranova and, thus, they are appropriate subjects for an experiment regarding object-oriented conceptual modelling[111].

The course content and planning were updated in order to incorporate the experimental setup, still maintaining the original course objectives. The 2010 TAW course had 31 students that attended classes acted as experimental subjects.

7.5.1.2 Experimental design

We want to compare two treatments. Each treatment is a different technique for deriving conceptual models from requirements models:

- *Text-based derivation*. This technique takes as input a textual requirements model and, by means of a linguistic analysis, facilitates the identification of actors (agents), business objects (classes of objects), operations (class services), changes in the state of objects (states and transitions), etc. This way, a conceptual model is created. This is how the OO-Method has been taught for many years and how industrial practitioners work under conditions of practice.

[111] See the master website for more details on the course
http://www.popinformatica.upv.es/MISMFSI/asignaturas-old/TAW/

- *Communication-based derivation*. This technique takes as input a Communication Analysis requirements model and, by applying a set of transformation rules, a conceptual model is derived. The derivation technique is described in Chapter 5.

In order make the comparison, we planned a *multi-test within object study* [Wohlin, Runeson et al. 2000]: each subject (in our case a team of 1 or 2 students acting as surrogates for analysts) uses exactly one of the techniques to create a conceptual model and then we assess the quality of this model; we intended to use a single case description as input for all the teams. This design implies only one round.

Table 71. Multi-test within object study

Subjects	Treatment	Object
Group TS	Text-based derivation	Photography Agency case
Group CA	Communication-based derivation	

The experimental object is the description of a photography agency (as in the experiment described in Section 6.5); however, given the different nature of the two derivation techniques being compared, two versions of the object were prepared (both containing the very same information); namely, a textual requirements model and a Communication Analysis requirements model. To measure the size of the case, we applied a functional size measurement procedure to the reference conceptual model for this case (using the corresponding tool of the Integranova suite); it indicates that the conceptual model has a size of 537 IFPUG function points.

7.5.1.3 Variables

We identified three types of variables [Juristo and Moreno 2001]:

Response variables

The response variables (a.k.a. dependent variables) are related to *actual effectiveness*; more specifically, to the semantic quality of the conceptual models. We decompose it based on the framework by Lindland et al. (see the detailed description of the variables in Section 6.4).

- *Semantic completeness*. The metric used to measure this variable is the degree of completeness of the conceptual model (*degCompl*), which is the result of a careful model inspection by an expert reviewer. To do the measurement the reviewer has a template that lists each and every statement about the domain that should be included in a complete model, and checks the conceptual model against this list.
- *Semantic validity*. The metric used to measure this variable is the number of validity errors in the conceptual model *(errValid*), as a result of the expert reviewer inspection. To do the measurement the reviewer notes down each invalid statement found in the conceptual model, while reviewing it to assess the semantic completeness.

Factors

The following factors (a.k.a. independent variables) are considered. We want to assess their impact on the response variables.

- *Derivation technique*. Two different sets of guidelines for the derivation of conceptual models from requirements specifications are used: communication-based derivation guidelines and text-based derivation guidelines. Each set of guidelines assumes a specific requirements modelling language: textual specification and Communication Analysis model, respectively. Thus, these two requirements modelling languages have been applied to create the requirements models that are input for the derivation task.
- The OO-Method *modelling competence level* of the team. We expect that all the subjects will start the training without any prior competence in the OO-Method modelling language and the Integranova tool (in any case, this is assessed by a questionnaire). Although we always attempt to offer a good OO-Method training that helps every subject improve their abilities, it always turns out that teams acquire different levels of modelling competence during the training. Thus, we plan to assess their competence and, if significant differences arise, to block them in two groups: high-competence and average-competence teams. We want to later investigate the interaction effect with the derivation technique. The metric used to measure this variable is an indirect metric that aggregates three metrics measuring: (i) the outcome of some OO-Method training exercises (the average of several training exercises contributes up to 0.2 points to the competence level), (ii) the result of the 30-question OO-Method knowledge test (contributes up to 0.3 points), (iii) and the completeness of a conceptual model made by the team (contributes up to 0.5 points). The result of these measures (*modCompet*, which stands for *mod*elling *compet*ence) is a normalised value ranging from 0 to 1.

Parameters

The variables that we do not want to influence the experimental results have been fixed. However, we measure some of them in order to ascertain the homogeneity of the subjects and, in case of some kind of inevitable heterogeneity, whether this heterogeneity has a significant impact on the response variables.

- *Problem domain*. The general domain is information systems analysis; more specifically, a particular business domain is chosen; namely, the Photography Agency case.
- *Conceptual modelling language* used for the target conceptual model. OO-Method is used as a target language for creating the conceptual model of the photography agency.
- *Conceptual modelling tool*. Integranova is used as a tool to support the OO-Method conceptual models.
- *Requirements engineering competence*. We expect that all the subjects will start the training with a similar level of expertise in requirements engineering (in any case, this is assessed by a questionnaire).
- *Domain competence*. All the subjects will start the training without any prior competence in the photography agency domain. However, we plan to assess this fact by asking them whether they are familiar with this domain.
- *Amount of information included in the requirements specifications*. The same information is included in both requirements specifications. The only difference lies in how it is expressed (the modelling language and the structure of the specification).

7.5.1.4 Hypotheses

The hypotheses formulated from the research questions defined above are the following (a suffix is added to the variable name to indicate the technique: *CA* stands for Communication Analysis, *TS* stands for textual specification)

Hypothesis 1 (*RQ1*)

H_{10} (null hypothesis): $degCompl\_CA\_High = degCompl\_TS\_High$

For high-competence teams, the text-based and the communication-based derivation techniques allow obtaining conceptual models with same degree of semantic completeness.

H_{11} (alternative hypothesis): $degCompl\_CA\_High > degCompl\_TS\_High$

For high-competence teams, the communication-based derivation technique allows obtaining conceptual models with greater degree of semantic completeness.

Hypothesis 2 (*RQ1*)

H_{20} (null hypothesis): *degCompl_CA_Avg* = *degCompl_TS_Avg*

For average-competence teams, the text-based and the communication-based derivation techniques allow obtaining conceptual models with same degree of semantic completeness.

H_{21} (alternative hypothesis): *degCompl_CA_Avg* > *degCompl_TS_Avg*

For average-competence teams, the communication-based derivation technique allows obtaining conceptual models with greater degree of semantic completeness.

Hypothesis 3 (*RQ2*)

H_{30} (null hypothesis): *errValid_CA_High* = *errValid_TS_High*

For high-competence teams, the text-based and the communication-based derivation techniques allow obtaining conceptual models with same number of validity errors.

H_{31} (alternative hypothesis): *errValid_CA_High* < *errValid_TS_High*

For high-competence teams, the communication-based derivation technique allows obtaining conceptual models with a lower number of validity errors.

Hypothesis 4 (*RQ2*)

H_{40} (null hypothesis): *errValid_CA_Avg* = *errValid_TS_Avg*

For average-competence teams, the text-based and the communication-based derivation techniques allow obtaining conceptual models with same number of validity errors.

H_{41} (alternative hypothesis): *errValid_CA_Avg* < *errValid_TS_Avg*

For average-competence teams, the communication-based derivation technique allows obtaining conceptual models with a lower number of validity errors.

7.5.1.5 Experimental procedure

The experimental procedure is depicted in Figure 179 and explained next.

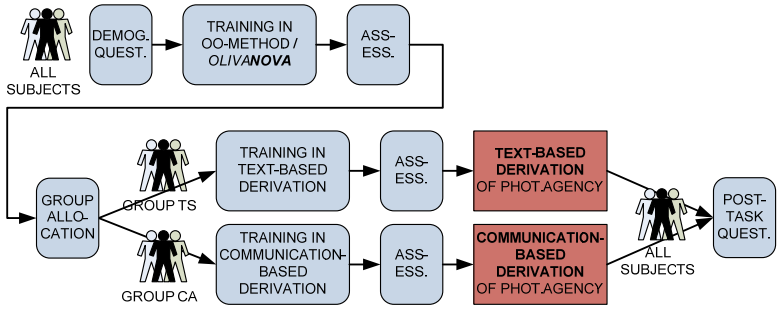


Figure 179. Experimental procedure

Demographic questionnaire

The experiment was initiated with the subjects answering the demographic questionnaire, which was supported by SurveyGizmo (a website that offers support to online surveys). Each subject has an identifier number ranging from 1 to 41 (however, there were some dropouts during the course).

A demographic questionnaire was applied with the purpose of identifying the background and experience using different specification techniques, such as Data Flow Diagram (DFD), Entity Relationship Diagram (ERD), Activity Diagram (AD), Class Diagram (CD), Use Case Diagram (UCD), State Transition Diagram (STD), Business Process Modeling Notation (BPMN), Relational Model for database design (RM) and Communication Analysis models (CA). The subjects rated in a 5-point Likert scale their knowledge level of the syntax of the different specification techniques (where 1 means "low" and 5 means "high"). They also rated their experience in applying the techniques (where 1 means "I have never user this technique" and 5 means "I have solved real cases professionally").

Table 72 shows the mean value and standard deviation of the group *knowledge of* and *experience in* the specification techniques. We consider that a subject reports having good knowledge of the syntax of a specification technique when s/he rates it over 3 in the 5-point Likert scale. We make a similar consideration for the experience in applying the techniques. Results indicate that, in general, the subjects report not being very knowledgeable of the syntax of the techniques. The better-known technique are the Class Diagram and the Relational Model; 15 subjects (57,7% of the total amount of subjects) have reported having a moderately high or a high knowledge level of their syntax.

Table 72. Descriptive statistics of demographic data (N=26)

Knowledge in	*DFD*	*ERD*	*AD*	*CD*	*UCD*	*STD*	*BPMN*	*RM*	*CA*
Mean	3.04	3.58	2.81	3.85	3.69	3.12	2.12	3.69	1.73
Std. deviation	0.91	1.02	1.05	1.08	1.01	1.07	1.30	0.97	0.96
Rating > 3	8	11	4	15	14	7	6	15	1
(num. and %)	30.8%	42.3%	15.4%	57.7%	53.8%	26.9%	23.1%	57.7%	3.8%

Experience in	*DFD*	*ERD*	*AD*	*CD*	*UCD*	*STD*	*BPMN*	*RM*	*CA*
Mean	2.92	3.46	2.50	3.65	3.38	2.77	1.92	3.46	1.54
Std. deviation	0.84	0.94	1.02	0.79	0.94	0.90	1.32	0.94	0.81
Rating > 3	5	10	4	16	13	5	3	12	1
(num. and %)	19.2%	38.5%	15.4%	61.5%	50.0%	19.2%	11.5%	46.2%	3.8%

The Use Case Diagram technique is also well known; 14 subjects (53,8%) report knowing the syntax well. The last technique that is worth mentioning is the Entity-Relationship Diagram, with 11 subjects (42,3%) reporting being knowledgeable of its syntax.

With regards to their experience in applying the techniques, the Class Diagram has been used by 16 subjects (61,5%) to solve moderately complex cases or even real cases in a professional environment. Use Case Diagrams, Relational Model and Entity Relationship Diagrams have also been used in moderately complex or real cases by 13 subjects (50,0%), 12 subjects (46,2%) and 10 subjects (38,5%), respectively.

We offer three charts showing the distribution of knowledge and experience ratings. According to our experience, the chart that corresponds to the Data Flow Diagram is quite representative of what master students usually claim to know about methods and techniques they have been taught during their studies.

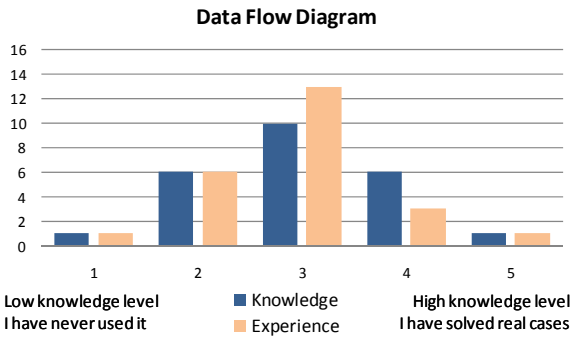


Figure 180. Distribution of knowledge and experience in the Data Flow Diagram

Both the knowledge and the experience follow a bell-shaped curve, the mode and the mean being around 3 (this varies among the techniques; see the means in Table 72). This means that they consider that they learnt the syntax good enough to pass their exams, and that they have applied them to solve small exercises that lack the complexity of real-world problems. The Activity Diagram and the State Transition Diagram have a similar distribution.

The distribution for the Class Diagram indicates a higher knowledge level and practical experience than for the previous techniques; this is convenient for our experiment since we expect them to create OO-Method conceptual models and we will place special interest in the Object Model (an extended Class Diagram). The Use Case Diagram and, to a lesser extent, the Entity Relationship Diagram and the Relational Model have a similar distribution.

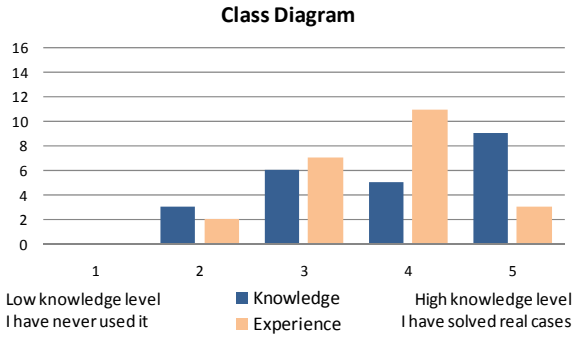


Figure 181. Distribution of knowledge and experience in the Class Diagram

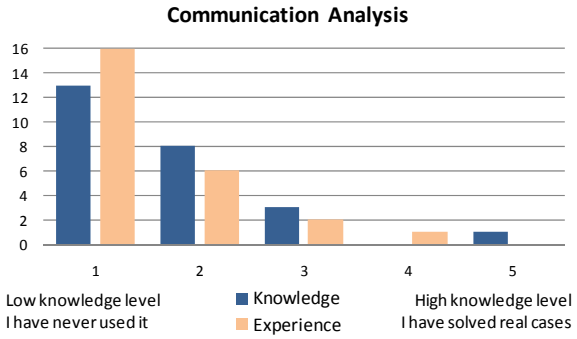


Figure 182. Distribution of knowledge and experience in Communication Analysis modelling techniques

The knowledge level of the subjects in the Communication Analysis modelling techniques is, as expected, very low. The reason is the fact that it is hardly known; unless the students have attended the course in which it is taught, the chances they have ever heard of the method are very low. Surprisingly, the Business Process Modeling Notation has a similar distribution.

It must be noticed that, among the 26 respondents, 6 subjects have been taught Communication Analysis in the past. Although only one of them reports having a good knowledge level of the syntax of Communication Analysis modelling techniques and having applied them in moderately complex cases, we want to take advantage of their knowledge in our experiment, as explained later on.

OO-Method training

Then the students were trained in the OO-Method, covering the concepts and modelling primitives of the Object Model, the Dynamic Model and the Functional Model. The students were also trained in using Integranova to create conceptual models. The training lasted 21 hours over 12 sessions (6 weeks), including theory and exercises with and without the tool).

OO-Method modelling competence assessment

After the OO-Method training, the students were evaluated in order to assess their knowledge level and their modelling competence. To assess the knowledge level of the subjects in the OO-Method (actually, knowledge and comprehension according to Bloom's taxonomy [Bloom, Engelhart et al. 1956]) a 30-question test was used. To assess their modelling competence (covering the application, analysis and -to some extent- synthesis levels of Bloom's taxonomy), they were handed out a small[112] case description and they created its corresponding conceptual model. To create this conceptual model they freely teamed up in pairs, and the same teams were maintained for the rest of the experiment. Each team has an identifier team number (ranging from 1 to 17). Two researchers (the course teacher and, me, his assistant) assessed the completeness of the models. As explained above, we aggregated in a variable the results of the training exercises, the knowledge test and the model quality assessment. This variable allowed us to classify teams into two levels of OO-Method modelling competence: high and average. We considered a team to have a high OO-Method modelling competence when the value of the competence variable is higher than or equal to 0.75, whereas the team has an average modelling competence hen the value of the competence variable is lower than 0.75. Since the variable ranges from 0 to 1 but teams having a score lower than 0.5 would be discarded, 0.75 constitutes a reasonable threshold.

[112] The model used as a reference solution has 5 classes, 16 attributes, 36 services and 5 relationships.

Allocation of teams to treatment groups

Then the teams were split into two treatment groups: group *TS* and group *CA* (whose names stand for textual specification and Communication Analysis, respectively). We applied blocking in order to be able to compare the effect of the OO-Method modelling competence on conceptual model quality, as well as the interaction of the effects of the OO-Method modelling competence and the derivation technique. For this comparison to be possible, we ensured balance between both treatment groups.

A purely random and balanced allocation can be done as follows. A list of the subjects is made (actually teams, in our case); the ordering criteria of the list (if any) are irrelevant. A random sequence of distinct numbers is generated (e.g. using the services of random.org, a website that generates random numbers using atmospheric noise). Each team in the list is assigned a number of the randomly generated sequence (the first team in the list is assigned the first random number in the sequence, and so on). Then, the list of teams is ordered according to the random number. Finally, each team of the list is alternatively allocated to a group (the first team to first group, the second team to the second group, the third team to the first group again, and so on). In order to decide which group is allocated a team in the first place, a coin is flipped.

If blocking is intended, then the procedure is similar, but the teams are first split into the different blocks according to the blocking parameter (we defined two levels of OO-Method modelling competence, in our case). The above mentioned allocation procedure is performed for each of the blocks separately.

These procedures are ideal; when working with human subjects, reality imposes some constraints that need to be dealt with flexibly, trying not to compromise or bias the allocation. The allocation procedure we ended up applying is described in Figure 183. We only clarify some steps.

In our case we needed to purposely allocate some teams to a specific treatment group (step 4). We refer to this as *fixed allocation* and it should be avoided unless there is a good reason to do it, since it violates allocation randomness and could eventually violate the balance among treatment groups. We did a fixed allocation for two reasons:

– The teams in which at least one of the subjects knew about Communication Analysis were allocated in treatment group *CA* straight away; namely, teams 4, 8, 9, 14 and 15. The rationale for doing this fixed allocation is that these subjects had already been trained in Communication Analysis, including some heuristic guidelines concerning the derivation of data models; thus, had they been allocated to the treatment group *TS*, we would not have been able to prevent them from applying their knowledge of these guidelines instead of the textual-based derivation guidelines.

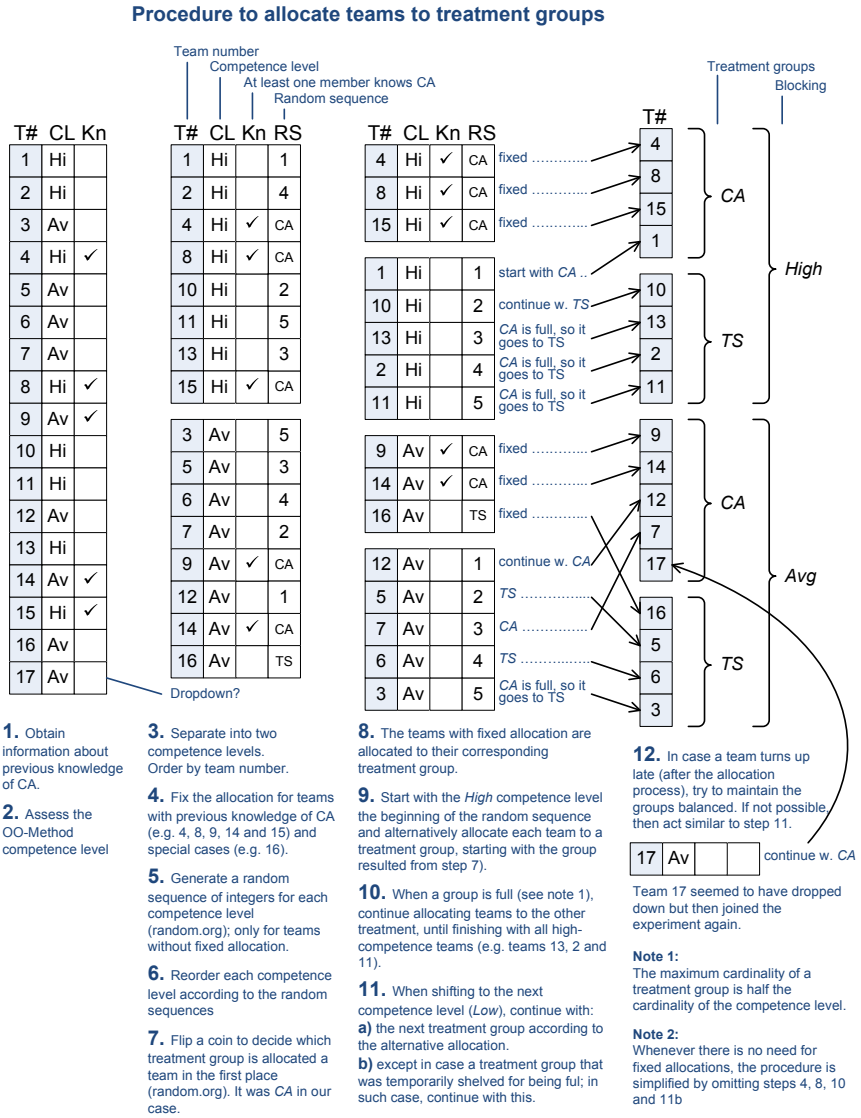


Figure 183. Procedure to allocate teams to treatment groups

– When we informed that the class would be split into two groups with different dates for the training sessions, one team reported not being able to assist one of the dates; namely, team 16.

In our case the fixed allocation was possible because there were only 5 teams that knew about Communication Analysis, 3 of them having a high competence level

and 2 of them having an average competence level. Therefore, the balance between both treatment groups (also taking into account the blocking of competence levels) was not compromised. The special case did neither compromise the balance.

We used the services of random.org (a website that generates random numbers using atmospheric noise) to generate two random sequences of numbers, one for each competence level: 1, 4, 2, 5, 3 and 5, 3, 4, 2, 1. Also, we used random.org to flip a coin in order to select a treatment group to start the random allocation with: *CA* was selected.

Treatment: training in a derivation technique

Then each treatment group was trained in a different technique for deriving conceptual models from requirements models:

- The teams in group *TS* were trained in text-based derivation of conceptual models.
- The teams in group *CA* were trained in communication-based derivation of conceptual models.

In both trainings, derivation guidelines and heuristics are offered. The Pedagogical Reinforcement Pattern was used to design the training material [Berenbach and Konrad 2008]. First the concepts and the guidelines are explained and illustrative examples are offered, then the subjects solve small exercises. Once the subjects know the derivation technique, they apply it to a mid-sized example; namely, the SuperStationery case. The solution to this case is discussed later on in class. The training for each group amounted 3.5 hours over two sessions (1 week).

Experimental task: derivation of a conceptual model

After the training sessions, both groups were merged once again and they were assigned a case to solve; namely, the Photography Agency case. The input for this task is different for each treatment group. Group *CA* receive a Communication Analysis requirements model of the case, whereas group TS receive a textual requirements model of the case. Both models contain the same information but expressed differently. The time allotted for deriving the conceptual model of the case and creating it with the Integranova Modeler was 7 hours over 4 sessions (2 weeks). After each session, the teams had to send a snapshot of their model to the researchers. The models sent after the fourth session are considered the actual result from this task and they are the main artefact being analysed in the experiment.

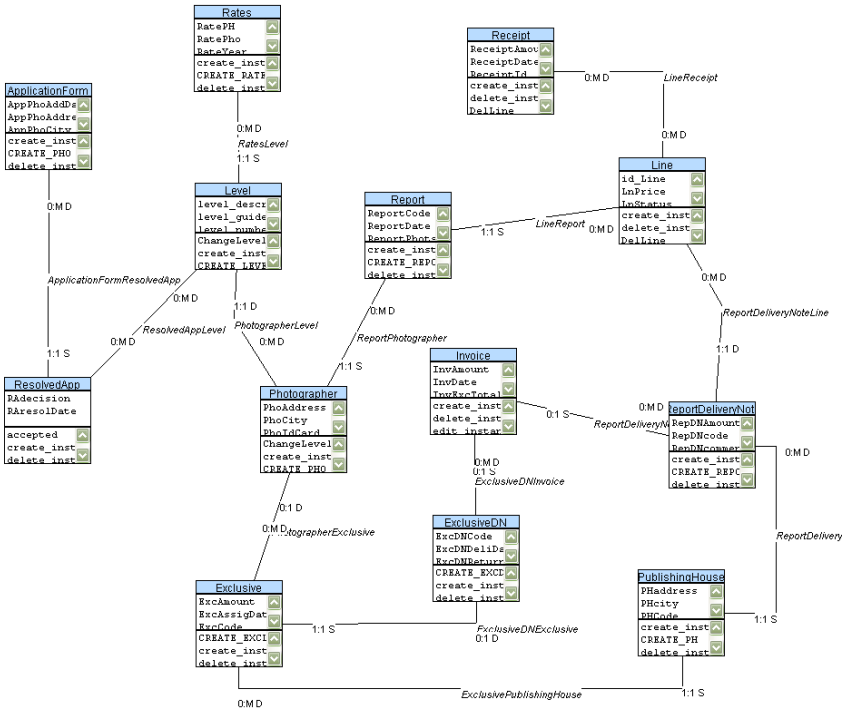


Figure 184. A model of the Photography Agency created with Integranova Modeler by an experimental subject

Post-task questionnaire and discussion

After the modelling task, the subjects answer a questionnaire along with a discussion to gather some feedback on the derivation technique. This post-task questionnaire was supported online by SurveyGizmo and asked for the three bigger advantages of applying the derivation technique (compared to not having any derivation technique at all), the three main drawbacks of the derivation technique (compared to having complete freedom and lack of rules for creating the conceptual model), and in which aspects the subjects thought the derivation technique could be improved (for instance, indicating what guidelines they missed in the derivation technique, or which modelling primitives of the conceptual model were the most difficult to derive from the requirements model). The aim was to elicit some feedback from the subjects. We later treated the qualitative data to draw some conclusions.

7.5.2 Results analysis and interpretation

7.5.2.1 OO-Method modelling competence of the teams

The OO-Method modelling competence of the teams was calculated by means of metric *numCompetNor*, which normalises the numeric indicator between 0 and 1. Table 73 shows the descriptive statistics of the competence level, both disregarding the treatment (for the whole sample of teams) and taking it into account (splitting by treatment). Note that one team was not assessed (it seemed a dropdown at that moment).

Table 73. Descriptive statistics of the competence level

Measure	N	Mean	Std. deviation
modCompet	16	0.7196	0.09439
modCompet_CA	8	0.7341	0.05554
modCompet_TS	8	0.7050	0.12459

Figure 185 shows the boxplot of the modelling competence, depending on the treatment. The outlier in group TS corresponds to a team that did not perform well in the competence assessment.

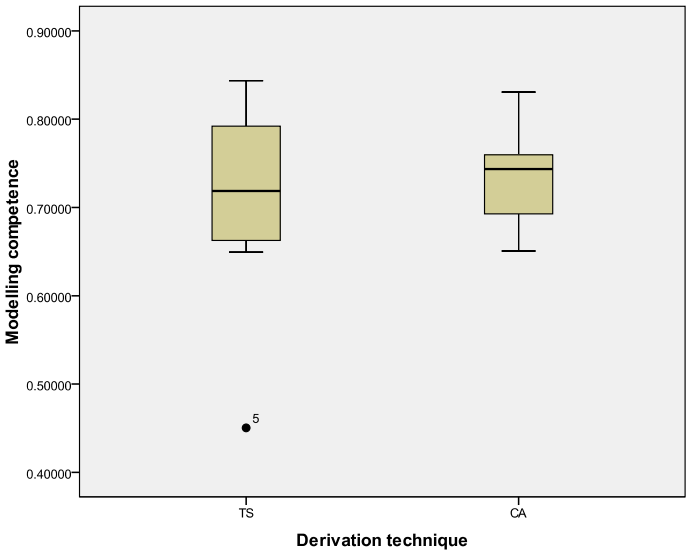


Figure 185. Boxplot of the OO-Method modelling competence (*modCompet*)

We expect that the difference between the modelling competences of the teams belonging to the two treatment groups is not significant. Otherwise, it would be a threat to the validity of the treatment comparison. To compare the means an independent-samples T-test can be applied, as long as its three assumptions are fulfilled.

Assumption 1: The two samples are independent of one another. Given the experimental design, both treatment groups are independent samples (each team was trained in only one derivation technique).

Assumption 2: The data from the two samples are both normally distributed. We apply the Shapiro-Wilk test of normality to *modCompet* and it indicates that the distribution is normal both for group TS (p=0.281>0.05) and group CA (p=0.925>0.05). The Kolmogorov-Smirnov test of normality is usually applied to larger samples; in any case it also indicated that the distribution is normal (p=0,200>0.05).

Assumption 3: The two samples have approximately equal variance on the dependent variable. According to the Levene test for the equality of variances, both competence levels have approximately equal variance on the response variable treatment (p=0.134>0.05).

Since the three assumptions are fulfilled, we apply the test and it indicates that the difference between the means is not significant (t=0.603 and p=0.556>0.05). In other words, the difference between the modelling competences of the teams belonging to the two treatment groups is not significant. Thus, we can be confident that the allocation did not introduce any bias with regards to the modelling competence.

7.5.2.2 Actual effectiveness of the derivation techniques

An expert reviewer used the reference model and a correction template to review the conceptual models. The conceptual models of the 17 teams where thoroughly reviewed; the mean review time was 1:03:30 (ca. 1 hour and 3 minutes). The measures obtained by the expert reviewer are analyzed in the following. See Table 74.

Table 74. Descriptive statistics considering treatment

Measure		**N**	**Mean**	**Std. deviation**
Degree of completeness	*degCompl_CA*	9	0.6815	0.07616
	degCompl_TS	8	0.6436	0.08585
Validity errors	*errValid_CA*	9	15.56	9.475
	errValid_TS	8	20.50	13.690

The degree of conceptual model completeness (*degCompl*) is 68.15% for the communication-based derivation technique and 64.36% for the text-based derivation technique. With regards to validity errors (*errValid*), the conceptual models created by the teams applying communication-based derivation had an average of 15.56 errors, whereas those created by teams applying text-based derivation had an average of 20.50 errors. However, it is necessary to further analyse the significance of these differences.

However, as discussed above, we should take into account the blocking and analyse separately the high-competence teats and the average-competence teams.

7.5.2.2.1 Completeness of the models (blocked comparison)

We investigate now the difference in the means of the degree of completeness but now blocking by modelling competence level. We compare the performance of teams with high and average OO-Method modelling competence separately. Table 75 shows the descriptive statistics.

It appears that the difference in the effect of the derivation technique over the degree of completeness is bigger for high-competence teams than for average-competence teams. Within the high competence level, the models created by the teams applying communication-based derivation (CA) have a mean degree of completeness of 74.38% whereas those created by the teams applying text-based derivation (TS) have a mean degree of completeness of 65.17%; there is a difference of 9.22% in favour of the communication-based derivation technique. Within the average competence level, the mean degree of completeness for group CA is 63.16% whereas for group TS is 63.55%; there is a difference of 0.38% in favour of the text-based derivation technique. Figure 186 shows the corresponding boxplots.

Table 75. Descriptive statistics of model completeness (*degCompl*), blocking by competence level

Competence level	Measure	N	Mean	Std. deviation
High	*degCompl_CA_High*	4	0.7439	0.05256
	degCompl_TS_High	4	0.6517	0.05191
Average	*degCompl_CA_Avg*	5	0.6316	0.05027
	degCompl_TS_Avg	4	0.6355	0.11969

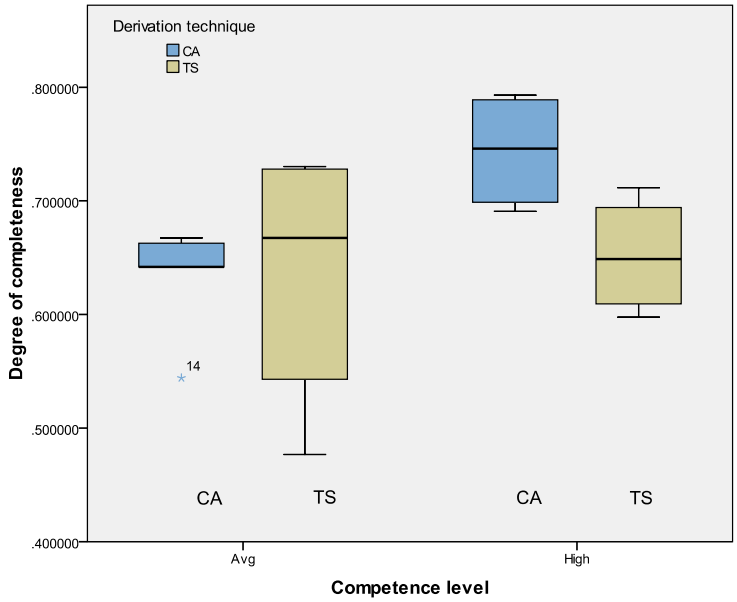


Figure 186. Boxplot of the model completeness (*degCompl*) for both treatment groups, blocking by competence level

High-competence teams

To investigate whether the difference for the high competence level is significant we intend to apply the independent-samples T-test, which relies on three assumptions.

Assumption 1: The two samples are independent of one another. This is true, see above.

Assumption 2: The data from the two samples are both normally distributed. The sample is too small to outline a bell curve, so we omit analysing the histograms and perform a more formal test to assess normality. The Shapiro-Wilk test on normality indicates that the distribution is normal (for TS, p=0.771>0.05; for CA, p=0.187>0.05).

Table 76. Tests of normality for the degree of completeness (*degCompl*) for the high competence level

Derivation technique	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
TS	0.224	4	.	0.949	4	**0.711**
CA	0.282	4	.	0.837	4	**0.187**

[a]. Lilliefors Significance Correction

Table 77. Mean comparison for the degree of completeness for the high competence level (independent samples T-test for *degCompl*)

	Levene's test for equality of variances		**T-test for equality of means**						
								95% confidence interval of the difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean differ.	Std. error differ.	Lower	Upper
=	0.079	**0.789**	-2.495	6	**0.047**	-0.0921	0.036935	-0.182529	-0.001773
≠			-2.495	5.99	0.047	-0.0921	0.036935	-0.182532	-0.001770

= Equal variances assumed, ≠ Equal variances not assumed

Assumption 3: The two samples have approximately equal variance on the dependent variable. According to the Levene test for the equality of variances (see Table 77), both competence levels have approximately equal variance on the response variable treatment (p=0.789>0.05).

Since the three assumptions are fulfilled, we can apply an independent samples T-test to verify the null hypothesis H_{10}. As shown also in Table 77, the test results indicate that the difference is indeed significant (p=0,047<0,05). Thus the null hypothesis H_{10} is refuted with a 95% confidence and the alternative hypothesis H_{11} is corroborated. This means that the communication-based derivation technique, when applied by analysts with a high competence in OO-Method modelling, allows obtaining conceptual models with greater degree of completeness than the text-based derivation technique.

Average-competence teams

With regards to the average competence level, the T-test cannot be applied because one of the two samples does not have a normal distribution (see Table 78); that is, the sample for group CA does not pass the Shapiro-Wilk test of normality (for TS, p=0.304>0.05; but for CA, p=0.029≤0.05).

Table 78. Tests of normality for the degree of completeness (*degCompl*) for the average competence level

Derivation technique	**Kolmogorov-Smirnov[a]**			**Shapiro-Wilk**		
	Statistic	df	Sig.	Statistic	df	Sig.
TS	0.224	4	.	0.872	4	**0.304**
CA	0.381	5	0.017	0.748	5	**0.029**

[a]. Lilliefors Significance Correction

Table 79. Results of the Mann-Whitney U test on the degree of completeness (*degCompl*) for the average competence level

Ranks

Derivation technique	N	Mean rank	Sum of ranks
TS	4	5.25	21.00
CA	5	4.80	24.00
Total	9		

Test Statistics[b]

Mann-Whitney U	9.000
Wilcoxon W	24.000
Z	-0.246
Asymp. Sig. (2-tailed)	0.806
Exact Sig. [2*(1-tailed Sig.)]	0.905[a]

[a]. Not corrected for ties.

[b]. Grouping variable: derivation technique

Therefore, we apply a non-parametric test for unpaired samples that allows for non normal distributions (see Table 79). The Mann-Whitney U test indicates that the difference is not significant (p>0.05) so we cannot refute H_{20}, that is, we cannot conclude that the derivation technique has a significant effect on the degree of completeness of the models created by analysts of an average OO-Method modelling competence.

Discussion

Summing up, we can conclude that the communication-based derivation technique has a significant effect on the degree of completeness of the derived conceptual models, but only when the team of analysts has a high competence level on the conceptual modelling technique. This result can be explained by our observations during the experiment on how the subjects interiorised the derivation technique. Those subjects with a higher competence in the OO-Method seemed to grasp the concepts and guidelines of the communication-based derivation technique better that those subjects with lower competence. It may be due to the fact that highly the outstanding modellers are those analysts with better capacity for abstraction; they would be better suited to take advantage of the systematic derivation rules. On the contrary, the subjects in the average competence level, could be those with more difficulty to apply abstract thinking; if that was so, they could sense the communication-based derivation technique as a burden, yet another complex task to solve. We acknowledge that this just an ad-hoc hypothesis that needs to be further investigated and, eventually, perhaps corroborated.

7.5.2.2.2 Validity of the models (blocked comparison)

We investigate now the difference in the means of the degree of completeness blocking by modelling competence level. We compare the performance of teams with high and average OO-Method modelling competence separately. Table 80 shows the descriptive statistics. The means show differences favourable to the communication-based derivation technique, but their significance needs to be assessed.

Table 80. Descriptive statistics of the validity errors (*errValid*), blocking by competence level

Competence level	Measure	N	Mean	Std. deviation
High	*errValid_CA_High*	4	16.00	11.690
	errValid_TS_High	4	21.75	20.172
Average	*errValid_CA_Avg*	5	15.20	8.758
	errValid_TS_Avg	4	19.25	5.123

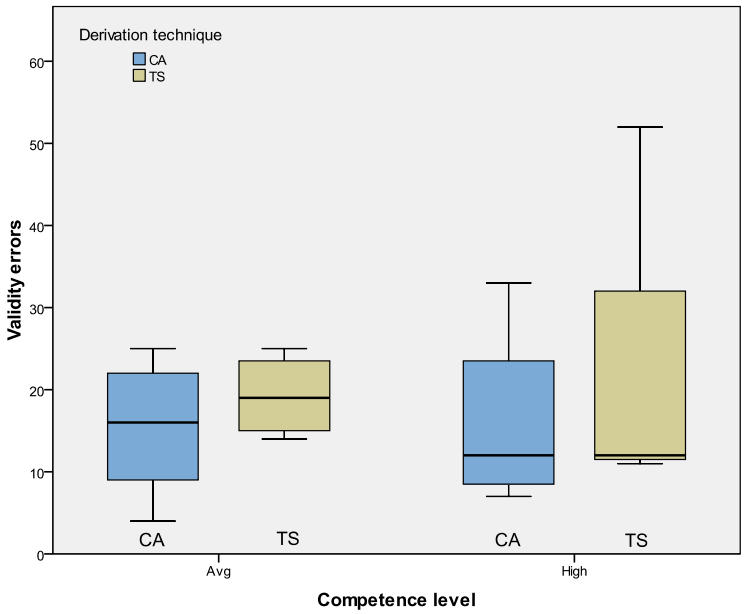


Figure 187. Boxplot of the validity errors (*errValid*) for both treatment groups, blocking by competence level

High-competence teams

We first investigate the teams with a high competence level. The distribution of the validity errors is not normal (see Table 81), according to the Shapiro-Wilk test of normality (it fails for TS, p=0.003≤0.05).

Therefore, we apply a non-parametric test for unpaired samples that allows for non normal distributions (see Table 82). The Mann-Whitney U test indicates that the difference is not significant (p>0.05) so we cannot refute H_{30}, that is, we cannot conclude that the derivation technique has a significant effect on the validity errors of the models created by analysts of a high OO-Method modelling competence.

Table 81. Tests of normality for the validity errors (*errValid*) for the high competence level

Derivation technique	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
TS	0.436	4	.	0.650	4	**0.003**
CA	0.318	4	.	0.838	4	**0.189**

[a]. Lilliefors Significance Correction

Table 82. Results of the Mann-Whitney U test on the validity errors (*errValid*) for the average competence level

Ranks

Derivation technique	N	Mean rank	Sum of ranks
TS	4	5.00	20.00
CA	4	4.00	16.00
Total	8		

Test Statistics[b]

Mann-Whitney U	6.000
Wilcoxon W	16.000
Z	-0.581
Asymp. Sig. (2-tailed)	0.561
Exact Sig. [2*(1-tailed Sig.)]	0.686[a]

[a]. Not corrected for ties.

[b]. Grouping variable: derivation technique

Average-competence teams

We now investigate the teams with an average competence level. We intend to apply a parametric test to compare the means.

Assumption 1: The two samples are independent of one another. True (above).

Assumption 2: The data from the two samples are both normally distributed. The Shapiro-Wilk test on normality indicates that the distribution is normal (see Table 83; for TS, p=0. 594>0.05; for CA, p=0. 755>0.05).

Assumption 3: The two samples have approximately equal variance on the dependent variable. According to the Levene test for the equality of variances (see Table 84), both competence levels have approximately equal variance on the response variable treatment (p=0.245>0.05).

Since the three assumptions are fulfilled, we can apply an independent samples T-test to verify the null hypothesis H_{40}. As shown also in Table 84, the test results indicate that the difference is not significant (p=0,443≥0,05). Thus the null hypothesis cannot be refuted. This means that we cannot conclude that the derivation technique has a significant effect on the validity errors of the models created by analysts of an average OO-Method modelling competence.

Table 83. Tests of normality for the validity errors (*errValid*) for the average competence level

Derivation technique	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
TS	0.237	4	.	0.930	4	**0.594**
CA	0.181	5	0.200*	0.953	5	**0.755**

[a]. Lilliefors Significance Correction

*. This is a lower bound of the true significance.

Table 84. Mean comparison for the validity errors for the high competence level (independent samples T-test for *errValid*)

	Levene's test for equality of variances		T-test for equality of means						
								95% confidence interval of the difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean differ.	Std. error differ.	Lower	Upper
=	1.607	**0.245**	0.813	7	**0.443**	4.050	4.978	-7.722	15.822
≠			0.865	6.555	0.417	4.050	4.978	-7.171	15.271

= Equal variances assumed, ≠ Equal variances not assumed

Discussion

Our procedure for measuring semantic completeness was very thorough: we assessed each and every requirement that ought to be represented in the conceptual model. On the contrary, our procedure for measuring validity we less systematic: we only counted invalidities that we came across while measuring completeness; we expect to have found many of them but we cannot ensure that all invalidities were spotted. We can assume that we found approximately the same percentage of invalidities in each model; however, we can neither be sure of this. Therefore, there are threats to the validity of the results concerning model validity (if you will forgive the repetition).

Despite the non-significant statistical results, we still have two ad-hoc hypotheses that need to be further investigated and, eventually, perhaps corroborated.

A possible explanation is that the communication-based derivation technique is still not covering all the primitives of the OO-Method and, whenever the subjects intend to increase the completeness of their models and try to model some statements about the domain for which the derivation technique does not offer guidelines, they make mistakes that increase the invalidity. Improving and extending the communication-based derivation technique could lead to a significant difference in the validity of the models resulting from both derivation techniques.

Or it could actually occur that, if a more thorough procedure for measuring validity is followed, then the results will become significantly different. The difficulty lies in the fact that covering all invalidities is too time-consuming when models as big as the Photography Agency are considered. Such an investigation should be made using a smaller case.

7.5.3 Validity evaluation

This section discusses issues with the potential to threaten the validity of the experiment [Wohlin, Runeson et al. 2000].

7.5.3.1 Conclusion validity

A random heterogeneity of subjects could give rise to greater variability in the measures, due to confounding factors. However, we verified that the subjects had a quite homogeneous background by means of a questionnaire, so there is no threat. As a trade-off, homogeneity limits external validity.

However, due to different OO-Method modelling competences of the subjects, we opted for applying blocking: we classified teams into two levels of modelling

competence. This way we could assess the effect of the modelling competence in the quality of the resulting models, as well as investigate the interaction of the modelling competence with the treatment (the conceptual model derivation technique). This provides a more detailed analysis of the mechanisms in play.

The proposed measures related to actual efficacy have been widely used by many researchers, since they come from a well-known quality framework by Lindland et al. [1994]. However, the instruments used to measure the quality are subject to threats (see below). Also, each model was only assessed by one reviewer. We acknowledge that it is convenient to have the models reviewed by more expert reviewers and to calculate the inter-reviewer agreement. This was not possible due to resource constraints. Note that 17 models where thoroughly reviewed; the mean review time was 1:03:30 (ca. 1 hour and 3 minutes). We plan, however, to research on different ways of reviewing the models so as to decrease the review time, while still obtaining a precise measure of the quality.

7.5.3.2 Internal validity

Instrumentation is the effect caused by the instruments used in the experiment. In previous experiments, we had experienced the error-proneness of paper form surveys (see Section 6.5.5.2); to avoid it, we used SurveyGizmo as a support for web-based forms. This allows enforcing the compulsoriness of certain answers and minimises transcription errors (since the results can be directly downloaded as a spreadsheet).

There is a risk of the subjects not understanding textual parts of the instruments correctly, due to the fact that it is in English and the mother tongue of the subjects is Spanish. Our impression during the experiment operation was that some aspects of the case description were not fully grasped. Whenever we found a team having a misconception due to a misinterpretation of the text, we helped them understand the meaning of the text; but some misconceptions may have remained unnoticed.

There is a risk related to the allocation of subjects to groups. We applied a random allocation procedure mixed with fixed allocations. Although complex, this procedure was designed with the honest intention of avoiding bias. Since in this experiment the treatment is a method, the only sensible decision with respect to those teams that already knew one of the methods was to allocate them to the group that received that treatment. Otherwise, their combined knowledge of both methods would have had an uncontrolled influence on the results. With respect to the only team that requested being allocated to a specific treatment, it should be noted that the team did not know at that moment, what the treatment was; their request was due to their unavailability to attend the training corresponding to the other treatment. Since we were aware of their commitment to the course,

we did not sense in this action any threat to the validity. It is simply a contingency related to using students as experimental subjects.

7.5.3.3 Construct validity

As in previous experiments (see Section 6.5.5.3), two of the researchers involved in the experiment are authors of Communication Analysis who have expectancies about this method performing better. In order to reduce the threat of bias, two experimenters without expectancies have been involved.

The experiment includes a single information system description so it may under-represent the construct of all information systems.

7.5.3.4 External validity

With respect to the use of students as experimental subjects, several works suggest that, to a great extent, the results can be generalised to industry practitioners [Runeson 2003]. In any case, we are aware that more experiments with a larger number of subjects are necessary and, that using practitioners would lead to more reliable results.

We thoughtfully selected a representative problem statement. However, more empirical studies with other requirements specifications are necessary. Although it is a big effort, we are currently building a repository of cases to use in experiments.

7.5.4 Discussion

The experiment has allowed us to test the conceptual model derivation technique with subjects that have learnt the OO-Method for the first time. The subjects teamed up in pairs and were trained in the OO-Method. Since this method, although it is based in the UML, contains specific languages such as the Functional Model, its training takes several sessions. Our many-years experience in OO-Method education and training shows that not all students/practitioners grasp the concepts and guidelines easily. This is a risk when attempting to use inexperienced analysts as experimental subjects. We are not actually investigating the features of the OO-Method in the experiment, but of the compared derivation techniques; however, the conceptual modelling method is the foundation stone of the experiment and the subjects are required at least a minimum expertise in it. For this reason, we assessed the OO-Method modelling competence of each team after the training. We differentiate two blocks: teams with a high competence and teams with an average competence. Then the teams were allocated into two treatment groups, preserving the blocking. Unlike in the experiment reported in

Section 6.5, the allocation procedure achieved more balanced groups. The treatment consisted in training the teams in one of the two conceptual model derivation techniques being compared: either text-based derivation or communication-based derivation. During the experimental task, the teams derived an OO-Method conceptual model from a requirements model. The semantic quality attributes measured in the models were completeness and validity.

The analysis of results has shown that, with regards to high-competence teams, the there is a significant difference between semantic completeness of the OO-Method conceptual models created by teams that applied the communication-based derivation technique (see Chapter 5) and those created by teams applying the text-based derivation technique. The more requirements from the Photography Agency that are supported by the conceptual model, the higher the semantic completeness. This statistical result coincides with our appreciation that the teams applying the technique were acting more thoroughly and systematically that those of the other block, which we interpret that has led to the higher completeness.

The difference in model completeness within the average-competence teams, although again favourable to the communication-based derivation technique, is not statistically significant. A hypothesis for this phenomenon is that, unless a team has enough competence to feel comfortable with OO-Method modelling, the derivation guidelines entail an excessive cognitive workload. Some teams with an average competence level got stuck during the experimental task, trying to find out how to express a precondition or a transaction formula. Their inexperience seems to have led them to spend more time than highly competent teams to construct the conceptual model. Since the duration of the experimental task was restricted, some average competence teams reported having to prioritise in order to first attempt to model the statements of the requirements model they were confident to be able to express.

In any case, we claim that the teams of the high-competence block have more similarities to real OO-Method practitioners than the teams of the average-competence block (actually real practitioners are far more competent that the most competent team participating in the experiment). Thus, we are confident that their performance is closer to the performance that would have resulted in case the experimental subjects had been real practitioners.

Besides the quantitative analysis, some interesting qualitative information was also gathered during the experiment. We noted down which aspects of the technique were more difficult to grasp by the subjects, which misconceptions were more frequent. Also, after the task, an open discussion was held so as to gather some feedback regarding the experience. Additionally, a survey with open questions asked for benefits and drawbacks of the derivation technique. This

qualitative information has been analysed and categorised in order to conclude which conceptual modelling primitives the derivation technique facilitates identifying, and which are the primitives for which some improvement is needed. For each subject, it was identified which modelling primitives were mentioned (in a positive or in a negative sense), and totalisations were made. Figure 188 shows the perceptions of the subjects that applied the communication-based derivation technique with regards to the strengths and weaknesses of the approach. The green bars indicate how many subjects mentioned that the technique helped them to identify a given conceptual modelling primitive. For instance, 10 subjects reported that it facilitated the identification of classes; the modelling of attributes, relations and agents was also benefited. For instance, one subject commented that "It is clear how to structure classes and attributes." On the other hand, it became evident that there was still space for improvement, as shown by the red bars. 5 subjects reported to have difficulties in identifying restrictions such as relationship cardinalities and service preconditions; the subjects also experienced trouble identifying services. As one subject commented, "the primitives that were most difficult to derive are restrictions and services." Many subjects also appreciate that the approach is systematic: "the communicative event diagram is very useful to define the order in which the model is created."

Another interesting remark made by the subjects was the fact that the "derivation rules restrict many design decisions" and limit the modeller creativity. On the other hand, many subjects expressed that they had found the technique intuitive.

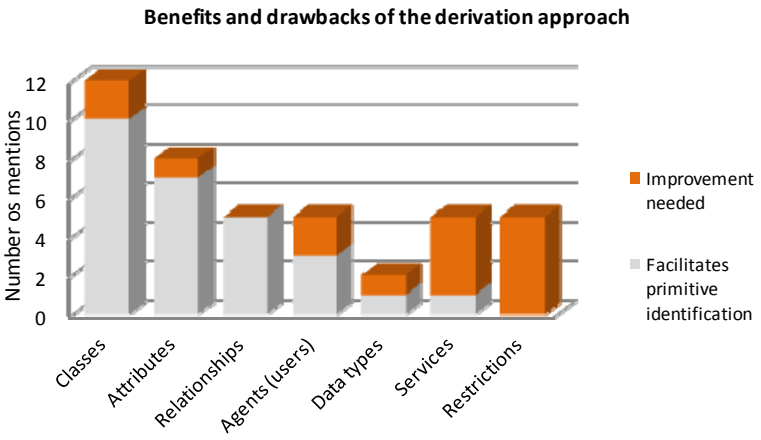


Figure 188. Subjective feedback from the 14 subjects using the derivation technique

According to our observations and the subjects' feedback, the guidelines that the teams had more trouble to grasp and apply correctly are, by far, those related to class extension. Note for instance the fragment of a model shown in Figure 189 where the team created a separate class for each communicative event affecting the exclusive report business object, instead of extending the class derives in the locally initiatory event; namely EXCLUSIVEREPORT. They were unable to figure out how to relate the classes. This of course, is an extreme, infrequent, case. Figure 190 shows a fragment of a correct model, in which class EXCLUSIVEREPORT has been properly extended while processing the communicative events. It corresponds to a version of the reference model created with Integranova Modeler.

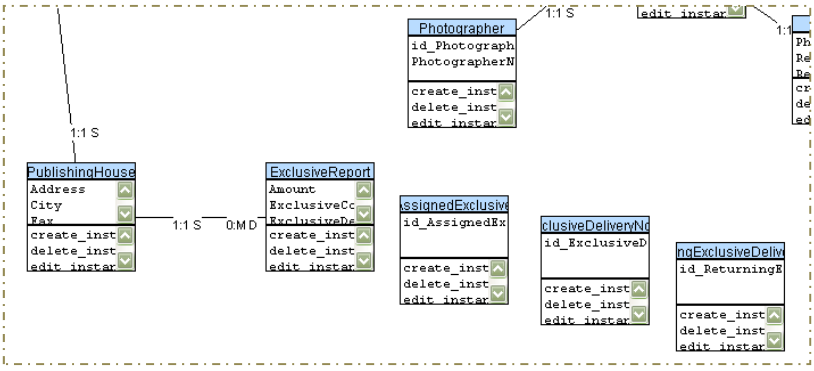


Figure 189. Fragment of a model resulting from the incorrect application of class extension derivation rules

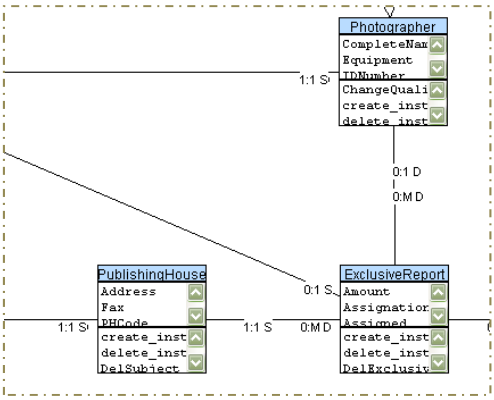


Figure 190. Fragment of a model in which class extension derivation rules have been correctly applied

In the second version of the derivation technique, all the feedback gathered was used to improve the derivation rules. The extension of classes has been expressed in clearer terms, leading to rules **OM15** and **OM16**. The second version of the rules has been put to test in a replication of the experiment described above, run in the same context in the year 2011. However, due to the time-consuming procedure for the assessment of the conceptual model quality, the results are not available yet.

7.6 Summary

In Chapter 5, a technique for deriving OO-Method conceptual models from Communication Analysis requirements models has been presented. In Chapter 7, we have undertaken the validation of this derivation technique in order to ascertain its feasibility and compare it to the current practice of OO-Method practitioners.

The experience with the lab demos (see several of them described in Section 7.3) has allowed the researchers to demonstrate that the approach is feasible in practice. An advantage of following a systematic procedure to create conceptual models is that it allows to ground conceptual modelling decisions in the information from the requirements model (diagrammatic elements, textual statements, etc.). The lab demos not only have served as a proof of concept, but have also provided very useful feedback (see, e.g. Section 7.3.4) that has been used to improve the derivation technique iteratively.

We have also designed and conducted a controlled experiment (see Section 7.5) in order to investigate how other analysts apply the technique and to compare the communication-based derivation technique with a text-based derivation technique that resembles how OO-Method practitioners create conceptual models in real development projects (however our experimental setting in a master course greatly differs from a real, industrial setting). The results show that, for the teams of analysts with high-competence in the OO-Method, there is a significant difference in the conceptual model completeness that can be attributed to the applied derivation technique. The teams applying the communication-based derivation technique produced models that are 9.22% more complete that the models produced by the teams applying the text-based derivation technique. Besides the quantitative results, the experiment also allowed us to gather qualitative data that we later used to improve the derivation technique.

Summing up, in our research effort to provide a model-driven information system development method that covers from requirements engineering to automatic software-code generation, we acknowledge that there is still space for

improvement, that there are still some loose ends that need to be tied up. However, the validations described in this chapter have shown evidences that the approach is promising, that it can be successfully applied (not only by the researchers, but also by master students) and that it performs better than a text-based derivation. We still investigate the application of our derivation technique under conditions of practice (e.g. conducting action research). However, we consider that the technique has been sufficiently validated to consider the implementation of tool support. In Section 8.3, we describe the implementation of the derivation rules using ATL Transformation Language. The transformation rules take as input Communication Analysis models created by the modelling tool whose implementation is described in Section 8.2.

PART IV.
SOLUTION IMPLEMENTATION

Chapter 8
Technological support

"At each increase of knowledge, as well as on the contrivance of every new tool, human labour becomes abridged."

Charles Babbage

8.1 Motivation

In software development, methods are important because they facilitate working systematically (e.g. grounding decisions, achieving repeatable results). In model-driven development (MDD), models play a pivotal role and methods usually operate on them, facilitating their creation, maintenance, analysis, etc. Moreover, MDD methods must be carefully designed taking into account the alignment of concepts that belong to different abstraction layers, so as to allow for model transformations. However, models can become a heavy burden unless the appropriate technological support is provided. There exist many activities that need to be supported by tools (e.g. requirements verification and validation, cost estimation, model-based testing), but due to time constraints we have restricted the technological support to the creation of Communication Analysis requirements models (see Chapter 4) and to the automatisation of the OO-Method conceptual model derivation technique (see Chapter 5)[113]. With regards to traceability management, we will support the creation of trace links during model transformations, but the later management of these links is not considered.

Nowadays, there exist numerous MDD tools that support modelling of the computerised information system. Talking in terms of the Model-Driven

[113] Actually, a simplified version of Communication Analysis requirements models is targeted and only the Object Model derivation rules are considered.

Architecture, most tools address the platform independent and/or the platform specific views, but disregard the computation independent view. Talking in terms of the Communication Analysis requirements structure (see Section 4.2), most tools address levels L4 and L5, but disregard levels L1, L2 and L3. There is indeed little support to requirements engineering in industrial MDD tools, especially with regards to model transformation (they generally cover textual specification and traceability management). In academia, some proposals start to appear. Figure 191 shows the results of the systematic review of articles concerning model-driven requirements engineering methods by [Loniewski, Insfran et al. 2010], with regards to the automation level of transformation rules. However, the percentage of automatic exogenous transformations addressing business requirements is probably much lower (the authors did not perform this analysis). We intend to contribute to filling this gap in the model-driven development technology.

The remaining of the chapter is structured as follows. Section 8.2 describes the technological framework that we have developed to support Communication Analysis; this includes the adaptation of an office suite (specifically, the tools Microsoft Visio and Microsoft Word) for organisations not willing to change their technology, and the implementation of a modelling tool based on Eclipse Modeling Framework. Section 8.3 describes the implementation of the transformation rules that automate the derivation technique, using the ATL Transformation Language. Section 8.4 concludes with a summary of the chapter.

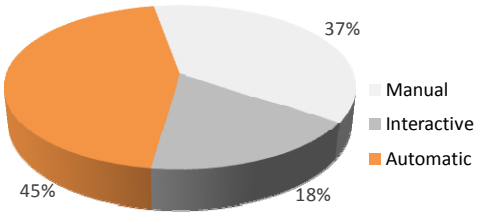


Figure 191. Automation level of transformations in model-driven requirements engineering, according to [Loniewski, Insfran et al. 2010]

8.2 *Support for Communication Analysis*

8.2.1 Adaptation of an office suite

One of the several underlying practical intentions within this thesis is to foster the adoption of Communication Analysis by organisations currently applying the OO-Method. Another intention is to foster the adoption of model-driven development practices by the organisations currently applying Communication Analysis. We are aware that some organisations currently applying Communication Analysis might want to keep their customary way of working. This involves using office-suite tools such as general-purpose diagramming tools and word processors to create requirements models, instead of using CASE tools or model-driven development frameworks. We also wish to provide a better support for the method for such organisations and not only for organisations willing to take a leap into the model driven development paradigm.

This way, we have provided support for Communication Analysis (according to the improved definition presented in Chapter 4) based on Microsoft Office, versions 2007 onwards. The documentation of the requirements models are supported by means of Microsoft Word templates, most of them have already been shown in previous sections (e.g. see the process specification template in Figure 69, and an example of the event specification template in Figure 67).

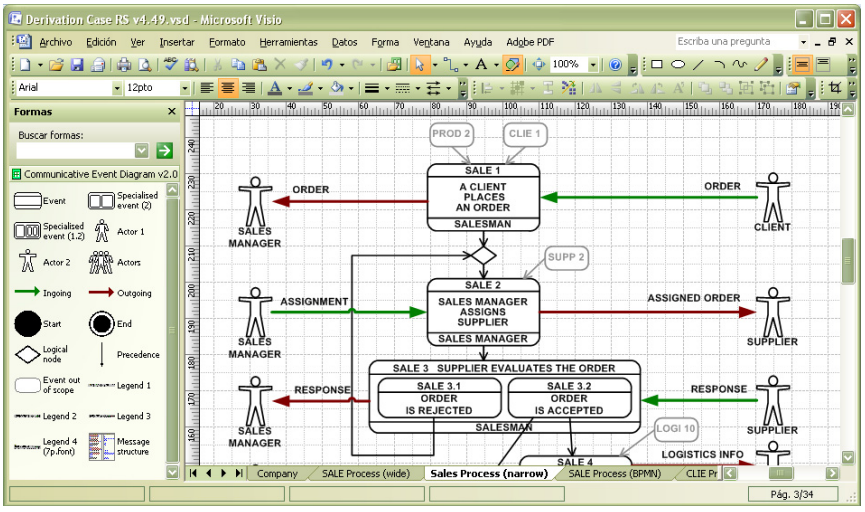


Figure 192. A Microsoft Visio stencil for Communicative Event Diagram

With regards to Communicative Event Diagram graphical notation, we have developed a Microsoft Visio stencil that contains a set of shapes for creating such diagrams (see Figure 192). By means of stencils, Visio offers the functionality for defining the concrete and abstract syntaxes of domain-specific languages [Kelly and Tolvanen 2008].

This stencil is an evolution of a previous one; the shapes have been redesigned to increase usability and facilitate modelling. Among other features, some shapes have been defined behaviour, configuration parameters, etc. For instance, Figure 193 shows that a logical node displays a different symbol depending on whether it is defined to have *or* semantics or *and* semantics.

General purpose diagramming tools such as Visio do not necessarily enforce the notation restrictions. This is both a strength and a weakness. On the one hand, this is an advantage during model construction. The analyst can freely arrange model elements disregarding whether modelling guidelines are being violated (even the most basic grammar rules). According to our experience, this allows for quick drafts that foster creativity and provide short feedback loops with the stakeholders. Also, it allows fine tuning of models for presentation and communication purposes (e.g. font types and properties), as well as applying ad-hoc secondary notation guidelines (e.g. highlighting a path in the business process model by means of changing the line width or the background colour of certain model elements. In fact, most of the models included in this thesis have been created using Microsoft Visio (e.g. the communicative event diagrams have been created using the above-mentioned stencil).

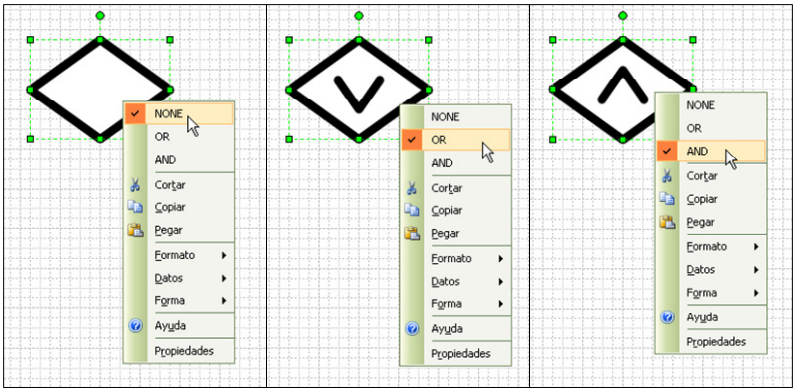


Figure 193. The logical node displays a different symbol depending on the selected option

On the other hand, the lack of rule enforcement turns out to be a strong disadvantage during requirements analysis. The responsibility of model verification falls on the analyst, which can be an unmanageable task if the model is complex.

With regards to the model designer, although it is possible to develop full-featured add-ons for Microsoft Visio (e.g. ViFlow[114], SemTalk[115], TrisoTech[116], ITpearls[117]), the framework is not especially suited for the purpose of creating model-driven development tools. All the code must be programmed ad hoc and there is no open-source or academic community around to facilitate developments.

8.2.2 Implementation of a modelling tool

If we really intend to integrate Communication Analysis into a model-driven development framework, Communication Analysis models should be operable and, preferably, created as an instantiation of a metamodel. Among the existing frameworks that allow developing a model-driven development tools; some focus on domain-specific modelling languages (e.g. MetaEdit+[118]) and others allow for general-purpose modelling languages (e.g. Eclipse Modeling Framework[119]). We selected Eclipse Modeling Framework (EMF) because of its versatility but mainly for practical reasons: our research centre is involved in the development of an Eclipse-based CASE tool named MOSKitt and we managed to get funds to hire a master student as a developer.

8.2.2.1 An EMF-based tool to support (part of) Communication Analysis

Selection of the platform

We have chosen the Eclipse framework to start developing a CASE tool intended to support the creation and management of Communication Analysis requirements models, and to serve as a departure point for the integration with the OO-Method. There exist several tools based on Eclipse; some are CASE tools

[114] http://www.vifow.com

[115] http://www.semtalk.com

[116] http://www.businessprocessincubator.com/modeler

[117] http://www.itpearls.com

[118] http://www.metacase.com

[119] http://www.eclipse.org/modeling/emf

(e.g. MOSKitt[120]) and some are CASE tool development environments (e.g. Eclipse Modeling Framework[121], Epsilon[122], openArchitectureWare[123]).

Modeling Software KIT (MOSKitt) is a free CASE tool, which is being developed by the Conselleria de Infraestructuras, Territorio y Medio Ambiente (CIT), an organisation of the Valencian Government (Spain). MOSKitt was developed with the aim to support the CIT adaptation of the Spanish software projects management standard Métrica Versión 3 [Ministerio de Administraciones Públicas 2000]; however, its plug-in architecture makes it suitable for developing extensions for different purposes related to modelling. Therefore, we have chosen to create an extension of MOSKitt, using the Eclipse Modeling Framework (EMF) [Steinberg, Budinsky et al. 2008].

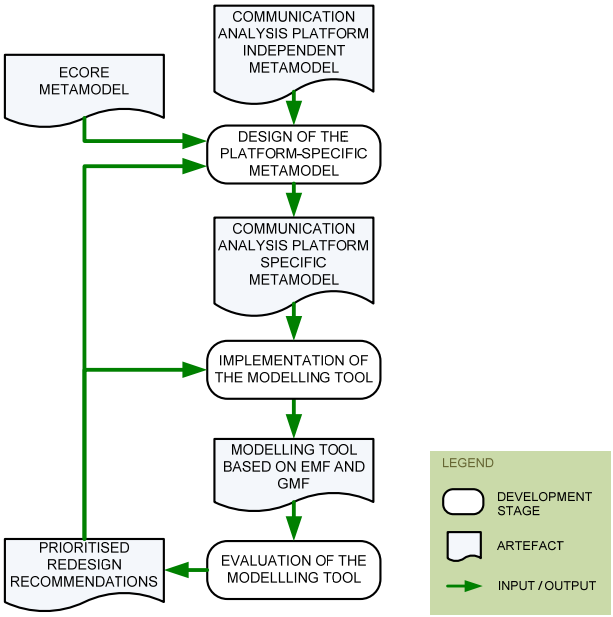


Figure 194. Stages of the modelling tool development

[120] http://www.moskitt.org

[121] http://www.eclipse.org/modeling/emf

[122] http://www.eclipse.org/gmt/epsilon

[123] http://www.eclipse.org/workinggroups/oaw

A simplified view of the stages followed during the modelling tool development is shown in Figure 194. First, a Communication Analysis platform specific metamodel has been designed to adapt the platform independent metamodel (see Section 4.5) to the particularities of the EMF; consequently, the metamodel has been expressed using the Ecore metamodel. Then the modelling tool has been developed, observing EMF common practices; the graphical notation layer has been implemented by means of the Eclipse Graphical Modeling Framework (GMF). The development has been iterative and incremental, undergoing several evaluations that provided feedback to improve the modelling tool.

Design of the platform specific metamodel

The platform specific metamodel contains most of the metaclasses that belong to the platform independent metamodel but also includes some which are platform-oriented; that is, some metaclasses that are not ontologically founded but that are necessary to develop the modelling tool in EMF or to add features and improve qualities related to computerisation (i.e. qualities such as usability, performance). This metamodel is shown in Figure 195.

Note that some design decisions have been made. For instance, the textual requirements taxonomy has been replaced by a metamodel of such. The partial views of the Communication Analysis metamodel corresponding to levels L3 (Figure 51) and L4 (Figure 53) proposed a taxonomy expressed in terms of a specialisation hierarchy. This has now been replaced by an enumeration named REQUIREMENT_TYPE. TEXTUAL_REQUIREMENT now applies to Element, instead of applying to COMMUNICATIVE_EVENT. This solution provides more flexibility with regards to the taxonomy and more expressiveness. In any case, for the purpose of automating the model transformations later on, we rather have the necessary information in an operable way, instead of textually specified (it is out of the scope of this thesis to automatically interpret and process textual requirement).

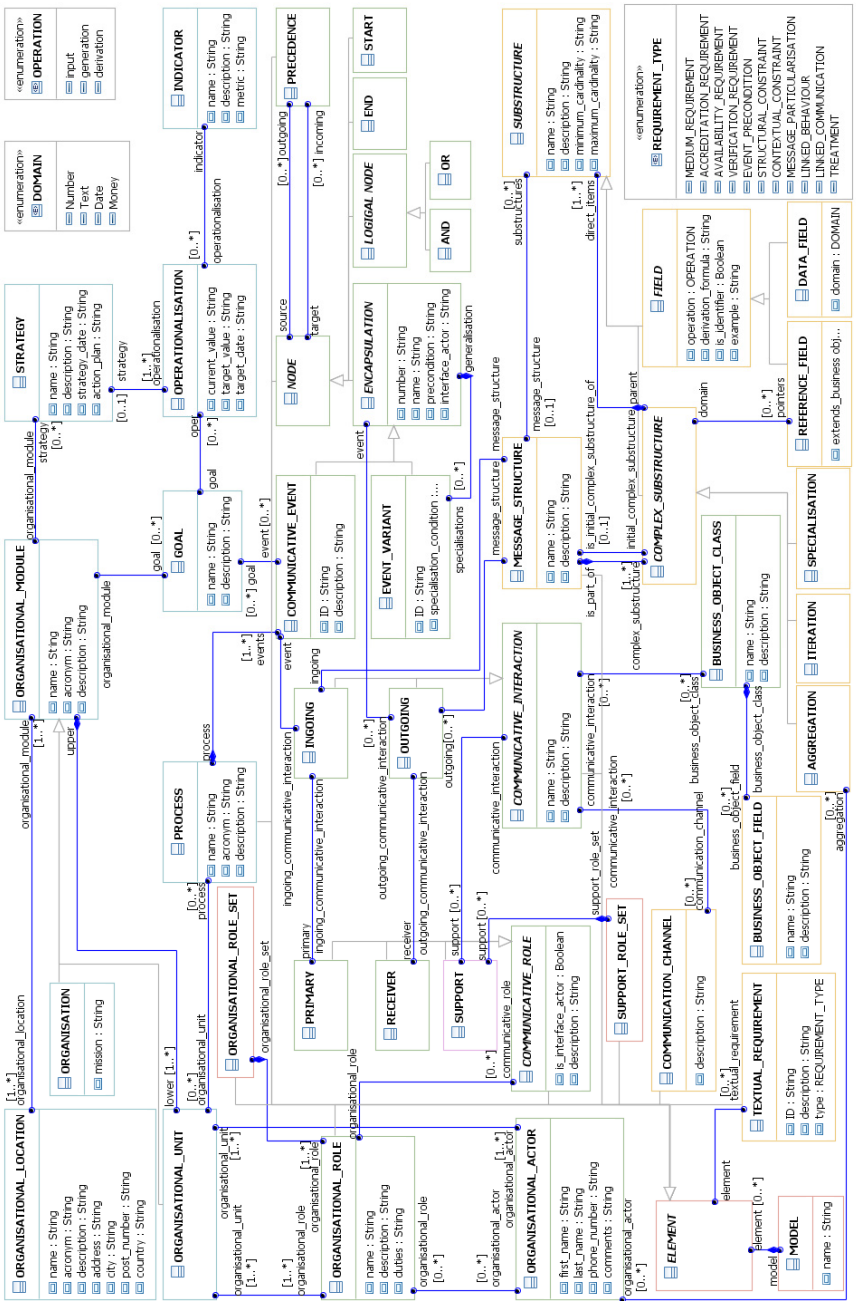


Figure 195. Communication Analysis EMF platform-specific metamodel

The metaclasses of the platform specific metamodel are specified in detail in [Ruiz 2011], following the common practices of metamodel specifications (as done in, e.g., [OMG 2010b] and [Gonzalez-Perez and Hug 2011]). The specification template includes a short description explaining meaning of the metaclass, one or more examples of instances, the metamodel view that concerns the metaclass (i.e. the metaclass being specified and its neighbouring metaclasses), a table describing the metaclass attributes, a table describing the relationships with other metaclasses, the constraints affecting the metaclass, and a description of a proposed graphical primitive. We provide an example in the following, adapted from [Ruiz 2011].

COMPLEX_SUBSTRUCTURE Metaclass

Definition

This metaclass is part of the support of the Message Structures modelling technique, which allows abstractly specifying the message associated to a communicative interaction (see Section 4.3.2). A message structure is said to be composed of substructures (i.e. its components), which can be fields (informally, elementary pieces of data) or complex substructures (informally, components that have further composition).

A complex substructure is a substructure (see SUBSTRUCTURE) that is composed of one or several substructures. This composition can be of different types: an aggregation, an iteration or a specialisation.

Examples

- *Aggregation substructure*. It specifies a composition of several substructures in a way that they are grouped as a whole. It is represented by angle brackets < >. For instance, LINE=<Product+Price+Quantity> specifies that an order line consists of information about a product, its price and the quantity that the client requests.
- *Iteration substructure*. It specifies a set or repetition of the substructures it contains. It is represented by curly brackets { }. For instance, LINES is an iteration substructure that represents a set of order lines: LINES={LINE=<Product+Price+Quantity>}.
- *Iteration substructure*. It specifies one or more variants (i.e. structural alternatives). For instance MSG=<a+b+OPTION=[c|d]+e> specifies that both <a+b+c+e> and <a+b+d+e> are valid messages.

Metamodel view for COMPLEX_SUBSTRUCTURE

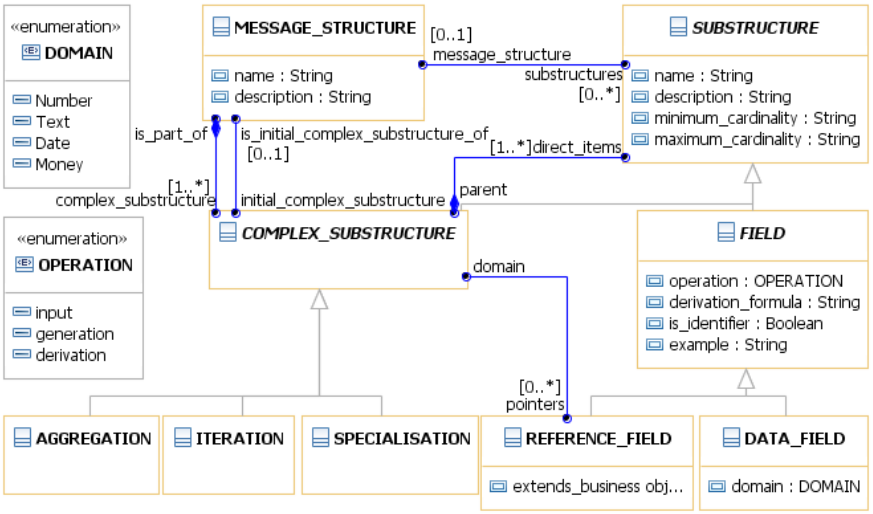


Figure 196. COMPLEX_SUBSTRUCTURE platform-specific metamodel view

Attributes

Name	Data type	Description
name	String	The name of the complex substructure. For instance, LINE. [Inherited from SUBSTRUCTURE]
description	String	A natural language explanation of the complex substructure. [Inherited from SUBSTRUCTURE]
minimum_cardinality	String	The minimum cardinality of the complex substructure. [Inherited from SUBSTRUCTURE]
maximum_cardinality	String	The maximum cardinality of the complex substructure. [Inherited from SUBSTRUCTURE]

Relationships

Role	Type	Target participant	Description
n/a	Inheritance	SUBSTRUCTURE	COMPLEX_SUBSTRUCTURE is a specialisation of SUBSTRUCTURE.
is_initial_complex_substructure_of	Association	MESSAGE_STRUCTURE	In case the complex substructure is an initial substructure, it indicates the message substructure. Otherwise, no link exists.
is_part_of	Composition	MESSAGE_STRUCTURE	It indicates which message

			structure contains the complex substructure. It is redundant with the role mess_struc.
mess_struc	Association	MESSAGE_STRUCTURE	The message structure to which the complex substructure belongs. [Inherited from SUBSTRUCTURE]
direct_items	Composition (components role)	SUBSTRUCTURE	The substructures that compose the complex substructure (only direct components, disregarding subsequent nested levels).
parent	Composition (compound role)	COMPLEX_ SUBSTRUCTURE	The complex substructure that contains the complex substructure (if any). [Inherited from SUBSTRUCTURE]
pointers	Association	REFERENCE_FIELD	The reference fields that refer (or point) to the complex substructure.

Constraints

#	Constraint description	Enforcement level
Imposed by the method		
C1	If is_part_of exists (i.e. the complex substructure is an initial substructure) and is_initial_complex_substructure_of exists (i.e. the link has been created), then both is_initial_complex_substructure_of and is_part_of point to the same instance of MESSAGE_STRUCTURE.	Prescribe
C2	The minimum cardinality of a complex substructure should be 1.	Recommend
Imposed by the transformation rules[124]		
C3	The complex substructure must be specialised in ITERATION or AGGREGATION, not in SPECIALISATION.	Prescribe

Graphical primitive

This metaclass has no graphical primitive of its own (graphical in the sense of visual but not textual). It has indeed a concrete syntax of textual nature, which has been described above.

[124] These constraints are not imposed by the Message Structures modelling technique, but by the limitations of the transformation rules in their current state. Thus, they should only be enforced in case the derivation of the OO-Method object model from the Communication Analysis requirements model is intended; even in that case, the constraints are temporary (until we improve the transformation rules).

Implementation of the modelling tool

To implement the modelling tool we have used the following complementary frameworks:

- Eclipse Modeling Framework, which concerns the definition of metamodels (a.k.a. abstract syntax) and the creation of tree-view editors.
- Eclipse Graphical Modeling Framework, which concerns the definition of the notation (a.k.a. concrete syntax) and the creation of graphical editors

The result is a graphical editor that allows creating communicative event diagrams (see Figure 197), as well as specifying other model elements (e.g. organisational roles, textual requirements). It also allows specifying message structures by means of the tree-view editor, which is not very usable and has led to the implementation of another tool (see Section 8.2.2.2).

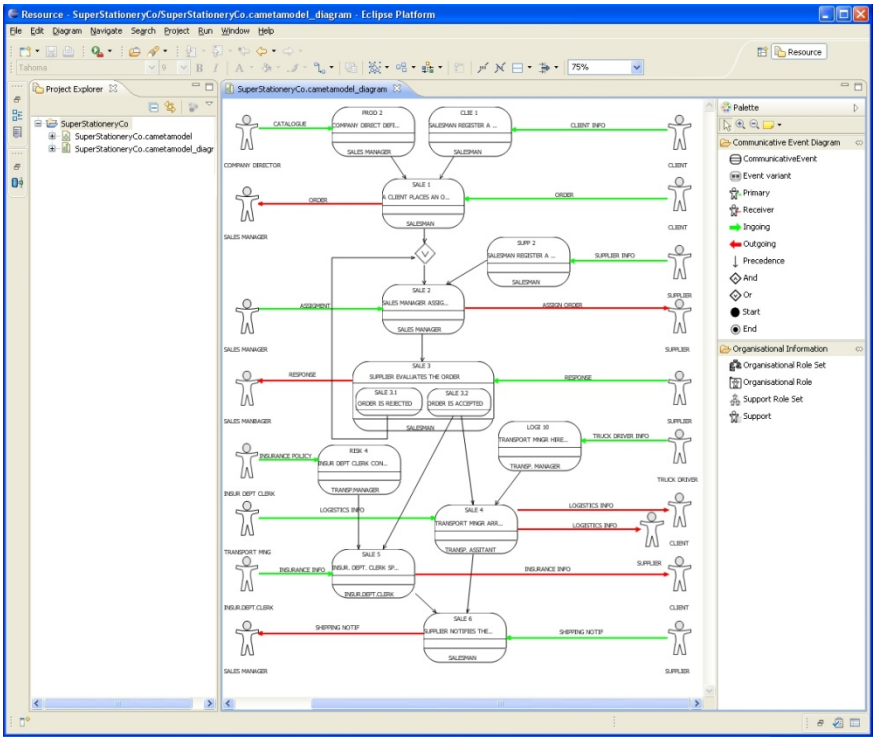


Figure 197. Sales management business process modelled in the graphical editor

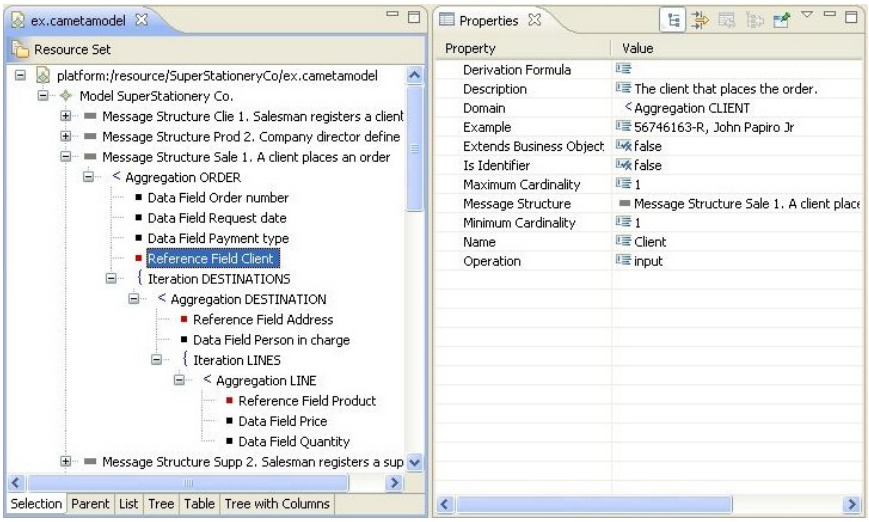


Figure 198. Example of message structure supported by the EMF environment

Figure 198 shows specification of the message structure ORDER (see Table 10) as an instantiation of the metamodel, in the form of an Ecore tree. The tree graphically represents the composition of complex substructures, leaving the operators = and + implicit. Note that each field has properties with representative names; for instance, the reference fielf Client has as domain a CLIENT, a Maximum Cardinality 1, and the acquisition Operation input. For the sake of simplicity, the marks for model transformation have been implemented as properties (e.g. Extends Business Object and Is Identifier), rather that as separate metaclases.

Evaluation of the modelling tool

The modelling tool was developed following an iterative and incremental lifecycle. The preliminary validations carried on during the first iterations were mainly focused on the expressiveness of the metamodel. Additionally, in one occasion, an additional evaluation was performed: a heuristic usability evaluation. With regards to the assessment of the expressiveness, a report with the observed limitations was issued and fed back to the design of the platform-specific metamodel. With regards to the heuristic evaluation, the classification of usability problems (CUP) scheme was used to record the encountered problems [Hvannberg and Law 2003]. As recommended in [Vilbergsdóttir, Hvannberg et al. 2006], the scheme was adapted to suit our needs; in particular, we allowed for the categorisation of the encountered usability problems according to the ten principles proposed by Nielsen [Nielsen 1994].

Additionally, a lab demo was performed. The SuperStationery case was modelled using the tool, as shown in Figure 197 and Figure 198. This also

provided valuable feedback concerning both the expressiveness of the platform-independent metamodel and the usability of the tool.

When the tool had undergone several iterations, a user testing was carried out. The intention was to put the tool to test with users not involved in the design of the method or in the development of the tool. The selected subjects were 23 master students in their final year, attending an optional course on human-computer interaction by Professor Jean Vanderdonckt. The user testing included a demographic questionnaire, a demonstration of the tool, some time for the subject to freely explore the application, the modelling task, a CSUQ usability questionnaire [Lewis 1995], and a focus group intended to elicit the users' impressions on the tool.

The quantitative data showed that most of the users (20 out of 23) were able to successfully complete the task. All in all, the mean completion was 97.61% (standard deviation 8.51%). The CSUQ index was 4.57 (it is an aggregated value averaging the responses of the users in a 1 to 7 Likert scale, with 7 as the highest perceived usability), which is an acceptable value for a prototype.

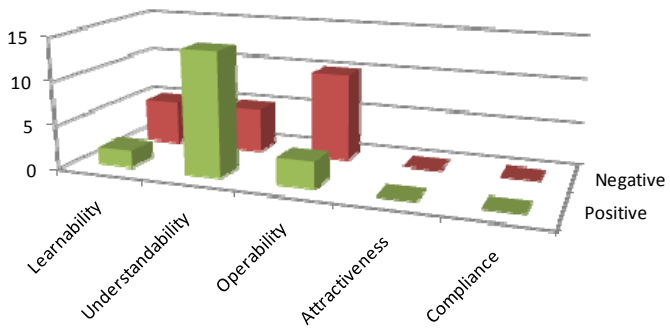


Figure 199. Amounts of positive and negative comments on the tool

Furthermore, the qualitative data confirms that the overall user experience was positive. Qualitative data came from different sources (open questions added to the usability questionnaire, focus group proceedings). This data was analysed with two purposes. First, prioritised list of usability problems and design recommendations (by the users) was compiled. Second, the responses of the users to the open questions were categorised according to the taxonomy of usability attributes in [España, Pederiva et al. 2006] (also presented in [Abrahão and Insfran 2006]). As shown in Figure 199, the majority of positive comments address the understandability of the tool. The majority of negative comments concern the operability. Provided that we keep the understandability high, the improvement of the existing features and the addition of new ones, will probably have a positive impact on the user experience.

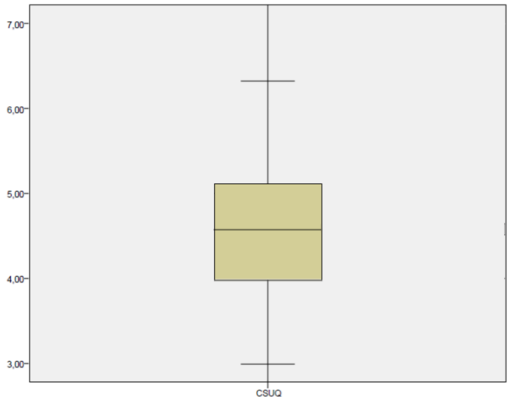


Figure 200. Boxplot of the usability indicator CSUQ

For a detailed report on the evaluations and their results, see [Ruiz 2011].

8.2.2.2 An Xtext-based tool for Message Structures

Our experience with the modelling tool revealed that the support for Message Structures could be improved. Creating the message structures with the tree-view editor of EMF is a cumbersome task. Fortunately, the modelling technique is based on structured text and can be specified using the extended Backus-Naur Form notation (see Table 19). This characteristic facilitates the development of a domain-specific language (DSL) tool based on Xtext[125], an Eclipse-based environment for the development of textual DSLs [Effting and Völter 2006; Behrens, Clay et al. 2010]. Table 85 shows the Message Structure grammar as defined in Xtext. This environment allows the automatic generation of textual editors for the defined DSLs.

Figure 201 shows the specification of the message structure ORDER using the Xtext tool. An advantage of this environment is that it can be integrated with other Eclipse-based modelling tools.

[125] http://www.eclipse.org/Xtext

Table 85. Domain-specific language definition in Xtext for Message Structures

```
grammar org.xtext.example.mydsl.CAMS with
org.eclipse.xtext.common.Terminals
generate cAMS "http://www.xtext.org/example/mydsl/CAMS"
MessageStruc:
strucName +=StrucName
(initialSubstruc +=InitialSubstruc);
StrucName:
strucName=ID '=';
InitialSubstr:
(aggregationSubstruc +=AggregationSubstruc) |
(iterationSubstruc +=IterationSubstruc);
AggregationSubstruc:
'<'(substrucList +=SubstrucList)'>';
IterationSubstruc:
'{'(substrucList +=SubstrucList)'}';
SpecialisationSubstruc:
'['(substrucList +=SubstrucList)
('|' (substrucList +=SubstrucList))']';
SubstrucList:
(substruc+=Substruc) ('+'(substruc+=Substruc))*;
Substruc:
(field +=Field) |
substrucName=ID'='(complexSubstruc+=ComplexSubstruc);
Field:
fieldName=ID;
ComplexSubstruc:
(aggregationSubstruc+=AggregationSubstruc)|
(iterationSubstruc+=IterationSubstruc)|
(specialisationSubstruc+=SpecialisationSubstruc);
```

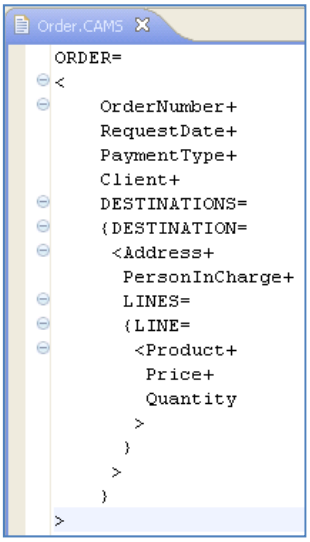


Figure 201. Example of message structure supported by the Xtext environment

8.2.3 Discussion

With regards to offering two alternative supports (a general-purpose diagramming tool and a modelling tool), we claim that it is convenient. At the end of the day, it is the organisation that will decide whether to enter the model-driven development paradigm or not. In any case, we provide support for both alternatives. There even exists the possibility of achieving interoperability between the Visio-based and the Eclipse-based solutions by means of meta-metamodel based transformations [Kern and Kühne 2009]. So, in the future, we could attempt to bridge the gap between both tools.

With regards to the modelling tool, it has many current limitations, but it has demonstrated that the approach is also technologically feasible. We have even provided two alternatives for modelling message structures. On the one hand, the implementation in Xtext ensures the compliance with the EBNF grammar for Message Structures and it offers an editorial environment that is more efficient and usable. On the other hand, the implementation in EMF extends the CASE tool for Communication Analysis; moreover, its Ecore metamodel offers the possibility of defining model to model transformations using languages such as ATL Transformation Language (ATL [Jouault and Kurtev 2005]) or Query/View/Transformation (QVT [OMG 2008]). In any case, both implementation approaches are complementary, since both environments (Xtext and EMF) can be integrated (this is planned as future work).

8.3 *Support for model transformation*

8.3.1 Implementation of an ATL transformation

We have chosen Atlas Transformation Language (ATL) to implement the transformation rules that support the conceptual model derivation technique presented in Chapter 5. ATL is a hybrid model transformation language developed as a part of the ATLAS Model Management Architecture [Jouault and Kurtev 2005]. It allows for declarative transformation rules based on pattern matching and for imperative transformation rules that can be explicitly invoked.

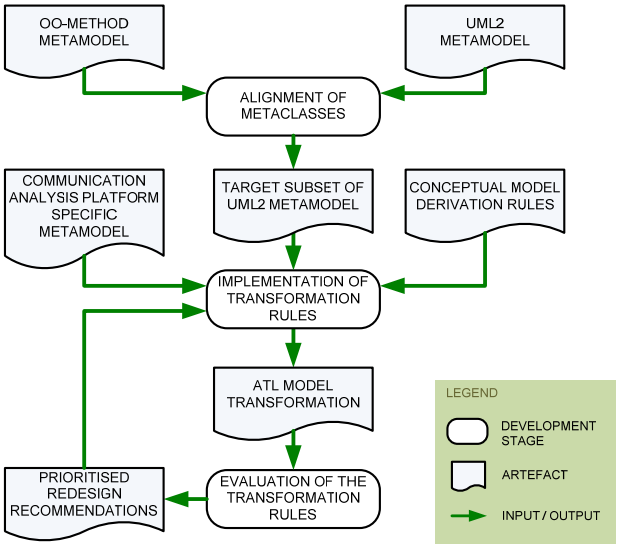


Figure 202. Stages of the ATL transformation development

A simplified view of the stages followed during the modelling tool development is shown in Figure 202. The inputs for the implementation were the derivation rules, the Communication Analysis platform-specific metamodel and the UML 2.0 Class Diagram metamodel. Both metamodels are expressed using Ecore. We opted for targeting the UML 2.0 Class Diagram for practical reasons: Eclipse EMF in general and MOSKitt in particular offer good support for creating and visualising UML 2.0 class diagrams. The OO-Method Object Model extends the UML 2.0 Class Diagram with several metaclasses and properties, but both metamodels share the same essential metaclasses. By selecting the UML 2.0 Class Diagram metamodel as target metamodel we needed to restrict ourselves to a

subset of the OO-Method modelling primitives (those that lie in the intersection of both metamodels).

This limitation can be overcome by defining a profile for UML 2.0 Class Diagrams that account for the particularities of the OO-Method Object Model, and extending the transformation rules to address the profile. The interoperability between UML profiles and the OO-Method has been solved by Giachetti et al. [Giachetti, Albert et al. 2010].

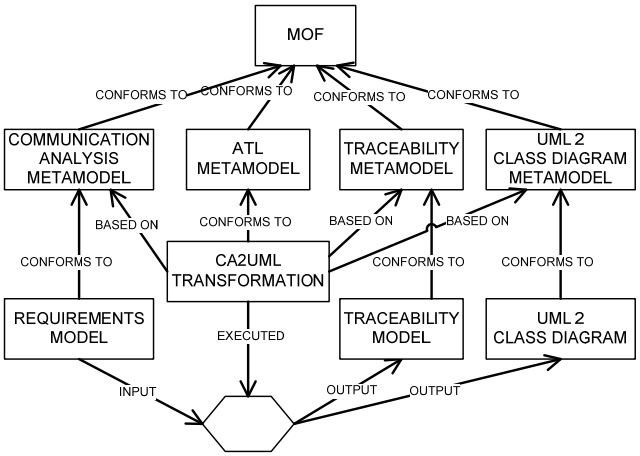


Figure 203. Operational context of the implemented transformation

With regards to the transformation rules, given that the communicative events need to be processed in order, we have opted for an imperative form (in any case, we plan to investigate the declarative form for the same transformation rules and compare their performance, as future work). Figure 203 shows the operational context of the transformation. Note that a traceability model is also produced as output of the transformation. The traceability metamodel is based on the work of Jouault [2005] on loosely coupled traceability. The traces are related to model elements in a very simple manner. The metaclass ELEMENT is a specialisation of the Ecore class EOBJECT and the metaclasses intended to be traced are specialisations of EOBJECT as well.

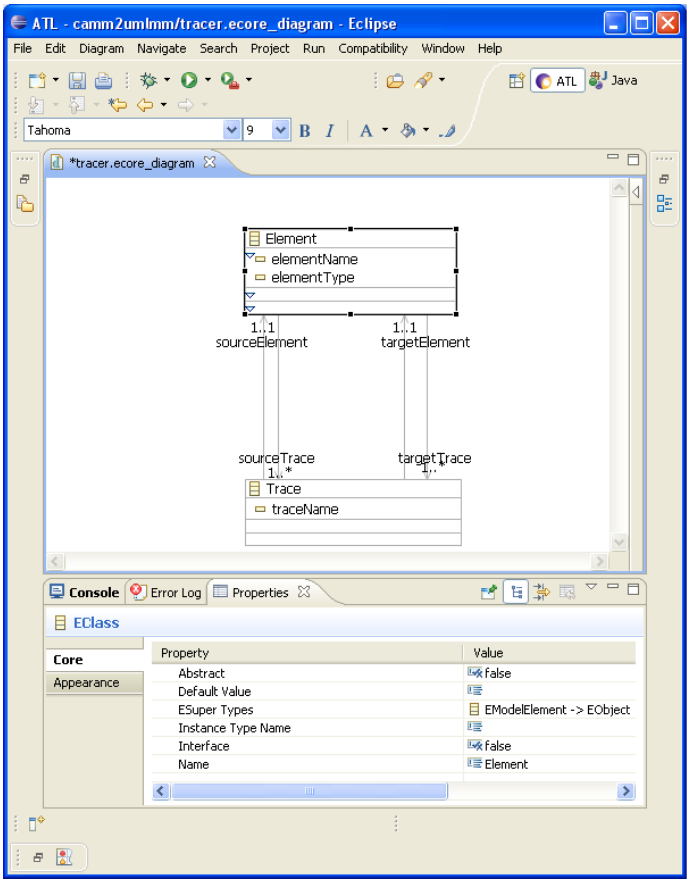


Figure 204. Traceability metamodel

In the following, we present a small sample of the transformation rules.

```
module ca2uml;
create OUT: uml, trace: tracer from IN: cametamodel;

rule process_create_class(active_substructure: cametamodel!Substructure){
   to
      class: uml!Class(name<-active_substructure.name),
      elementSrc: tracer!Element(elementName<-active_substructure.name),
      elementTrg: tracer!Element(elementName<-class.name,
         targetTrace<-Sequence{elementTrg}),
      trace: tracer!Trace(traceName<-'Create Class',
         targetElement <- elementTrg)
```

```
    do {
        --Define the newly-created class as the active_class
        self.active_class <- class;
        --Traceability
        elementSrc.refSetValue('sourceTrace', Sequence{elementSrc});
        trace.refSetValue('sourceElement', elementSrc);

    self.subs_classes_set.add(self.set_tuple_sub2class(active_substructure,
            self.active_class));
        --Add the new class to the Model
        self.conceptual_model.packagedElement.add(self.active_class);
    }
}

rule process_create_atrribute(active_item: cametamodel!DataField){
    to
        property: uml!Property(name <- active_item.name),
        elementSrc: tracer!Element(elementName<- active_item.name),
        elementTrg: tracer!Element(elementName<- active_item.name,
            targetTrace<-Sequence{elementTrg}),
        trace: tracer!Trace(traceName<- 'Create Attribute',
            targetElement<- elementTrg)
    do{
        --Define the newly-created attribute as the active_ attribute
        self.active_attribute <- property;
        --Body
        self.active_class.ownedAttribute.add(self.active_attribute);
        self.attribute_count <- self.attribute_count + 1;
        if(active_item.domain = 'text'){
            self.active_attribute.type<-self.data_type_string;
        }else if (active_item.domain = 'number'){
            self.active_attribute.type<-self.data_type_integer;
        }else if(active_item.domain = 'boolean'){
            self.active_attribute.type<-self.data_type_boolean;
        }else if(active_item.domain = 'money'){
            self.active_attribute.type<-self.data_type_real;
        }else if(active_item.domain = 'date'){
            self.active_attribute.type<-self.data_type_date;
        }else if(active_item.domain = 'time'){
            self.active_attribute.type<-self.data_type_date;
        }
        --Traceability
        trace.refSetValue('sourceElement', elementSrc);
        elementSrc.refSetValue('sourceTrace', Sequence{elementSrc});
        self.datF_attr_set.add(self.set_tuple_datf2attr(active_item,
            self.active_attribute));
    }
}
```

As a result of applying the transformation rules to the requirements model of the SuperStationery case (see Figure 197), a UML 2.0 class diagram is obtained. Figure 205 the resulting class diagram in the default Eclipse tree view, whereas Figure 206 shows the same diagram represented graphically (using Eclipse UML2 Tools).

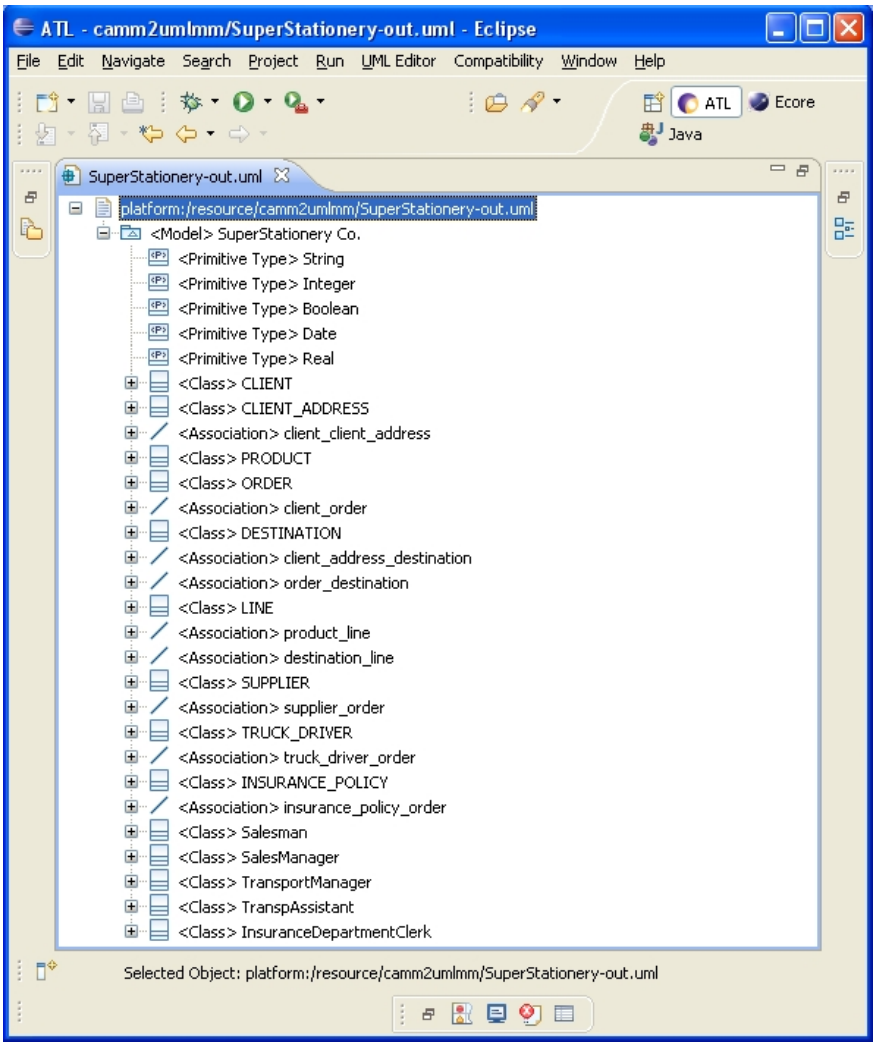


Figure 205. Tree view of the SuperStationery Co. class diagram

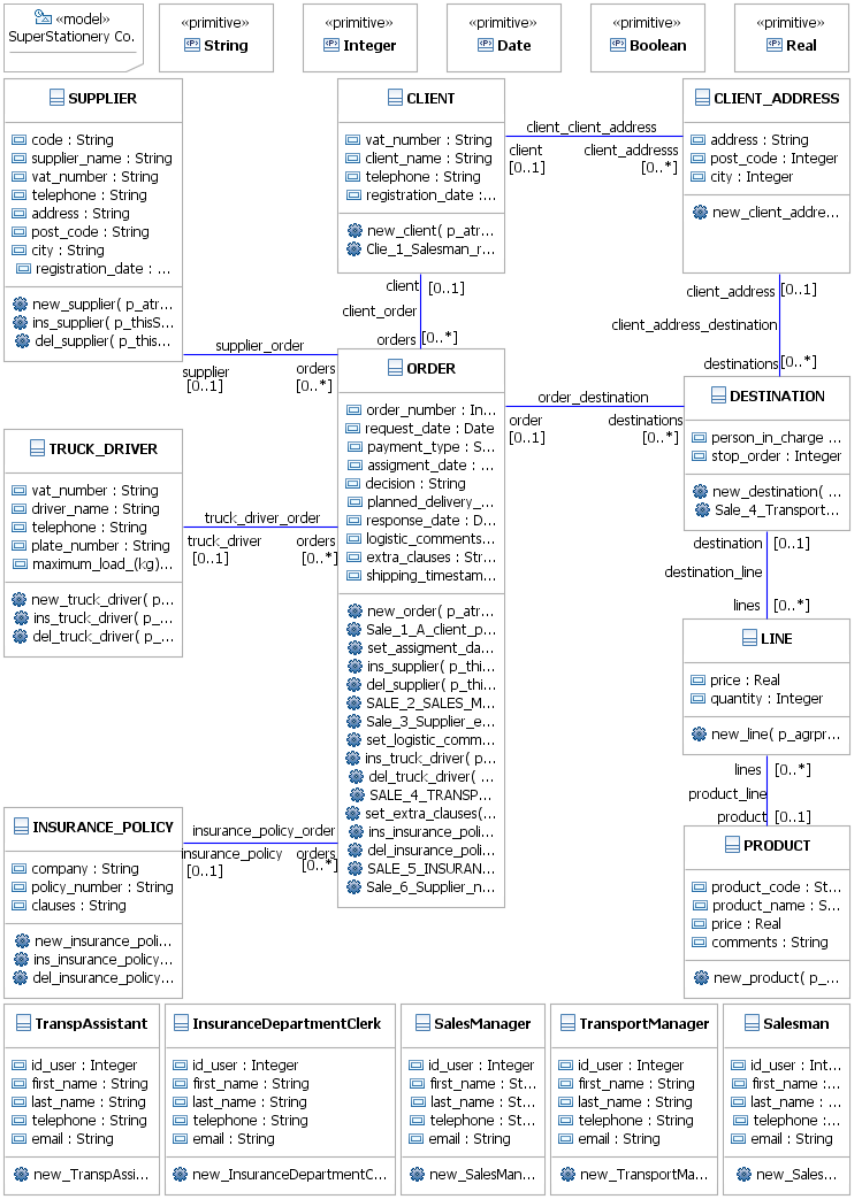


Figure 206. Graphical view of the SuperStationery Co. class diagram

Since we use the UML 2.0 Class Diagram metamodel to create the output class diagram, it lacks some of the OO-Method modelling primitives (e.g. agent relationships). We have also implemented transformation rules that create a to-do list (see . The to-do list includes a reminder of those task that the analyst still needs to do manually when creating the conceptual model using the Integranova Modeler (e.g. create agent relationships). In the future, when an appropriate OO-Method EMF metamodel is available, these tasks will easily be automated.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:todo="http:///todo.ecore">
    <todo:Element elementName="ASSIGMENT" elementType="Aggregation substructure" todoRefSource="/0" />
    <todo:Element elementName="SALE_2._SALES_MANAGER_ASSIGNS_SUPPLIER" elementType="Transaction"
      todoRefTarget="/1" />
    <todo:ToDo description="Add to the transaction formula the related services: set_assigment_date ins_supplier"
      elementSource="/0" elementTarget="/1" />
    <todo:Element elementName="LOGISTICS INFO" elementType="Aggregation substructure" todoRefSource="/3" />
    <todo:Element elementName="SALE_4._TRANSPORT_MANAGER_ARRANGES_LOGISTICS" elementType="Transaction"
      todoRefTarget="/4" />
    <todo:ToDo description="Add to the transaction formula the related services: set_logistic_comments
      ins_truck_driver" elementSource="/3" elementTarget="/4" />
    <todo:Element elementName="INSURANCE INFO" elementType="Aggregation substructure" todoRefSource="/6" />
    <todo:Element elementName="SALE_5._INSURANCE_DEPARTMENT_CLERK_SPECIFIES_CLAUSES"
      elementType="Transaction" todoRefTarget="/7" />
    <todo:ToDo description="Add to the transaction formula the related services: set_extra_clauses
      ins_insurance_policy" elementSource="/6" elementTarget="/7" />
    <todo:ToDo description="Create an agent relationship between the agent-class : Salesman and the event:
      new_client" />
```

Figure 207. Fragment of the to-do list for the SuperStationery Co. case

The trace links are stored in a separate traceability model (see an XML view of this model for the SuperStationery Co. case in Figure 208).

```
traceability-2.xml
1<?xml version="1.0" encoding="ISO-8859-1"?>
2<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:tracer="tracer">
3   <tracer:Element sourceTrace="/0" elementName="CLIENT" elementType="Aggregation substructure"/>
4   <tracer:Element targetTrace="/1" elementName="CLIENT" elementType="Class"/>
5   <tracer:Trace traceName="Create Class" targetElement="/1" sourceElement="/0"/>
```

Figure 208. Example of traceability information for SuperStationery Co. case

8.3.2 Discussion

Planas et al [2011] propose a lightweight method that performs a static analysis of ATL rules with respect to two correctness properties: weak executability and coverage.

- *Weak executability of an ATL rule* R analyzes whether a rule R may be safely applied without violating any target metamodel integrity constraints. A rule not being at least weakly executable is useless; each time the rule is executed, an error would arise because the target model violates some of its integrity constraints.

- *Covering of an ATL rule set* concerns whether a set of rules allow addressing all elements of the source and target metamodels; hence, this needs to be analysed from two viewpoints:
  - *Source coverage* refers to whether all the elements of the source metamodel (these include metaclasses, their properties and their associations) can be navigated through the execution of the set of rules.
  - *Target coverage* refers to the degree to which all the elements of the target metamodel can be created and properly initialised through the execution of the set of rules.

We are aware that the set of ATL rules presented in Section 8.3.1 neither covers all the elements of the source metamodel, nor all the elements of the target metamodel. This is due to the fact that (i) the derivation rules in Section 5.5 deliberately discarded some modelling primitives of the OO-Method and (ii) the ATL rules are an implementation of the derivation rules and, again, deliberately restrict themselves to an even smaller subset of the OO-Method metamodel. Our plan is to address the automatic transformation from Communication Analysis requirements models to OO-Method conceptual models in two stages. In this, the first stage, we focus on the subset of metamodel elements of the OO-Method Object Model that have an alignment with the metamodel elements of the UML 2.0 Class Diagram (this way the approach is more general). In the second stage, we plan to extend the ATL transformations to cover OO-Method-specific metamodel elements.

8.4 Summary

In Chapter 4, a detailed specification of Communication Analysis has been presented, including a platform-independent metamodel. Chapter 5 has presented a technique to manually derive OO-Method conceptual models from Communication Analysis requirements models. In this chapter we have addressed the implementation of a technological framework that supports both the Communication Analysis modelling techniques (see Section 8.2) and the model derivation (see Section 8.3). We offer two supports to Communication Analysis models. We provide a solution based on Microsoft Office (see Section 8.2.1): requirements models are created using Microsoft Word templates and the communicative event diagrams are created using Microsoft Visio (we provide a stencil with the necessary shapes). We also provide a more versatile modelling tool based on Eclipse (see Section 8.2.2), including a graphical editor for communicative event diagrams and two alternative means of specifying message structures. The advantage of the modelling tool is that it is based on metamodelling and, therefore, it facilitates the implementation of model transformation rules. The transformation rules have been implemented using ATL Transformation Language (see Section 8.3.1).

The implemented tools have been evaluated in different ways, ranging from expressiveness assessments to user testing. Although the technological framework cannot be considered full-fledged, it is a promising start that can be further improved. Another plausible scenario is to apply our knowledge to extend other existing CASE tools to support our techniques.

PART V.
CONCLUSION

Chapter 9
Final discussion

"Finally, in conclusion, let me say just this."

Peter Sellers

9.1 Contributions and some lessons learned

Information systems development and computerisation is a wicked problem[126]. To a large extent, this is due to their socio-technical nature [Lockemann and Mayr 1986] and to the intervention of multiple stakeholders with often conflicting needs and world views. To overcome conflicts and to reach agreement, stakeholders' perceptions have to be placed in a knowledge base that is shared with Information system developers. Requirements engineering (RE) facilitates this process by offering techniques for the discovery and description of stakeholders' needs.

Requirements engineering for enterprise information systems mainly deals with engineering business process models and requirements models. These artefacts serve several purposes. For instance, they are used to specify the needs of an organisation with regards to communication and information processing. This way, they serve as input for later software development activities such as conceptual modelling and information system design. Since the business strategy and the information technology are expected to be aligned [Henderson and Venkatraman 1999], data models and business logic must be designed so that they ensure the information system support to business processes.

[126] Among other characteristics, wicked problems do not have a unique solution and their statement is not clear until they are solved (in part due to stakeholder discrepancies) [Rittel and Webber 1973].

In this thesis we explore how to take advantage of an emerging software development paradigm to leverage requirements engineering practices: model-driven software development (MDD). MDD is a very active area of research, that slowly but surely transfers technology to industry. The advent of de facto standards facilitates the adoption of MDD methods. For instance, the Model-Driven Architecture (MDA) [OMG 2003] (as a development paradigm) and the Unified Modeling Language (UML) [OMG 2010b] (as a general-purpose notation) have facilitated the development of many MDD CASE tools. MDA distinguishes four modelling layers: the Computation Independent Model (CIM) describes a system disregarding whether it will be computerised of not, the Platform Independent Model (PIM) describes the system disregarding the specific technology that will support it, the Platform Specific Model (PSM) takes into account the target platform, and the Code Model (CM) corresponds to the final software source code.

Viable code-generation strategies start to appear but they are still *rarae aves*. For instance, the OO-Method is a UML-compliant object-oriented MDD method whose transformation strategy allows automatically generating a fully-functional software code from a conceptual model at the PIM layer [Pastor and Molina 2007]. The OO-Method conceptual model consists of four complementary models that cover the structural (Object Model), behavioural (Functional Model and Dynamic Model) and interactive (Presentation Model) aspects of the computerised information system under construction. The *OLIVANOVA* technology provides industrial support to the OO-Method, both for conceptual modelling and for multiplatform code generation (a.k.a. model compilation) [CARE Technologies].

Moreover, in the MDD area many research challenges remain open. For instance, much effort has been made lately to define transformation strategies that include the CIM layer (i.e. defining a systematic derivation of conceptual models from requirements models) [Loniewski, Insfran et al. 2010; Yue, Briand et al. 2010]. This is the gap this thesis intends to bridge.

This work is part of a long-term research effort to obtain a model-driven information systems development method that covers from business process modelling and requirements engineering to software-code generation. For this purpose, the thesis proposes the integration of two methods: Communication Analysis, a communication-oriented BPM and RE method [España, González et al. 2009], and the OO-Method (see Figure 209). Our proposal adds to previous ones (see Section 2.4), being the first attempt that covers several views of the OO-Method Conceptual Model (i.e. Object Model, Dynamic Model and Functional Model) and provides an implementation of the transformation rules.

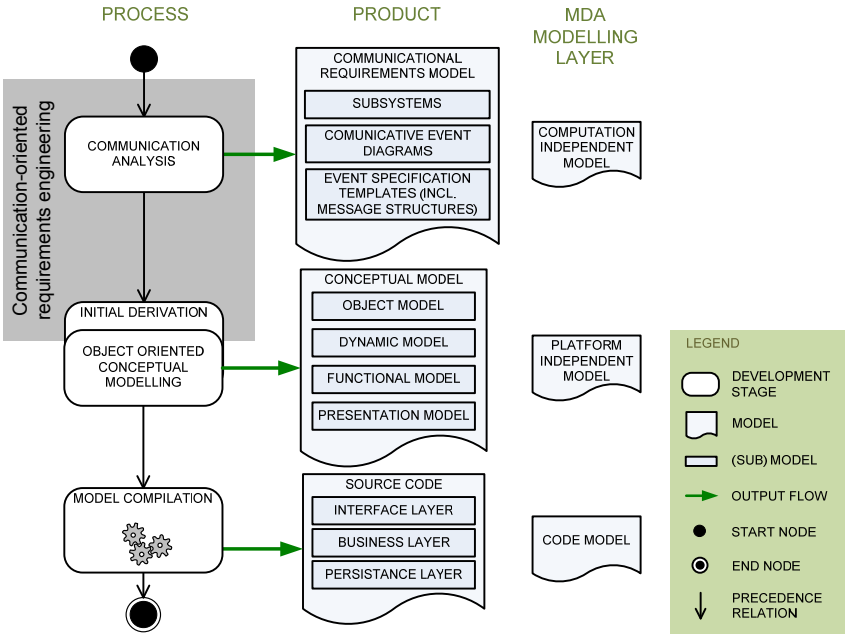


Figure 209. The proposed model-driven requirements engineering method

A communication-oriented requirements engineering approach

Attending to academic literature and industrial practice, various conceptions of information system are found. Some authors consider information systems as a mere representation of reality [Wand and Weber 1995]. Under this perspective, information systems can be described following an ontological approach; that is, focusing on the objects perceived in the universe of discourse (e.g. OO-Method). Other perspectives focus on organisational intentions (e.g. i* [Yu 1997], Maps [Rolland 2007]), value object exchanges (e.g. e3-value [Gordijn and Wieringa 2003]), etc. The authors view an information system as a support for organisational communications [Langefors 1977; Lockemann and Mayr 1986]. Therefore, a communicational approach to information systems analysis is necessary; that is, we claim that information systems requirements engineering should take into account users' communicational needs.

We propose Communication Analysis as a method for the development and computerisation of enterprise information systems. It focuses on communicative interactions that occur between the information system and its environment. The method stemmed from academic research on information systems foundations and it evolves by means of the collaboration with industry. Communication Analysis is currently being used by important Spanish enterprises and governmental institutions.

A theoretical framework: building on firm foundations

One of the most relevant challenges for information systems basic research is to construct a theoretical framework that comprises the concepts that underlie all information systems, disregarding their specific supporting technology. Ontology plays a fundamental role in this area. When designing a method intended for information systems development, it is convenient to adopt an ontology in order to rigorously ground method design decisions. We have built our research and method design upon FRISCO, a framework of information systems concepts [Falkenberg, Hesse et al. 1998] that adopts a constructivist stance by which conceptions are our means to reason about reality, and a semiotic approach by which representations of the conceptions are our means to communicate our knowledge. Moreover, we have also made the following contributions:

- **Extension of FRISCO**. We have picked up the work where the authors left it. We have redefined 1 concept (i.e. organisational system), we have made explicit 8 definitions that remained implicit in the report (e.g. non-human actor) and we have added 19 concept definitions. Some of the new definitions belong to the fundamental layers (e.g. adjacency relationship), but others account for a multiple perspective that FRISCO missed: information systems are indeed a support for organisational communication, but FRISCO did not distinguish the different semiotic layers from which the communicative interactions with information systems can be conceived. We have distinguished the following views: (i) syntactic-semantic view (e.g. interface actor, input message), (ii) pragmatics view (e.g. regulated-transition observer, regulated-transition information), and (iii) social world view.
- **Definition of a conceptual framework for model modularity**. We consider that modularity is a key issue in modelling for it deals with the level of abstraction with which models are created, allowing for different levels of detail. Step-wise refinement for instance, is a classical application of modularity to information system models; it can be applied to data-flow diagrams as well as to BPMN diagrams. We have provided 11 concept definitions (e.g. unitive criteria, module), with detailed explanations that include diagrams *à la FRISCO*.
- **Definition of conceptual framework for model-driven system development**. Since the thesis addresses the integration of Communication Analysis into a model-driven development framework, we believe that the fundamental concepts of this area sought to be investigated and clarified. We have provided 23 concept definitions (and 2 synonyms) concerning not only model driven development (e.g. modelling-language conceiving action, model transformation rule, model transformation), but also method engineering in general (e.g. method engineer, method, technique).

We acknowledge that extension to FRISCO is subject to improvement. After all, even in the FRISCO report the authors expressed their personal opinions and recognised some loose ends of their work. I share Falkenberg's impression that "the metaphor of the *semiotic ladder* is quite well-chosen. The ladder must be very high indeed: The higher you climb, the more nebulous the views get." [Falkenberg, Hesse et al. 1998, p.169].

With regards to the other two conceptual frameworks on model modularity and model-driven development, they have shed light on two important topics addressed in this thesis: the communicative event unity criteria and the integration of Communication analysis into the OO-Method, respectively. We have built these contributions upon the frameworks. Apart from that, the frameworks are to be taken as a proof of concept that demonstrates the extensibility of FRISCO to address important and highly topical research subjects.

Improving the specification of Communication Analysis

As discussed in Chapter 1, Communication Analysis was born as a result of the academic and industrial work of Arturo Gonzalez. Being close to the way FRISCO viewed information systems, he investigated their fundamentals for years, and eventually adapted available information system analysis techniques to suit his vision. At the same time, he was involved in many projects related to business process reengineering and information system development, to which he applied the method now called Communication Analysis. He spread the method among many of his collaborators. For instance, the staff of the software development department of Anecoop S.Coop. are now knowledgeable of the method and apply it in their daily work practice; some of them have even made practical contributions to the method [González, España et al. 2008b]. However, at the moment this thesis was started, we felt that the method specification was capable of improvement, especially if the integration of Communication Analysis with the OO-Method was intended.

Thus, in this thesis, we present a detailed specification of Communication Analysis. Taking as input all the available documentation about the requirements engineering method, we have created an improved documentation that defines the following aspects of the method:

- The **requirements structure**, including the concepts ascribed to each requirements level. We are aware that applying a taxonomy of requirements is often hard in practice, but analysts should at least be provided some guidance to structure their requirements models appropriately.
- The modelling techniques that can be applied to model business processes and information system requirements; for instance, **Communicative Event Diagram, Message Structures** and **Event Specification Templates**.

- The elicitation techniques that can be applied to discover requirements. We refer as **generative analysis** to a connection of techniques to gather data from the users, analyse business forms and existing process specifications. We have distinguished several types of business forms, defining how organisations use them and indicating how the analyst should analyse them.
- The requirements analysis techniques that can be applied to spot incompleteness, to discover new requirements and to design an effective support to organisational communications; we refer as **analysis of communicative behaviour** to this compendium of techniques.
- A **platform-specific metamodel** of Communication Analysis that paves the way to its integration into a model-driven development framework.

These aspects do not cover all the necessary practices within requirements engineering, but we have focused on those for which the communicational point of view had a greater impact.

Integration of Communication Analysis and the OO-Method

This thesis presents a technique to derive OO-Method conceptual models from Communication Analysis requirements models. Actually three views of the conceptual model are addressed by the derivation rules:

- **Object Model** (an extended UML Class Diagram that specifies the memory of the computerised information system). The technique offers a systematic way of tackling the derivation problem by sorting the communicative events and processing them in order (see Section 5.5.1). For each communicative event, a class diagram view is derived. This is achieved by processing the message structure that corresponds to the event (i.e. a specification of the message conveyed by the actor to the information system, in the form of structured text). Each class diagram view constitutes a portion of the class diagram, so the class diagram is incrementally constructed by integrating the views.
- **Dynamic Model** (a collection of extended UML State Machines, a.k.a. state-transition diagrams). The derivation of the Dynamic Model (which specifies e.g. the different states of a client order) is also addressed (see Section 5.5.3). The importance of the state-transition diagrams (as applied by the OO-Method) lies in the fact that they restrict the behaviour of class instances to ensure the fulfilment of the business process specifications. Traceability relationships produced during the derivation of the Object Model allow determining which communicative events affect a given class. Then this subset of events is processed to obtain the state-transition diagram of the class.
- **Functional Model** (specifications of the semantics of the class services using an abstract, platform-independent pseudocode). By deriving valuation rules and transaction formulas, we provide the semantics of the class services (see

Section 5.5.2). The technique allows processing the communicative events to support recording the information conveyed by the primary actor.

These derivations provide enough information (structural and behavioural aspects of the system) to the conceptual model to allow for automatic code generation with Integranova. The Presentation Model (which is a specification of the computerised information system interface in terms of abstract interface patterns) has been purposely left out of the scope of this thesis. Also, the derivation technique has some limitations in its current state that will be dealt with by future works.

Chapter 5 presents the derivation technique as a set of derivation rules intended to be applied manually by an analyst. Chapter 8 describes the implementation of an ATL model transformation that automates the derivation.

Validation of the proposals

The main novelty of this thesis is the definition of a derivation technique that bridges the gap between two different information system modelling orientations and abstraction layers: Communication Analysis deals with business process modelling and requirements engineering from a communicational perspective, whereas the OO-Method deals with object oriented conceptual modelling. Thus, we had initially planned validating the derivation technique. However, since Communication Analysis had never been presented to the academy before this thesis began, we also felt the need to demonstrate the qualities of the method. Several investigations were carried out to validate both proposals, trying to apply distinct kinds of validation techniques.

- **Theoretical validation**. We have applied an ontology analysis to check whether the concepts and the metamodel of Communication Analysis were well defined (see Section 6.2), as well as whether the integration of Communication Analysis and the OO-Method is theoretically sound (see Section 7.2). We have chosen FRISCO as the reference ontology for obvious reasons.
- **Lab demos**. We have reported four lab demos, in which we have applied our proposals (both the requirements modelling method and the derivation technique) to a realistic example in an artificial environment. Sections 6.3 and 7.3 describe the experience and the lessons learned.
- **Experiments**. We have compared the quality of the models produced by subjects applying Communication Analysis with that of models produced by subjects applying Use Cases (see Section 6.5. We have also compared the communication-based derivation technique with a text-based derivation technique, with regards to the completeness of the resulting models (see Section 7.5).

Our conclusion is that lab demos allow the researchers to rapidly test the feasibility of their solutions and that it is an appropriate validation technique to establish a feedback loop by which to incrementally improve the proposal. We also recommend it for communication purposes, but we suggest creating a full-detail technical report and make it available online, instead of only describing it briefly in an article (e.g. [España, González et al. 2011]). Experiments offer the chance to compare observable results of several treatments (e.g. methods) and test the statistical significance of the differences, as well as to investigate the mechanisms by which these differences appear. We provide the following recommendations for newbie experimenters (we learned some of these lessons the hard way):

- Aim for a simple experiment; start with a single well-defined research question (at most two) and state the hypotheses clearly in plain words. Do not attempt to answer many questions or you may end up overwhelmed by a complex experimental procedure.
- Define the variables (dependent and independent) and the parameters of the experiment in plain words, but also provide a metric. Sometimes indirect metrics are needed but their definition is not straightforward. However, if they are not explicitly defined, complications may appear during measurement or during data processing; moreover, it can be difficult for future readers to understand how a variable was measured (not to mention replicating the experiment).
- If you are an author of one of the compared methods, involve some researchers who have no expectancies and take their suggestions into account. This way you decrease the threat to the validity of the experiment.
- Always test the instrumentation before using it. For instance, ambiguities or defects in a questionnaire may invalidate the gathered data, so have at least some colleague proof-read it in advance. Also, with regards to texts, beware of not using the mother-tongue of your subjects or at least assess their reading (writing, hearing) comprehension.
- Plan for contingencies. Things do not always occur as planned and you need to react quickly and smartly not to threaten the validity of the experiment. For instance, if you intend to use a web-based post-task questionnaire, also have hardcopies readily available just in case the network connection fails.
- Keep a log of the experiment, especially during its operation, or you may forget why you made certain on-the-fly decisions.

A theoretical validation is a powerful technique but it is more easily applicable if the method concepts have been precisely defined. Unexpectedly, it has very practical applications; for instance, your method metamodel can end up being improved and this will have an impact of the method quality. When applied to validate the integration of Communication Analysis and the OO-Method, we

came across some difficulties due to the fact that both methods deal with a different level of abstraction. All in all, it has been a fruitful experience.

Implementation of the proposals

We have provided technological support for both Communication Analysis and the derivation technique. The creation of Communication Analysis models is supported by two alternative solutions. A set of Microsoft Word templates and a Microsoft Visio stencil are intended to provide support to companies willing to adopt Communication analysis as a standalone method or, at least, keeping their work practice based on office suites. A modelling tool based on Eclipse Modeling Framework provides a model-driven development framework that is based on an Ecore metamodel for Communication Analysis. An extra tool has been implemented in Xtext to provide a friendlier editor for Message Structures.

The derivation technique has been automated by means of a model transformation implemented in ATL Transformation Language. A design decision has led to choose UML 2 Class Diagram as a target metamodel. The practical implication is that, for the time being, only a subset of the OO-Method metaclasses is covered by the transformation. However, this limitation can be solved as described in [Giachetti, Albert et al. 2010] (it defines a systematic method to ensure the interoperability).

Lastly, Table 86 presents the application of each of the criteria defined in [Loniewski, Insfran et al. 2010] to the proposal of this thesis.

Table 86. Classification of our proposal according to the criteria by [Loniewski, Insfran et al. 2010]

Criterion		**Value**
1	Requirements type	Business
2	Requirements structure	Non-standard model
3	Type of models	Behavioural
4	Transformations provided	Yes
5	Transformations level	Exogenous
6	Standard transformations	Yes (ATL)
7	Transformations automation	Manual and automatic
8	Traceability requirements	To analysis
9	Traceability automation	Automatic
10	Tool support	Traceability & transformation
11	Type of validation	Experiments
12	Approach scope	Academic

We have proposed a model-driven development approach that deals with *business requirements* (Communication Analysis mainly deals with organisational communication and includes some aspects of organisational modelling and business process modelling). The requirements structure is a *non-standard model* (e.g. Communicative Events Diagram, Message Structures); however, we are currently adopting BPMN as a notation for business processes. With regards to the type of models, although we consider the criterion to be ill defined, we believe that the term *behavioural* appropriately describes the Communication Analysis approach: it actually focuses on the communicative behaviour of organisations. We provide *model transformations*. The transformation is *exogenous*; more specifically, it is of type *synthesis* because a higher-level, more abstract, specification is used to obtain a lower-level, more concrete, one (see [Mens, Czarnecki et al. 2005]). Also according to [Mens, Czarnecki et al. 2005], the transformation is *vertical* because the source and target models reside at different abstraction levels. We have implemented the transformations following the MDA principles and using a *standard language* (i.e. ATL). However, we provide both a set of *manual* derivation rules and the *automatic* transformation. With regards to traceability, we provide traceability from requirements models to *analysis* models in an *automatic* way. The tools support both *traceability and transformation*. We have validated our proposals by means of controlled *experiments* (as well as lab demos and ontological analyses). All in all, our approach is *academic* and has still not been applied in industrial settings.

9.2 Thesis impact

The impact of this thesis is argumented in this section on the basis of these criteria: the scientific publications that are related to the thesis (see Section 9.2.1); 2) the academic and industrial projects in which I have participated (see Section 9.2.2); 3) the collaborations with researchers of other institutions during the development of the thesis, including the research stays (see Section 9.2.3).

There are other merits and duties that are not discussed herein but are also relevant to judge the work of this thesis. I have been involved in reviewing conference (e.g. ICECCS) and journal (e.g. Journal of Web Engineering) articles (more than 30 in total). I am a member of the program committee of several workshops and conferences; namely, ONTOSE since 2009, VORTE since 2009, HWID 2009 and 2011. I have been involved in the organisation of several workshops and conferences; for instance, I am operating committee chair of RCIS 2012 and CAiSE 2013. I have taught my own research work in several seminars and courses; for instance "Model-driven requirements engineering" in 2011 (6 hour course in UPV for master students from Université Paris 1), "Model-driven

Systems Development“ in 2009 (40 hour course for bachelor students in University of Twente), “Requisitos de uso (Usage requirements)” in 2007 and 2008 (16 hour courses for industry practitioners in UPV).

9.2.1 Publications

The set of publications that are related to this thesis consists of twenty-two articles that have been accepted in three international journals, one national journal, eleven international conferences, five international workshops, one national conference and two national workshops, and of one book chapter. I have also co-directed a Master thesis. The publications (in chronological order of publication, within each category) are the following ones:

International journal articles

- España, S., N. Condori-Fernández, A. González and O. Pastor (2010). "An empirical comparative evaluation of requirements engineering methods." Journal of the Brazilian Computer Society **16**(1): 3-19.
- Pastor, O., S. España, J. Panach and N. Aquino (2008). "Model-Driven Development: piecing together the MDA jigsaw." Informatik-Spektrum. Special issue on Modelling **31**(5): 394-407.
- Panach, J., S. España, I. Pederiva and O. Pastor (2008). "Capturing interaction requirements in a model transformation technology based on MDA." Journal of Universal Computer Science. Special issue "Designing the Human-Computer Interaction: Trends and Challenges" **14**(9): 1480-1495.

National journal articles

- España, S., J. I. Panach, N. Aquino, F. Valverde and Ó. Pastor (2010). "Propuestas para la captura de requisitos y el modelado de la interacción en el marco MDA." NOVÁTICA **202**: 61-67.

International conference articles

- España, S., M. Ruiz, Ó. Pastor and A. González (2011). Systematic derivation of state machines from communication-oriented business process models. IEEE Fifth International Conference on Research Challenges in Information Science (RCIS 2011). Guadeloupe - French West Indies, France, IEEE.
- España, S., N. Condori-Fernández, A. González and Ó. Pastor (2009). Evaluating the completeness and granularity of functional requirements specifications: a controlled experiment. 17th IEEE International Requirements Engineering Conference (RE'09). Atlanta, Georgia, USA, IEEE: 161-170.
- España, S., A. González and Ó. Pastor (2009). Communication Analysis: a requirements engineering method for information systems. 21st International

Conference on Advanced Information Systems (CAiSE'09). Amsterdam, The Netherlands, Springer LNCS 5565: 530-545.

- González, A., S. España and Ó. Pastor (2009). Unity criteria for Business Process Modelling: A theoretical argumentation for a Software Engineering recurrent problem. Third International Conference on Research Challenges in Information Science (RCIS 2009). Fes, Morocco, IEEE: 173-182.
- González, A., S. España and Ó. Pastor (2008). Towards a communicational perspective for enterprise information systems modelling. IFIP WG 8.1 Working Conference on The Practice of Enterprise Modeling (PoEM 2008). J. Stirna and A. Persson. Stockholm, Sweden, Springer LNBIP 15: 62-76.
- Panach, J. I., S. España, A. Moreno and Ó. Pastor (2008). Dealing with usability in model transformation technologies. 27th International Conference on Conceptual Modeling (ER 2008). Q. Li, S. Spaccapietra, E. Yu and A. Olivé. Barcelona, Spain, Springer LNCS. 5231: 498-511.
- Pastor, O., S. España and A. González (2008). An ontological-based approach to analyze software production methods. Information Systems and e-Business Technologies. United Information Systems Conferences (UNISCON 2008). R. Kaschek, C. Kop, C. Steinberger and G. Fliedl. Klagenfurt, Austria, Springer Lecture Notes in Business Information Processing (LNBIP). 5: 258-270.
- Pederiva, I., J. Vanderdonckt, S. España, J. I. Panach and O. Pastor (2007). The beautification process in model-driven engineering of user interfaces. XI eleventh IFIP TC13 International Conference on Human-Computer Interaction (INTERACT 2007), Rio de Janeiro, Brazil, Springer LNCS.
- España, S., J. I. Panach, I. Pederiva and Ó. Pastor (2006). Towards a holistic conceptual modelling-based software development process. 25th International Conference on Conceptual Modeling (ER 2006). D. W. Embley, A. Olivé and S. Ram. Tucson, Arizona, USA, Springer LNCS 4215: 437-450.
- España, S., I. Pederiva and J. Panach (2006). Integrating model-based and task-based approaches to user interface generation. Conference on Computer-Aided Design of User Interfaces (CADUI 2006). G. Calvary, C. Pribeanu, G. Santucci and J. Vanderdonckt. Bucharest, Romania, Springer: 253-260.
- España, S., I. Pederiva, J. I. Panach, S. Abrahao and O. Pastor (2006). Linking requirements specification with interaction design and implementation. 1st IFIP TC 13.6 WG Conference: Designing for Human Work (HWID 2006). T. Clemmensen, P. Campos, R. Orngreen, A. M. Pettersen and W. Wong. Madeira, Portugal, Springer. 221: 123-133.

International workshop articles

- González, A., S. España, M. Ruiz and Ó. Pastor (2011). Systematic derivation of class diagrams from communication-oriented business process models. 12th Working Conference on Business Process Modeling, Development, and

Support (BPMDS'11). T. A. Halpin, S. Nurcan, J. Krogstieet al. London, United Kingdom, Springer LNBIP. 81: 246-260 (extended version selected for special issue in International Journal of Information System Modeling and Design, IJISMD).

- González, A., M. Ruiz, S. España and Ó. Pastor (2011). Message Structures: a modelling technique for information systems analysis and design. 14th Workshop on Requirements Engineering (WER 2011). M. Lencastre and H. Estrada. Rio de Janeiro, Brazil.
- Gailly, F., S. España, G. Poels and Ó. Pastor (2008). Integrating business domain ontologies with early requirements modelling. 2nd International Workshop on Requirements, Intentions and Goals in Conceptual Modeling (RIGiM 2008); 27th International Conference on Conceptual Modeling (ER 2008). Barcelona, Spain, Springer LNCS. 5232: 282-291.
- Panach, J. I., S. España, I. Pederiva and O. Pastor (2007). OO-Sketch: una herramienta para la captura de requisitos de interacción. X Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software (IDEAS 2007), Isla Margarita, Venezuela.
- España, S., I. Pederiva, J. I. Panach, S. Abrahao and Ó. Pastor (2006). Evaluación de la usabilidad en un entorno de arquitecturas orientadas a modelos. 9° Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software (IDEAS 2006), La Plata, Buenos Aires, Argentina.

National conference articles

- Panach, J. I., I. Pederiva, S. España and Ó. Pastor (2006). Generación automática de interfaces a partir de patrones estructurales de tareas. VII Congreso Internacional de Interacción Persona-Ordenador (INTERACCION 2006). Puertollano, Ciudad Real, Spain.

National workshop articles

- España, S., M. Ruiz, A. González and Ó. Pastor (2010). Integrando técnicas avanzadas de modelado de requisitos en MOSKitt. MOSKittDay - eclipseDay 2010. Valencia, España.
- Ruiz, M., S. España, A. Gonzalez and O. Pastor (2010). Análisis de Comunicaciones como un enfoque de requisitos para el desarrollo dirigido por modelos. VII Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM 2010), Jornadas de Ingeniería de Software y Bases de Datos (JISBD) O. Avila-García, J. Cabot, J. Muñoz, J. R. Romero and A. Vallecillo. Valencia, España: 70-77.

Book chapters

- Pastor, O., A. González and S. España (2007). Conceptual alignment of software production methods. Conceptual modelling in information systems engineering. J. Krogstie, A. L. Opdahl and S. Brinkkemper (ed.). Heidelberg, Germany, Springer: 209-228.

Master thesis

I have co-directed a Master thesis, along with Prof. Óscar Pastor. The Master thesis is closely related to the work in my PhD thesis; it deals with the operational support of Communication Analysis and the conceptual model derivation technique.

- Ruiz, M. (2011). Communication Analysis: a requirements engineering approach for model driven development. Master thesis, DSIC, Universitat Politècnica de València, Valencia, Spain.

Also, we have issued five technical reports, available online[127], that offer full detail about some of the topics in this thesis (e.g. the SuperStationery Co. case).

Table 87 shows the rating of some of the forums at which the publications of the thesis have been accepted. The rating for journals is mainly the yearly Science Citation Index (SCI), in which the Journal Citation Report (JCR) is based. The rating for conferences is based on international conference rankings that are used by the Spanish Government and universities to asses the quality of a publication; namely, the ranking by the Computing Research and Education Association of Australasia[128] (CORE), the Conference Ranking in Computer Science[129] (CSCR), and the ranking by Citeseer[130] (Citeseer).

One of the four journals is included in the JCR; namely, the Journal of Universal Computer Science.

Of the nineteen conference and workshop articles, eight of them have been accepted in forums that are included in some of the above-mentioned rankings. Three of the conferences are considered top forums (CAiSE, RE and ER), thus four of the publications can be regarded as very high quality publications. The conferences are very competitive; for instance, CAiSE 2009 had an acceptance ratio of 15% (among 230 submissions) and RE 2009 of 21% (among 170).

[127] http://www.pros.upv.es/index.php/en/informes-tecnicos Accessed 11-2011

[128] http://core.edu.au/index.php/categories/conference%20rankings Accessed 11-2011

[129] http://www.grc.upv.es/localdocs/Conference_category_CS.pdf Accessed 11-2011

[130] http://citeseerx.ist.psu.edu/stats/venues Accessed 11-2011

Table 87. Rating of some of the forums where the work has been published

Journal	SCI (2008)	SCI (2010)
Journal of Universal Computer Science	0.488	0.578

Conference	Acceptance ratio (submiss.)	CORE	CSCR	Citeseer
CAiSE	15% (230)	A	0,91	209/581
RE	21% (170)	A	0,88	448/581
ER (x2)		A		147/581
INTERACT				491/581
RCIS (x2)	36% (115)			
BPMDS		C		

9.2.2 Academic and industrial projects

The results of this thesis have been applied to some degree in the following projects. Research projects are funded by governmental institutions and aim at increasing the knowledge of a scientific field and providing/improving methods and tools to solve a given problem, whereas industrial projects are funded by companies with the aim to have some of their specific problems solved.

Research projects

- "DESTINO: Desarrollo de e-Servicios para la nueva sociedad digital". National CYCIT project referenced as TIN2004-03534.
- "Towards the Maturation of Usability Evaluation (MAUSE)". European Union COST Action 294 (http://www.cost294.org/). Years 2004-2017. Ijnternational project leaders Effie Lai-Chong Law (ETH, Zurich), Ebba Thora Hvannberg (University of Iceland, Iceland). UPV coordinator Oscar Pastor López.
- "Integración metodológica del Análisis de Comunicaciones en un marco de producción de software dirigido por modelos". National FPU Project referenced as AP2006-02323.
- "SESAMO: Construcción de Servicios Software a partir de Modelos". National CYCIT project referenced as TIN2007-62894.
- "From Business Objectives to Information Systems / De los Objetivos de Negocio a los Sistemas de Información". Integrated action Spain-Italy referenced as HI2008-0190.
- "ProsREQ - Producción de Software Orientado a Servicios basada en Requisitos: La parte Funcional". National CYCIT project referenced as TIN2010-19130-C02-02. Years 2011-2014. Project leader Óscar Pastor. It is complemented by project TIN2010-19130-C02-01 led by Xavier Franch, Universitat Politècnica de Catalunya (Spain).

- "Anytime Universal Intelligence" National EXPLORA project referenced as TIN2009-06078-E/TIN. Years 2009-2011. Project leader José Hernández Orallo.
- "M-Learning Portal for/based on Web Engineering Techniques". Socrates European Union Program. Years 2004-2007. UPV coordinator Óscar Pastor López.
- "ATENEA, Arquitectura Middleware y Herramientas". Ministerio de Industria, Turismo y Comercio project FIT-340503-2006. Years 2006-2008. Local project coordinator Vicente Pelechano Ferragud.
- "WEE-NET: Web Engineering Network of Excellence". European Unio ALPHA project. Years 2005-2007. UPV coordinator Óscar Pastor López.

Industrial projects

- "Asesoramiento e Integracion de Sistemas Informaticos de las Cooperativas B2B Colaborativo" (Consulting and integration of B2B collaborative information systems for fruit and vegetables cooperatives). Anecoop Sociedad Cooperativa. Years 2003-2007. Coordinator Arturo González del Río Rams.
- "Ingenieria del Software Avanzada para la Sociedad de la Informacion" (Advanced software engineering for information society). CARE Technologies S.A. Years 2003-2007. Coordinator Óscar Pastor López.

9.2.3 Collaborations

During the development of this thesis, we have collaborated with other researchers that share or complement our research interests, and they belong to the following institutions:

- University of Twente (The Netherlands)
- Universidade Federal de Pernambuco (Brazil)
- Università degli Studi di Trento (Italy)
- Universiteit Gent (Belgium)
- Université Catholique de Louvain (Belgium)
- Monash University (Australia)
- Universitat Politècnica de Catalunya (Spain)
- Universidad Politécnica de Madrid (Spain)
- Universidad Complutense de Madrid (Spain)

Moreover, a three-month stay in 2009 was performed in the Information Systems Group, at University of Twente (The Netherlands). A bachelor course on Model-driven system development was taught, which was used as a setting to conduct the experiment on the impact of requirements engineering on conceptual model quality (see Section 7.4).

Chapter 10
Outlook

"A work is never completed except by some accident such as weariness, satisfaction, the need to deliver, or death: for, in relation to who or what is making it, it can only be one stage in a series of inner transformations."

Paul Valéry

This thesis has introduced a model-driven requirements engineering approach for the information systems domain. The approach is the result of the integration of Communication Analysis and the OO-Method. Communication Analysis (see Chapter 4) enables analysts to elicit and model information system requirements of an organisational system, as well as to analyse and edit the requirements model in order to improve its completeness and even perform (non-extreme) business process reengineering. The communicational perspective on business process modelling has proved to be valuable if information system development is intended. By means of the derivation technique presented in Chapter 5, takes Communication Analysis requirements models as input and produces an OO-Method conceptual model as output. Several issues remain open.

First of all, we acknowledge that practitioners are usually reluctant to use non-standard notations. We therefore plan adopt the Business Process Modeling Notation (BPMN) [OMG 2011] to support Communication Analysis business process modelling. The BPMN Choreography Diagram is the first candidate to represent communicative event diagrams. Three challenges stem from this decision. First, a careful investigation needs to be carried out to adopt the notation while preserving the concepts and criteria of the method. Second, the derivation technique needs to be adapted to deal with the new notation. Third, a proper tool support needs to be provided, in order to facilitate validation, promote adoption, etc. With regards to the latter challenge, there are plenty of business process management suites in the wild these days; among them, Oryx[131]

[131] http://oryx-project.org

stands as our favourite candidate because of its open, academic nature. Modelio[132] has recently moved to open source and is therefore another candidate.

We are also aware that the specification of the Communication Analysis presented in Chapter 4 is too academic to use it for dissemination purposes in industrial environments. We actually plan to write a handbook to explain basics of the method, present the modelling techniques (probably already using BPMN as notation), and provide concise but useful guidelines to perform requirements elicitation, specification and analysis from a communicational perspective. This implies a shift in the linguistic register, as well as in the intention and the level of detail. For instance, the unity criteria for communicative events need to be reworded, a checklist or a decision tree needs to be provided to aid their application (see the RAM how-to guide in [Gorschek and Wohlin 2005]), and concise illustrative examples should be included.

We plan to investigate how other analytical perspectives (e.g. goal or value orientation) may extend our approach and become useful under certain project circumstances. In Communication Analysis, business processes are means to fulfil the business goals (see Defs. 162 and 167); however, no methodological support is provided to elicit or model goals. We indeed envision a methodological relation between goal modelling and business process modelling by which the goals serve as a rationale for process design but this is still to be investigated in depth. There exist previous proposals that intend to bridge this gap systematically [Kueng and Kawalek 1997], but this research line is still active due to its many open challenges [Guizzardi, Guizzardi et al. 2010; Cardoso, Almeida et al. 2011], especially if a communicational perspective is targeted. We also plan to investigate how the non-functional requirements (NFR) framework [Chung and do Prado Leite 2009] can be applied in conjunction with Communication Analysis; for this purpose, now that the method has been integrated into a model-driven development framework, we can build upon the work of Ameller, Franch and Cabot [2010].

Communication Analysis is currently being applied to big projects in industrial environments; e.g. the integration of Anecoop S.Coop (a Spanish major distributor of fruit and vegetables) with its associated cooperatives. We plan to describe this industrial experience by means case study reports. As mentioned in Section 4.4.2.2.1, previously-created business process models and requirements specifications are often reengineered to conform to Communication Analysis guidelines and criteria. New models are elicited, created and analysed as described in Chapter 4. However, the evolution and maintenance of requirements models still needs some improvement, so we plan to investigate how to deal with this issues (see, e.g., [Ali, Dalpiaz et al. 2011]).

---

[132] http://www.modeliosoft.com

The derivation technique covers three out of four views of the OO-Method Conceptual Model; namely, the Object Model, the Functional Model and the Dynamic Model. We plan to design and evaluate an extension to cover the Presentation Model. To do this, we plan to build upon previous work regarding interface modelling within Communication Analysis [España 2005] and the results of current research lines in out research centre [Panach, Aquino et al. 2011], such as transformation templates for concrete interface modelling within the OO-Method [Aquino, Vanderdonckt et al. 2010].

Moreover, further validations of the derivation technique are planned, including an action research application of the derivation technique in a pilot development project. This will probably be carried out in the context of GEM Biosoft[133], a spin-off of the Research Center on Software Production Methods.

[133] http://www.gembiosoft.com

REFERENCES

Abrahão, S. and E. Insfran (2006). Early usability evaluation in Model Driven Architecture environments. Sixth International Conference on Quality Software (QSIC'06). Beijing, China, IEEE**:** 287-294

Abrahão, S., E. Insfran, J. A. Carsí and M. Genero "Evaluating requirements modeling methods based on user perceptions: A family of experiments." Information Sciences **181**(16): 3356-3378

Abrahão, S., G. Poels and O. Pastor (2006). "A functional size measurement method for object-oriented conceptual schemas: design and evaluation issues." Software and Systems Modeling **5**(1): 48-71.

Achour-Salinesi, C. B., A. Opdahl and M. Rossi (2002). "REFSQ '2001 workshop summary: Seventh International Workshop on Requirements Engineering: Foundations for Software Quality." SIGSOFT Softw. Eng. Notes **27**(2): 35-49.

Achour, C. B., C. Rolland, C. Souveyet and N. A. Maiden (1999). Guiding use case authoring: results of an empirical study. 4th IEEE International Symposium on Requirements Engineering, IEEE Computer Society**:** 36-43.

Ågerfalk, P. J. and B. Fitzgerald (2006). Exploring the concept of method rationale: A conceptual tool for method tailoring. Advanced Topics in Database Research. Vol 5. K. Siau (ed.). Hershey, PA., Idea Group.

Ågerfalk, P. J. and J. Ralyté (2006). Situational requirements engineering processes: reflecting on method engineering and requirements practice. Software Process: Improvement and Practice., Wiley.

Albert, M., J. Cabot, C. Gómez and V. Pelechano (2011). "Generating operation specifications from UML class diagrams: A model transformation approach." Data & Knowledge Engineering **70**(4): 365-389.

Alférez, M., U. Kulesza, A. Sousa, J. Santos, A. Moreira, J. Araújo and V. Amaral (2008). A model-driven approach for software product lines requirements engineering. 20th International Conference on Software Engineering & Knowledge Engineering (SEKE'2008). San Francisco, USA**:** 779-784.

Ali, R., F. Dalpiaz, P. Giorgini and V. E. S. Souza (2011). Requirements evolution: from assumptions to reality. Enterprise, business-process and information

systems modeling. T. Halpin, S. Nurcan, J. Krogstieet al (ed.), Springer LNBIP. **81:** 372-382.

Ameller, D., X. Franch and J. Cabot (2010). Dealing with non-functional requirements in model-driven development 18th IEEE International Requirements Engineering Conference (RE 2010). Sydney, Australia, IEEE Computer Society**:** 189-198.

Anda, B., D. I. K. Sjøberg and M. Jørgensen (2001). Quality and understandability of Use Case models. 15th European Conference on Object-Oriented Programming (ECOOP 2001), Springer**:** 402-428.

Antón, A. I., J. H. Dempster and D. F. Siege (2001). Deriving goals from a use-case based requirements specification. Requirements Engineering Journal., Springer-Verlag. **6:** 63-73.

Aquino, N., J. Vanderdonckt and O. Pastor (2010). Transformation templates: adding flexibility to model-driven engineering of user interfaces. 25th ACM Symposium on Applied Computing (SAC 2010). S. Y. Shin, S. Ossowski, M. Schumacher, M. J. Palakal and C.-C. Hung. Sierre, Switzerland, ACM**:** 1195-1202.

Armistead, C. G. and P. Rowland (1996). Managing business processes: BPR and beyond. Chichester, John Wiley.

Artale, A., E. Franconi and N. Guarino (1996). "Part-whole relations in object-centered systems: an overview." Data & Knowledge Engineering **20**(3): 347-383.

August, J. H. (1991). Joint Application Design: the group session approach to system design. Upper Saddle River, NJ, USA, Yourdon Press.

Auramäki, E., E. Lehtinen and K. Lyytinen (1988). "A speech-act-based office modeling approach." ACM Transactions on Information Systems **6**(2): 126-152.

Austin, J. L. (1962). How to do things with words, Oxford University Press.

Ayala, C. P., C. Cares, J. P. Carvallo, G. Grau, M. Haya, G. Salazar, X. Franch, E. Mayol and C. Quer (2005). A comparative analysis of i*-based agent-oriented modeling languages. 17th International Conference on Software Engineering and Knowledge Engineering (SEKE 2005). W. C. Chu, N. Juristo and W. E. Wong**:** 43-50.

Backus, J. W., F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. v. Wijngaarden and M. Woodger (1963). "Revised report on the algorithm language ALGOL 60." Commun. ACM **6**(1): 1-17.

Baldwin, C. Y. and K. B. Clark (1999). Design rules: the power of modularity. Cambridge, MA, USA, MIT Press.

Ballmer, T. T. and W. Brennenstuhl (1981). Speech act classification: A study of the lexical analysis of English speech activity verbs. Berlin, Springer.

Baniassad, E. and S. Clarke (2004). Theme: an approach for aspect-oriented analysis and design. Proceedings of the 26th International Conference on Software Engineering (ICSE 2004), IEEE Computer Society**:** 158-167.

Bart, C. K. (1997). "Industrial firms and the power of mission." Industrial Marketing Management **26**(4): 371-383.

Basili, V. and H. Rombach (1988). "The TAME project: towards improvement-oriented software environments." IEEE Transactions on Software Engineering **14**: 758-773.

Batini, C., M. Lenzerini and S. B. Navathe (1986). "A comparative analysis of methodologies for database schema integration." ACM Comput. Surv. **18**(4): 323-364.

Behrens, H., M. Clay, S. Efftinge, M. Eysholdt, P. Friese, J. Köhnlein, K. Wannheden, S. Zarnekow and and contributors (2010). Xtext user guide (v 1.0.1) http://www.eclipse.org/Xtext/documentation/1_0_1/xtext.pdf, Accessed 2010-11.

Berenbach, B. and S. Konrad (2008). The Reinforcement Pedagogical Pattern for industrial training. Requirements Engineering Education and Training (REET '08)**:** 1-5.

Bertalanffy, L. v. (1975). Perspectives on General Systems Theory. Scientific-Philosophical Studies. E. Taschdjian (ed.). New York, George Braziller.

Beynon-Davies, P. (2007). "Informatics and the Inca." International Journal of Information Management **27**(5): 306-318.

Beynon-Davies, P. (2009). "Neolithic informatics: The nature of information." International Journal of Information Management **29**(1): 3-14.

Bhat, J. M. and N. Deshmukh (2005). Methods for modeling flexibility in business processes. Sixth Workshop on Business Process Modeling, Development, and Support (BPMDS'05). Porto, Portugal.

Blair, G., J. Gallagher, D. Hutchison and D. Sheperd (1991). Object-oriented languages, systems and applications. New York, Halsted Press.

Bloom, B. S., M. D. Engelhart, E. J. Furst, W. H. Hill and D. R. Krathwohl (1956). Taxonomy of educational objectives: The classification of educational goals. New York, McKay**:** 201-207.

Boehm, B., R. K. McClean and D. B. Ufrig (1975). "Some experience with automated aids to the design of large-scale reliable software." IEEE Trans. Softw. Eng. **1**(1): 125-133.

Booch, G. (1994). Object-oriented analysis and design with applications (2nd ed.), Benjamin-Cummings Publishing Co., Inc.

Booch, G., J. Rumbaugh and I. Jacobson (1999). The Unified Modeling Languge, Addison-Wesley.

Braun, H. v., W. Hesse, U. Andelfinger, H.-B. Kittlaus and G. Scheschonk (2000). Conceptions are social constructs: towards a solid foundation of the FRISCO

approach. Proceedings of the IFIP TC8/WG8.1 International Conference on Information System Concepts: An Integrated Discipline Emerging (ISCO 1999), Kluwer, B.V.**:** 61-73.

Bresciani, P., A. Perini, P. Giorgini, F. Giunchiglia and J. Mylopoulos (2004). "Tropos: an agent-oriented software development methodology." Autonomous Agents and Multi-Agent Systems **8**(3): 203-236.

Brinkkemper, S. (1996). "Method engineering: engineering of Information Systems development methods and tools." Information and Software Technology **38**(4): 275-280.

Brinkkemper, S., M. Saeki and F. Harmsen (1999). "Metamodelling based assembly techniques for situational method engineering." Inf. Syst. J. **24**(3): 209-228.

Brown, A., J. Conallen and D. Tropeano (2005). Introduction: models, modeling, and Model-Driven Architecture (MDA). Model-driven software development. S. Beydeda, M. Book and V. Gruhn (ed.). Berlin, Springer.

Brown, J., C. Cooper and M. Pidd (2006). "A taxing problem: the complementary use of hard and soft OR in the public sector." Eur J Oper Res **172**(2): 666-679.

Bubenko, J. A. (1995). Challenges in requirements engineering. Proceedings of the Second IEEE International Symposium on Requirements Engineering, IEEE Computer Society.

Bubenko, J. A., D. Brash and J. Stirna (1998). EKD User GuideDept. of Computer and Systems Science, Royal Institute of Technology (KTH) and Stockholm University, Stockholm, Sweden, http://people.dsv.su.se/~js/ekd_user_guide.html

Bunge, M. (1998). Philosophy of science: from explanation to justification, Transaction Publishers.

Bunt, H. C. (1994). "Context and dialogue control." Think Quarterly **3**(1): 19-31.

Bunt, H. C. (1995). Dynamic interpretation and dialogue theory. The structure of multimodal dialogue. M. Taylor, F. Néel and D. G. Bowhuis (ed.), John Benjamins. **II:** 139-188.

Cambridge (2003). Cambridge advanced learner's dictionary, Cambridge University Press.

Cardoso, E., J. P. A. Almeida, R. S. S. Guizzardi and G. Guizzardi (2011). "A method for eliciting goals for business process models based on non-functional requirements catalogues." International Journal of Information System Modeling and Design **2**(2): 1-18.

CARE Technologies - Integranova Model Execution System http://www.integranova.es/, Accessed 11-2011.

Castro, J., M. Kolp and J. Mylopoulos (2002). "Towards requirements-driven information systems engineering: the Tropos project." Information Systems **27**: 365-389.

Cockburn, A. (1997). "Structuring use cases with goals." Journal of Object-Oriented Programming(Sep-Oct, 1997 y Nov-Dic, 1997).

Cockburn, A. (2000). Writing effective use cases, Addison-Wesley Professional.

Codd, E. F. (1970). "A relational model of data for large shared data banks." Commun. ACM **13**(6): 377-387.

Codd, E. F. (1974). Recent investigations in relational data base systems. IFIP Congress 1974. Stockholm, Sweden, North-Holland**:** 1017-1021.

Collongues, A. (1986). Merise. Methode de conception. Paris, France, Dunod.

Constantine, L. L. and L. A. D. Lockwood (1999). Software for use: a practical guide to the models and methods of usage-centered design. Reading, USA, ACM Press/Addison-Wesley Publishing Co.

Constantine, L. L. and E. Yourdon (1975). Structured design, Yourdon Press.

Costal, D., A. Olivé and M.-R. Sancho (1997). Temporal features of class populations and attributes in conceptual models. 16th International Conference on Conceptual Modeling (ER '97), Springer.

Cox, K. and K. Phalp (2000). "Replicating the CREWS Use Case authoring guidelines experiment." Empirical Software Engineering **5**(3): 245-267.

Cronholm, S. and G. Goldkuhl (2004). Communication Analysis as perspective and method for requirements engineering. Requirements engineering for socio-technical systems. J. L. Mate and A. Silva (ed.), Idea Group, Inc.**:** 340-358.

Chan, S. L. and C. F. Choi (1997). "A conceptual and analytical framework for business process reengineering." International Journal of Production Economics **50**(2-3): 211-223.

Chang, M. K. and C. C. Woo (1994). "A speech-act-based negotiation protocol: design, implementation, and test use." ACM Trans. Inf. Syst. **12**(4): 360-382.

Checkland, P. (1981). Systems thinking, systems practice, Wiley.

Chonoles, M. J. and J. A. Schardt (2003). UML 2 for Dummies, John Wiley & Sons.

Chung, L. and J. do Prado Leite (2009). On non-functional requirements in software engineering. Conceptual modeling: foundations and applications: Essays in honor of John Mylopoulos. A. Borgida, V. Chaudhri, P. Giorgini and E. Yu (ed.), Springer. **5600:** 363-379.

Daniels, C. B. and W. J. LaMarsh (2007). Complexity as a cause of failure in information technology project management. IEEE International Conference on System of Systems Engineering, 2007 (SoSE '07).

Dardenne, A., A. van Lamsweerde and S. Fickas (1993). "Goal-directed requirements acquisition." Sci Comput Program **20**(1-2): 3-50.

Davenport, T. (1993). Process innovation: reengineering work through information technology. Boston, Harvard Business School Press.

Davenport, T. and J. E. Short (1990). "The new industrial engineering : information technology and business process redesign." Sloan Management Review **31**(4): 11–27.

Davis, A. M. (1990). Software requirements: analysis and specification, Prentice Hall Press.

Davis, A. M. (2003a). "The art of requirements triage." Computer **36**(3): 42-49.

Davis, A. M. (2003b). "System phenotypes." IEEE Softw. **20**(4): 54-56.

Davis, A. M., S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta and M. Theofanos (1993). Identifying and measuring quality in a software requirements specification. First International Software Metrics Symposium. Baltimore, MD, USA, IEEE**:** 141-152.

Davis, F. D., R. P. Bagozzi and P. R. Warshaw (1989). "User acceptance of computer technology: a comparison of two theoretical models." Management Science **35**(8): 982-1003.

Davis, G. B. (1982). "Strategies for information requirements determination." IBM Systems Journal **21**(1): 4-30.

de la Vara, J. L. and J. Sánchez (2007). Business process-driven requirements engineering: a goal-based approach. 8th Workshop on Business Process Modeling, Development, and Support (BPMDS'07), 19th International Conference on Advanced Information Systems Engineering (CAiSE'07), Trondheim, Norway.

de la Vara, J. L. and J. Sánchez (2009). BPMN-based specification of task descriptions: approach and lessons learnt. 15th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'09). Amsterdam, Holland, Springer LNCS.

de la Vara, J. L. and J. Sánchez (2010). System modeling from extended task descriptions. 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010). Redwood City, San Francisco Bay, USA**:** 425-429.

de la Vara, J. L., J. Sánchez and O. Pastor (2008). Business process modelling and purpose analysis for requirements analysis of information systems. 20th International Conference on Advanced Information Systems Engineering (CAiSE'08), Montpellier, France, Springer LNCS.

Debnath, N., M. C. Leonardi, M. V. Mauco, G. Montejano and D. Riesco (2008). Improving Model Driven Architecture with requirements models. Fifth International Conference on Information Technology: New Generations, IEEE Computer Society**:** 21-26.

DeMarco, T. (1979). Structured analysis and system specification, Yourdon Press.

Dennis, A. R., G. S. Hayes and R. M. Daniels (1999). "Business process modeling with group support systems." Journal of Management Information Systems **15**(4): 115-142.

Díaz, I., F. Losavio, A. Matteo and O. Pastor (2004). "A specification pattern for use cases." Information & Management **41**(8): 961-975.

Díaz, I., J. Sánchez and A. Matteo (2005). Conceptual modeling based on transformation linguistic patterns. 24th International Conference on Conceptual Modeling (ER 2005). Klagenfurt, Austria**:** 192-208.

Dieste, O., M. Lopez and F. Ramos (2008). Updating a systematic review about selection of software requirements elicitation techniques. 11th Workshop on Requirements Engineering (WER 2008) Barcelona, Spain

Dietz, J. L. G. (1999). Understanding and modelling business processes with DEMO. 18th International Conference on Conceptual Modeling (ER 1999). Paris, France, Springer-Verlag**:** 188-202.

Dietz, J. L. G., G. Goldkuhl, M. Lind and V. E. van Reijswoud (1998). The Communicative Action Paradigm for business modelling - a research agenda. 3rd International Workshop on the Language Action Perspective on Communication Modelling (LAP 1998). G. Goldkuhl, M. Lind and U. Seigerroth, Jönköping International Business School.

Dijkstra, E. W. (1974). On the role of scientific thought. E. W. Dijkstra Archive. Austin, The Center for American History, University of Texas.

Dijkstra, E. W. (1976). A discipline of programming. Englewood Cliffs, NJ, Prentice Hall.

Ding, F. and L. Jie (2008). An empirical study of flexible business process based on Modularity System Theory. Third International Multi-Conference on Computing in the Global Information Technology (ICCGI '08), Athens, Greece IEEE.

Dissanayake, K. and M. Takahashi (2006). "The construction of organizational structure: connections with autopoietic systems theory." Contemporary Management Research **2**(2): 105-116.

Dobing, B. and J. Parsons (2006). "How UML is used." Commun. ACM **49**(5): 109-113.

Duhem, P. (1954). The aim and structure of physical theory, Princeton University.

Edwards, P. (1985). Systems analysis, design, and development with structured concepts. New York, Holt, Rinehart & Winston.

Effting, S. and M. Völter (2006). oAW xText: a framework for textual DSLs. Modeling Symposium at Eclipsecon Summit Europe.

Escalona, M. J., J. J. Gutiérrez, L. Rodríguez-Catalán and A. Guevara (2009). Model-driven in reverse: the practical experience of the AQUA project. Euro American Conference on Telematics and Information Systems: New

Opportunities to increase Digital Citizenship. Prague, Czech Republic, ACM**:** 1-6.

España, S. (2005). Guía metodológica para especificación de interfaces gráficas de usuario basada en estructura de adquisición de datos, Methodological guide for the specification of graphical user interfaces based in Message Structures (in Spanish). Bachelor thesis, Facultad de Informática de Valencia, Universitat Politècnica de València, Valencia, Spain.

España, S. (2008). A generic model of Information Systems. Master thesis, Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información, Valencia, España.

España, S., N. Condori-Fernández, A. González and O. Pastor (2010). "An empirical comparative evaluation of requirements engineering methods." Journal of the Brazilian Computer Society **16**(1): 3-19.

España, S., N. Condori-Fernández, A. González and Ó. Pastor (2009). Evaluating the completeness and granularity of functional requirements specifications: a controlled experiment. 17th IEEE International Requirements Engineering Conference (RE'09). Atlanta, Georgia, USA, IEEE**:** 161-170.

España, S., N. Condori, R. Wieringa, A. González and Ó. Pastor (2011). Model-driven system development: Experimental design and report of the pilot experiment. Technical report ProS-TR-2011-12, ProS Research Centre, Universitat Politècnica de València, Spain, http://arxiv.org/abs/1111.0562

España, S., A. González and Ó. Pastor (2009). Communication Analysis: a requirements engineering method for information systems. 21st International Conference on Advanced Information Systems (CAiSE'09). Amsterdam, The Netherlands, Springer LNCS 5565**:** 530-545.

España, S., A. González, Ó. Pastor and M. Ruiz (2011). Integration of Communication Analysis and the OO-Method: Manual derivation of the conceptual model. The SuperStationery Co. lab demo. Technical report ProS-TR-2011-01, ProS Research Centre, Universitat Politècnica de València, Spain, http://arxiv.org/abs/1101.0105

España, S., I. Pederiva, J. I. Panach, S. Abrahao and Ó. Pastor (2006). Evaluación de la usabilidad en un entorno de arquitecturas orientadas a modelos. 9° Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software (IDEAS 2006), La Plata, Buenos Aires, Argentina.

España, S., M. Ruiz, Ó. Pastor and A. González (2011). Systematic derivation of state machines from communication-oriented business process models. IEEE Fifth International Conference on Research Challenges in Information Science (RCIS 2011). Guadeloupe - French West Indies, France, IEEE.

Estrada, H., A. Martínez and Ó. Pastor (2003). Goal-based business modeling oriented towards late requirements generation. 22nd International Conference

on Conceptual Modeling (ER 2003), Chicago, Illinois, USA, Lecture Notes in Computer Science 2813, Springer-Verlag.

Estrada, H., A. Martínez, Ó. Pastor and J. Mylopoulos (2006). An empirical evaluation of the i* framework in a model-based software generation environment. CAiSE 2006, Lecture Notes in Computer Science 4001, Springer Verlag.

Eveleens, J. L. and C. Verhoef (2010). "The rise and fall of the Chaos Report figures." IEEE Softw. **27**(1): 30-36.

Falkenberg, E., W. Hesse, P. Lindgreeen, B. Nilsson, J. L. H. Oei, C. Rolland, R. K. Stamper, F. VanAssche, A. Verrijn-Stuart and K. Voss (1998). FRISCO. A Framework of Information Systems Concepts (IFIP WG 8.1 Task Group Report), D:_DATOS\BIBLIOTHEQUE\Concepts\Information Systems\Falkenberg, Hesse, Lindgreen et al - A framework of information system concepts (FRISCO 1998).pdf

Fiedler, F. E. (1964). "A contingency model of leadership effectiveness." Advances in Experimental Social Psychology **1**: 149-190.

Flores, F. and J. Ludlow (1980). Doing and speaking in the office. Decision Support Systems: issues and challenges. G. Fick and R. H. Sprague (ed.). New York, USA, Pergamon Press**:** 95-118.

Fons, J., V. Pelechano, M. Albert and Ó. Pastor (2003). Development of web applications from web enhanced conceptual schemas. 22nd International Conference on Conceptual Modeling (ER 2003), Springer LNCS 2813.

Fortuna, M., C. Werner and M. Borges (2007). Um modelo integrado de requisitos com casos de uso. X Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software (IDEAS 2007). Isla Margarita, Venezuela.

Fortuna, M., C. Werner and M. Borges (2008). Info Cases: integrating use cases and domain models. 16th International Requirements Engineering Conference (RE'08). Barcelona, Spain, IEEE**:** 81-84

Fortuna, M. H. and M. R. S. Borges (2005). Modelagem informacional de requisitos. VIII Workshop em Engenharia de Requisitos (WER 2005). J. Araújo, A. D. Toro and J. F. e. Cunha. Porto, Portugal**:** 269-280.

Foucaut, O. and C. Rolland (1978). Concepts for design of an information system conceptual schema and its utilization in the remora project. 4th International Conference on Very Large Data Bases. West Berlin, Germany, VLDB Endowment.

Fox, M. S. and M. Gruninger (1998). "Enterprise modeling." AI Magazine **Fall 1998**.

Franch, X. (2010). Incorporating modules into the i* framework. 22nd International Conference in Advanced Information Systems Engineering (CAiSE 2010). B. Pernici. Hammamet, Tunisia, Springer LNCS 6051.

Galliers, R. D. (1994). "Information systems, operational research and business re-engineering." International Transactions in Operational Research **1**(2): 159-167.

Gangemi, A., D. M. Pisanelli and G. Steve (2000). Understanding systematic conceptual structures in polysemous medical terms. 2000 AMIA Fall Symposium.

Garson, D. (2008). Scales and standard measures, North Carolina State University, http://www2.chass.ncsu.edu/garson/pa765/standard.htm, Accessed 05-2011.

Génova, G., M. Valiente and M. Marrero (2009). "On the difference between analysis and design, and why it is relevant for the interpretation of models in Model Driven Engineering." Journal of Object Technology **8**(1): 107-127.

Giachetti, G., M. Albert, B. Marín and O. Pastor (2010). "Linking UML and MDD through UML profiles: a practical approach based on the UML association." Journal of Universal Computer Science **16**(17).

Giachetti, G., B. Marin, N. Condori-Fernández and J. C. Molina (2007). Updating OO-Method function points. 6th International Conference on the Quality of Information and Communications Technology (QUATIC 2007).

Glass, R. L. (2006). "The Standish Report: does it really describe a software crisis?" Commun. ACM **49**(8): 15-16.

Goedertier, S. and J. Vanthienen (2007). Declarative process modeling with business vocabulary and business rules. Workshop on Object-Role Modeling (OTM 2007 Workshops). Vilamoura, Portugal, Springer**:** 603-612.

Goldkuhl, G. (1996). Generic business frameworks and action modelling. International workshop on the Language Action Perspective on Communication Modelling (LAP 1996). Tilburg, The Netherlands.

Gonzalez-Perez, C. and C. Hug (2011). ConML - Conceptual Modelling Language technical specification, version 1.1.0The Heritage Laboratory (LaPa), Spanish National Research Council (CSIC), Galicia, Spain,

González, A. (2002). Análisis basado en sucesos. Technical Report DSIC-II/25/02Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain,

González, A. (2004). Algunas consideraciones sobre el uso de la abstracción en el análisis de los sistemas de información de gestión (PhD thesis) Some considerations on the use of abstraction in management information systems analysis (in Spanish). PhD thesis thesis, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia.

González, A. (2005). Estructuras de adquisición. Technical Report DSIC-II/08/05Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia,

González, A., S. España and Ó. Pastor (2008a). Towards a communicational perspective for enterprise information systems modelling. IFIP WG 8.1

Working Conference on The Practice of Enterprise Modeling (PoEM 2008). J. Stirna and A. Persson. Stockholm, Sweden, Springer LNBIP 15**:** 62-76.

González, A., S. España and Ó. Pastor (2009). Unity criteria for Business Process Modelling: A theoretical argumentation for a Software Engineering recurrent problem. Third International Conference on Research Challenges in Information Science (RCIS 2009). Fes, Morocco, IEEE**:** 173-182.

González, A., S. España and D. Ruiz (2008b). Análisis de requisitos basado en comunicaciones DSIC-II/01/08, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, http://www.dsic.upv.es/docs/bib-dig/informes/etd-02062008-094803/0708requisitosB0017041207.pdf

González, A., S. España, M. Ruiz and Ó. Pastor (2011). Systematic derivation of class diagrams from communication-oriented business process models. 12th Working Conference on Business Process Modeling, Development, and Support (BPMDS'11). T. A. Halpin, S. Nurcan, J. Krogstieet al. London, United Kingdom, Springer LNBIP. **81:** 246-260.

Gordijn, J. and R. J. Wieringa (2003). A value-oriented approach to e-business process design. 15th International Conference on Advanced Information Systems Engineering (CAiSE 2003). J. Eder and M. Missikoff. Klagenfurt/Velden, Austria, Springer LNCS. **2681:** 390-403.

Gorschek, T. and C. Wohlin (2005). "Requirements Abstraction Model." Requirements Engineering **11**(1): 79-101.

Gotel, O. C. Z. and C. W. Finkelstein (1994). An analysis of the requirements traceability problem. 1st International Conference on Requirements Engineering.

Grau, G., C. Cares, X. Franch and F. J. Navarrete (2006). A comparative analysis of i*agent-oriented modelling techniques. Eighteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'06). San Francisco, California, USA.

Gray, J. (1981). The transaction concept: virtues and limitations. 7th International Conference on Very Large Data Bases. Cannes, France, VLDB Endowment**:** 144-154.

Gruber, T. (1995). "Toward Principles for the Design of Ontologies Used for Knowledge Sharing." International Journal Human-Computer Studies **43**(5-6): 907-928.

Guarino, N. and C. Welty Identity and subsumption. The Semantics of relationships: an interdisciplinary perspective. R. Green, C. Bean and S. Myaeng (ed.), Kluwer.

Guizzardi, G. (2005). On a unified foundational ontology and some applications of it in business modeling. Ontologies and business systems analysis. M. Rosemann and P. Green (ed.), IDEA Publisher.

Guizzardi, G. (2006). On Ontology, ontologies, conceptualizations, modeling languages, and (meta)models. 7th International Baltic Conference on Databases and Information Systems (DB&IS'2006), IOS Press**:** 18-39.

Guizzardi, G., L. F. Pires and M. van Sinderen (2005). An ontology-based approach for evaluating the domain appropriateness and comprehensibility appropriateness of modeling languages. ACM/IEEE 8 th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2005)**:** 691-705.

Guizzardi, R. S. S., G. Guizzardi, J. P. A. Almeida and E. C. S. Cardoso (2010). Bridging the gap between goals, agents and business processes. 4th International i* Workshop. J. Castro, X. Franch, J. Mylopoulos and E. Yu. Hammamet, Tunisia, CEUR-WS.org. **586**

Gupta, D. and N. Prakash (2001). "Engineering methods from method requirements specifications." Requir. Eng. **6**(3): 135-160.

Haerder, T. and A. Reuter (1983). "Principles of transaction-oriented database recovery." ACM Comput. Surv. **15**(4): 287-317.

Halliday, M. A. K. and J. R. Martin (1993). Writing science: literacy and discursive power. London, The Falmer Press.

Hammer, M. and J. Champy (1993). Re-engineering the corporation: a manifesto for business revolution. New York, Harper Business.

Henderson-Sellers, B., C. Gonzalez-Perez and J. Ralyté (2008). Comparison of method chunks and method fragments for Situational Method Engineering. 19th Australian Conference on Software Engineering (aswec 2008). Perth, Australia 479-488.

Henderson-Sellers, B., C. González-Perez and J. Ralyté (2007). Situational Method Engineering: fragments or chunks? CAiSE Forum 2007. J. Eder, S. L. Tomassen, A. L. Opdahl and G. Sindre, CEUR-WS.org. **247:** 89-92.

Henderson, J. C. and N. Venkatraman (1999). "Strategic alignment: leveraging information technology for transforming organizations." IBM Syst. J. **38**(2-3): 472-484.

Hesse, W. and A. Verrijn-Stuart (2000). Towards a theory of Information Systems: the FRISCO approach. 10th European-Japanese Conference on Information Modeling and Knowledge Bases. Saariselkä, Finland.

Hevner, A. R., S. T. March, J. Park and S. Ram (2004). "Design science in information systems research." MIS Quarterly **28**(1): 75-105.

Höfer, A. and W. F. Tichy (2007). Status of empirical research in Software Engineering Empirical Software Engineering Issues. Critical Assessment and Future Directions, Springer LNCS 4336**:** 10-19.

Høydalsvik, G. M. and G. Sindre (1993). On the purpose of object-oriented analysis. Proceedings of the eighth annual conference on Object-oriented

programming systems, languages, and applications. Washington, D.C., United States, ACM Press: 240-255.

Hvannberg, E. T. and L.-C. Law (2003). Classification of Usability Problems (CUP) Scheme. Human-Computer Interaction INTERACT 2003, Zurich, Switzerland.

IDS Scheer (2005). ARIS Platform - System white paper, http://www.ids-scheer.com/set/6473/ARIS_Platform_SWP_en_2008-06.pdf

IEEE (1990). Standard glossary of software engineering terminology IEEE Std 610.12-1990,

Iivari, J. and P. Kerola (1983). A sociocybernetic framework for the feature analysis of information systems design methodologies. Information systems design methodologies: a feature analysis. T. W. Olle, H. G. Sol and C. J. Tully (ed.). Amsterdam, North-Holland.

Insfrán, E. (2003). A requirements engineering approach for object-oriented conceptual modeling. thesis, DSIC, Universidad Politécnica de Valencia, Valencia.

Insfrán, E. (2004). Requirements Engineering TOol (RETO) http://reto.dsic.upv.es/reto/, Accessed 02/2008.

Insfrán, E., Ó. Pastor and R. Wieringa (2002). "Requirements engineering-based conceptual modelling." Requirements Engineering **7**(2): 61-72.

ISO (1987). Information processing systems - Concepts and terminology for the conceptual schema and the information base. ISO/TR 9007:1987. J. J. v. Griethuysen (ed.), ISO TC97/SC5/WG3.

ISO/DIS (2009). Language resource management - Semantic annotation framework (SemAF) - Part 2: Dialogue acts ISO/DIS 24617-2,

ISO/IEC (1996). Information technology -- Syntactic metalanguage -- Extended BNF ISO/IEC 14977:1996,

ISO/IEC (2007a). International vocabulary of metrology -- Basic and general concepts and associated terms (VIM) ISO/IEC Guide 99:2007,

ISO/IEC (2007b). Software engineering - Metamodel for development methodologies ISO/IEC 24744,

IT Cortex IT Cortex project failure statistics http://www.it-cortex.com/Stat_Failure_Cause.htm, Accessed 10/2010.

Jacobson, I. (1987). Object-oriented development in an industrial environment. Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'87). Orlando, Florida, USA, ACM: 183-191.

Jacobson, I. (2004). "Use cases - Yesterday, today, and tomorrow." Software and Systems Modelling (SoSyM) **3**(3): 210-220

Jakobson, R. (1990). The speech event and the functions of language. On language. M. Monville-Burston and L. R. Waugh (ed.). Cambridge, Harvard University Press: 69-79.

Jouault, F. (2005). Loosely coupled traceability for ATL. Traceability Workshop at European Conference on Model Driven Architecture (ECMDA). Nuremberg, Germany.

Jouault, F. and I. Kurtev (2005). Transforming models with ATL. Satellite Events at the MoDELS 2005 Conference. Montego Bay, Jamaica**:** 128-138.

Juristo, N. and A. Moreno (2001). Basics of software engineering experimentation. Boston, Kluwer.

Juristo, N., A. Moreno and A. Silva (2002). "Is the European industry moving toward solving requirements engineering problems?" IEEE Software **19**(6): 70-77.

Kabanda, S. and M. Adigun (2006). Extending model driven architecture benefits to requirements engineering. 2006 Annual Research Conference of the SAICSIT on IT Research in Developing Countries. Somerset West, South Africa, South African Institute for Computer Scientists and Information Technologists**:** 22-30.

Kahn A.B. (1962). Topological sorting of large networks. Communications of the ACM. New York. **5:** 558-562.

Kaindl, H., S. Brinkkemper, J. A. Bubenko, B. Farbey, S. J. Greenspan, C. L. Heitmeyer, J. C. S. d. P. Leite, N. R. Mead, J. Mylopoulos and J. Siddiqi (2002). "Requirements Engineering and technology transfer: obstacles, incentives and improvement agenda." Requirements Engineering **7**(3): 113-123.

Kaplan, R. S. and D. P. Norton (1996). The Balanced Scorecard: translating strategy into cction, Harvard Business School Press.

Karlsson, F. and P. J. Ågerfalk (2004). "Method configuration: adapting to situational characteristics while creating reusable assets." Information and Software Technology **46**(9): 619-633.

Karlsson, F. and K. Wistrand (2006). "Combining method engineering with activity theory: theoretical grounding of the method component concept " European Journal of Information Systems **15**(1): 82-90.

Kelly, S. and J.-P. Tolvanen (2008). Domain-specific modeling: enabling full code generation, Wiley-IEEE.

Kent, W. (1978). Data and reality: basic assumptions in data processing reconsidered. New york, Elsevier.

Kern, H. and S. Kühne (2009). Integration of Microsoft Visio and Eclipse Modeling Framework using M3-level-based bridges. 2nd ECMDA Workshop on Model-Driven Tool & Process Integration. Enschede, Netherlands.

Kherraf, S., É. Lefebvre and W. Suryn (2008). Transformation from CIM to PIM using patterns and archetypes. 19th Australian Conference on Software Engineering (ASWEC 2008). L. Eric and S. Witold**:** 338-346.

Kitchenham, B. (2004). Procedures for performing systematic reviewsKeele University and NICTA

Klima, G. (2008). The medieval problem of universals. The Stanford Encyclopedia of Philosophy (Spring 2008 Edition). E. N. Zalta (ed.), http://plato.stanford.edu/archives/spr2008/entries/universals-medieval/.

Kock, N., J. Verville, A. Danesh-Pajou and D. DeLuca (2009). "Communication flow orientation in business process modeling and its effect on redesign success: Results from a field study." Decision Support Systems **46**(2): 562-575.

Kock, N. F. and R. J. McQueen (1996). "Product flow, breadth and complexity of business processes: An empirical study of 15 business processes in three organizations." Business Process Management Journal **2**(2): 8-22.

Koch, N., G. Zhang and M. J. Escalona (2006). Model transformations from requirements to web system design. 6th International Conference on Web Engineering. Palo Alto, California, USA, ACM**:** 281-288.

Krogstie, J., O. I. Lindland and G. Sindre (1995). Towards a deeper understanding of quality in requirements engineering. 7th International Conference on Advanced Information Systems Engineering**,** Springer LNCS 932**:** 82-95.

Krogstie, J., G. Sindre and H. Jorgensen (2006). "Process models representing knowledge for action: a revised quality framework." Eur J Inf Syst **15**(1): 91-102.

Kroll, P. (2004). Dr. Process: How many use cases should you have in a system?, IBM Rational developerWorks documentation, Accessed 02-2009.

Kueng, P. and P. Kawalek (1997). "Goal-based business process models: creation and evaluation." Business Process Management Journal **3**(1): 17-38.

Kulak, D. and E. Guiney (2000). Use cases: requirements in context, Addison-Wesley.

Kumar, K. and R. J. Welke (1992). Method engineering, a proposal for situation-specific methodology construction. Systems analysis and design : a research agenda. Cotterman and Senn (ed.), Wiley**:** 257-268.

Laguna, M. A. and B. Gonzalez-Baixauli (2005). Requirements variability models: meta-model based transformations. 2005 Symposia on Metainformatics (MIS '05). Esbjerg, Denmark, ACM**:** 9.

Langefors, B. (1977). Theoretical analysis of information systems (4th ed). Lund, Sweden, Studentlitteratur.

Langlois, R. N. (2002). "Modularity in technology and organization." Journal of Economic Behavior & Organization **49**(1): 19-37.

Lapouchnian, A., Y. Yu, S. Liaskos and J. Mylopoulos (2006). Requirements-driven design of autonomic application software. Conference of the Center for Advanced Studies on Collaborative research. Toronto, Canada, ACM**:** 7.

Larman, C. (1997). Applying UML and patterns, Prentice Hall.

Lauesen, S. (2003). "Task descriptions as functional requirements." IEEE Softw. **20**(2): 58-65.

Lawley, M., R. Topor and M. Wallace (1993). Using weakest preconditions to simplify integrity constraint checking. 4th Australian Database Conference. E. Orlowska and M. Papazoglou**:** 161-170.

Le Moigne, J. L. (1985). "Towards new epistemological foundations for information systems." Systems Research and Behavioral Science **2**(3): 247-251.

Leplin, J., Ed. (1984). Scientific realism, University of California Press.

Leśniewski, S. (1992). Collected works, Kluwer.

Lewis, J. R. (1995). "IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use." Int. J. Hum.-Comput. Interact. **7**(1): 57-78.

Limaye, M. R. and D. A. Victor (1991). "Cross-cultural business communication research: state of the art and hypotheses for the 1990s." Journal of Business Communication **28**(3): 277-299.

Lindland, O. I., G. Sindre and A. Sølvberg (1994). "Understanding quality in conceptual modeling." IEEE Softw. **11**(2): 42-49.

Lockemann, P. C. and H. C. Mayr (1986). Information system design: techniques and software support. Information Processing 86. H.-J. Kugler (ed.). Amsterdam, The Netherlands, North-Holland.

Loniewski, G., E. Insfran and S. Abrahão (2010). A systematic review of the use of requirements engineering techniques in model-driven development. Model driven engineering languages and systems. D. Petriu, N. Rouquette and Ø. Haugen (ed.), Springer**:** 213-227.

Loux, M. J. (2001). The problem of universals. Metaphysics: contemporary readings. M. J. Loux (ed.). London, Routledge.

Lundell, B. and B. Lings (2004). "Method in action and method in tool: a stakeholder perspective." Journal of Information Technology **19**(3): 215-223.

Lyytinen, K. (1987a). "Different perspectives on information systems: problems and solutions." ACM Comput. Surv. **19**(1): 5-46.

Lyytinen, K. (1987b). "Two views of information modeling." Inf. Manage. **12**(1): 9-19.

Lyytinen, K. and R. J. Welke (1999). "Guest editorial: Special issue on meta-modelling and methodology engineering." Inf. Syst. J. **24**(2): 67-69.

Machado, R. J., J. M. Fernandes, P. Monteiro and H. Rodrigues (2005). Transformation of UML models for service-oriented software architectures. 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS '05)**:** 173-182.

Maiden, N., S. Manning, S. Robertson and J. Greenwood (2004). Integrating creativity workshops into structured requirements processes. Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques. Cambridge, MA, USA, ACM**:** 113-122.

Mammar, A., F. Gervais and R. Laleau (2006). Systematic identification of preconditions from set-based integrity constraints. XXIVème Congrès INFORSID. Hammamet, Tunisia**:** 595-610.

Mazón, J.-N., J. Trujillo, J. Lechtenbörger and rger (2007). "Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms." Data & Knowledge Engineering **63**(3): 725-751.

McGuinness, D. L. (2003). Ontologies come of age. Spinning the Semantic Web: Bringing the World Wide Web to its full potential. D. Fensel, J. i. Hendler, H. Lieberman and W. Wahlster (ed.). Cambridge, Massachussets, USA, MIT Press.

McRae, K. B. and D. A. J. Ryan (1996). "Design and planning of long-term experiments." Canadian Journal of Plant Science **76**: 595-601.

Medina-Mora, R., T. Winograd, R. Flores and F. Flores (1992). The action workflow approach to workflow management technology. Proceedings of the 1992 ACM conference on Computer-supported cooperative work. Toronto, Ontario, Canada, ACM**:** 281-288.

Melão, N. and M. Pidd (2000). "A conceptual framework for understanding business processes and business process modelling." Inform Syst J **10**(2): 105-129.

Mendling, J., H. A. Reijers and W. M. P. v. d. Aalst (2010). "Seven process modeling guidelines (7PMG)." Information and Software Technology **52**(2): 127-136.

Mens, T., K. Czarnecki and P. V. Gorp (2005). A taxonomy of model transformations. Language engineering for model-driven software development. J. Bezivin and R. Heckel (ed.). Dagstuhl, Germany, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI).

Menzies, T. (2003). "Editorial: model-based requirements engineering." Requirements Engi **8**(4): 193-194.

Milton, S. and E. Kazmierczak (2004). "An ontology of data modelling languages: a study using a common-sense realistic ontology." Journal of Dababase Management **15**(2): 19-38.

Milton, S., E. Kazmierczak and C. Keen (2001). An ontological study of data modelling languages using Chisholm's ontology. 11th European-Japanese Conference Information Modelling and Knowledge Bases, Maribor.

Milton, S., E. Kazmierczak and L. Thomas (2000). Ontological foundations of data modeling in information systems. Sixth Americas Conference on Information Systems.

Ministerio de Administraciones Públicas (2000). Métrica versión 3: Metodología de planificación, desarrollo y mantenimiento de sistemas de información, Gobierno de España.

Mintzberg, H. (1979). The structuring of organizations. Englewood Cliffs, N.J., Prentice-Hall.

Mirbel, I. and J. Ralyté (2005). "Situational Method Engineering: combining assembly-based and roadmap-driven approaches." Requir. Eng. **11**(1): 58-78.

Molina, J. C. (2003). Applications development with *OLIVA**NOVA*** Model Execution: a proven approach, CARE Technologies, http://www.ivory.nl/bestanden/WP-BenchmarkGartner%5B1%5D.pdf, Accessed 2010-10.

Moody, D. L. (2003). The Method Evaluation Model: A theoretical model for validating information systems design methods. 11th European Conference on Information Systems (ECIS 2003). Naples, Italy.

Moody, D. L. (2005). "Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions." Data Knowl. Eng. **55**(3): 243-276.

Moody, D. L. and G. G. Shanks (1994). What makes a good data model? Evaluating the quality of Entity Relationship models. 13th International Conference on the Entity-Relationship Approach. Manchester, U.K., Springer**:** 94-111

Moody, D. L., G. Sindre, T. Brasethvik and A. Solvberg (2002). Evaluating the quality of process models: empirical testing of a quality framework. 21st International Conference on Conceptual Modeling, Springer-Verlag.

Moody, D. L., G. Sindre, T. Brasethvik and A. Solvberg (2003). Evaluating the quality of information models: empirical testing of a conceptual model quality framework. 25th International Conference on Software Engineering (ICSE 2003). Portland, USA**:** 295-305.

Morgan, T. (2002). Business rules and information systems - Aligning IT with business goals, Addison-Wesley.

Mumford, E. (1994). "New treatments or old remedies: is business process reengineering really socio-technical design?" Journal of Strategic Information Systems **3**(4): 313-326.

Muñoz, J. and V. Pelechano (2005). Building a software factory for pervasive systems development 17th International Conference on Advanced Information Systems Engineering (CAiSE 2005), Springer LNCS 3520**:** 329-343.

Myers, D. and T. Hathaway (2005). On business requirements and technical specifications: a requirements taxonomyRequirements Solutions Group, LLC, Tampa,

Mylopoulos, J., P. A. Bernstein and H. K. T. Wong (1980). "A language facility for designing database-intensive applications." ACM Trans. Database Syst. **5**(2): 185-207.

Naur, P. and B. Randell (1969). Software engineering. Report on a conference sponsored by the NATO Science CommitteeNATO Scientific Affairs Division, Brussels, Belgium,

Nielsen, J. (1994). Heuristic evaluation. Usability inspection methods. J. Nielsen and R. L. Mack (ed.). New York, USA, John Wiley and Sons.

Nunes, N. J. and J. F. e. Cunha (2000). "Wisdom: a software engineering method for small software development companies." IEEE Software **17**(5): 113-119.

OASIS (2007). Web Services Business Process Execution Language version 2.0 - OASIS standard, OASIS Web Services Business Process Execution Language (WSBPEL) TC, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, Accessed 07-2011.

Olivé, A. (2005). Conceptual Schema-Centric Development: a grand challenge for information systems research. 16th Conference on Advanced Information Systems Engineering. Ó. Pastor and J. F. e. Cunha (ed.). Porto, Portugal, Lecture Notes in Computer Science vol 3520, Springer-Verlag**:** 1-15.

Olivé, A. (2007). Conceptual modeling of information systems. Berlin, Springer.

Olivé, A. and R. Raventós (2006). "Modeling events as entities in object-oriented conceptual modeling languages." Data & Knowledge Engineering **58**(3): 243-262.

OMG (2003). MDA Guide Version 1.0.1 http://www.omg.org/docs/omg/03-06-01.pdf, Accessed 12-2010.

OMG (2006a). Business Process Modeling Notation specification v1.0 http://www.bpmn.org, Accessed 03-2008.

OMG (2006b). Meta Object Facility (MOF) core specification v2.0 http://www.omg.org/cgi-bin/apps/doc?formal/06-01-01.pdf, Accessed 03-2008.

OMG (2008). Meta Object Facility (MOF) 2.0 Query/View/Transformation specification (version 1.0) http://www.omg.org/spec/QVT/1.0/, Accessed 05-2010.

OMG (2009). OMG Unified Modeling Language (OMG UML), Superstructure, V2.2 http://www.omg.org/cgi-bin/doc?formal/09-02-02, Accessed 02-2010.

OMG (2010a). Business Motivation Model, version 1.1 http://www.omg.org/spec/BMM/1.1/, Accessed 06-2011.

OMG (2010b). OMG Unified Modeling Language (OMG UML), Superstructure, V2.3 http://www.omg.org/spec/UML/2.3/Superstructure/PDF, Accessed 11-2010.

OMG (2011). Business Process Modeling Notation (BPMN) version 2.0 http://www.omg.org/spec/BPMN/2.0/, Accessed 04-2011.

Opdahl, A. L. and B. Henderson-Sellers (1999). Evaluating and improving OO modelling languages using the BWW-Model. Ontology, Semiotics and

Practice 1999 - Information Systems Foundations Workshop, Sydney, Australia, Macquarie University.

Opdahl, A. L. and B. Henderson-Sellers (2002). "Ontological evaluation of the UML using the Bunge–Wand–Weber model." Software and Systems Modelling **1**(1): 43-67.

Opdahl, A. L., B. Henderson-Sellers and F. Barbier (1999). An ontological evaluation of the OML metamodel. IFIP TC8/WG8.1 International Conference on Information System Concepts: an integrated discipline emerging. E. Falkenberg, K. Lyytinen and A. Verrijn-Stuart. Univerity of Leiden, The Netherlands, Kluwer**:** 217-232.

Opdahl, A. L., B. Henderson-Sellers and F. Barbier (2000). An ontological evaluation of the OML metamodel. Proceedings of the IFIP TC8/WG8.1 International Conference on Information System Concepts: An Integrated Discipline Emerging, Kluwer, B.V.

Opdahl, A. L. and A. Sølvberg (1992). A framework for performance engineering during information system development. Advanced Information Systems Engineering (CAiSE 1992). Manchester, UK, Springer. **593:** 65-87.

Övergaard, G. and K. Palmkvist (2004). Use Cases: patterns and blueprints, Addison-Wesley.

Panach, J., N. Aquino and O. Pastor (2011). Interaction modeling at PROS research center. 13th IFIP TC13 Conference on Human-Computer Interaction (INTERACT 2011). P. Campos, N. Graham, J. Jorgeet al. Lisbon, Portugal, Springer LNCS. **6949:** 677-678.

Panach, J., S. España, I. Pederiva and O. Pastor (2008). "Capturing interaction requirements in a model transformation technology based on MDA." Journal of Universal Computer Science. Special issue "Designing the Human-Computer Interaction: Trends and Challenges" **14**(9): 1480-1495.

Parnas, D. L. (1972). "A technique for software module specification with examples." Commun. ACM **15**(5): 330-336.

Pastor, O., S. España and A. González (2008). An ontological-based approach to analyze software production methods. Information Systems and e-Business Technologies. United Information Systems Conferences (UNISCON 2008). R. Kaschek, C. Kop, C. Steinberger and G. Fliedl. Klagenfurt, Austria, Springer Lecture Notes in Business Information Processing (LNBIP). **5:** 258-270.

Pastor, Ó., J. Gómez, E. Insfrán and V. Pelechano (2001). "The OO-method approach for information systems modeling: from object-oriented conceptual modeling to automated programming." Information Systems **26**(7): 507-534.

Pastor, O., A. González and S. España (2007). Conceptual alignment of software production methods. Conceptual modelling in information systems engineering. J. Krogstie, A. L. Opdahl and S. Brinkkemper (ed.). Heidelberg, Germany, Springer**:** 209-228.

Pastor, Ó., F. Hayes and S. Bear (1992). OASIS: An object-oriented specification language. CAiSE 1992, Manchester, United Kingdom, Springer-Verlag.

Pastor, Ó., E. Insfrán, V. Pelechano, J. Romero and J. Merseguer (1997). OO-Method: an OO software production environment combining conventional and formal methods. 9th Conference on Advanced Information Systems Engineering (CAiSE'97). A. Olive and J. A. Pastor (ed.). Barcelona, Spain, Springer LNCS.

Pastor, O. and J. C. Molina (2007). Model-Driven Architecture in practice: a software production environment based on conceptual modeling. New York, Springer.

Patching, D. (1995). "Business process re-engineering: don't scare the horses." Management Services **39**(4): 8-11.

Peffers, K., T. Tuunanen, M. Rothenberger and S. Chatterjee (2008). "A Design Science research methodology for information systems research." J. Manage. Inf. Syst. **24**(3): 45-77.

Peirce, C. S. (1998). What is a Sign? The essential Peirce. Selected philosophical writings. Vol 2. (1893-1913). Indiana, USA, Indiana University Press.

Peterson, J. L. (1981). Petri Net theory and the modeling of systems. Englewood Cliffs, Prentice-Hall.

Piaget, J. (1967). Logique et connaissance scientifique, Gallimard.

Piattini, M. (1994). Definición de una metodología de desarrollo para bases de datos orientadas al objeto fundamentadas en extensiones del modelo relacional (in Spanish). PhD thesis, Universidad Politécnica de Madrid Spain.

Pisanelli, D. M., M. Battaglia, A. Gangemi and G. Steve (2002). Ontological analysis for the unification of biology. 2002 AMIA Fall Symposium.

Planas, E., J. Cabot and C. Gómez (2011). Two basic correctness properties for ATL transformations: executability and coverage. 3rd International Workshop on Model Transformation with ATL (MtATL-2011). I. Kurtev, M. Tisi and D. Wagelaar. Zürich, Switzerland**:** 1-9

Plato (2000). The Republic, G. R. F. Ferrari (ed.), Tom Griffith (transl.), Cambridge University Press (originally written ca. 380 BC).

Pohl, K. (1994). The three dimensions of requirements engineering: a framework and its applications. 5th International Conference on Advanced Information Systems Engineering, Paris, France, Pergamon Press.

Raj, A., T. V. Prabhakar and S. Hendryx (2008). Transformation of SBVR business design to UML models. Proceedings of the 1st India Software Engineering Conference. Hyderabad, India, ACM**:** 29-38.

Ralyté, J., S. Brinkkemper and B. Henderson-Sellers, Eds. (2007). Situational method engineering: fundamentals and experiences. IFIP International Federation for Information Processing, v. 244, Springer.

Rashid, A., P. Sawyer, A. Moreira and J. Araújo (2002). Early Aspects: a model for aspect-oriented requirements engineering. 10th Anniversary IEEE Joint International Conference on Requirements Engineering, IEEE Computer Society**:** 199-202.

Raskin, J. (2000). The humane interface: new directions for designing interactive systems, Addison Wesley.

Rational (2003). Rational Unified Process version 2003.06.00.65, Rational Software Corporation.

Reijers, H. and J. Mendling (2008). Modularity in process models: review and effects. 6th International Conference on Business Process Management (BPM 2008). Milan, Italy, Springer LNCS 5240**:** 20-35.

Rescher, N. (1977). Methodological pragmatism: systems-theoretic approach to the theory of knowledge. Oxford, Basil Blackwell.

Rittel, H. and M. Webber (1973). "Dilemmas in a general theory of planning." Policy Sciences **4**: 155-169.

Rolland, C. (2007). Capturing system intentionality with Maps. Conceptual modelling in information systems engineering. J. Krogstie, A. L. Opdahl and S. Brinkkemper (ed.), Springer**:** 141-158.

Rolland, C. and C. Ben Achour (1998). "Guiding the construction of textual use case specifications." Data Knowl. Eng. **25**(1-2): 125-160.

Ruiz, M. (2011). Communication Analysis: a requirements engineering approach for model driven development. Master thesis, DSIC, Universitat Politècnica de València, Valencia, Spain.

Ruiz, M., S. España, A. Gonzalez and O. Pastor (2010). Análisis de Comunicaciones como un enfoque de requisitos para el desarrollo dirigido por modelos. VII Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM 2010), Jornadas de Ingeniería de Software y Bases de Datos (JISBD) O. Avila-García, J. Cabot, J. Muñoz, J. R. Romero and A. Vallecillo. Valencia, España**:** 70-77.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy and W. Lorensen (1991). Object-oriented modeling and design. Englewood Cliffs, New Jersey, USA, Prentice Hall.

Runeson, P. (2003). Using students as experiment subjects - an analysis on graduate and freshmen student data. 7th International Conference on Empirical Assessment & Evaluation in Software Engineering (EASE). Staffordshire, UK, Keele University**:** 95-102.

Runeson, P. and M. Höst (2009). "Guidelines for conducting and reporting case study research in software engineering." Empirical Software Engineering **14**(2): 131-164.

Russo, N., J. Wynekoop and D. Walz (1995). The use and adaptation of system development methodologies. International Conference of IRMA (International Resources Management Association), Atlanta.

Saab, Y. (2001). "A fast and effective algorithm for the feedback arc set problem." Journal of Heuristics **7**(3): 235-250.

Sager, J. C. (1990). A practical course in terminology processing. Amsterdam/Philadelphia, John Benjamins Publishing.

Santos, J., A. Moreira, J. Araújo, V. Amaral, M. Alférez and U. Kulesza (2008). Generating requirements analysis models from textual requirements. First International Workshop on Managing Requirements Knowledge, IEEE**:** 32-41.

Scheer, A.-W. (2000). ARIS - Business process modeling, 3rd edition. New York, Springer.

Schilling, M. A. (2000). "Towards a general modular systems theory and its application to inter-firm product modularity." Academy of Management Review **25**(2): 312-334.

Schuette, R. (1999). Architectures for evaluating the quality of information models - a meta and an object level comparison. 18th International Conference on Conceptual Modeling (ER 1999), Springer-Verlag**:** 764-765.

Schuette, R. and T. Rotthowe (1998). The Guidelines of Modeling - an approach to enhance the quality in information models Conceptual Modeling (ER'98). T. W. Ling, S. Ram and M. L. Lee. Singapore, Springer LNCS 1507**:** 240-254.

Schuler, D. and A. Namioka (1993). Participatory design: principles and practices, L. Erlbaum Associates Inc.

Searle, J. R. and D. Vanderveken (1985). Foundations of illocutionary logic, Cambridge University Press.

Sendall, S. and A. Strohmeier (2000). From use cases to system operation specifications. 3rd International Conference on the Unified Modeling Language. York, UK, Springer-Verlag**:** 1-15.

Shanks, G. G. and P. Darke (1997). Quality in conceptual modelling: linking theory and practice. Asia-Pacific Conference on Information Systems (PACIS 1007). Brisbane**:** 805-814.

Shannon, C. E. (1948). "A mathematical theory of communication." Bell System Technical Journal **27**: 379-423 & 623-656.

Shaw, M. E. and J. M. Wright (1967). Scales for the measurement of attitudes New Yoyrk, McGraw-Hill.

Siau, K. and M. Rossi (1998). Evaluation of information modeling methods - a review. 31st Hawaii International Conference on Systems Science (HICSS 1998) Volume 5. Kohala Coast, USA, IEEE**:** 314-322.

Silver, M. S., M. L. Markus and C. M. Beath (1995). "The information technology interaction model: a foundation for the MBA core course." MIS Quarterly **19**(3): 361-390.

Simon, H. (1976). Administrative behavior. New York, Free Press.

Simons, A. J. H. (1999). Use Cases considered harmful. Technology of Object-Oriented Languages and Systems (TOOLS 1999). Nancy, France, IEEE Computer Society**:** 194-203.

Snoeck, M., C. Michiels and G. Dedene (2003). Consistency by construction: the case of MERODE. Conceptual Modeling for Novel Application Domains. Proceedings of ER 2003 Workshops ECOMO, IWCMQ, AOIS, and XSDM, Chicago, IL, USA.

Sommerville, I. (2006). Software Engineering, Addison Wesley Longman Ltd.

Stamper, R. K. (1985). Management epistemology: garbage in, garbage out. Knowledge Representation for Decision Support Systems. Durham, Elsevier.

Stamper, R. K. (1986). A logic of social norms for the semantics of business information. Database Semantics (DS-1). T. B. Steel and R. Meersman (ed.). Amsterdam, North-Holland**:** 233-253.

Stamper, R. K. (1997). Organizational semiotics. Information Systems: an emerging discipline. F. Stowell and J. Mingers (ed.). London, McGraw Hill**:** 267-283.

Steinberg, D., F. Budinsky, M. Paternostro and E. Merks (2008). EMF: Eclipse Modeling Framework, Addison-Wesley Professional.

Stewart, C. J. and W. B. Cash (2002). Interviewing: principles and practices. New York, McGraw-Hill.

The Standish Group (1995). Chaos Report 1994The Standish Group International, Inc, www.standishgroup.com

The Standish Group (2008). CHAOS Summary 2008The Standish Group International, Inc,

The Standish Group (2010). CHAOS Summary 2010The Standish Group International, Inc,

Van Gigch, J. P. and J. L. Le Moigne (1989). "A paradigmatic approach to the discipline of information systems." Systems Research and Behavioral Science **34**(2): 128-147.

van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. 5th IEEE International Symposium on Requirements Engineering (RE 2001). Toronto, Ont., Canada, IEEE Computer Society**:** 249-262.

Vilbergsdóttir, S. G., E. T. Hvannberg and E. L.-C. Law (2006). Classification of usability problems (CUP) scheme: augmentation and exploitation. 4th Nordic Conference on Human-Computer Interaction: changing roles. Oslo, Norway, ACM**:** 281-290.

Volere (2006). Requirements specification template & Atomic requirements template, Atlantic Systems Guild, www.volere.co.uk, Accessed 05/2008.

Wand, Y., D. Monarchi, J. Parsons and C. Woo (1995). "Theoretical foundations for conceptual modeling in information systems development." Decision Support Systems **15**.

Wand, Y. and R. Weber (1990). "An ontological model of an information system." IEEE Transactions on Software Engineering **16**(11): 1282 -1292.

Wand, Y. and R. Weber (1993). "On the ontological expressiveness of information systems analysis and design grammars." Information Systems **3**: 217-237.

Wand, Y. and R. Weber (1995). "On the deep structure of information systems." Inf. Syst. J. **5**: 203-223.

Weber, R. (1997). Ontological foundations of information systems. Queensland, Australia, Coopers & Lybrand.

Weber, R. (1999). The Information Systems discipline: The need for and nature of a foundational core. IS Foundations Workshop; Ontology, Semiotics and Practice, Macquarie University.

Weigand, H., P. Johannesson, B. Andersson, M. Bergholtz, A. Edirisuriya and T. Ilayperuma (2006). On the notion of value object 18th International Conference on Advanced Information Systems Engineering (CAiSE 2006). Luxembourg, Belgium, Springer LNCS**:** 321-335.

White, S. and M. Edwards (1995). A requirements taxonomy for specifying complex systems. Proceedings of the 1st International Conference on Engineering of Complex Computer Systems, IEEE Computer Society.

Wieringa, R. (2009). Design science as nested problem solving. 4th International Conference on Design Science Research in Information Systems and Technology. Philadelphia, Pennsylvania, ACM**:** 1-12.

Wieringa, R. J. (1996). Requirements engineering: frameworks for understanding, Wiley.

Wieringa, R. J. (1998). "A survey of structured and object-oriented software specification methods and techniques." ACM Comput. Surv. **30**(4): 459-527.

Wieringa, R. J. (2002). Design methods for reactive systems: Yourdon, Statemate and the UML, Morgan Kaufmann.

Wieringa, R. J. (2008). Requirements Engineering research methodology: principles and practice (RE 2008 tutorial) http://wwwhome.cs.utwente.nl/~roelw/DesignScienceMethodology-handout.pdf, Accessed 02-2009.

Wieringa, R. J. and J. M. G. Heerkens (2004). Evaluating the structure of research papers: A case study. Second International Workshop in Comparative Evaluation of Requirements Engineering (CERE'04). Kyoto, Japan, IEEE Press**:** 41-50.

Wieringa, R. J. and J. M. G. Heerkens (2006). "The methodological soundness of requirements engineering papers: a conceptual framework and two case studies." Requir. Eng. **11**(4): 295-307.

Winograd, T. (1986). A language/action perspective on the design of cooperative work. 1986 ACM conference on Computer-supported cooperative work. Austin, Texas, ACM.

Winograd, T. and F. Flores (1987). Understanding computers and cognition: A new foundation for design. Boston, Addison-Wesley.

Winston, M. E., R. Chaffin and D. Herrmann (1987). "A taxonomy of part-whole relations." Cognitive Science **11**(4): 417-444.

Wirth, N. (1971). "Program development by stepwise refinement." Commun. ACM **14**(4): 221-227.

Wohed, P. and B. Andersson (2005). Meta-modelling as a means for improved communication and interoperability - The case of FRISCO. Enterprise Modelling and Ontologies for Interoperability (EMOI-INTEROP'05), co-located with CAiSE'05. Porto, Portugal.

Wohlin, C., P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell and A. Wesslén (2000). Experimentation in software engineering: an introduction, Kluwer.

Yadav, S. B., R. R. Bravoco, A. T. Chatfield and T. M. Rajkumar (1988). "Comparison of analysis techniques for information requirement determination." Commun. ACM **31**(9): 1090-1097.

Yourdon, E. (1989). Modern structured analysis. Upper Saddle River, New Jersey, USA, Yourdon Press Computing Series.

Yu, E. (1997). Towards modelling and reasoning support for early-phase requirements engineering. 3rd IEEE Intenational Symposium on Requirements Engineering (RE'97). Washington D.C., USA**:** 226-235.

Yu, E. and J. Mylopoulos (1994). From E-R to "A-R" - Modelling strategic actor relationships for business process reengineering. Proceedings of the13th International Conference on the Entity-Relationship Approach. Manchester, Springer-Verlag**:** 548-565.

Yuditskii, S. A. (2003). "Behavioral models of business systems " Autom. Remote Control **64** (2): 310-321

Yue, T., L. Briand and Y. Labiche (2010). "A systematic review of transformation approaches between user requirements and analysis models." Requirements Engineering: 1-25.

Zave, P. and M. Jackson (1997). "Four dark corners of requirements engineering." ACM Transactions on Software Engineering and Methodology (TOSEM) **6**(1): 1-30.

Zhang, L. and W. Jiang (2008). Transforming business requirements into BPEL: a MDA-based approach to web application development. IEEE International Workshop on Semantic Computing and Systems, IEEE**:** 61-66.

zur Muehlen, M. (1999). Resource modeling in workflow applications. Workflow Management Conference. Münster**:** 137-153.

INDEX OF CONCEPT DEFINITIONS

D

E

G

H

I

K

L

M

N

O

P

R

S

T

U

Note: When this thesis was printed, the *OLIVA**NOVA*** technology had its name recently changed to Integranova. We have updated most references to this technology in the thesis. However, the snapshots of the *OLIVA**NOVA*** Modeler still correspond a previous release.

Keep up with Sergio at http://es.linkedin.com/in/sergioespana
Find his articles at http://www.citeulike.org/user/sergioespana
His personal webpage is http://www.pros.upv.es/index.php/en/sespana
Feel free to drop an email at sergio.espana@pros.upv.es

To be continued...

Arturo enjoys software development projects that entail complex challenges. He created Communication Analysis by investigating information system fundamentals and by revising existing analysis methods.

Óscar is a strong believer of model-driven development. He created the OO-Method to make his dream of automatic software generation come true. Currently, he is applying conceptual modeling to the human genome.

About the author

While studying Computer Science at Universidad Politécnica de Valencia, Sergio got interested in information systems analysis and design. Under the supervision of Arturo, he participated in a big development project applying Communication Analysis and initiated research in the field of software development methods. He got so hooked on research that Óscar didn't have trouble persuading him to join the ProS research centre, the craddle of the OO-Method. It wasn't long before he embarked on integrating Communication Analysis and the OO-Method.

This thesis is the result of this five-year endeavour.

Feel free to drop an email at sergio.espana@pros.upv.es