A service-oriented approach for the *i** framework

By Hugo Estrada Esquivel

PhD Thesis

Presented to the Department of Information Systems and Computation of the Valencia University of Technology, Spain, and to the Department of Information and Communication Technology of the University of Trento, Italy in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science

September 2008

The research reported in this thesis has been financially supported by the SUPERA project in Mexico, the Valencia University of Technology, Spain, the enterprise CARE Technologies S.A., and the University of Trento, Italy.

Printed in Spain

Valencia

Thesis Advisors:

Dr. Oscar Pastor López, Valencia University of Technology, Spain

Dr. Paolo Giorgini, University of Trento, Italy

Thesis Committee:

Dr. John Mylopoulos, University of Trento, Italy

Dr. Xavier Franch, University of Catalonia, Spain

Dr. Jaelson Castro, University of Pernambuco, Brazil

Dr. Juan Sánchez, Valencia University of Technology, Spain

Dr. Vicente Pelechano, Valencia University of Technology, Spain

Abstract

New application areas such as e-Business, application service provision and peer-to-peer computing all call for very complex software systems which effectively support "on-line" enterprise processes. To build such systems, practicing software engineers are discovering the effectiveness of using organizational modeling techniques to facilitate the elicitation of requirements for information systems and also for guiding and supporting the software production process.

In this context, the *i** Framework is one of the most well-founded organizational modeling techniques in use today. It mainly focuses on: a) the representation of social and intentional relationships among the network of actors of an enterprise, and b) the representation of the internal behaviors required to satisfy actor dependencies. The *i** framework supports the description of organizational networks made up of social actors that have freedom of action, but that depend on other actors to achieve their objectives and goals.

Despite the well-known advantages of the *i** modeling approach, there are certain issues that still need to be improved to assure their effectiveness in practice. In order to accurately identify areas of strength as well as weaknesses of *i** in real case studies, empirical evaluations of this framework must be conducted in practice.

One of the objectives of this thesis is to present an empirical evaluation that enables us to identify and to understand what the practical problems of *i** are. We present the lessons learned, both in terms of the strengths of *i** and in terms of the detected weak points that need to be overcome. Solutions for these weak points are also proposed as an initial response to the results of the empirical evaluation.

We consider that service-orientation is currently considers as a very promising paradigm to deal with the complexity of modeling the IT systems. In this sense, the main objective of the thesis is to define a service-oriented architecture to solve the problem of *i** complexity in real-life cases. The proposed architecture distinguishes three abstractions levels (services, process and protocols) and describes a methodological approach to align the business models produces at these abstraction levels.

Our service-oriented approach considers the following aspects: a) A conceptual modeling language, based on the *i** primitives, which defines the modeling concepts and their corresponding relationships. b) A service-oriented architecture specific for the *i** framework that define the service components and the modeling diagrams. c) A business modeling method to represent services at the organizational level.

The business services and the service components have been precisely defined in terms of properties and relationships. It is important to point out that services components have been designed based on the intended use of this abstraction in representing services at the organizational level.

With the extensions proposed in this thesis, our intention is to overcome the current limitations that practitioners face when using *i** in its current state. In fact, these extensions are intended to both, solve the problems that were detected, and to make the practical application of the method easier.

Resumen

Nuevas áreas de aplicación como comercio electrónico, aplicaciones para provisión de servicios y computación P2P (peer-to-peer) requieren de sistemas de software complejos que puedan soportar procesos de negocio "en línea". Actualmente, los ingenieros de software han descubierto la efectividad de usar técnicas de modelado organizacional para guiar el proceso de producción de este tipo de sistemas complejos.

En este contexto, el framework *i** es una de las técnicas de modelado organizacional mejor fundamentadas hoy en día. *i** se enfoca en dos aspectos principales: a) la representación de las relaciones sociales e intencionales que existen entre la red de actores de un negocio. b) la representación del comportamiento interno requerido para satisfacer las dependencias entre actores. El framework *i** permite describir una organización como una red de actores que tienen libertad de acción, pero que dependen de otros actores para lograr sus metas y objetivos.

Sin embargo, a pesar de las bien conocidas ventajas de *i**, existen ciertos problemas que necesitan ser resueltos para asegurar su efectividad en ambientes reales de desarrollo. En este sentido, el framework necesita ser evaluado en la práctica con el objetivo de identificar sus fortalezas y debilidades en casos de estudio reales.

Uno de los objetivos de esta tesis fue realizar una evaluación empírica que nos permitiera identificar y analizar los problemas prácticos de *i**. Se presentan las lecciones aprendidas en términos de fortalezas y de puntos débiles que necesitan ser resueltos. Además, la tesis presenta soluciones a los puntos débiles que fueron detectados en la evaluación empírica.

Consideramos que la orientación a servicios es un paradigma muy prometedor para enfrentar la complejidad del modelado de sistemas de tecnologías de información actuales. En este sentido, el principal objetivo de esta tesis fue definir una arquitectura orientada a servicios que nos permitiera resolver los problemas de

complejidad de *i** en la práctica. La arquitectura propuesta distingue tres niveles de abstracción complementarios (servicios, procesos y protocolos) y describe un enfoque metodológico para alinear los modelos de negocios producidos en cada nivel de abstracción.

Nuestro enfoque orientado a servicio considera los siguientes aspectos: a) un lenguaje de modelado conceptual basado en las primitivas de modelado de *i**, el cual define los conceptos y sus correspondientes relaciones. b) Una arquitectura orientada a servicios, específica para el framework *i**, que define los componentes del servicio y los diagramas de modelado y finalmente c) un método de modelado para representar servicios a nivel organizacional.

Los servicios de negocio y los componentes del servicio han sido definidos en forma precisa en términos de propiedades y relaciones. Es importante hacer notar que los componentes del servicio han sido diseñados teniendo en cuenta su utilidad para representar servicios al nivel organizacional.

Nuestra intención al proponer extensiones al framework *i** es dar solución a las actuales limitaciones que tienen los analistas al usar *i** como lenguaje de modelado es su estado actual. Con esto se intenta dar una solución a los problemas detectados y hacer más simple la tarea de modelado organizacional.

Sommario

Nuovi indrizzi di applicazione come il e-commerce, aplicazioni per la provisione di servizi e computo P2P (peer-to-peer) richiedono di sistemi di software complessi che possono supportare procesi di negozi "on line". Nella attualitÃ , i software engineers hanno scoperto la efficacia del utilizzo di tecniche di modelazione organizzativa per la guida del proceso di produzione di questo tipo di sistemi complessi. In questo contesto, il framework i* Ã¨ una delle tecnique di modelazione organizzativa meglio fondamentata oggi. i* se focalizza in due aspetti principali: a) la rappresentazione delle relazioni sociali ed intenzionali che esistono tra la rete di attori di un affare. b) la rappresentazione del comportamento interno richiesto per soddisfare le dipendenze tra gli attori. Il quadro i* permette describere una organizzazione come una rete di attori per ottenere il loro obiettivo. Tuttavia, esistono dei problemi che richiedono soluzione per assicurare la efficacia in ambienti di sviluppo reali. In questo senso, il quadro richiede di essere valutato nella pratica con lo scopo di riconoscere i suoi ponti di forza e debolezze nei casi di studi reali. Uno degli obiettivi di questa tesi fu la realizzazione di una valutazione empirica che ci consente la identificazione ed analisi dei problemi pratici di i*. Sono presentati le lessione imparate in termini di ponti di forza e debolezze che richiedono soluzione. Inoltre, questa tesi presenta soluzioni ai ponti di debolezza che furono individuati nella valutazione empirica.

Consideriamo la orientazione ai servizi un paradigma molto prometente per affrontare la complessita della modelazione di sistemi di tecnologie della informazione. In questo senso, l'obiettivo principale di questa tesi è quello di definire una architettura orientata ai servizi per risolvere i problemi della complessita' di i* nella pratica. L'architettura proposta distingue tre livelli (servizi, processi e protocolli) di astrazione e descrive un approccio metodologico per allineare i modelli di negozio prodotti

e questi livelli di astrazione. Nostro approccio orientato ai servizi considera i seguenti aspetti: a) un linguaggio di modellazione concettuale, sulla base di primitive di modellazione di i*, che definisce i concetti di modellazione e dei loro rapporti, b) una archittetura orientata ai servizi, specifica per il quadro i* che definiscono i componenti di servizio e i diagrammi di modellizzazione. c) un metodo di modellizzazione di negozio per rappresentare i servizi a livello organizzativo.

I servizi di negozio e le componenti di servizio sono stati definiti con precisione in termini di proprietà e rapporti. E importante sottolineare che le componenti di servizio sono stati progettati sulla base della loro utilità per rappresentare servizi a livello organizzazionale.

La nostra intenzione è di superare gli attuali limiti che gli utenti di i* affrontano oggi. Con questo si cerca di risolvere i problemi che sono stati rilevati, e rendere più semplice il compito di modellazione organizzativa.

Resum

Noves àrees d'aplicació com comerç electrònic, aplicacions per a provisió de serveis i computació P2P (peer-to-peer) requereixen de sistemes de programari complexos que puguin suportar processos de negoci "en línia". Actualment, els enginyers de programari han descobert l'efectivitat d'usar tècniques de modelització organitzacional per a guiar el procés de producció d'aquest tipus de sistemes complexos.

En aquest context, el framework i* és una de les tècniques de modelització organitzacional millor fonamentades avui dia. i* s'enfoca en dos aspectes principals: a) la representació de les relacions socials i intencionals que existeixen entre la xarxa d'actors d'un negoci. b) la representació del comportament intern requerit per a satisfer les dependències entre actors. El framework i* permet descriure una organització com una xarxa d'actors que tenen llibertat d'acció, però que depenen d'altres actors per a assolir les seves metes i objectius.No obstant això, a pesar de les avantatges bé conegudes de*i, existeixen certs problemes que necessiten ser resolts per a assegurar la seva efectivitat en ambients reals de desenvolupament. En aquest sentit, el framework necessita ser avaluat en la pràctica amb l'objectiu d'identificar les seves fortaleses i debilitats en casos d'estudi reals.

Un dels objectius d'aquesta tesi va ser realitzar una avaluació empírica que ens permetés identificar i analitzar els problemas pràctics de *i . Es presenten les lliçons apreses en termes de fortaleses i de punts febles que necessiten ser resolts. A més, la tesi presenta solucions als punts febles que van ser detectats en l'avaluació empírica. Creiem que l'orientació a serveis és un paradigma molt prometedor per a enfrontar la complexitat de la modelització de sistemes de tecnologies d'informació actuals.

En aquest sentit, el principal objectiu d'aquesta tesi va ser definir una arquitectura orientada a serveis que ens permetés resoldre els problemes de complexitat de* i en la pràctica. L'arquitectura proposada distingeix tres nivells d'abstracció complementaris (serveis, processos i protocols) i descriu un enfocament metodològic per a alinear els models de negocis produïts en cada nivell d'abstracció. Aquest enfocament orientat a servei considera els següents aspectes: a) un llenguatge de modelització conceptual basat en les primitives de modelització de *i , el qual defineix els conceptes i les seves corresponents relacions. b) Una arquitectura orientada a serveis, específica per al framework i*, que defineix els components del servei i els diagrames de modelització i finalment c) un mètode de modelització per a representar serveis a nivell organitzacional.

Els serveis de negoci i els components del servei han estat definits en forma precisa en termes de propietats i relacions. És important fer notar que els components del servei han estat dissenyats tenint en compte la seva utilitat per a representar serveis al nivell organitzacional.

La nostra intenció al proposar extensions al framework i* és donar solució a les actuals limitacions que tenen els analistes a l'usar i* com llenguatge de modelització en el seu estat actual. Amb això s'intenta donar una solució als problemes detectats i fer més simple la tasca de modelització organitzacional.

Acknowledgments

This thesis could not have been completed without the precious support of many people. Above all, I want to thank to Oscar Pastor, my thesis advisor, who gave me guidance and support throughout the entire thesis process. I thank him for being the perfect partner in this scientific adventure. In fact, I am truly positive this one was just the beginning. Also I want to thank Oscar for all the affection and friendship we shared during this years. Thanks Oscar for being our friend.

I would like to express my gratefulness to John Mylopoulos and Paolo Giorgini for giving me the opportunity to develop my research work in their prestigious research group. Thanks both for the hours of intellectually stimulating discussions in Trento.

A special thank to Jaelson Castro for his contributions, advice and friendship. It is a pleasure to work with you Jaelson.

I would like to express my warmest appreciation to all my friends who have supported me through joy and who make my life richer. Thanks to Javier, Azucena, Juan, Max, Toño, Manuel, Jose Luis, Vero, Carlos, my Mexican friends. Also thanks to Isabel, Juan, Manoli, Pele, Joan, Pedro, Ana and Nelly, who was our support in the final steps of this process.

There are no words to write my love to my parents, Angel and Maria, and my brothers, Vicente and Paty. I want to say thanks for being always an isolated paradise to forget the problems. Thanks also to my family in Higueron, Tere, Lucio, Ari and Ervin because the life is easier with you.

This thesis is dedicated to my definitively complement in the life, Alicia, thanks for being here in the good and bad times of my life. Thanks Alicia, you know that today, you are all I need to be happy. I love you Alicia.

Finally (sorry, but I need to write this part in Spanish) a mi hija Denisse (mi chicota) por ser mi ejemplo de superación y de valentía ante la vida. Por haber soportado mis ausencias y estar siempre ahí, dispuesta a darme todo su amor sin esperar nada a cambio. Por demostrarme que sólo falta una actitud positiva para enfrentar cada nuevo reto en la vida. A ti Denisse que sabes que te quiero con todo el corazón.

A mi hijo Iker (gatto) que nació durante el desarrollo de este trabajo y que es la fuente de agua fresca que nos renueva cada día. A ti, que ahora eres demasiado pequeño para entender la felicidad que has traído a nuestra casa. A ti Iker que cada día inventas una nueva forma de hacernos reír. A ti que ahora tienes más sonrisas que dientes. Te quiero gatto.

Thank you all.

Hugo Estrada Esquivel

Valencia, Spain, September 2008

A service-oriented architecture for the *i** Framework

Contents

Chapter 1

Introduction

This section briefly introduces the key aspects of the research efforts presented in this thesis and describes several problems and our solutions to them with an emphasis on the main research contributions. The structure of the thesis is also presented at the end of the section.

1.1 The Context

New application areas such as e-Business, application service provision and peer-to-peer computing all call for very complex software systems which effectively support "on-line" enterprise processes. To build such systems, practicing software engineers are discovering the effectiveness of using organizational modeling techniques to facilitate the elicitation of requirements for information systems and also for guiding and supporting the software production process.

In this context, the *i** Framework is one of the most well-founded organizational modeling techniques in use today. It mainly focuses on: a) the representation of social and intentional relationships among the network of actors of an enterprise, and b) the representation of the internal behaviors required to satisfy actor dependencies. The *i** framework supports the description of organizational networks made up of social actors that have freedom of action, but that depend on other actors to achieve their objectives and goals.

Despite the well-known advantages of the *i** modeling approach, there are certain issues that still need to be improved to assure their effectiveness in practice. In order to accurately identify areas of strength as well as weaknesses of *i** in real case studies, empirical evaluations of this framework must be conducted in practice.

First aim of this thesis is to present an empirical evaluation that enables us to identify and to understand what the practical problems of *i** are. We will present the lessons learned, both in terms of the strengths of *i** and in terms of the detected weak points that need to be overcome. Solutions for these weak points are also proposed as an initial response to the results of the empirical evaluation.

We consider that service-orientation is currently considers as a very promising paradigm to deal with the complexity of modeling the IT systems. In this sense, the main objective of the thesis is to define a service-oriented architecture to solve the problem of *i** complexity in real-life cases. The proposed architecture distinguishes three abstractions levels (services, process and protocols) and describes a methodological approach to align the business models produces at these abstraction levels.

Our service-oriented approach considers the following aspects: a) A conceptual modeling language, based on the *i** primitives, which defines the modeling concepts and their corresponding relationships. b) A service-oriented architecture specific for the *i** framework that define the service components and the modeling diagrams. c) A business modeling method to represent services at the organizational level.

The business services and the service components have been precisely defined in terms of properties and relationships. It is important to point out that services components have been designed based on the intended use of this abstraction in representing services at the organizational level.

With the extensions proposed in this thesis, our intention is to overcome the current limitations that practitioners face when using *i** in its current state.

1.2 The Problem

In this section, we present the main problems addressed by this work, namely: the lack of evaluations of the *i** framework and the lack of methodological extensions to improve the *i** framework.

1.2.1 The lack of evaluations of the *i** framework

*i** Framework is one of the most well-founded frameworks for organizational modeling. It uses strategic relationships to model the social and intentional context of an enterprise, and has been used widely in research and in some industrial projects. In this context, the *i** Framework and its methodological extensions (such as GRL (Liu and Yu 2003) and Tropos (Bresciani, Perini, Giorgini, Giunchiglia, and Mylopoulos 2004)) have been used as a powerful analysis technique in a wide range of application domains: Business Modeling (Kolp, Giorgini, and Mylopoulos 2003), Object-Oriented System Development (Castro, Alencar, Filho, and Mylopoulos 2001), (Martinez, Castro, Pastor, and Estrada 2003), Software Requirements Elicitation (Maiden, Jones, Manning, Greenwood, and Renou 2004), (Estrada, Martinez, and Pastor 2003), Agent System Development (Bresciani, et al. 2004) (Bastos and Castro 2003), Selection of Components (Carvallo, Franch, Quer, and Rodriguez 2004), Non-Functional Requirements (Chung, Nixon, Yu, and Mylopoulos 2000), Security, Trust, Dependability and Privacy (Yu and Liu 2001), (Giorgini, Massacci, Mylopoulos, and Zannone 2005), etc. In all these works, the research has been oriented to extend and enrich the semantic of concepts so to be used in different domains and for different applications.

However, up to now, no empirical evaluation has been proposed and this makes very difficult to confirm and argue about the practical *i** usefulness. The thesis presents the results of an empirical evaluation of *i** using industrial case studies. We go a further step in determining the modifications that are necessary to ensure the real applicability of the *i** framework.

1.2.2 The lack of methodological extensions to improve the *i** framework

The *i** modeling concepts have been used in a wide range of application domains. In all applications, the *i** concepts have been used to capture the social and intentional elements of each specific domain, but just a little attention has been paid to propose mechanisms to manage the complexity of the modeling activity and to improve the usability and scalability of the *i** models.

Practical experiences have revealed that there are certain issues that need to be improved to ensure their effectiveness in practice. Particularly, new modeling primitives and mechanisms are needed to handle the complexity management of large organizational models. In order to do this, we have proposed a method based on business services as building blocks that encapsulate organizational behaviors.

1.3 The Solution

This section briefly presents our approach to solve the problems discussed in the previous section.

1.3.1 The lack of evaluations of the *i** framework

In this thesis, an empirical evaluation of *i** using industrial case studies was carried in collaboration with an industrial partner who uses an object-oriented model-driven approach for software development.

The evaluation of *i** uses a feature-based framework that captures relevant characteristics in industrial settings. By performing the evaluation, the evaluators assign a judgment (value) to specify how well or badly each evaluated feature is supported by the *i** framework. The evaluation framework has been designed keeping in mind that it is to be used within model-based software development environments.

One contribution of this work is the definition of a consensus to explain the reason for assigning a certain value to the analyzed issues. As a result of the empirical evaluation, the thesis reports on lessons learned from this experience, both in terms of strengths and detected weaknesses. Another contribution is the definition of a set of results that can play a relevant role in guiding future extensions of the *i** framework.

1.3.2 The lack of methodological extensions to improve the *i** framework

As main result of our practical evaluation, what is clearly required is the need to extend the *i** framework with mechanisms to manage granularity and refinement in real-life projects. These mechanisms must allow us to create and represent an organizational model in a modular way. As a solution, in this thesis we propose a service-oriented approach that deals with the current drawbacks of *i**.

We can characterize the service-oriented paradigm by the explicit representation of the externally observable properties of a system. Thus, a system (an enterprise in our case) can be described based on the description of its external properties, that we called business services. In this sense, organizational behaviors can be encapsulated based on the description of the business services. Thus, services can be used as basic buildings blocks where business analysts do not need to have knowledge about the internal implementation of the services that offer and expose an enterprise. The services are the mechanism to map the abstract definition of business functionalities with the internal protocols needed to operationalize the services.

The main idea is the representation of an organizational model as a composite of business services, where these services represent the functionalities that the organization offers to potential customers. Business services become the building blocks that allow us to represent a business model in a high-level, three-tiered conceptual architecture. Business services, business processes, and business protocols are the hierarchically interrelated tiers that make up our service-oriented architecture.

In the proposed approach, the organizational modeling process starts with the definition of a high-level view of the services offered and requested by the organization. Each business service is then refined into more concrete process models according to the business service method introduced. The main advantage is that it provides a solution to manage granularity, refinement and reuse. This results essential when *i** is applied in real-life, complex projects.

We aim at making the modeling process simple by making the social and intentional characteristics of *i** hidden for novel analysts, at least in the early elicitation stages. To do this, the method uses a well-known elicitation mechanism, such as goal analysis, in order to define a goal structure that is built in such a way that it contains the organizational knowledge without explicit social relationships. Thus, a method is proposed, as part of the service-oriented method, in order to transform the goal structure into *i** business models.

We argue that the expressive power of the conceptual primitives that we have introduced in the *i** enables the analyst to better manage the complexity of organizational modeling in practice. However, we also consider that an analysis of the original *i** notation must be done in order to "clean" its syntax and semantics. This is the reason why a revisited version of the *i** modeling concepts is proposed in this thesis, with the objective of defining the service-oriented modeling language proposed for the *i** framework.

Furthermore, the proposed method makes it feasible to use *i** as the starting point for a full software production process. In this process, the elaboration of the organizational model can be the cornerstone of the software process because requirements modeling and conceptual modeling will be the result of a precise model transformation process, where organizational aspects are correctly represented in the corresponding lower-level models. Given the advanced model-based software production tools that currently exist in the market, having extended tools to support a full software process that covers all the activities from organizational modeling to its corresponding final software product can become a reality.

1.4 Innovative Aspects

The key innovative aspects of the thesis can be summarized as follows:

1. ***An empirical evaluation of the i* framework***

One of the contributions of this thesis is the description of an empirical evaluation framework that uses well-defined features to asses *i** with real projects in a software development company that uses model-driven tools for software development. Another contribution is to provide consensual explanations of the reasons to assign a specific value to each one of the analyzed issues. To do this, several meetings were held with designers and users of *i** and Tropos in order to make a judgment about the features values of the evaluation framework. Last contribution of this thesis section is the definition of a set of conclusions of the practical evaluation of *i** to be considered in the definition of new versions of this framework.

2. ***The definition of a modeling language based on i* notation***

In our proposal, *i** is used as business modeling language in order to take advantage of its powerful means for representing the social an intentional setting of an enterprise. With regard to the modeling language definition, one of the contributions of this work is the analysis of the current *i** modeling concepts in order to propose a revisited version that overcomes some of the problems that have been detected in the empirical evaluation.

3. ***A service-oriented method for the i* framework***

Finally, with regard to the service modeling method, the contribution of this thesis is the definition of a new methodological approach to address the enterprise modeling activity using *i**. The new approach is based on the use of business services as building blocks for encapsulating organizational behaviors. We propose a specific business modeling method in accordance with the concept of business service. The use of services as building blocks enables the analyst to represent new business functionalities by composing models of existing services. We propose, as first modeling step, an elicitation technique to find actual implementations of the services offered and requested by the analyzed enterprise, where goals will play a very relevant role in the discovering process.

As a contribution of this work, we introduced a formal definition of the basic concepts of the service-oriented architecture. This architecture can be summarized with three modeling diagrams that capture the service composition, service variability, service objectives, services resources and service behaviors.

As a key point of the method, we propose an extensive use of goals structures as an elicitation mechanism instead of starting the modeling process directly with the intentional concepts of *i**. The idea of hiding the intentional characteristics of *i** (at least in early elicitation stages) is to make the method more suitable for non-expert analysts in the use of *i** concepts. Therefore, another contribution is the definition of a method to derive goal-refinement structures into *i** business models in an automatic way. This proposal, which joins a goal-based elicitation process with the social aspects of the *i** strategic models, represents one of the contributions of this thesis over the current goal modeling techniques.

1.5 Structure of the Thesis

The presentation of this thesis is organized in the following Chapters:

Chapter 1. Introduction

This introductory Chapter provides a brief overview of the issues analyzed in this thesis and we explain the context in which the thesis is developed. To do this, the relevance of the service-oriented architecture is pointed out as a solution to the problems detected in the empirical evaluation of the *i** framework.

Chapter 2. State of the art

This Chapter presents the state of the art in the fields that are relevant to this thesis: evaluations of the *i** framework, goal modeling methods, and service-oriented computing approaches.

Chapter 3. The Empirical Evaluation of the *i framework**

This Chapter details the empirical evaluation of the *i** framework in a specific model-driven software generation environment. The Chapter presents the framework used to carry out the evaluation and the strategy that was used to lead the experiment. We detail the features that integrate the framework and, finally, we present an explanation of agreed upon values assigned to each one of the selected features of the framework.

Chapter 4. The modeling language definition.

In this Chapter, a revisited version of the *i** modeling concepts is presented. The objective of the proposed revisited version is to address the issues detected in the empirical evaluation. The revisited version will be the basis for the definition of the service-oriented modeling language proposed in this thesis.

Chapter 5. The Service-oriented architecture for *i Framework**

This Chapter details the components of the service-oriented architecture: modeling concepts and diagrams. It presents in detail the definition of business services as a key mechanism for encapsulating organizational behaviors.

Chapter 6. The Service-oriented modeling method for the *i Framework**

This Chapter presents the business modeling method based on the concept of services. We detail the set of steps that allow an enterprise to be represented using the concept of business service.

Chapter 7. The Service-oriented Method: a case study

An empirical evaluation of the proposed methodology is presented in this Chapter. The aim is to empirically demonstrate that the proposal overcomes the issues detected in the empirical evaluation of the *i** framework.

Chapter 8. Conclusions and further work.

This Chapter presents the main contributions and relevant work of this thesis. Future work is also presented.

Chapter 2

2. The State of the Art

The main objective of this thesis is to improve the current *i** business modeling process. The first step in achieving this objective was to determine the issues of *i** that need to be solved in order to ensure the use of *i** in practice. To do this, an empirical evaluation with industrial cases studies was performed to detect the strengths and weaknesses of this modeling framework.

Two complementary solutions have been given for the detected issues: first, a revisited version of the *i** modeling concepts has been developed to overcome the repeatability problems found in the empirical evaluation. Then, as a second solution, a service-oriented architecture has been placed at the top of these modeling primitives in order to solve the modularity and refinement issues. This service-oriented approach allows us to encapsulate organizational behaviors in well-defined building blocks.

One of the objectives of this proposal is to make the modeling process simple by making the intentional characteristics of *i** hidden for novel analysts. To be able to do this, a method to transform an elicitation goal structure into the *i** strategic models has been proposed. This transformation method represents one of the contributions of this thesis.

This section introduces a brief overview of the state-of-the art in the research areas that are considered to be relevant to this work: evaluations of the *i** framework and goal modeling methods. A brief review about service-oriented technologies is also presented.

2.1 Evaluations of the *i** framework

Nowadays, the *i** Framework and its methodological derivations, such as GRL (Liu and Yu 2003) and Tropos (Bresciani, Perini, Giorgini, Giunchiglia, and Mylopoulos 2004) are considered to be among the most relevant agent-modeling techniques. In this context, several research efforts have been made to evaluate and compare them with other relevant agent-based techniques.

2.1.1 Shehory and Sturm research works

Shehory and Sturm (2001) propose a feature-based framework for evaluating and comparing agent-oriented methodologies. The framework examines various aspects of each methodology: concepts and properties, notations and modeling techniques, processes, and pragmatics. The authors propose a set of criteria to evaluate the quality of the target methodologies from the software engineering viewpoint. Following, the definition of the concepts is presented according to Shehory and Sturm (2001):

Preciseness: the semantics of a modeling technique must be unambiguous in order to avoid misinterpretation of the models (of the modeling technique) by those who use it.

Accessibility: a modeling technique should be comprehensible to both experts and novices.

Expressiveness: a modeling technique should be able to present: the structure of the system; the knowledge encapsulated within the system; the data flow within the system; the control flow within the system; the interaction of the system with external systems.

Modularity: a modeling technique should be expressible in stages. That is, when new specification requirements are added, there should be no need to modify pervious parts, and these may be used as part of the new specification.

Complexity management: a modeling technique should be expressed, and then examined, at various levels of detail. Sometimes, high-level requirements are needed, while in other situations, more detail is required. Examination and development of all levels should be facilitated.

Executability: either a prototyping capacity or a simulation capacity should be associated with at least some aspects of the modeling technique. That is, the modeling technique has related tools that allow (possibly inefficient) computation for sample input. These tools should demonstrate possible behaviors of the system being modeled and help developers determine whether the intended requirements have been expressed.

Refinability: a modeling technique should provide a clear path for refining a model through gradual stages to reach an implementation, or at least for clearly connecting the implementation level to the design specification.

Analyzability: a methodology, or, preferably, an associated tool is available to check the internal consistency or implications of the models, or to identify aspects that seem to be unclear, such as the interrelations among seemingly unrelated operations. Such tools encourage both consistency and coverage.

Openness: a modeling technique should provide a good basis for modeling agent-based systems without coupling them to a specific architecture, infrastructure or programming language.

Shehory and Sturm also propose the following criteria to evaluate the quality of modeling methodologies according to the desired characteristics for agent-based systems:

Autonomy: unlike objects, agents may be active and are responsible for their own activities: the agent has control over both its reactive and proactive behaviors. The modeling technique should support the capability of describing an agent's self-control feature.

Complexity: agent-based systems are basically sets of components (agents) that interact with each other in order to achieve their goals. These systems may consist of decision-making mechanisms, learning mechanisms, reasoning mechanisms and other complex algorithms. Modeling complex algorithms and mechanisms requires strong expressive power and many layers of detail. A modeling technique should support such expressiveness in order to model the functionality of agent-based systems. Moreover, the complexity feature requires that the modeling techniques should be modular, support complexity management and should describe the complex nature of an agent.

Adaptability: agent-based systems have to be flexible in order to adjust their activities to the dynamic environmental changes. The adaptability feature may require that a modeling technique be modular and be able to activate each component according to the environmental state.

Concurrency: an agent may need to execute several activities/tasks at the same time. The concurrency feature raises the requirement that some agent-based systems must be designed as parallel processing systems. This requires the ability to express parallelism and concurrency in the design and implementation/deployment stages.

Distribution: multi-agent systems are sometimes working on different hosts and should be distributed over a network. This requires the ability to express distribution in the design and implementation/deployment stages.

Communication richness: a basic definition of an agent consists of its autonomous activity. As such, the agent must establish communication with its environment. The environment may include other agents and information sources. The communication is characterized by its type (either inter-agent communication or intra-agent communication), its content, and its architecture (e.g. client-server, peer-to-peer). This requires that a modeling technique be able to express the communication characterization in order to produce agent communication command or sentences during the implementation stage.

All the properties defined by Shehory help the analyst to decide the correct technique to be used to model a specific problem domain.

Shehory and Sturm evaluated the selected agent techniques (AOM, ADEPT, and DESIRE) via a case study for a single agent application.

This first evaluation proposed by Shehory and Sturm did not consider the *i** framework; however, we consider this work to be relevant because it analyzes the relevant characteristics to evaluate modeling techniques. This work was later used in the evaluation of a more extensive list of agent-based techniques.

In (Sturm and Shehory 2003), Shehory and Sturm used the features catalog represented in their framework to perform an empirical evaluation of the GAIA methodology.

More recently, the same authors (Sturm, Dori and Shehory 2005) used the proposed framework in addition to an empirical evaluation based on case studies to carry out an evaluation and comparison analysis of several agent-oriented methodologies including Tropos (Gaia, Tropos, MaSe, and OPM/MAS). The case studies employed students taking a computer science course. An important contribution of this work is the use of a framework, that is based on a set of pre-defined criteria (features), for evaluating and comparing agent-oriented methodologies.

One of the problems of this work is the subjectivity of the evaluation. This is because some of the authors of the evaluation are also authors of the AOMT methodology, which was one of the methodologies analyzed. Therefore, some of the selected features are well addressed by their agent-based technique.

2.1.2 Dam and Winikoff research works

Dam and Winikoff (2003) also performed a feature evaluation analysis of Agent methodologies (MaSe, Prometheus, and Tropos) using an attribute-based evaluation framework. The evaluation was carried out by comparing the strengths and weaknesses of each evaluated methodology based on the set of relevant features. These authors have selected a set of features according to its relevance for the following issues: concepts, properties, modeling, notation, processes, and pragmatics of the technique.

The features that were evaluated for each topic are the following:

Features about concepts and properties:

Autonomy
Mental attitudes
Proactive
Reactive
Concurrency
Teamwork
Protocols
Situated
Clear concepts

Features about modeling and notation:

Static/dynamic
Syntax defined
Semantic defined
Clear notation
Easy to use
Easy to learn
Different views
Expressiveness
Traceability
Consistency check
Refinement
Modularity
Reuse
Hierarchical modeling

Features about process:

Requirements
Architectural design
Implementation
Testing and debugging
Deployment
Maintenance

Features about pragmatics:

Quality	Used by non-creators
Cost estimation	Domain specific
Management decision	Scalable
Applications	Distributed
Real applications	

The evaluation was carried out by comparing the strengths and weaknesses of each evaluated methodology based on the set of relevant features. In this evaluation, a group of summer students developed the same case study using different methodologies. The students then filled out a questionnaire to give feedback about their experience in understanding and using the methodologies based on the selected features. The authors of this evaluation also collected comments from the authors of the methodologies using the same questionnaire that the summer students had completed. One of the interesting elements of this work is the attempt to eliminate misconceptions by taking into account comments from the authors of each methodology.

One of the main problems of this work is that the evaluation was made using academic case studies developed by students. We consider this to be an important limitation of this work. Academic case studies usually do not reflect the real complexity of real projects in software industries. Also, the different rates of knowledge and experience of student compared with real analysts can affect the results of the evaluation.

2.1.3 Sudeikat research works

Along similar lines, Sudeikat, Braubach, Pokahr, and Lamersdorf (2004) presented an evaluation framework for the evaluation of agent-oriented methodologies that takes platform-specific criteria into account. The specific objective of this study was to determine how the methodologies under evaluation (Mase, Tropos and Prometeus) match up with the Jadex agent platform.

The Sudeikat works place emphasis on developing evaluations that correctly match the methodologies and platforms. Following this approach, it is possible to compare one methodology to many platforms, several methodologies to one specific platform and many-to-many evaluations where several methodologies are associated to several platforms.

The features of the Sudeikat works were separated into four groups: concepts, notation, process and pragmatics:

Features about concept	**Features about notation**
Internal architecture	Usability
Social architecture	Expressiveness
Communication	Refinement
Autonomy	Dependency of models
Pro-activity	Traceability
distribution	Clear definitions
	Modularity
Features about process	**Features about pragmatics:**
Coverage of workflows	Tool support
Management	Connectivity
Complexity	Documentation
Properties of process	Usage in projects

One of the main contributions of Sudeikat´s work is the evaluation of agent methodologies according to specific criteria rather than their analysis in the abstract. Therefore, the result of the evaluation provides more precise information about the strengths and weak points of a modeling technique.

2.1.4 Summary of issues in the evaluation of *i**

The main problem with the current analysis of the *i** framework and its methodological extension is that the evaluations have been developed by computer science students using academic (toy) cases studies. Evaluations in industrial contexts are needed in order to evaluate *i** in practice with real analysts.

2.2 Goal modeling proposals

Traditionally, requirements engineering has been defined as the systematic process of identification and specification of the expected functions of a software system. However, this approach has certain weaknesses. McDermind (McDermind 1994) indicates that when the functional specification of the software system is the focal point of the requirements analysis, requirements engineers tend to establish the scope of the software system before having a clear understanding of the user's real needs. This constitutes a very important reason to explain why many of the systems developed from a requirements model that focuses only on the functionality of the software system do not comply with their correct role within the organization.

It is important to point out that the main objective of an information system is to automate certain tasks or activities in a business process, allowing the organizational actors to reach their particular goals, as well as the general goals of the organization. In this context, there are research works that highlight the importance of using goal modeling as the starting point for the software development process. Goal modeling allows the analyst to create a model that describes the relationship among the strategic objectives of the managers and the specific goals of the enterprise stakeholders. In this context, it is possible to evaluate if the current organizational tasks correspond with the objectives of the enterprise.

The most significant works in Goal-oriented requirements engineering are: a) the Teleological approach (Loucopoulos and Kavakli 1995): a modeling technique for eliciting the organizational setting based on a set of complementary modeling diagrams, b) GBRAM (Anton 1996): a Goal- Based Requirements

Analysis Method to represent the goals in an approach that is less formal but more focused on user needs, c)KAOS (Dardenne, Lamsweerde and Fickas 1993): a formal framework based on temporal logic to elicit and represent the goals that the system software should achieve, and d) EKD (Bubenko and Kirikova 1995): a technique based on the creation of a set of sub-models that provide a different, but complementary, view of the business model.

A brief description of these relevant goal-oriented modeling techniques is presented below.

2.2.1 The Teleological approach for business modeling (Loucopoulos and Kavakli 1995)

This proposal is based on the explicit modeling of the organizational objectives, the social roles and the operations from the Teleological point of view. One of the main premises of this proposal is that an organizational model is relevant if it allows us to provide explanations about the behavior of the enterprise. Teleological proposal establishes the analysis of goals and the analysis of organizational dependencies as the first step for and in-depth understanding of the enterprise. This approach, which has been called teleological, is useful for capturing the reasons that exist behind the business task and also for explaining how a certain activity has been assigned to a specific organizational actor.

The teleological technique is composed of five basic elements: goals, roles, actors, processes and resources. The goals are the core of the modeling process because they provide clear explanations about the current and future configuration of the enterprise. The concept of actor considers people as organizational units and as basis constructs. Processes are the mechanisms that permit changes of states in the organizational system. Finally, resources are the informational or physical means that are produced as result of the business processes.

Teleological approach includes three complementary views for representing an organizational model: the teleological, social, and process views. Each phase is described below:

The Teleological view: the goals of the stakeholders are represented in this view. The goals imply intentions and also represent solutions to the problems of the enterprise. The constraints, which are operational goals, must be formulated in terms of precise properties and actions (Figure 2.1).

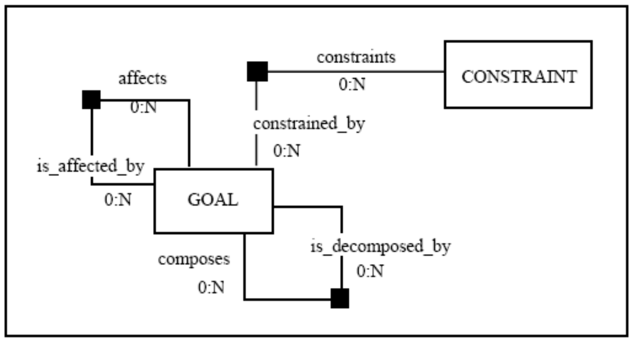


Figure 2.1 The teleological view of the enterprise modeling

Social view: the organizational actors and their interactions are detailed in this view (Figure 2.2). The actor is a key modeling factor since the actor is the entity responsible for executing the organizational activities. An actor can be and individual (person, a software system, etc) or an organizational unit (department, division, section, etc). The roles are a set of processes that are assigned to a specific agent. This assignation is dependent on their goals and capabilities. An actor can play several roles at the same time.

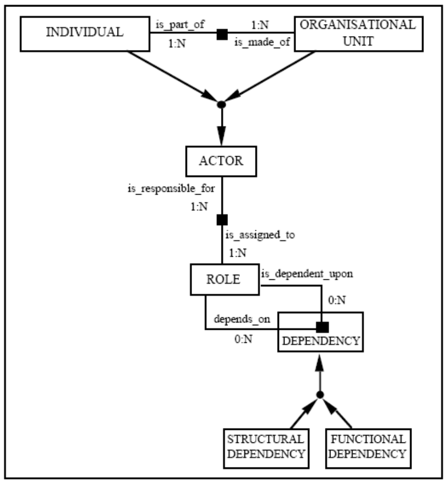


Figure 2.2 Meta-model of the social view

Process view: it provides a general view of the current process in the enterprise (Figure 2.3). This view also considers the resources that are relevant for the execution of the processes. The process view permits the representation of triggers that correspond to changes in the business. The events represent the dynamic dependencies among the processes. The events can be generated by processes or by temporal conditions.

The advantages of this proposal are the following:

- The views of the teleological model can be very useful for constructing an initial set of requirements for either the business model as for the software system.
- The proposal considers a well-defined graphical notation for each business view. The views consider only a small number of modeling elements.
- The technique enables us to define functional and structural dependencies. This characteristic is useful for determining when the tasks of a certain actor influence the execution of tasks of other organizational actors.

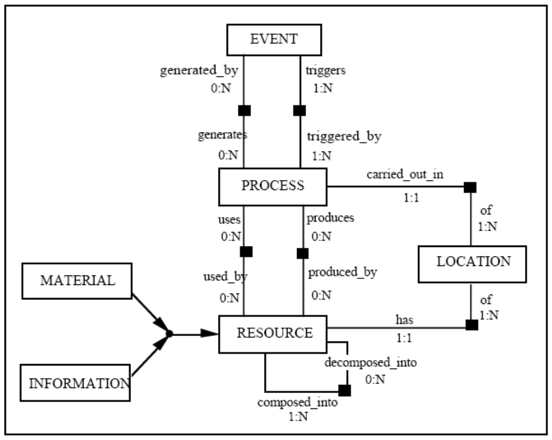


Figure 2.3 Meta-model of the process view

On the other side, the main issues of this proposal are the following:

- Two kinds of analyses must be carried out. The first is the determination of the high-level objectives of the enterprise and their refinement until the operational activities are elicited (prescriptive analysis). The second analysis concerns the details of the operations of the current business processes (descriptive analysis). However, no details are given in order to reconcile the two specifications when there is no precise match between them.
- There is only a brief explanation about goal decomposition. No details are given about conflicting or redundant goals. Also, there is no formal description of the elicited goals, which makes it difficult to validate the goal model.
- Only a brief explanation of the traceability among the different views of the proposal is given.

- There is not an explicit association between the goal model and the process model. This makes it difficult to identify the processes that give support to a specific enterprise goal.
- The complete explanation of the business model implies the analysis of the three models. Therefore, it is not possible to have a unique global view of the current business process, which can be very useful for business process reengineering.

2.2.2 The GBRAM approach for requirements analysis (Anton 1996)

In the GBRAM approach (Goal-Based Requirements Analysis Method), the goals are used as the appropriate mechanisms to identify and justify the requirements of a software system according to the business model.

In this technique, a button-up approach must be followed to elicit the requirements. This is because the goals are obtained from the description of the current processes and also from the descriptions of the stakeholders.

GBRAM is composed of two main processes: goal analysis and goal-refinement (Figure 2.4). These processes are detailed below:

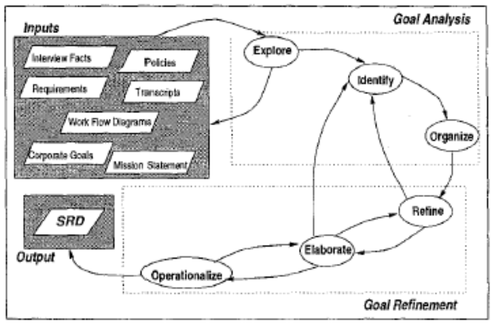


Figure 2.4 GBRAM modeling activities

GBRAM Goal Analysis. Goal analysis is the process of exploring the documentation associated with the enterprise in order to identify and clarify the business goals. This approach places emphasis on the description of the abstract goals and the specific behavior that the stakeholders expect for the system. The main steps of the goal analysis are the following: a) exploration of activities to look for relevant information. b) identification of goals and responsible actors, and c) organization of activities associated with goals according to goal dependencies.

GBRAM Goal Refinement. The objective of this phase is the identification of high-level goals of the enterprise until the level of operational goals is reached. To do this, two kinds of steps must be performed: a) *Elaboration*: this step permits the identification of obstacles to the fulfillment of goals and also permits the identification of restrictions in order to make an operational description of the elicited goals. B) *Refinement*: implies the determination of the pre and post conditions needed to obtain operational goals.

In the GBRAM proposal, the goals are classified into two different kinds: maintenance and achievement goals. The maintenance goals reflect the most abstract objectives of the enterprise. The achievement goals describe actions that prescribe the current behavior of the business processes.

The advantages of this proposal are the following:

- One of the main contributions of this work is the definition of a clear method to elicit the abstract goals in order to define a set of operational goals which will lead the requirements for the information system. This approach makes it possible to define of the reasons for the existence of business activities.
- GBRAM offers appropriate mechanisms to detect redundant goals and to consolidate equivalent goals. The goal restrictions are used as "finishing" mechanisms. This

helps the analyst to determine when the goal-refinement process must end.

- This proposal considers the definition of the pre- and post-conditions needed for goal fulfillment.

On the other side, the main issues of this proposal are the following:

- There is no a formalization of the elicited goals. Therefore, the description of the goals is made in natural language. This is a disadvantage because natural language cannot be used to perform formal verifications or reasoning about the elicited goals.
- This approach does not propose a graphical notation for the proposed goal category. Therefore, the only unique material available to analysts is the natural language goal definition.
- The modeling process of GBRAM ends when the operational goals have been elicited. Therefore, this technique does not offer mechanisms to define a business model that explicitly associates the business process model with the elicited goal model.

2.2.3 KAOS: Goal-based requirements elicitation Dardenne, Lamsweerde and Fickas 1993)

The KAOS approach (Knowledge Acquisition in an automated Specification) is a method for requirements elicitation based on goals, agents and restrictions. These concepts must be presented in a graph where the nodes represent an abstraction (goal, action, restriction, and object) and where the arcs capture the semantic links among these abstractions.

KAOS follows a top-down strategy for the requirements elicitation process starting from the abstract goals and refining them until the operational level is reached.

In KAOS, the goals are non-operational objectives to be reached by an enterprise. This means that the objective cannot be formulated in terms of objects and actions available for a specific agent in the system. A category of goals has been defined in order to guide the analyst in the elicitation process. The goals are classified into five well-established patterns: achieve, cease, maintain, prevent, and optimize. These patterns have a direct impact on the possible behaviors of the system. The aspects regarding behavior generation are analyzed with achievement and cease goals. The maintenance and avoid goals are used for restricting behaviors, and finally, the optimization goals concern comparison behavior.

In this proposal, the actions are mathematical relations over the set of objects of the system. The application of actions defines the state transitions. Also, the specification of the actions must include preconditions, triggers, and postconditions. The pair precondition - postcondition captures the state transitions produced by the application of actions.

The restrictions in this proposal are operational objectives to be reached by the enterprise. The restrictions must be formulated in terms of objects and actions available for a specific agent in the system. Constraints are ensured by restricting existing actions and objects (through strengthened preconditions, invariants, etc.) or through the introduction of new actions and objects.

In KAOS, the agents, events, entities and relations are joined in the category of concerned objects. Figure 2.5 shows a fragment of a KAOS meta-model where the relations among these concepts are defined.

The KAOS approach offers a well-founded set of modeling primitives that allows us to formally specify the requirements of an information system.

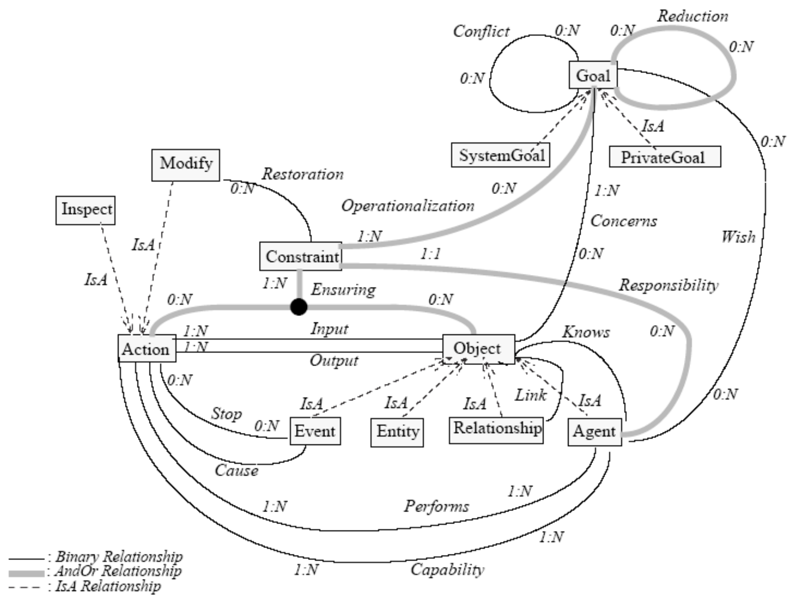


Figure 2.5 A fragment of the KAOS Meta-model (Dardenne, Lamsweerde and Fickas, 1993)

KAOS provides a well-founded method to derive high-level goals into operational restrictions. The main steps for requirements´ elicitation are the following:

Step 1. Identifying goals from initial documents.

Step 2. Formalizing goals and identifying objects.

Step 3. Eliciting new goals through WHY questions.

Step 4. Eliciting new goals through HOW questions.

Step 5. Deriving agent interfaces.

Step 6. Identifying operations.

Step 7. Operationalizing goals.

Step 8. Anticipating obstacles.

Step 9. Handling conflicts

The formal focus of the KAOS approach enables us to make formal analyses of the goal specification before determining the requirements of the information system. This is very useful for reasoning activities.

The advantages of this proposal are the following:

- An exhaustive treatment of the business goals has been carried out in this proposal. The goals can be categorized into different levels of abstraction. This focus enables the analysts to build very complete and precise goal models.
- A specific method for each stage of modeling is presented. This constitutes one of the main advantages of this technique since precise guidelines are provided to build the modeling diagrams.
- All the elements of the KAOS meta-model have a corresponding formalization in temporary logic. This formal-based approach enables us to perform automatic reasoning about the goal specification to detect inconsistencies and redundancies.

On the other side, the main issues of this proposal are the following:

- There is no precise graphical notation that gives support to the KAOS goal analysis method
- There is not a simple view of the set of actions involved in the business processes.
- The top-down strategy of KAOS could cause descriptive models that do not reflect the current organizational setting.
- It is not possible to analyze the complex relationship among the organizational actors.
- KAOS does not consider the definition of an explicit business model to reflect the organizational setting. It only considers the elicitation of requirements until the

operational level is reached. In this context, the technique does not provide well-defined mechanisms for business reengineering.

2.2.4 EKD: Enterprise Modeling (Bubenko and Kirikova 1995)

This technique is based on the creation of a set of sub-models that provide different, but complementary, views of the business model. The starting point of the modeling process is the determination of the goals of the enterprise. A set of complementary models are defined from these goals (in a top-down approach) in order to refine the goals until the description of low-level goals is reached.

The EKD approach proposes six different models (Figure 2.6) for enterprise modeling: the goal model, the business rules model, actor and the resource model, the concept model, the business process model, and the requirements model.

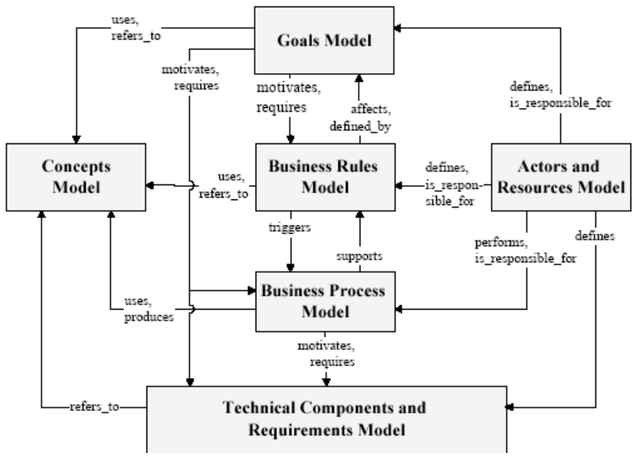


Figure 2.6 The sub-models comprising the enterprise model

Goal model. This model represents the goals of the enterprise. This model permits the identification of relevant properties of the goals such as criticism, priority, relationships, and relevance.

Business rules model. This model is used to represent the set of restrictions that affect the satisfaction of a specific goal of the goal model.

Concept model. This model is used to precisely define the objects and behaviors that are relevant to business processes. In this model, entities, attributes and relationships are represented as concerned objects.

Business process model. This model is a data-flow-like specification used to represent the dynamic aspects of the business model.

Resource and actor model. This model describes the type of relationship among actors and resources identified in the goal and process model.

Requirements model. This model focuses on the elicitation of the requirements for the system-to-be from the previous modeling stages.

The advantages of this proposal are the following:

- The EKD approach, which is based on multiple and complementary views, approaches the modeling process in an incremental way.
- There are well-defined graphical notations for each one of the views that makes up the business model.

On the other side, the main issues of this proposal are the following:

- The proposal only provides a brief definition about the generation of a business goal model. This makes it difficult for inexperienced analysts to differentiate among the several types of goals defined in the proposal: abstract goals, operational goals, etc.

- The goals are defined in natural language. The lack of a formal definition of the goals represented in the model makes the detection of inconsistencies, redundancies and conflicting goals difficult.
- The semantics of the organizational model must be represented using a large number of models, which makes the practical application of this proposal difficult.
- There is not a well-defined method that allows us to derive the general goals of the enterprise from the operational goals of the stakeholders. Only a description of each sub model is presented in the proposal.

2.2.5 Goal reasoning with Tropos (Giorgini et al. 2002)

In this work, Giorgini (Giorgini et al. 2002) presents a formal framework for goal reasoning. Specifically, this work introduces a qualitative and a numerical axiomatization for goal modeling primitives. Label propagation algorithms, which are shown to be sound and complete according to their respective axiomatizations, are also proposed in this work.

The work of Giorgini has been developed in the context of the Tropos methodology (which adopts the *i** modeling concepts). Tropos is an agent-oriented software methodology based on concepts such as actors, goals, and social dependencies.

The advantages of this proposal are the following:

- The main advantage of this approach is the use of a quantification-based approach to evaluate the degree of goal accomplishment. This characteristic enables the analyst to evaluate various alternatives to satisfy the enterprise goals with the highest probability of success.
- This approach offers a well-founded set of axioms for defining goal relationships. It also provides axioms to lead the qualitative and quantitative reasoning with goal models.

- This approach introduces a well-defined goal relationship to indicate both the positive and negative contributions of the satisfaction of a goal to the satisfaction of other goals in the model.

On the other side, the issues of this proposal are the following:

The main disadvantages of the works of Giorgini´s work is that there are no mechanisms to associate the goal structure generated by the application of their technique with the strategic models of the Tropos framework. This is a consequence of the modeling strategy, where the focus is placed on the analysis of the goals in the abstract, without considering the specific actors that are responsible for satisfying the elicited goals. Therefore, for inexperienced analyst in Tropos, it could be complicated to take the design decisions to assign a certain goal to a specific actor in the enterprise.

In two of the above-mentioned research works (GBRAM and KAOS) and in other goal-based approaches such as (Bolchini and Paolini 2002), the software requirements are directly obtained from the operations or restrictions that satisfy the goals. The operations and restrictions are mapped into use case model specifications or into services of the information system-to-be. This approach allows us to carry out the elicitation process at an abstraction level that is closer to the final users. However, business analyses, such as business process reengineering analysis, dependency analysis, and workflow analysis can be carry out that are fundamental to obtaining requirements that reflect the functionality expected by the users of the information system. In this way, a more complete method must consider an intermediate step among goals and requirements of the information system-to-be. However, the transition among these models is not straightforward.

In the other two research works (Teleological approach and EKD) the representation of the business process that supports the enterprise goals is too limited. These descriptions are only based on the procedural aspects of the business processes, which permits a simple view of the processes, but does not enable deeper analysis of the business, such as dependency analysis, roles analysis, vulnerability analysis, performance analysis, etc. Therefore, we consider that these goal-based techniques offer the appropriate mechanism to have a simple view of the enterprise; however, they do no offer the needed mechanisms to carry out deeper business analysis, such as business process reengineering.

2.3 The service-oriented proposals

Service-oriented computing (SOC) is one of the fastest emerging paradigms in software development today. Service-oriented mechanisms have been the dominating technology for the next years.

One of the definitions that reflects the current perception about services associates them with autonomous platform-independent computational elements that can be described, published, discovered, orchestrated and programmed using XML artifacts for the purpose of developing massively distributed interoperable applications. Although this definition represents the current state of service technology, it is only a partial view of the potential of service-oriented computing for characterizing the static and dynamic semantics of different application domains.

In this sense, several definitions of services have been given according to the domain context where the service is used as a representation mechanism. Further on, we present the concept of services at the implementation, conceptual and organizational levels.

2.3.1 Services at the implementation level

In this level, we have found the well-established technology for Web Services. Web services are the appropriate mechanisms for implementing e-services. The definition of web service provided by the Stencil Group seems to be one of the most representative of what a web service is: web services are “loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols” (Stencil Group 2001).

This application domain presents the Web Service Definition Language defined by the World Wide Web Consortium (W3C Working Group 2004) a standard reference about Web Services.

According to the schema proposed by the W3C, the definition of a Web service must include the following components:

- message construction (envelope, header, body)
- message exchange patterns (MEP)
- processing model for messaging: originator, intermediaries, destination
- extensibility mechanism
- fault system
- bindings to transport protocols (HTTP, SMTP, ...)
- message(s) accepted and emitted: abstract description (XML Schema)
- network protocol(s) and message format(s)
- operation: exchange of messages
- port type: collection of operations
- port: implementation of a port type
- service: collection of ports

Some of the technological problems that lead the web services are the following:

- message structure and infrastructure
- describing what messages a service accepts and sends

- routing messages
- describing message exchange patterns (choreography, workflow)
- finding services to exchange messages with
- security, authorization, access control
- transactions (succeed or rollback)
- asynchronous messaging (needs reliable messaging)
- caching and cache control
- correlation: tying messages together into a sequence

As commented above, there is a consolidated technology for each and every possible aspect of Web Services.

2.3.2 Services at the conceptual modeling level

In addition to the consolidation of the technology for implementing web services, there is an emerging set of modeling techniques for characterizing the compositional aspects of the service integration, and also for the definition of transactional properties that must be defined on top of the basic web service standards. In this context, several emerging standards, such as BPEL4WS: Business Process Execution Language for Web Services (Andrews et al. 2005), propose modeling mechanisms to represent the services as an entity in a conceptual model that gives a more abstract view of the problem domain.

The BPEL4WS is an XML-based standard for defining how you can combine Web services to implement business processes. It builds upon the Web Services Definition Language (WSDL) and XML Schema Definition (XSD). Thus, the BPEL4WS extends the Web Services interaction model and enables it to support business transactions.

BPEL4WS is said to be a modeling technique in the organizational level; however, we argue that this is an inappropriate viewpoint. In fact, almost all the definitions about this emerging standard suggest that BPEL4WS offers semantics for specifying business process behaviors. However, BPEL4WS works with elements that are essentially software. In this context, BPEL4WS models the composite of element software components to fit the organizational structure. BPEL4WS does not characterize the enterprise itself, but it models the appropriate configuration of software services that give support to the organizational structure.

A real technique for business process modeling must offer mechanisms for describing actors and describing organizational chart that organizes the actors hierarchically; it must offer mechanisms for representing the chain value, the dependencies among actors, the resources produced by the organizational tasks, etc. The specifications of BPEL4WS lack of this semantics and only represent the organizational activities with a one-to-one correspondence with web services. In this sense, the following definition puts BPEL4WS in the context of software artifacts modeling: "The BPEL4WS process itself is basically a flow-chart like expression of an algorithm. Each step in the process is called an activity. There is a collection of primitive activities: invoking an operation on some Web service (<invoke>), waiting for a message to operation of the service's interface to be invoked by someone externally (<receive>), generating the response of an input/output operation (<reply>), waiting for some time (<wait>), copying data from one place to another (<assign>), indicating that something went wrong (<throw>), terminating the entire service instance (<terminate>), or doing nothing (<empty>)." (Sanjiva and Curbera 2002)

In this context, we can establish that the modeling task in BPEL4WS corresponds to an intermediate level between the organizational model and the implementation level.

BPEL4WS provides an XML notation and semantics for specifying business process behavior based on Web Services. A BPEL4WS process is defined in terms of its interactions with partners. A partner may provide services to the process, require services from the process, or participate in a two-way interaction with the process. Thus, BPEL orchestrates Web Services by specifying the order in which it is meaningful to call a collection of services, and assigns responsibilities for each of the services to partners. (Mantell 2005)

According to the schema proposed by BPEL4WS, the definition of a service composition must include the following components:

- Definition of abstract processes.
- Sequencing of process activities, especially Web Service interactions.
- Correlation of messages and process instances.
- Recovery behavior in case of failures and exceptional conditions.
- Bilateral Web Service based relationships between process roles.

In a situation similar to the definition of services in the implementation level, the technology for defining services in the conceptual modeling level is currently in a consolidation stage. The definition of emerging standards makes it possible to consider a possible consolidation of this technology in the following years.

2.3.3 Services at the organizational modeling level:

This is the most rapidly emerging research field in service-oriented modeling. The focus of this phase consists of the definition of the services that are offered by an enterprise. In contrast to the definition of service in the conceptual or implementation level, the definition of services at the organizational level does not necessarily imply the definition of a software system that gives support to organizational tasks. Therefore, in these modeling phases, the focus is placed on the definition of abstract functionalities provided (manual or automatically) by a supplier to potential customers. This specification, which reflects the current situation of an enterprise, must be the source for the generation of software services that give support to the organizational activities. At the organizational modeling level we also found a scarcity of methods or mechanisms to model the enterprise following a service-oriented approach. The main contribution of this thesis is to provide more powerful and useful mechanisms to support services at the organizational level. Following, we present some of the more relevant works in this area.

One of the few existing proposals is *On demand Business Service Architecture* (Cherbakov et al. 2005). In this proposal, the authors explore the impact of service orientation at the business level. The services represent functionalities offered by the enterprise to the customers. It considers the definition of complex services composed of low-level services.

One of the contributions of the Cherbakov work is that the services are represented from the customer point of view. One of the main weaknesses is the lack of mechanisms to model the complex internal behavior needed to satisfy the business services. The services are represented as “black boxes” where the internal details of the implementation of each service are not represented. This makes difficult to apply the technique to address business model reengineering tasks, which are mainly based on the operational aspects of the business processes. Figure 2.7 presents an example of the definition of complex services that are redefined into more concrete atomic services.

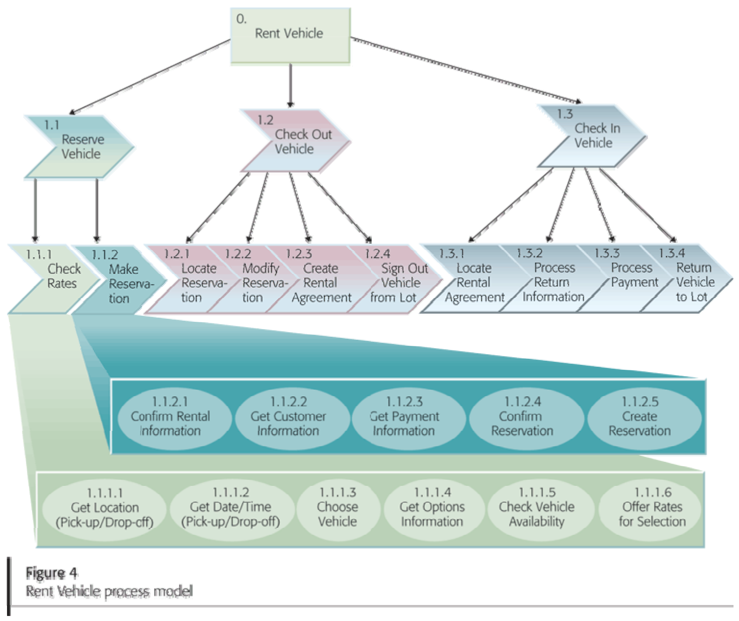

0. Rent Vehicle
1.1 Reserve Vehicle
1.2 Check Out Vehicle
1.3 Check In Vehicle
1.1.1 Check Rates
1.1.2 Make Reserva-tion
1.2.1 Locate Reserva-tion
1.2.2 Modify Reserva-tion
1.2.3 Create Rental Agreement
1.2.4 Sign Out Vehicle from Lot
1.3.1 Locate Rental Agreement
1.3.2 Process Return Information
1.3.3 Process Payment
1.3.4 Return Vehicle to Lot
1.1.2.1 Confirm Rental Information
1.1.2.2 Get Customer Information
1.1.2.3 Get Payment Information
1.1.2.4 Confirm Reservation
1.1.2.5 Create Reservation
1.1.1.1 Get Location (Pick-up/Drop-off)
1.1.1.2 Get Date/Time (Pick-up/Drop-off)
1.1.1.3 Choose Vehicle
1.1.1.4 Get Options Information
1.1.1.5 Check Vehicle Availability
1.1.1.6 Offer Rates for Selection
Figure 4
Rent Vehicle process model

Figure 2.7 Definition of complex enterprise services (Cherbakov et al, 2005)

Another example of the use of services at business level is the proposal of Software-aided Service Bundling (Baida 2006). The main contribution of this research work is the definition of an ontology –a formalized conceptual model– of services to develop software for service bundling. A service bundle consists of elementary services, where service providers can offer service bundles via the Internet. The ontology describes services from a business value perspective. Therefore, the services are described by the exchange of economic values between suppliers and customers rather than describing services by physical properties.

The *service value* perspective is a demand-side, customer perspective. It describes the service from the point of view of the customers in terms of their needs and demands, their quality descriptors and their acceptable sacrifice, in return for obtaining the service (including price, but also intangible expenses such as inconvenience costs and access time) (Baida 2006).

The modeling language proposed in this work places emphasis on the satisfaction of complex customer needs through complex services. Complex services are composed of several elementary services that are packaged in order to provide a value to the final customers. Figure 2.8 presents different examples of schemas of the service bundling approach.

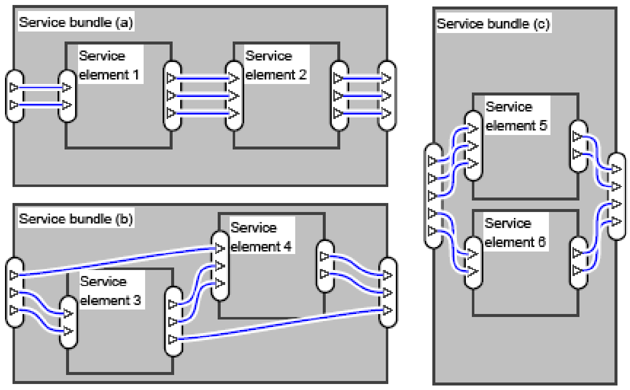


Figure 2.8 Different configurations of service bundling (Baida 2006)

One industrial version of the software-aided service bundling is e3value (Gordijn and Akkermans 2001). It has been developed to put into practice the concept of services bundling. Some practical cases have been developed to demonstrate the advantages of this proposal (Gordijn and Akkermans 2003).

This modeling technique shares the same problem as the proposal of *on demand business service*. The services are defined as black boxes, where the main focus is on the definition of the set of input and outputs of the service. This has been done in order to make service bundling possible by matching the inputs and outputs of the services to be composed.

One of the main consequences of not having mechanisms to describe the internal behavior of the services is that it is impossible to relate the services offered with the strategic objectives of the enterprise. Therefore, it could be difficult to define the alternative services that better satisfy the goals of the enterprise.

No matter what the services are analyzed in, in all cases there is a strong dependency between the concept of services and the concept of business functionalities. However, this key aspect of service modeling has been historically neglected in the literature. At present, there is only a partial solution to the problem of representing services at the organizational level, in the same way as the services are perceived by the final customers.

2.3.4 The *i** proposals for representing services

The *i** framework, and its methodological extension, Tropos, have been used in several research works related to service-oriented computing. Following, we present the most representative works in this area.

Lau and Mylopoulos (Lau and Mylopoulos 2004) propose a design methodology for Web services with Tropos. This work uses goals to determine the space of alternative solutions to satisfy these goals, where the solutions are represented by web services. The generated web services are expected to accommodate as many of those solutions as possible. This proposed design methodology supports early and late requirements as well as architectural and detailed design. One of the main issues of this work is the use of the "pure" *i** concepts to capture the requirements for the web services. The use of the original definition of the *i** concepts raises to non-service-oriented description models, so the mapping between the Tropos social view of the modeling phases with Tropos and the web service description is not straightforward analysts who are inexperienced in Tropos.

The research works by Colombo (Colombo, Mylopoulos and Spoletini 2005) presents a methodological framework that supports the modeling and formal analysis of service composition. In this work, the *i** framework has been extended with a complementary process perspective. Specifically, the *i** concepts have been used to create a social specification of a service composition. This social view enables the analyst to represent the specification of market actors and dependencies among them. This view also permits the refinement of business relationships. This kind of analysis is developed using the "pure" *i** notation. Later on, the *i** specification is used to define an operational view of the enterprise process. This is done in order to perform an operationalization of the intentional elements and also to be sure that the process model composition complies with both the social model and the policies that restrict the composition. As in the case of Lau and Mylopoulos work (Lau and Mylopoulos 2004), the main issue of this Colombo work is the use of a non-service-oriented version of *i** for the initial elicitation tasks.

Finally, in Kazhamiakin (Kazhamiakin, Pistore and Roveri 2004) has proposed changes to the *i** notation in order to generate web services. The authors propose a methodology for business requirements modeling that use the Tropos framework to capture the strategic goals of the enterprise. This method enables the analyst to produce a concrete BPEL4WS description based on the abstract description of business process with the *i** concepts.

The modifications made to the *i** framework consist in the definition of separate layers to represent the strategic, activity and message levels. Thus, the activities support the goal fulfillment. However, the strategic model is used only for analysis and design purposes because it is not possible to map the goals with any implementation primitive for the generated web services. This is the reason why, only the activity and message levels are used to generate the web service specification. Even though the proposed modifications extend the capabilities of *i** to represent the process needed to satisfy the enterprise objectives in a complementary view, the analyst still does not have a complete notation and a technique to elicit the organization requirements in a service-oriented approach. Therefore, there is still an abrupt transition from a non-service-oriented business model to a complete service-oriented web service description.

The analysis of different proposals to address services with *i** have revealed the need to give the *i** modeling framework a service-oriented orientation in order to permit a softer transition between the enterprise modeling phase and web service modeling.

2.4 Final Considerations

In this Chapter, the relevant works in areas of interest in this thesis has been pointed out. First we present the research works in evaluation of the *i** framework: Sheory and Sturm works, Dam and Winikoff works and Sudeikat works. We pointed out the differences of these works with the research presented in this thesis.

We have analyzed the most influent works in goal modeling: Teleological approach, GBRAM, KAOS and EKD. We have pointed out in the advantages and disadvantages of these research works.

Finally, we have presented the proposals to represent services at the different abstraction levels, implementation, conceptual and business level.

Chapter 3

3. The Empirical Evaluation of the *i** Framework

This section introduces the empirical evaluation of the *i** framework, focusing on the strengths and weaknesses of *i** in industrial case studies of a software development company that uses a Case tool for automatic software generation. The evaluation was supported by an evaluation framework that considers relevant features to be measured in practical experimentation.

3.1 Introduction

One of the main contributions of this Chapter is the description of an empirical evaluation of the *i** Framework for real projects in a software development company that uses model-driven tools for software development. The objective of the evaluation was to accurately detect the strengths and weaknesses of the *i** Framework in practice and to provide recommended solutions for the issues that were detected. The evaluation framework has been designed keeping in mind that it is to be used within model-based software development environments with analysts who have no previous knowledge of *i**. Another contribution of this work is the definition of a consensus to explain the reason for assigning a certain value to the analyzed issues. Finally, the last contribution of the Chapter is the definition of a set of conclusions to be considered in the definition of future versions of *i**.

The empirical evaluation of *i** was conducted in Care Technologies Inc. (http://www.care-t.com), a software development Company that uses OO-Method (a well-founded Model Transformation and Conceptual Schema Centric Case Tool

(Pastor et al. 2001)) to automatically generate complete information systems from object-oriented conceptual models.

Following, a brief explanation of the *i** Framework is presented that details the modeling primitives and the modeling diagrams.

3.2 An overview of the *i** Framework

The *i** framework (Yu 95) is a language for supporting goal-oriented modeling and reasoning of requirements. The *i** modeling framework views organizational models as networks of social actors that have freedom of action, and depend on each other to achieve their objectives and goals, carry out their tasks, and obtain needed resources.

The *i** modeling is different that traditional mechanisms to requirements specification that are based on the description of what must be done in order to accomplish an organizational process. *i** is well equipped to exposing why business processes are executed in a specific way, and also it permits the explicit representation of the space of alternatives that exist for fulfilling a business goal. The *i** Framework permits omitting the operational details of the processes by reducing the complexity of the business model This allows us to have a high level representation of the current enterprise situation. This abstraction level is also useful to make analysis of the future enterprise situation.

3.2.1 The *i** modeling primitives

The modeling primitives of *i** are the following: actor, goal, task, resource and dependency (Asnar et al. 2006).

An ***actor*** represents an entity that has strategic goals and intentionality within the system or the organizational setting. In *i** the concept of actor can be specialized into agent, role and position. An agent represents a specific instance of the actor's class. Therefore, the agent represents a physical actor. The role represents and abstract characterization of the behavior of a social actor within a specific context. A position represents a set of roles. Thus, an actor can be instantiated into a specific agent, this agent can occupy a certain position, where it can play different roles according to its position.

A ***goal*** represents the strategic interest of a business actor. In *i** we distinguish hard goals from softgoals. How the goal is to be achieved is not specified, allowing alternatives to be considered. A goal can be either a business goal or a system goal. A softgoal represent a goal where fulfillment conditions can be clearly established. Softgoals are typically used to model non-functional requirements.

Task specifies a particular course of action that produces a desired effect. Tasks can also be seen as the solutions in the target system that allow, totally or partially, fulfilling a goal. These solutions provide operations, processes, data representations, structuring, constraints and agents in the target system to meet the needs stated in the goals and softgoals. The execution of a task can be a means for satisfying a goal or a softgoal.

A ***resource*** represents a physical or informational entity.

A ***dependency*** between two actors indicates that one actor depends, for some reason, on the other in order to attain some goal, execute some task, satisfy a softgoal, deliver a resource, or provide a service. The former is called *depender* and the latter is called the *dependee*. The object around which the dependency centers is called the *dependee*, which can be a goal, resource or plan.

3.2.2 The *i** modeling diagrams

The *i** Framework is made up of two models that complement each other: the *strategic dependency model* for describing the network of inter-dependencies among actors, as well as the *strategic rationale model* for describing and supporting the reasoning that each actor goes through concerning its dependencies on other actors. These models have been formalized using intentional concepts from Artificial Intelligence, such as *goal*, *belief*, *ability*, and *commitment*.

A **strategic dependency model** (SD) is a graph involving *actors* who have strategic dependencies among each other. A dependency describes an "agreement" (called *dependum*) between two actors: the *depender* and the *dependee*. Dependencies have the form *depender* → *dependum* → *dependee*.

The type of the dependency describes the nature of the agreement: goal, task, resource and softgoal dependencies (Grau, Horkoff and Yu 2006).

- In goal dependency the *depender* depends on the *dependee* to bring about a certain state of affairs in the world. In goal dependency, all decisions about fulfilling the goal need to be taken by the *dependee*, therefore, the *depender* doesn't care about how the *dependee* goes about achieving the goal. In goal dependency, the *depender* delegates the responsibility for fulfilling the goal to the *dependee*, who is the new goal owner.
- In softgoal dependency a *depender* depends on the *dependee* to satisfy a non functional requirement. Softgoal are similar that goal dependencies, but in the case of the former, the fulfillment conditions cannot be precisely defined (e.g., because it is subjective and/or partial).
- In task dependency a *depender* depends on the *dependee* to execute a given activity. The *depender* is the actor that prescribes the procedure to execute the delegated task, in this sense; the *dependee* has already made decisions about how the task needs to be carried out.
- In resource dependency a *depender* depends on the *dependee* to provide a resource. In resource dependency

- we assume that there are no open issues to be addressed about resource production or resource delivery.

In *i** diagrams, actors are represented as circles; goals, softgoals, tasks and resources are respectively represented as ovals, clouds, hexagons and rectangles. In the SD model, the internal goals, plans, and resources of an actor are not explicitly modeled. The SD model is focused on representing the external relationships among actors.

The **strategic rationale model** (SR) is a graph that focuses on providing a representational structure for expressing the rationales behind dependencies. The key idea of this model is the representation of the actors behaviors needed to satisfy each actor dependency.

The strategic rationale model is a graph with four types of nodes (goal, task, resource, and softgoal) and three types of internal links to the *i** actor (means end links, task decomposition links and contribution links). Following, a definition for these modeling primitives is presented based on the definitions of Grau (Grau, Horkoff and Yu 2006).

- **Goal**: represents an intentional desire of an actor. The name of the goal represents the desired state of affairs.
- **Task**: represents the desire of an actor to accomplish some specific task, performed in a specific way.
- **Resource**: represents information or information entities produced as a result of the organizational tasks.
- **Softgoal**: represents the quality attributes that the enterprise wants to fulfill by implementing a business process. The means to satisfy such goals are described using contribution links from the other modeling element.
- **Means-end links** represent the space of alternative ways to satisfy a goal. The end is represented as a goal and the means are represented by using the concept of task.

- **Decomposition links** represent necessary elements needed to satisfy a task. Four alternatives exist to apply decomposition links: task-goal decomposition, task-task decomposition, task-resource decomposition and finally, task-softgoal decomposition.
- **Contribution links** represent the positive and negative contributions to a different degree of goals/tasks

3.3 The context of the empirical evaluation

As commented before, the empirical analysis of *i** was made in the context of OO-Method, which can be viewed as a Computer-Aided Requirements Engineering (CARE) environment where the focus is placed on properly capturing the system requirements in order to manage the whole software production process. This is in contrast to the more conventional CASE - Computer-Aided Software Engineering- environments, where the correct representation of requirements is not the basic issue. OO-Method follows a model-driven development approach (MDD) to generate complete information systems based on the information contained in a conceptual model.

The key feature of OO-Method is the integration of formal specification techniques with conventional object-oriented modeling techniques. The main advantage of this is that the models are built using concepts that are much closer to the problem space domain. In addition, this integration avoids the complexity associated with the use of formal methods.

In a MDD approach, two main aspects must be clearly stated: which *conceptual modeling patterns* are provided by the method and which *notation* is provided to properly capture those conceptual modeling patterns. Regarding to ***conceptual modeling patterns***, OO-Method has adopted the well-known OMT strategy by dividing the conceptual modeling process into three complementary views: the object view, the dynamic view, and the functional model view (adding a fourth views to specify presentation patterns). When software engineers are specifying the system, what they are doing is capturing a formal specification of the system according to the *OASIS* formal specifications language (Pastor et al. 2001). This feature allows the introduction of a well-defined expressiveness in the specifications, which is often lacking in conventional methodologies.

The use of such a formal specification provides the context to validate the system in the *problem space*, obtaining a software product that is functionally equivalent to the specifications. This equivalence is achieved by creating a model compiler that implements all the mappings specified between the conceptual patterns that represent what the system is (*problem space* level) and their software representations (at the *solution space* level). The execution model is based on the idea of transforming a set of precise conceptual modeling constructs into their associated, concrete software representations. The implementation of the corresponding set of mappings between conceptual constructs and software representations constitutes the core of a Conceptual Model Compiler. The OO-Method approach provides a well-defined software representation of the required representations in the solution space. A concrete execution model based on a component-based architecture has been introduced to deal with the peculiarities of component-based systems. Naturally, we have had to introduce relevant information to address specific features of *OASIS* in these diagrams (Object Model, Dynamic Model, Functional Model, and Presentation Model). Nevertheless, this is always done preserving the external view that is compliant with the most extended modeling notation, which is the UML.

Hence, the subset of UML used in OO-Method is the one that is necessary to complete the information that is relevant for filling a class definition in *OASIS.* This specification constitutes a high-level data dictionary, which is the input for the final model transformation process that creates the software product. In this way, the arid formalism is hidden from the modelers when is describing the system by making it more comfortable to use a conventional notation. Another main objective in the design of OO-Method was to keep, modelers from having to learn another graphical notation in order to model an information system. Having a formal basis allows us to provide a modeling environment where the set of needed diagrams is clearly established.

The OO-Method model transformation process from problem space concepts to solution space representations opens the door to the generation of executable software components in an automated way. Taken together, these software components constitute a software product that is functionally equivalent to the requirements specification collected in the conceptual modeling step. A graphical representation of the strategy of the OO-Method approach is presented in Figure 3.1.

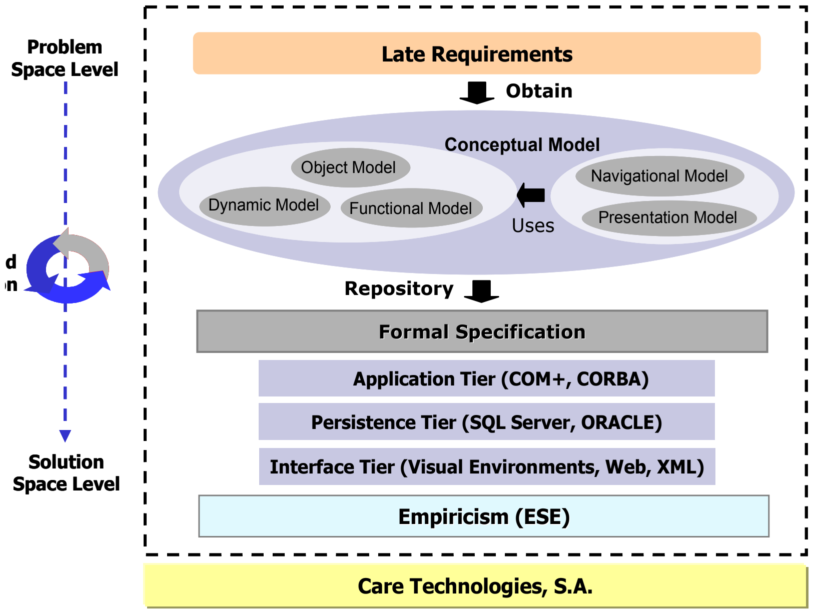


Figure 3.1 The OO-Method Approach for Model-Driven Software Development

Further on, we briefly introduce the four conceptual model views that exist in the OO-Method approach.

3.3.1 Object Model

The object model is a graphical model where system **classes** and **relationships** (association, aggregations, and inheritance) are defined. Additionally, **agent relationships** are specified to state the services that objects of a class are allowed to activate. These primitives capture the static point of view of the system. The corresponding UML-based diagram is the Class Diagram, where the additional expressiveness is introduced by defining the corresponding stereotypes.

3.3.2 Dynamic Model

The system class architecture has been specified using the Object Model. Additionally, basic features (such as which object life cycles can be considered valid and which inter-object communication can be established) have to be introduced in the system specification. To do this, OO-Method provides a dynamic model. It uses two kinds of diagrams: State Transition Diagrams and Interaction Diagrams.

The **State Transition Diagram (STD)** is used to describe correct behavior by establishing valid object life cycles for every class. By valid life, we mean an appropriate sequence of service occurrences that characterizes the correct behavior of the objects that belong to a specific class. The corresponding UML based diagram is the State Diagram.

The **Interaction Diagram (ID)** specifies the inter-object communication. We define two basic interactions: **triggers**, which are object services that are activated in an automated way when a condition is satisfied, and **global interactions,** which are transactions involving services of different objects. The corresponding UML base diagram is the Collaboration Diagram where the context of the interaction is not shown.

3.3.3 Functional Model

A correct functional specification is a shortcut of many of the most extended OO Methods. Sometimes, the model used breaks the homogeneity of the OO models, as it happened with the initial versions of OMT, which proposed using the structured DFDs as a functional model. The use of DFD techniques in an object-modeling context has been criticized for being imprecise, mainly because it offers a perspective of the system (the functional perspective) that differs from the other models (the object perspective).

Other methods leave the free-specification of the system operations in the hands of the designer. The OO-Method functional model (FM) is quite different from these conventional approaches. In this model, the semantics associated with any change of an object state is captured as a consequence of a service occurrence. To do this, it is declaratively specified how the services change the object state depending on the arguments of the service involved and object's current state. A clear and simple strategy is given for dealing with the introduction of the necessary information. The relevant contribution of this functional model is the concept of *categorized attributes*.

3.3.4 Presentation Model

The object's society structure, behavior, and functionally are specified using the three conceptual models described above. The last step is to specify how users will interact with the system.

This is done by the Presentation Model through the definition of a set of Presentation Patterns. The Presentation Patterns capture the information required to characterize what appearance the application will have, and how the user will interact with the application.

Despite the major advantage of the OO-Method in automatically generating information systems, there are disadvantages as well. Specifically, there are currently no mechanisms for acquiring the requirements of an information system. Accordingly, the next step in developing further the OO-Method consists of adding a new phase of organizational modeling as a starting point to determine the correct requirements for the information system-to-be.

In performing the empirical evaluation, our objective was to determine possible extensions to *i** that would make it suitable for inclusion in the OO-Method modeling and methodological framework. The main idea of this approach is the generation of a modeling process that uses the intentional and social characteristics of *i** to determine the correct requirements for the information system-to-be. There are some preliminary results of this approach in (Martinez, Castro, Pastor, and Estrada 2003), (Estrada, Martinez and Pastor 2003). Consequently, the selected features for measurement in this empirical evaluation are inspired by model-driven approaches.

3.4 The contribution of the empirical evaluation

Our empirical evaluation is different than those performed by Shehory, Dam and Sudeikat (Sturm, Dori, and Shehory 2005). Our evaluation focuses on an in-depth analysis of a specific methodology (*i** Framework) rather than a comparison of several.

Moreover, our evaluation approach is also different from the one presented in Sudeikat´s works (Sudeikat et al. 2004), because our evaluation studies how well *i** matches a specific software development context (model-based software generation) in practice, rather than analyze *i** in the abstract.

Moreover, other evaluations of agent-oriented methodologies (including Tropos), involve academic case studies developed by students. This represents a major limitation of these studies (because students are novices, rather than professional analysts) and a major point of difference from our work.

There are also reported studies that use *i** for some application. In most of these studies, the modelers were well-acquainted with *i** concepts and their use. We have detected scarcity of experiments where *i** is evaluated in practice by modelers who are not used to working with *i** and who do not perform organizational modeling as a current task in their modeling activities. This Chapter presents such a practical evaluation that fills this gap.

3.5 Type of empirical evaluation

The empirical study of *i** was based on a feature-based framework. Such a framework consists of a set of features that can be properties, qualities, attributes, or characteristics. These features can describe the evaluated methodology well enough so that it can be assessed for a particular purpose (Dam 2003). The evaluation was conducted by *evaluators* who assigned a judgment (*value*) of how well each *feature* was supported by the *subject* of the evaluation. For our study, the features were selected on the basis of their relevance to model-driven software generation.

The feature-based evaluations can be useful for assessing how much support a methodology appears to provide for a specific domain. This is done by selecting features that are relevant to the application domain of interest and evaluating the methodology against this set of relevant features. Therefore, this kind of evaluation is appropriated for the objective of our research work, because we tried to evaluate a specific set of relevant features in the context of the Model-Driven Software Generation approach of the Care Technologies Enterprise, rather than making an evaluation of an extensive list of features.

3.6 The population background

The empirical evaluation was implemented using three real-life projects that were developed in parallel by three different development teams. The composition of the development teams was as follows:

Team 1: Three experts in requirements engineering. These analysts were experts in the use of advanced tools for generating conceptual schemas from requirements models, with a high degree of automation in the corresponding transformation process. At the beginning of the evaluation, this team had limited knowledge of *i**.

Team 2: Three experts in programming. These analysts were experts in the use of the CASE tool for automatically generating information systems from conceptual models. At the beginning of the evaluation, this team had no knowledge of *i**.

Team 3: Two experts in *i** Modeling. These analysts were experts in the use of *i** for organizational modeling.

In our evaluation, which took 9 months, the case studies were conducted in isolation, i.e., with no exchange of information among participant teams. This was done in order to avoid the empirical analysis being affected by the different levels of knowledge about *i** by the teams involved.

3.7 Evaluation design

The empirical evaluation of the *i** Framework was conducted in five steps:

The first phase of the empirical evaluation consisted of the determination of a set of relevant issues to be measured in the empirical evaluation. The relevance of the issues was given by the Model-Based transformational approach of the Company where the analysis was developed.

The second step consisted of training the three teams, where details about the concepts and proper use of *i** were given out, using original *i** sources and basic teaching support.

The third phase of the empirical evaluation consisted in the use of the *i** Framework to develop the selected case studies.

The fourth phase consisted in the evaluation of the results of each team. To accomplish this, each participating team evaluated *i** for each relevant feature.

The fifth phase consisted in analyzing the results and drawing conclusions about the strengths and weaknesses of *i**.

Figure 3.2 presents a graphical representation of the strategy selected to perform the empirical evaluation.

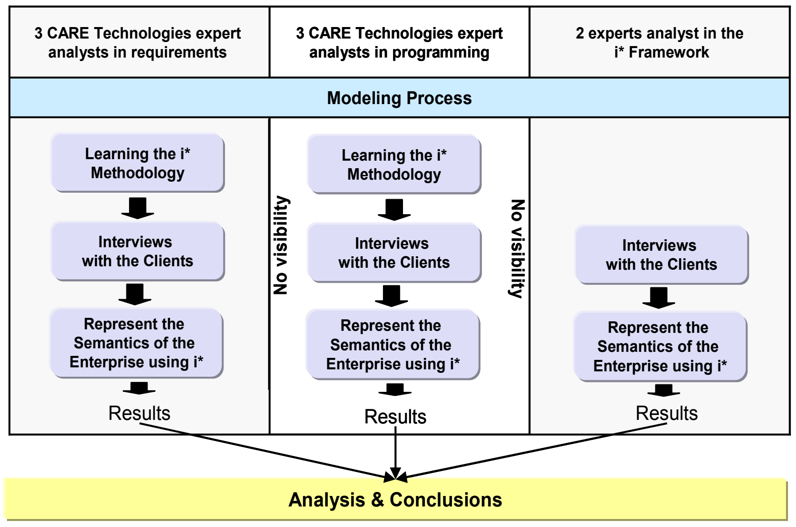

3 CARE Technologies expert analysts in requirements
3 CARE Technologies expert analysts in programming
2 experts analyst in the i* Framework
Modeling Process
Learning the i* Methodology
Interviews with the Clients
Represent the Semantics of the Enterprise using i*
Results
No visibility
Learning the i* Methodology
Interviews with the Clients
Represent the Semantics of the Enterprise using i*
Results
No visibility
Interviews with the Clients
Represent the Semantics of the Enterprise using i*
Results
Analysis & Conclusions

Figure 3.2 The strategy for the empirical analysis

3.8 The selected case studies

As mentioned above, the case studies are real projects of the Care Technology Company. Next, we briefly describe the case studies that were analyzed.

- Technical Meeting Management. This case study implied the modeling of the processes associated with review

- papers for a technical meeting as well as the processes to manage the operative aspects of the organization of the meeting.
- Golf Tournament Management. This project implied the modeling of the business processes for organizing Golf Tournaments validated by the Spanish Golf Federation. The case study included the processes for registering golfers, creating matches, assigning controllers to specific holes of the golf course as well as for obtaining and publishing partial and final results for each match.
- Car Rental Management. This project dealt with the modeling of the process for a car rental company in Alicante, Spain. The case study included the processes for renting cars and additional services as well as for buying new cars for the Rental Company.

The goal of the development teams was to represent relevant business processes for each project using *i**. For the Technical meeting management case study, the organizational environment involves a large number of interactions among participant actors, and a relatively small number of actors´ internal elements. For the Golf tournament management case study, the organizational environment concerns a large number of actors´ internal activities and a small number of actor interactions. On the other hand, the Car rental management case study involves an organizational context with a large number of actors' internal activities and actors' interactions. As such, the case studies had rather different organizational characteristics and ensured that our study would be biased because of similarities in the case studies chosen.

3.9 The evaluation framework

The empirical evaluation of *i** was based on a set of features that have been considered highly relevant in the context of a model-based software development environment. In this specific context, the modeling primitives of a model must provide precise, bidirectional traceability with subsequent stages of the modeling process. It is important to note that the experiment was designed for practicing analysts who are used to dealing with software production concepts such as model-driven architectures, code generation, object-oriented analysis and late (conventional) software requirements specifications, rather than analysts who are familiar with early requirements. After all, we expect that this will be the normal scenario for *i** use in software production companies. Therefore the determination of relevant features for the study was perhaps the most critical step in the whole evaluation process.

In order to assure the correct selection of those criteria to be evaluated, we based our evaluation on relevant features that have been proposed in the literature to evaluate agent-oriented methodologies. Specifically, to evaluate the *i** Framework, we based our framework on proposals from (Padgham et al. 2005) (Sturm and Shehory 2003) (Dam 2003), and (Dam and Winikoff 2003) to compare agent-oriented Methodologies. By including features used in three different studies, we have tried to avoid biases that arise from using a single set of features that might be well suited for *i**.

The empirical evaluation considered two main aspects of the *i** Framework: a) Modeling Language (Refinement, Modularity, Repeatability, Complexity Management, Expressiveness, Traceability, and Reusability) and b) Pragmatics of the Modeling Method (Scalability and Domain Applicability). The features selected for these aspects are listed below.

- **Refinement**: This feature measures the capability of the modeling method to refine a model gradually through stages until the most detailed view is reached (Bergenti, Gleizes and Zambonelli 2004). This is a relevant feature because it allows analysts to develop and fine-tune design artifacts at different levels of granularity during the development process (Dam and Winikoff 2003).

- **Modularity**: the degree to which the modeling language offers well-defined building blocks for building model. The building blocks should allow the encapsulation of internal structures of the model in a concrete modeling construct. This characteristic ensures that changes in one part of the model will not have to be propagated to other parts.
- **Repeatability**: the degree to which the modeling technique generates the same output (i.e., same models), given the same problem. This is a very relevant feature in the context of model-driven approaches, where each modeling element during a specific step of the modeling process corresponds to a modeling element in subsequent steps. Repeatability ensures that a correct result is obtained when a transformation between models is applied. We use this feature to evaluate whether we obtain the same *i** model when the same domain is modeled by different modelers.
- **Complexity Management**: This feature measures the capability of the modeling method to provide a hierarchical structure for its models, constructs and concepts. Model management is a fundamental problem in industrial project settings.
- **Expressiveness**: the degree to which the application domain is represented precisely in terms of the concepts offered by the modeling technique. More concretely, this feature measures the degree to which the modeling technique allows us to represent static, dynamic, intentional and social elements of the application domain.
- **Traceability**: the capability to trace modeling elements through different stages of the modeling process. This feature is important because it allows the user to verify that all elements of one model (e.g., capturing requirements) have corresponding elements during the

analysis and design stages, and vice versa. Traceability makes it possible for the analyst to move back and forth between models corresponding to different development stages (Dam and Winikoff 2003). From an organizational modeling perspective, this is basically oriented to assure that the late requirements model and the subsequent conceptual model are correct representations of the original organizational model.

- **Reusability**: the degree to which models can be reused. As with software code, this feature is causally related to modularity. If the modeling technique allows the definition of modules, general cases (patterns) can be defined for reuse.
- **Scalability**: the degree to which the modeling framework can be used to handle applications of different sizes. Scalability also measures the degree to which the inclusion of new modeling elements leaves unaffected the understandability of models (also known as extensibility). Scalability is related to refinement and modularity.
- **Domain Applicability:** the degree to which the modeling framework matches modeling requirements for a particular application domain.

This is the set of characteristics we selected to accomplish the evaluation tasks. It is true that, for some of the features chosen, one can evaluate *i** (or any other modeling framework, for that matter) on theoretical grounds alone. However, in our study of *i**, we wanted to include a practical evaluation as confirmation of any preliminary theoretical suppositions. Moreover, clearly the chosen features interact. For instance, better modularity management, obviously contributes to easier complexity management. Likewise, reusability contributes to scalability. Also, refinement is close related to traceability. We are studying such correlations and hope to integrate them in the evaluation framework for future studies. Here, we focus on the application of the proposed set of features in evaluating *i** in practice.

3.10 The evaluation results

The evaluation was conducted over a 9-month period. The average size of the models generated by the three teams had as follows: (i) Technical meeting management: 12 actors, 55 dependencies, 70 actors´ internal activities; (ii) Golf tournament management: 8 actors, 42 dependencies, 103 actors´ internal activities; (iii) Car rental management: 13 actors, 143 dependencies, 219 actors´ internal activities.

The evaluation assigned one of three possible values (*Well supported*, *Not well supported*, and *Not supported*) to each feature. Another output of the evaluation was a list of reasons given by the analysts for a judgment passed. In order to make the evaluation consensual, a meeting was held at the end of each case study. In these meetings, produced diagrams and personal evaluations were presented and discussed. The meetings included in-depth discussions for each feature in order to reach consensus and a final judgment.

One interesting result of the evaluation concerns the differences in the models produced by the participating teams. The members of team 1 were experienced in requirements modeling, although not used to modeling in terms of goals, actors and dependencies. They understood well the concepts underlying *i** (after all, requirements concepts match well *i** modeling), and were enthusiastic about using *i** in practice. In this case, resulting models were partially compliant with *i** philosophy. Moreover, the analysts of this team detected several areas where *i** lacked mechanisms to guarantee the usefulness of organizational models in generating system requirements.

In Team 2, the analysts were used to work with class diagrams, state and functional models as part of their on-going modeling activities. In this case, *i** social and intentional concepts were rather unfamiliar and the analysts tried to use the concepts in the same way they used the concepts they were accustomed to. In this case, resultant models were less compliant with *i** modeling philosophy. Moreover, these analysts had a lot to say about the lack of precise definitions for *i** concepts, and guidelines for generating *i** models.

The analysts for Team 3 were experienced *i** modelers. In this case, resulting models were completely compliant with *i** modeling philosophy. However, these models were often too abstract for generating software requirements.

Table 1 shows a synthesis of the results obtained in the empirical evaluation. The first column indicates the type of evaluation criteria (Modeling Language or Pragmatics), the second column indicates the feature evaluated, and finally, the third column indicates the judgment passed on each feature (Not supported, Not Well Supported, Well Supported).

Evaluation Criteria	Issue		Evaluation
Modeling Language	1	Refinement	Not Well Supported
	2	Modularity	Not Supported
	3	Repeatability	Not Well Supported
	4	Complexity management	Not well Supported
	5	Expressiveness	Well Supported
	6	Traceability	Not Well Supported
	7	Reusability	Not supported
Pragmatics	8	Scalability	Not supported
	9	Domain applicability	Well Supported

Table 1 Results of the empirical evaluation

Let us point out that one of the contributions of this empirical evaluation is the presentation of information about the reasons for the analysts to give a certain evaluation to each one of the selected issues. In this section, we present the arguments to justify the consensus reached when analyzing the values assigned to each one of the issues of the Evaluation Framework as a result of the performed experimentation.

It is important to point out that the evaluation of the *i** Framework was made in order to determine how this framework supports the relevant properties of a Model-Driven environment. The analysis of the features was made taking into account this objective. Therefore, the values assigned to the features (Not supported, Not Well Supported, Well Supported) only represents if the *i** modeling framework fits the specific model-driven environment, and they do no represent a global qualification of the i* framework.

This is the first evaluation of the *i** Framework focusing on a specific application domain. However, the results obtained in our empirical evaluation are similar to those obtained in research works where *i** and Tropos have been analyzed (with general features) together with other agent-oriented methodologies ((Dam and Winikoff 2003) and (Sturm and Shehory 2003)). The similarities in the results support the conclusions of our evaluation. An in-deep analysis must be done in order to determine if the results obtained of our empirical evaluation could be similar to those obtained applying the evaluation framework outside the context of a model-driven development process. At present there are no precise evidences to indicate that our result can be interpolated to other application domains.

Once the values for each feature were assigned by the participant teams, the next step was to understand and justify these values. To do this, an explanation for the assigned values was obtained by consensus of the participant teams. The explanation for each feature is presented below.

3.10.1 Feature: Refinement

Evaluation: Not Well Supported

Explanation: There are two types of refinement supported by *i**: (i) refinement of strategic dependency models in terms of a more detailed strategic rationale model, where one can see why actors depend on each other; (ii) 2) refinement of actor goals into more concrete subgoals. However, the literature using *i** includes many examples where a rationale model is not the result of a refinement of a dependency model. This kind of refinement can be performed in the boundaries of an actor model.

These types of refinement are useful when analyzing small case studies. However, they have severe limitations when the model grows in size and complexity. The dependency model is too concrete to serve as a starting point for the analysis of a large enterprise. In such cases, it may contain many actors with a large number of dependencies corresponding to different business processes, whose union constitutes a very complicated model to manage.

The current version of *i** does not include modeling primitives that allows us to start the modeling process of an enterprise with abstract concepts. These concepts would allow us to incrementally add more detail -- using other, more specific, modeling primitives -- until we reach concrete models of business processes and their actor dependencies. There are also no concepts to structure the different functional units of a complex organization. As a consequence of this absence of high-level refinement facilities, the modeling of complex systems that involve a large number of dependencies among many different actors is problematic for *i**.

3.10.2 Feature: Modularity

Evaluation: Not Supported

Explanation: Based on the empirical evaluation, it was concluded that modularity is not supported in *i**. This is the case because *i** doesn't have mechanisms for using building blocks that can be logically composed to represent different organizational fragments (e.g., business processes). In this context, if a new organizational process is added, this may affect all models constructed so far.

The lack of modularity mechanisms in *i** can be viewed as a consequence of its focus on actor modeling rather than on business process modeling. The modeling mechanisms of *i** are oriented towards the definition of the behavior of the organizational actors (to satisfy their goals and dependencies) rather than being oriented to the definition of high-level views of the organizational business processes.

Due to this the lack of modularity, rationale models represent a monolithic view where all elements of an enterprise are represented at the same abstraction level without considering any sort of hierarchy. Figure 3.3 shows an example for the Technical Meeting Management case study where the goal dependency "*obtain quality review*s" and other dependencies associated with this goal (the task dependency: "*send reviews on time"*, and the resource dependency: "*review"*) are represented at the same abstraction level. This makes it impossible to distinguish the hierarchical level of these concepts, which are represented as dependencies in the same diagram.

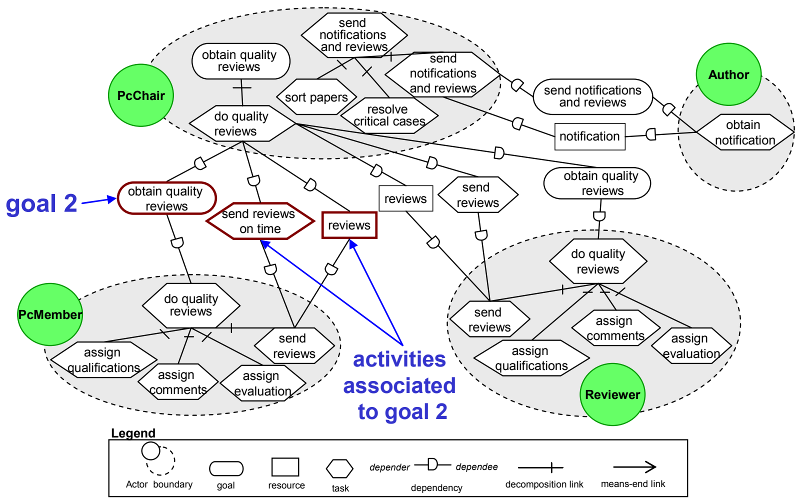


Figure 3.3 Example of the representation of concepts in the same abstraction level.

3.10.3 Feature: Repeatability:

Evaluation: Not Well Supported

Explanation: One of the key points for ensuring repeatability in a modeling method is the definition of a precise, formal semantics for the modeling constructs. In principle, the modeling constructs of *i** have been defined using formal descriptions and meta-modeling diagrams. These definitions are useful for expert analysts in early requirements. However, for those who are not experts in *i**, these definitions do not provide the necessary, precise support to determine which modeling construct to use when. This problem can also be noted in the *i** literature. There are several examples where very similar settings have been modeled using different primitives.

It is also possible to find in the literature examples of dependencies that do not satisfy the basic semantics of an actor dependency (vulnerable actor, actor who decides how to fulfill the dependency, type of *dependum*). For example, we found cases where the *dependee* of a dependency was incorrectly used as the vulnerable actor, instead of the *depender*. In another example, we found cases where the *dependee* of a dependency was incorrectly treated as the actor who prescribes the actions to execute for a delegated task (task dependency), instead of following the guidelines of always placing the *depender* as the actor that prescribes a task dependency. As a consequence of these situations, it is difficult to ensure that a reasonable degree of repeatability is achievable with *i**.

Figure 3.4 shows an example of these repeatability problems. In this example, taken from the Golf tournament management case study, the process for "Pay for registration in tournament" was represented in two different ways by the participating analysts: either as a task dependency, where the focus was placed on the activity to be executed; or as a resource dependency, where the focus was placed on the payment, which was viewed as a concrete resource relating the actors involved.

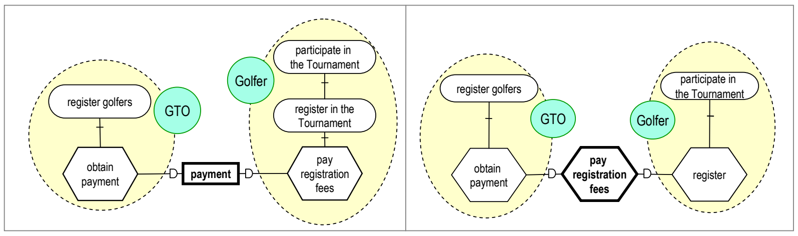


Figure 3.4 Example of two different representations for a given single process.

3.10.4 Feature Complexity Management:

Evaluation: Not Well Supported

Explanation: In the current version of *i**, it is possible to analyze an enterprise model using two different viewpoints: the strategic dependency model and the strategic rationale model. These viewpoints are useful for small cases, but they are not adequate for dealing with large and complex problems. There are no mechanisms for defining a high-level view of the whole process executed in the enterprise. This high-level view would be properly decomposed following a model-within-a-model strategy, where lower level descriptions are created separately, incorporating all relevant details.

The limitation in the mechanisms that are provided for managing the system complexity make modeling in *i** unnecessarily complicated. The lack of hierarchies leads to problems such as: a) difficulties to determine where to start the analysis; b) difficulties to determine the elements of the model that correspond to each organizational process and/or unit. The lack of hierarchies produces models where several business processes are represented and mixed all together in the same diagram, without any indication of the ownership of neither each low-level activity nor any information about the boundaries of each individual process.

Figure 3.5 shows the graphical representation of a model where several business processes are represented and mixed all together in the same diagram, without any indication of the ownership of the low-level activity or any information about the boundaries of each individual process.

3.10.5 Feature Expressiveness:

Evaluation: Well Supported

Explanation: There was unanimous agreement among all participants in this experiment that *i** indeed provides a very interesting set of conceptual primitives that make it possible to build pure organizational models on top of conventional requirements ones (mostly, use case-based models). Analysts also agreed on the importance of linking early requirements and late requirements, as a way of connecting software engineering practices with organizational design tasks that are too often performed in isolation by consultants.

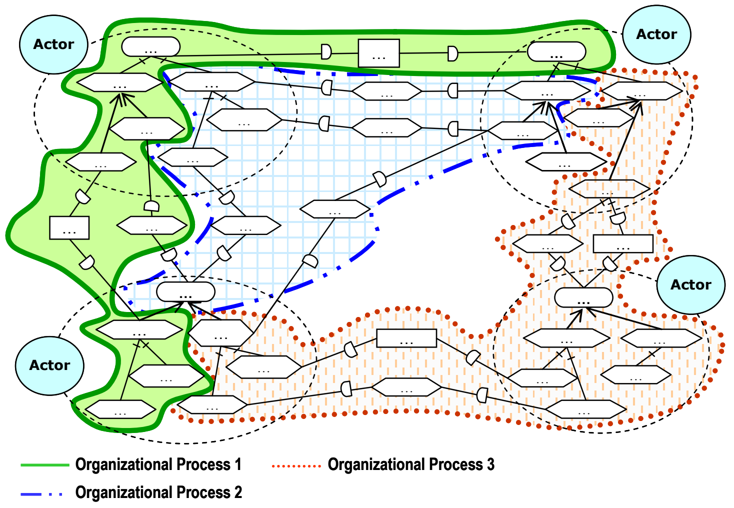


Figure 3.5 Representation of different processes in the same diagram

The *i** Framework was deemed adequate for capturing the relevant concepts of the enterprise, providing mechanisms for representing: a) the social structure of the enterprise, b) the intentional aspects of the organizational actors, c) the activities needed to satisfy the goals of the organizational actors, d) the relevant resources in the business processes, e) the ability to represent roles, positions and agents to describe the organizational actors, f) the architecture of the enterprise and g) the interaction between the system and external agents.

These conclusions account for the difference between *i** and other modeling techniques, which are not as well equipped to represent the social and intentional reasons that underlie the operation of an enterprise.

The static structure of the organization could be represented using the graphical representation of the *i** diagrams. These diagrams allow us to represent a static overview of how the organization works. These models show the actors, goals, tasks, dependencies, resources, and the boundaries of the organization. Whereas the graphical diagrams capture the static structure of the organization, the formal specification of the modeling concepts captures some aspects of the dynamic behavior of the system. To do this, the formal specification represents the pre- and post–conditions and triggers for the organizational tasks. The empirical evaluation allowed us to demonstrate that building an *i** organizational model is very useful for detecting the following problems:

Bottlenecks: This is the case when an actor concentrates a large number of incoming dependencies from other organizational actors. In this case, a failure or delay in this organizational actor could cause a chain reaction in the entire enterprise. The bottleneck problem could be detected by analyzing the dependencies where an actor plays the role of *dependee* of several dependency relationships. We are not aware of other modeling frameworks that account for this kind of analysis. Figure 3.6 shows a graphical representation of bottlenecks in a business process represented in the *i** Framework.

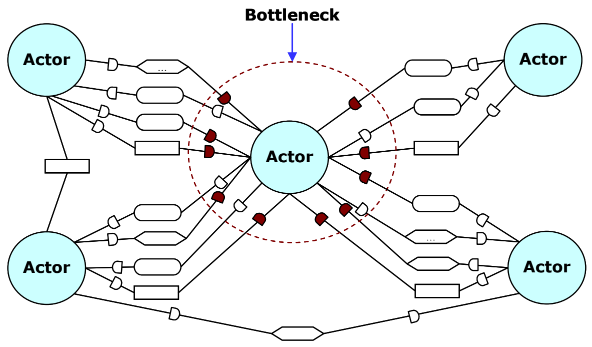


Figure 3.6 Representation of bottlenecks in a business process

Vulnerabilities: One of the key advantages of *i** is the explicit representation of vulnerabilities of organizational actors. In this case, if an actor participates in too many dependencies as *depender*, this actor could then become vulnerable if any of the *dependee* actors fail to deliver on their respective dependencies. Figure 3.7 shows a graphical representation of vulnerabilities in a business process represented in the *i** Framework.

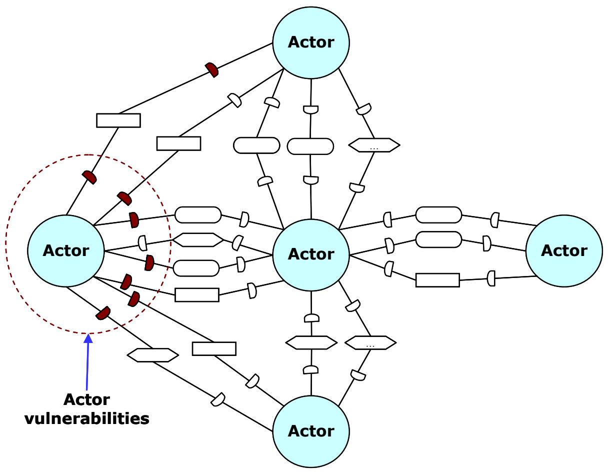


Figure 3.7 Representation of vulnerability in a business process

Critical responsibilities: This is the case where an actor concentrates many goal dependencies, which indicates that the actor has many critical responsibilities in the business process. In this case, it may be that the actor has excessive responsibilities and needs help, or at least monitoring. Figure 3.8 shows an example of this situation.

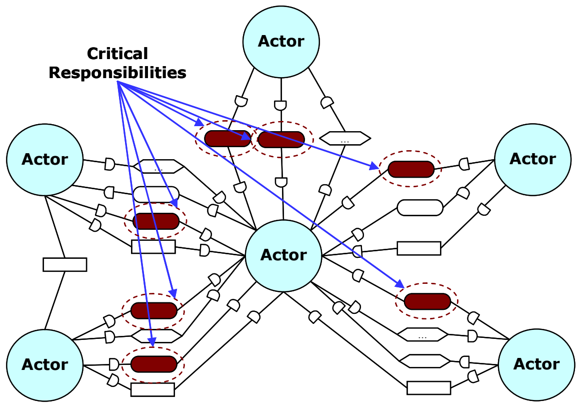


Figure 3.8 Representation of excessive responsibilities in a business process

The explicit representation of these organizational situations is the basis to carry out a useful business process reengineering analysis.

3.10.6 Feature: Traceability:

Evaluation: Not Well Supported

Explanation: *i** provides modeling flexibility for adding elements to an individual dependency and/or rational models. This means that new dependencies can be added to the rationale model that were not previously considered in the corresponding dependency model and vice versa. This is sometimes useful with respect to modeling flexibility. However, it is also true that this could have negative effects for model-driven approaches, where the elements of a model must have counterparts in previous models. We conclude that *i** does not have precise guidelines for deriving each element of the dependency model from corresponding elements in the rationale model.

3.10.7 Feature: Reusability:

Evaluation: Not Supported

Explanation: *i** does not offer clear mechanisms for properly managing reusability of parts of an organizational model. As mentioned earlier, the lack of good reusability capabilities is a consequence of the absence of mechanisms for modularization. The lack of conceptual building blocks with the required granularity makes it very complicated to reuse certain fragments of a model. Moreover, *i** lacks view definition mechanisms (in the sense of database views) for selecting parts of a monolithic model that capture new viewpoints.

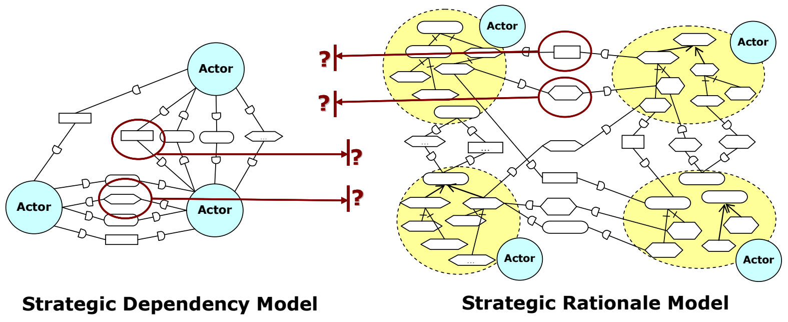


Figure 3.9 Representation of problems of traceability

As a consequence of this weakness, modeling projects using *i** must too often start from scratch, without taking advantage of previous projects for similar domains.

3.10.8 Feature: Scalability:

Evaluation: Not Supported

Explanation: This is probably the best-known and widely acknowledged problem of *i**. There are simply no clear mechanisms for managing the scalability of strategic models in *i**.

For small problems *i** clearly works fine. However, when the modeling problem grows in size and complexity, the large number of elements represented in the same diagram makes their systematic use and analysis very complicated, when not completely impossible. The scalability problem is also a direct consequence of the lack of mechanisms for modularization, and the inability to put together an abstract view of the high-level business processes of an enterprise. Consequently, all modeling elements for representing the semantics of a specific business process must be placed in the same diagram. Figure 3.10 shows an example of the high number of modeling elements in a diagram for only a fragment of a business process. And this is a very small fragment of the case study.

In summary, the lack of mechanisms for managing scalability is one of the greatest problems for the real applicability of *i** modeling.

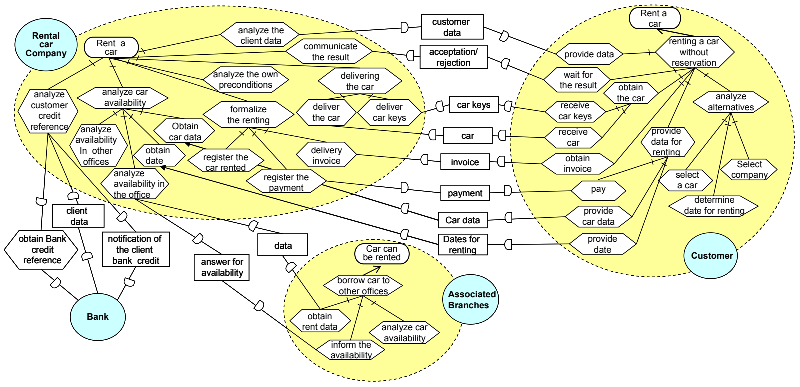


Figure 3.10 Fragment of the car renting process in the Car Rental Management case study

3.10.9 Feature: Domain applicability:

Evaluation: Well Supported

Explanation: The *i** Framework has a semantics and a corresponding *i** has an ontology and a corresponding notation that we found well suited for organizational modeling. It is also appropriate for the analysis of late requirements. The conceptual primitives are expressive enough to be applied in different domains, and they are appropriate for expressing properties that an organizational model must include. The semantics of the social concepts could also be applied, for example, to present dependencies within and between communities of systems, or even to represent the dependencies between an information system and its stakeholders.

3.11 Discussion

The main conclusion of this empirical evaluation is that *i** needs to be extended with mechanisms that manage granularity and refinement in models, as discussed below:

Granularity: Many of the negative results in the evaluation of *i** are related to the lack of mechanisms for defining granules of information at different abstraction levels, and composition mechanisms for composing these granules. This problem becomes evident when the modeling problem grows in size and complexity. In these cases, non-expert *i** users have difficulty with the scalability of their model. The result of this scenario is usually an overloaded monolithic model that contains all the relevant details of a social and intentional setting. Any activity that tries to extend, analyze, adapt or reuse parts of such a model is bound to be complicated and error-prone. To avoid this problem, it is necessary to provide precise conceptual constructs representing building blocks that break the monolithic structure of *i** models as well as composition mechanisms. Then, encapsulated model units could be created, analyzed and reused in an independent way. The practical implication of the granularity solution is the introduction of viewpoints that go beyond the actor viewpoint. For example, process viewpoints could give an orthogonal view for an organizational model. Note that for this extension, no modifications are needed to the original set of *i** modeling constructs.

Refinement: Apart from the definition of abstract primitives as building blocks, analysts must be provided with guidelines that allow them to structure a complete enterprise model. One way to achieve this consists of using concrete specification units to create the models following a refinement-based approach. In this way, the modeling process starts with a high-level view of the enterprise. Then, each element of this high-level view is refined into a more concrete model. Viewpoint mechanisms are a very promising direction to help manage the complexity of modeling activities. A viewpoint on a system involves a perspective that focuses on specific concerns regarding the system, while suppressing irrelevant details (Sinan 2003). A promising strategy in this direction would be to guide the organizational modeling process using selected viewpoints. The refinement process enables us to join the advantages of social modeling with a compositional approach to create the organizational models incrementally.

In order to propose a solution for the problems of refinement, modularity, complexity management, reusability and scalability found in the practical evaluation of *i**, we propose a Business Service Architecture for the *i** Framework. This proposal attempts to improve the current state of the *i** Framework so that it can be used in a Model-Driven approach. The detailed description of the service-oriented approach for the *i** Framework can be found in Chapters 5 and 6 of this thesis.

As a further step in ensuring the repeatability of the modeling results and also to ensure the traceability between models, we are developing a proposal to revisit the definitions of the modeling primitives of *i**/Tropos based on a multidimensional framework. The multidimensional framework captures relevant properties (dimensions) which allow us to characterize each modeling primitive of the *i/Tropos Framework. Thus, it is possible to clearly differentiate the modeling primitives of *i**. A detailed description of this work can be found in Chapter 4.

Chapter 4

4. The Modeling Language definition

This section introduces the definition of the modeling language for our service-oriented architecture. The modeling language is the result of revisiting and extending the semantic of the *i** modeling concepts. The Chapter presents the proposed syntax and semantics for association, aggregation, generalization and classification relationships that were adapted from the *i** relationships.

4.1 Introduction

In this thesis, the basic *i*/Tropos* modeling concepts have been adopted for organizational modeling purposes. However, a reviewed version of these modeling concepts has been developed in order to make the appropriate extensions for the service-oriented method. To do this, an initial analysis allowed us to precisely identify current issues in notation and semantics of the *i** modeling concepts. Another objective of revisiting the modeling construct is the attempt to overcome some of the problems detected in the empirical evaluation of the *i** framework. Our intention in reviewing the semantic of the *i** relationships is not the attempt to standardize the definition of the *i** relationship, but to provide a semantics clear for the purposes of our service-oriented method.

The modeling constructs of *i** have been defined using formal descriptions and meta-modeling diagrams. However, for those novel analysts in *i**, these definitions do not provide the necessary, precise support to determine which modeling construct to use when a specific semantic must be represented when facing real case studies. This problem was also found in bibliographical research in *i** literature. As a result of this situation, there are several examples where very similar settings have been modeled using different primitives. This situation makes it difficult to ensure an appropriate rate of repeatability and traceability in the modeling results.

As stated in empirical evaluation Chapter, at the present time it is difficult to ensure that novel analysts use the same modeling concept to represent similar semantics (repeatability problems). The traceability issue correlates directly with the repeatability factor. If repeatability cannot be ensured, then it will be difficult to perform the automatic translation among modeling diagrams. The weakness of *i** to manage repeatability and traceability has a more relevant impact on model-based development approaches, where automation of models transformation process is a key factor to ensure the correct model transition.

The basic components of a modeling language are primitive concepts and abstraction mechanisms (relationships). In this thesis, we focus on the analysis of the *i** relationship concepts. This is because modeling concepts of this kind are the main source of the feeling of ambiguity detected in practical experiences.

One of the objectives of this work is to review the semantics of *i** relationships to ensure that they will fit the needs of the analysts in practical case studies, This is because we found several cases where novel analysts had found it difficult to determine what situations to use each *i** concept must be used in, and to relate what kind of elements must be associated through a specific relationship. Therefore, instead of following the criteria of using the semantics of the modeling constructs according to a specific method (*i**, Tropos or GRL) we propose a specific semantic for our service-oriented method.

In this Chapter, we propose a specific semantics for association (*member-of*), aggregation (*part-of*), generalization (*is-a*) and classification (*instance-of*) relationships. To provide a formal definition of each type of relationship, a multi-property framework is proposed to provide our particular interpretation of the *i** abstraction mechanisms that overcome the current issues detected in the empirical evaluation. We have pointed out the differences of our reviewed version of the relations and the original syntax and semantics of the i* modeling concepts. It is important to point out that the analysis of the *i** primitive concepts (goal, softgoal, resource, and plan) is out of the scope of this thesis.

The *i** modeling elements (primitive concepts) and the proposed relationships (abstraction mechanisms) to associate modeling elements are presented below.

4.2 The *i** primitive concepts

The *i** Framework provides four basic modeling concepts: actor, goal, task and resource.

Actor**:* An actor represents an autonomous and social entity that has strategic goals and intentionality. ***Goal*:** A goal is a condition or state of affairs in the world that the stakeholders would like to achieve. A softgoal represents a goal that has no clear-cut definition and/or criteria as to whether it is satisfied. ***Task: A task specifies a particular way of doing something. ***Resource**:* A resource represents a physical or an informational entity.

We have determined that the current definition of the *i** primitive concepts gives the correct support for the service-oriented method proposed in this thesis.

4.3 The *i** abstraction mechanisms

The conceptual modeling techniques must offer semantic terms (primitive terms) for modeling an application (such as entity, activity, agent goal, etc). Moreover, they must offer a way to organize information in terms of *abstraction mechanisms* such as generalization, aggregation and classification (Mylopoulos 1998).

The abstraction mechanisms allow us to structure primitive term assemblies to represent a specific semantics of the problem space.

The *i** Framework proposes five mechanisms for associating basic concepts: decomposition links, means-end links, contribution links, dependency relationships and *is-a* relationships. However, to date, there is no analysis to associate the *i** relationships with the standard abstraction mechanisms (generalization, aggregation and classification). Mappings of this kind are needed in order to precisely define the characteristics of each modeling primitive according to standard attributes for abstraction mechanisms. By doing this, we give the analyst the knowledge to select the modeling primitive to use to represent a certain semantics. Therefore, the first objective of this work was to propose a specific mapping between the *i** relationships and the abstraction mechanisms.

We propose the following mapping schema between the abstraction mechanisms and the *i** modeling primitives; aggregation is implemented by using decomposition relationships, the association corresponds with means-end, contribution and dependency relationships; generalization is supported by the *is-a* relationship, finally, classification is implemented by *instance-of* link. The rationalities behind this specific mapping are presented in following sections.

As stated above, a particular interpretation of the *i** abstraction mechanisms is proposed in order to make it comply with our service-oriented method, and also to provide solutions to the issues detected in the empirical evaluation of the *i** framework. To do this, an in-depth analysis of each modeling concept has been made in order to clarify the ambiguities and inconsistencies detected in the empirical analysis. The strategy for characterizing each abstraction mechanism is presented in the following section.

4.4 The strategy to characterize abstraction mechanisms

The semantics of the abstraction mechanisms is influenced by a set of constraints or properties that restrict the way in which the elements can be associated. Therefore, the properties define the rules to associate the primitive terms using a specific abstraction mechanism.

In our proposal, the properties are used to define a multi-property framework that precisely characterizes our revisited version of the *i** abstraction mechanisms. In this way, the framework allows us to define our particular definition of each abstraction mechanism according with the properties of the framework. The framework captures the relevant constraints that must be expressive enough to ensure that the modeling concepts can be properly distinguished. By giving values to the constraints, we can establish the semantic of the proposed relationships.

Multi-property approaches have been successfully applied to define modeling constructs in several application domains. In OO-Method project, a multidimensional framework has been proposed to define the semantics of the relationships between classes (association, aggregation and composition) in object-oriented conceptual models (Albert et al. 2003). The purpose of this work was to clearly define the properties that enable the analyst to differentiate among UML relationships.

The properties defined in our framework explicit state the rules for using the *i** abstraction mechanisms in order to represent a certain semantics. Therefore, the proposed framework makes it possible to clearly differentiate the modeling primitives of *i** so that modelers get better guidance on what primitives to use in different situations.

In order to reach an agreement about the relevant properties for the modeling concepts, several meetings were held with designers and users of *i** and Tropos. In these meetings, the ambiguities that were detected in practical case studies were presented and discussed. We also performed an exhaustive review and analysis of the *i**/Tropos bibliography. By doing this, it was possible to reach a consensus about the values for the proposed properties.

In the proposed multi-property framework, the values of the properties are presented in a table. The columns indicate the modeling concept being analyzed. The rows of the table show the values for the proposed properties (Figure 4.1).

	Modeling concept
Dimension 1	
Dimension 2	
….	
Dimension n	

Figure 4.1The multi-property framework

One of the key points of this work is the definition of the set of constraints and properties that help us to represent the rules for using the *i** abstraction mechanisms. In order to avoid the ambiguity of selecting an arbitrary set of properties, we have based the selection of the properties on a set of well-known standard constraints for characterizing abstraction mechanisms.

The definition of the *i** abstraction mechanisms is composed by following elements: a) the standard definition of the relationship, b) the description of the *i** modeling construct that supports a specific abstraction mechanism, c) the definition of relevant properties for constructing the framework, d) the definition of the revisited interpretation of the relationship were values were assigned to the selected properties, e) the analysis about the values assigned to each one of the properties, and finally, f) a brief discussion of the abstraction mechanism being analyzed.

The definition of the abstraction mechanisms (aggregation, association, generalization and classification) for the *i** framework is presented below.

4.5 Aggregation (*part-of*) relationship

Aggregation, which refines association (Albert et al. 2003), (Saksena, France, Larrondo-Petrie 1999), (Ambler 2005), associates an *aggregate* (or whole or composite) to its *components* (or parts). Therefore, aggregation, which implies stronger coupling than association, specifies that instances of one class contain instances of the other class as parts. The aggregation has also been called *part-of* relationship (Whole←Part).

Semantically, *part-of* relationships can be distinguished by several constraints or properties based on *multiplicity*, *transitivity*, *reflexivity*, *symmetry*, *homogeneity*, *world assumption*, *shareability*, and *existence dependency.* Each of these influences how the "part" components relate to the "whole" component (Pardedel, Wenny, and David 2004). We have determined that along with the standard properties for the aggregation, additional constraints are also needed in our specific work for characterizing the aggregation in *i**. These additional properties are *boundary* and *operators*.

4.5.1 A multi-property framework to characterize aggregation in *i**

Following, we present the definition of the properties that integrate the framework for the aggregation relationship. The definition of the properties considers the following elements: i) the indication whether the property applies to the ends of the relationship or to the relationship itself. ii) the level at which properties apply (the instance or class level). iii) the intuitive meaning of the property. iv) the possible values for the property, and finally, v) the property formalization (when it is possible to formalize it). It is important to point out that the formal definitions of multiplicity and existence dependency have been adopted from the works proposed in (Albert 2006).

a) **Multiplicity**

Defined over: the ends of the relationship.

Applicability level: the class level.

Meaning: The multiplicity specified over an end *E1* of a relation *r* determines the minimum (*Min*) and maximum (*Max*) values of objects of the end *E1* that can be connected through the relationship *r* to an object of the opposite end *E2*. The value of the multiplicity must be determined for both sides of the relationship. The multiplicity defined at the class level restricts the association of elements at the instance level.

Values: non-negative integers.

Formalization (Albert 2006):

> $(\forall X)\ A(x) \Rightarrow$ smaller_equal_that(size(r(X), Max_A) $\wedge$
>
> greater_equal_than(size(r(X), Min_A)
>
> where:
>
> *A* is a predicate and *A(x)* is true if *x* is an object of the class *A*.
>
> *r(X)* takes an object and returns the set of objects associated with this object through the relationship r.
>
> *size(r(X))* returns the number of group elements.

b) **Transitivity**

Defined over: the relationship

Applicability level: the instance level.

Meaning: A relationship *R* is transitive if *xRy* and *yRz* together imply *xRz*

Values: transitive / non-transitive

Formalization:

Transitive relationship:

$\forall X, Y, Z, R(X,Y) \wedge R(Y,Z) \Rightarrow R(X,Z)$

Non-transitive relationship:

$\forall X, Y, Z, R(X,Y) \wedge R(X,Z) \Rightarrow \neg R(X,Z)$

c) **Reflexivity**

Defined over: the relationship

Applicability level: the instance level.

Meaning: Reflexivity specifies whether an instance of a modeling concept can be connected to itself. The value [Reflexive] indicates that this is possible; [Anti-Reflexive] indicates the contrary, and the value [Non-reflexive] indicates that it is possible but not obligatory to associate an element to itself.

Values: reflexive / anti-reflexive / non-reflexive

Formalization:

Reflexive relationship: $\forall X \Rightarrow R(X,X)$

Non-reflexive: $\forall X \neg R(X,X)$

d) **Symmetry**

Defined over: the relationship

Applicability level: the instance level.

Meaning: Symmetry specifies whether an instance of a modeling concept can be connected to another instance of a modeling concept which is already connected to it. If this is possible, the value of the property is [Symmetric]. If this is not possible, the value of the property is [Anti-symmetric].

Values: Symmetric / Anti-symmetric

Formalization:

Symmetric relationship

$\forall X,Y\ R(X,Y) \Rightarrow R(Y,X)$

Anti-Symmetric relationship

$\forall$ X,Y R(X,Y) $\Rightarrow \neg$R(Y,X)

e) **Homogeneity**

Defined over: the ends of the relationship

Applicability level: the instance level.

Meaning: homogeneity identifies whether the types of component that compose the relationship *R* are either homogeneous or heterogeneous. If the relationship permits to associate elements of different kinds, then the value of the property is [Heterogeneous]. If the relation only permits to associate elements of the same nature, then the value of the property is [Homogeneous].

Values: Homogeneous / Heterogeneous

Formalization:

Homogeneous relationship

$\forall$ X,Y R(X,Y) $\Rightarrow$ type(X) = type(Y)

Heterogeneous relationship

$\forall$ X,Y R(X,Y) $\Rightarrow$ type(X) $\neq$ type(Y)

f) **World assumption**

Defined over: the ends of the relationship

Applicability level: the instance level.

Meaning: The open world assumption is that the presumption that what is not stated is currently unknown. The closed world assumption is the presumption that what is not currently known to be true is false. Therefore, the world assumption identifies whether the specification of the relationship indicates an exhaustive set of associated elements. In this way, the relationship *R* must be composed of a predetermined set of elements and no others.

Values: open world assumption / close world assumption

Formalization:

Closed world assumption:

$R(\{X1,X2,...Xn\},Y) \wedge \ell(\partial (Y)) \Rightarrow \neg\exists Z\ R(Z,Y)$

Open close assumption:

$R(\{X1,X2,...Xn\},Y) \wedge \ell(\partial (X)) \wedge \exists Z\ R(Z,Y) \Rightarrow \text{True}$

where:

$\ell(X)$ is a predicate that is true when *X* exists, and it is an instance of a modeling element

$\partial(X)$ takes an object and returns the state of the object in the next state of the system.

g) **Shareability**

Defined over: the relationship

Applicability level: the instance level.

Meaning: shareability identifies whether instance(s) of end components can be shared by more than one instance of the other end of the relationship. If they can be shared, we call it a shareable aggregation.

Values: shareable / non-shareable

Formalization:

Shareable relationship:

$\forall\ X,Y\ R(X,Y) \wedge R(Z,Y) \wedge X \neq Z \Rightarrow \text{True}$

$\forall\ X,Y\ R(X,Y) \wedge R(X,Z) \wedge Y \neq Z \Rightarrow \text{True}$

Non-shareable relationship

$\forall\ X,Y\ R(X,Y) \wedge R(Z,Y) \wedge X \neq Z \Rightarrow \text{False}$

$\forall\ X,Y\ R(X,Y) \wedge R(X,Z) \wedge Y \neq Z \Rightarrow \text{False}$

h) **Existence dependency**

Defined over: the ends of the relationship

Applicability level: the instance level.

Meaning: this property identifies whether the components of the relationship must or must not coexist and adhere to each other. If the existence of one particular end of the relationship is totally dependent on the other end of the relationship, we call it an existence-dependent relationship. This means that removing the *end* component of the relationship will also remove all the associated *part* components. In the case of a non-existence dependency relationship, the deletion of the *end* component only implies removing the links with the *part* components.

Values: existence dependent / non-existence dependent

Formalization (Albert 2006):

existence-dependent relationship:

$\forall$ X,Y $R(X,Y) \wedge \neg \ell(\partial(X)) \Rightarrow \neg \ell(\partial(Y)) \wedge \neg R(\ell(\partial(X)), \ell(\partial(Y)))$

non-existence-dependent relationship:

$\forall$ X,Y $R(X,Y) \wedge \neg \ell(\partial(X)) \Rightarrow \neg R(\ell(\partial(X)), \ell(\partial(Y)))$

where:

$\ell(X)$ is a predicate that is true when *X* exists, and it is an instance of a modeling element

$\partial(X)$ takes an object and returns the state of the object in the next state of the system.

i) **Boundary**

Defined over: the relationship

Applicability level: the class and the instance level.

Meaning: Boundary specifies if the modeling construct can be used only in the actor's limits. If this is true, then the value of this dimension is [Internal]. If the modeling construct permits associate elements outside the actor's limits (the link crossing the actor boundaries), then the value of the construct is [External]. It is important to point out that, in this property, the reference point to indicate if the modeling construct is internal or external is the actor's boundary.

Values: internal / external

j) **Operators**

Defined over: the relationship

Applicability level: the class level.

Meaning: This dimension specifies the type of operator or quality metric supported by the modeling relationships.

Values: AND, OR, XOR, +, ++,-,--

Once the properties to characterize aggregation have been defined, the decomposition link, which is the *i** modeling concept that implements aggregation, is defined based on the proposed framework.

4.5.2 Decomposition links as an aggregation mechanism

We have determined that the *i** decomposition link can properly fit the semantics of the aggregation relationship. The decomposition (*task decomposition* in *i** and *AND/OR decomposition* in Tropos*)* allows for a decomposition of a root element into a set of leaf elements, where the elements of the compositions constitute an exhaustive set of elements that permit a root element to be achieved. In this context, the component elements are the parts that constitute the composite root basic concept.

The decomposition relationship implies full satisfaction. This indicates that, in the case of the AND decomposition, the satisfaction of the *part* components implies full satisfaction of the root element. In OR decomposition, the satisfaction of (at least) one of the possible alternatives, represented as *part* components, implies full satisfaction of the root elements.

In our proposal, the aggregation must be applied to the following sort of elements: goal, resource, plan and actor. In the case of the original definition of the *i** decomposition, it can only be applied to goals and tasks. We analyze the decomposition as an abstract modeling relationship and also the *AND/OR* decomposition as a specialized *part-of* relationship. The decomposition link can be denoted by: partof(X,Y), where *X* represents the *part* component and *Y* represents the *whole* component. Figure 4.2 presents the decomposition notation.

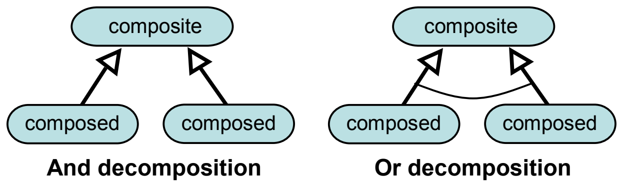


Figure 4.2The notation for decomposition link

4.5.3 The characterization of the decomposition based on the proposed framework for the aggregation

Once the properties to characterize aggregation have been introduced, the semantics of the *i** decomposition (represented as *partof* in the clauses) is defined by giving values to the framework properties. The first rows represent the standard aggregation definition.

Standard formalization of aggregation	$\forall$ X,Y partof(X,Z) $\wedge$ isa(Y,Z) $\Rightarrow$ part of(X,Y)
	$\forall$ X,Y partof(X,Y) $\Leftrightarrow$ wholeof(Y,X)
	partof(X,Y) $\wedge$ partof(Z,X) $\wedge$ Y $\neq$ Z $\Rightarrow$ True
Sort set	{goal, resource, plan, actor}
Multiplicity	(1,*), (1,*): A(whole)$\Rightarrow$smaller_equal_that(size(r(whole),*) $\wedge$ greater_equal_that(size(r(whole), 1) A(part)$\Rightarrow$smaller_equal_that(size(r(part),*) $\wedge$ greater_equal_that(size(r(part), 1)

Transitivity	Transitive: $\forall$X,Y,Z partof(X,Y) $\wedge$ partof(Y,Z) $\Rightarrow$ partof(Z,X)
Reflexivity	Non-reflexive: $\forall$X $\wedge$ partof(X,X) $\Rightarrow$False
Symmetry	Anti-symmetric: $\forall$ X,Y partof(X,Y) $\Rightarrow$ $\neg$partof(Y,X)
Homogeneity	Homogeneous: partof(X,Y) $\Rightarrow$ type(X) = type(Y)
World assumption	Closed world assumption: partof({D1,D2,...Dn},C) $\Rightarrow$ $\neg\exists$Z part of(Z,C)
Shareability	Shareable: partof(X,Y) $\wedge$ partof (Z,Y) $\wedge$ X $\neq$ Z $\Rightarrow$ True wholeof(X,Y) $\wedge$wholeof(X,Z) $\wedge$ Y $\neq$ Z $\Rightarrow$True
Existence dependency	Non-existence dependency : $\forall$ X,Y R(X,Y) $\wedge$ $\neg$ $\ell(\partial$ (X)) $\Rightarrow$ $\neg$R($\ell(\partial$ (X)), $\ell(\partial$ (Y)))
Boundary	Internal
Operators	AND, OR

4.5.3.1 The partAND relationship:

The partAND relationship is a specialization of the *partof* abstraction mechanism. This relationship allows us to decompose a root goal into a set of subgoals using an AND operator.

isa(partAND, partof)
Ins(X,C) $\wedge$ partAND({D1,…Dn}, C) $\Rightarrow$ (partof(X,X1) $\wedge$ ins(X1,D1)) $\wedge$ (partof(X,X2) $\wedge$ ins(X2,D2)) $\wedge$,… (part of(X,Xn) $\wedge$ ins(Xn,Dn))

4.5.3.2 The partOR relationship:

The partOR relationship is a specialization of the part/of abstraction mechanism. This relationship allows us to define a set of alternative goals needed to satisfy a root goal.

isa(partOR, part of)
$Ins(X,C) \wedge partOR(\{D1,\ldots Dn\}, C) \Rightarrow (partof(X,X1) \wedge ins(X1,D1)) \vee (partof(X,X2) \wedge ins(X2,D2)) \vee,\ldots (partof(X,Xn) \wedge ins(Xn,Dn))$

Following, we present the analysis of the decomposition based on the values of the multi-property framework. In this analysis, we present the semantic differences between the original *i** definition and the revised concept. It is important to point out that the original *i** definition has been taken from Yu´s thesis (Yu 1995).

a) *Multiplicity*: The value for the Multiplicity constraint for the original *i** concept is defined as follows:

decomposition (*i**)
(Root- leafs) (1:N, 1:1)

This value indicates that a root node can be associated with 1 or more leaf nodes; it also indicates that a leaf node can only be linked with a root node. In this context, the standard definition of the aggregation relationship does not imply a specific value for the multiplicity constraint, in contrast to the composition relationship (which refines the aggregation) in which the needed value of the multiplicity value must be (1, 1, *, *) to indicate the existence dependency of the *part* to the *whole*. Therefore, there are no restrictions for the value of multiplicity for the aggregation mechanism. However, we found it very useful to represent the situations where the same leaf node can part from more than one root node (for example: a task that is used to execute various high-level tasks). This situation can be found in several papers on business modeling (Aart, Wielinga and Schreiber 2004), (Van Welie, Van der Veer, and Eliëns 1998), (Decker, Erdman and Studer 1996).

The value of the Multiplicity constraint for our revised version of the decomposition links is defined as follows:

decomposition (revisited concept)
(Root- leafs) (1..N, 1..N)

As indicated above, the multiplicity property applies at the instance level. In the example, an instance of a root goal can be associated with several goal instances, and a leaf goal can be associated with more than one root goal (Figure 4.3).

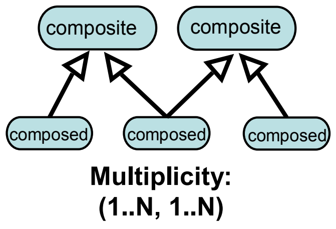


Figure 4.3 The multiplicity property for the decomposition link.

b) *Transitivity*: The *part-of* relationship is a transitive relationship (Terry 1998). Therefore, we can establish that if a modeling element C is part of another element B, and if this modeling element is also part of element A, then C is also part of element A.

There is no information about this topic in the original *i** definition of the decomposition relationship.

In the following example (Figure 4.4), the decomposition of goal A into goal C, and the later decomposition of this goal into goal E implies that goal E is also a decomposition of goal A. Transitivity applies to the instance level.

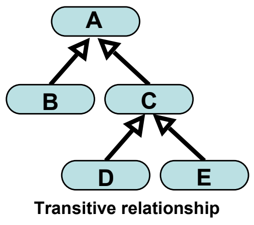
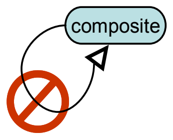


Figure 4.4 The transitivity property for the decomposition link.

c) *Reflexivity*: The *part-of* relationship is a non-reflexive relationship (Albert 2006). Therefore, it is not allowed to define a decomposition link that connects an intentional element to itself. This premise is only true at the instance level; in the definition of the meta-class level it is possible to connect classes of intentional elements to themselves in order to indicate, for example, that a class goal must be decomposed only into other goals.

This topic has not been analyzed before in the i* bibliography.

Figure 4.5 indicates that it is not possible to define a decomposition in which the *whole* and the *part* are the same instance of a goal class.

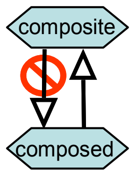


Non-reflexive relationship

Figure 4.5 The reflexivity property for the decomposition link.

d) *Symmetry*: The *part-of* relationship is a non-symmetric relationship (Albert 2006). Therefore, it is not possible to create a decomposition relationship between the intentional elements A and B, where B is already connected to A by a decomposition relationship. This assumption is true when applied to the instance level. In the class level, it is possible to define a symmetric relationship that indicates that a specific class type must be *part* and *whole* of the decomposition.

In Figure 4.6, we show that an instance of the task class cannot be *part* and *whole* at the same time for the same decomposition relationship.



Non-symmetric relationship

Figure 4.6 The symmetry property for the decomposition link.

e) *Homogeneity*: The And/Or Decomposition analysis is the appropriate modeling concept to be used to represent the decomposition of a high-level component in more concrete and specific sub-components (this is done in order to simplify the complexity of the semantics to be represented). Following, we present the values of the homogeneity in the original *i** definition of this modeling construct:

decomposition (*i**)
Non-homogeneous (composed of – composite to) Task – (Task, Goal, Resource, Softgoal)

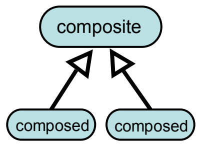
In the original definition of this concept, the root of the decomposition is always a task. This is the reason why the original name of this construct is Task-Decomposition. However, we have detected that, in practice, this specification avoids the specification of goal reduction, which is one of the basic analyses to determine how the strategic objectives of an enterprise are refined into more specific sub-goals and tasks. Therefore, we propose using the decomposition links to associate, not only tasks, but also goals, resources and actors. In this way it is possible, for example, to make an explicit reduction of the concept of high-level goal until the level of operational goals is reached. While it is true that at a certain point goals need to be operazionalized through the definition of tasks, it is also true that in some cases, the definition of the goals elicited by the analysts do not necessarily correspond with goals from low levels (those susceptible to be operationalized), instead it corresponds to abstract objectives that need to be refined into low-level goals. We propose the use of the decomposition link to implement the concept of refinement.

In order to make a consistent use of this concept, we have concluded that this kind of relationship can be used to decompose a modeling construct into subcomponents of the same nature (actor-actor, goal-goal, task-task, and softgoal-softgoal). In this way, it is not possible to represent decompositions that associate different kinds of primitive concepts. The main idea is to define a polymorphic modeling construct that could be applied for all *i** primitive concepts. The current decomposition link represents a mono-morphic representation in the sense that it can only be applied to decomposition with a task as root node. The proposed modification to the decomposition link (which is based on ontological foundations of aggregation) represents a significant improvement over the original *i** definition.

decomposition (revisited concept)
Homogeneous (composed of– composite to) Goal – Goal Plan – Plan Softgoal – Softgoal Actor – Actor

In the case of actor decomposition, this modeling construct is used to define generic actors that are refined until the level of specific actors is reached. The decomposition is used to define the structure of an organization by defining the subcomponents of an actor. In this particular case of actor decomposition, the OR-decomposition in the class level disappears at the instance level, in which a specific aggregation instance must be represented in the model.

In the case of task analysis, the decomposition is useful to represent the set of low-level activities that are needed to execute a high-level task. In the example presented in Figure 4.7, the decomposition link is used to associate goals, which indicate the refinement of an abstract goal into low level subgoals that refine the root node.



Homogeneous relationship

Figure 4.7 The homogeneity property for the decomposition link.

It is true that proposed semantics for the decomposition links is more restrictive that original definition, however, our intention with this semantic is the attempt to simplify the use of the modeling constructs for the specific context of our service-oriented method. In some other context, the flexibility of the original i* notation is required.

f) *World assumption*: In order to ensure the complete definition of *part* components that permit the fulfillment of the root node, the specification of decomposition must imply the definition of an exhaustive set of *part* components during analysis time. For this reason, it is never possible to incorporate new instances of *part* element that fulfill the root node. Therefore, the decomposition represents a closed world assumption relationship in the sense that only those *part* components that have been represented permit the satisfaction of the root goal. Those that are not currently specified during analysis time are false.

Again, there is no information about this topic in the original *i** definitions.

Figure 4.8, which identifies the world assumption property of the decomposition relationship, indicates that it is necessary to define an exhaustive set of leaf goal nodes “a priori” in order to decompose a root goal. Therefore, when a certain decomposition has been created in a time *t*, it is not possible to add new instances of *part* components in a time *t´*.

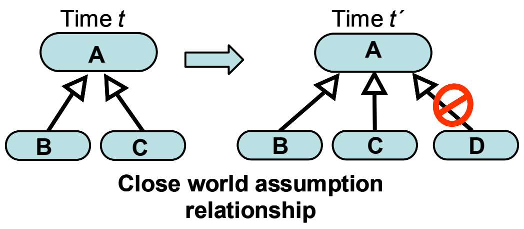


Figure 4.8 The close world assumption property for the decomposition link.

g) *Existence-dependency*: As commented above, the definition of aggregation does not imply a specific value for the existence dependency attribute. In this thesis, the And/Or Decomposition represents a non-existence dependency relationship, which indicates that the existence of the instances of the *part* components is not subordinated to the existence of the instances of the root node. One direct consequence of the existence dependency is the delete propagation schema. In the And/Or Decomposition, if an instance of a root node is eliminated, only the links among the root node and the leaf nodes of the relationship must be deleted.

The example presented in Figure 4.9, which represents the existence dependent property of the decomposition relationship, indicates that the existence of the leaf nodes is not subordinated to the root node, and therefore, the deletion of the root node A implies the deletion of the links that are part of the relationship.

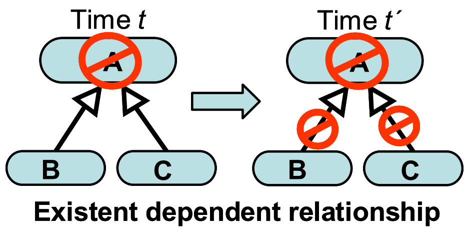


Figure 4.9 The existence dependency property for the decomposition link.

i) *Boundary*: When the decomposition mechanism is used to associate goals, tasks and resources, then it must only be used within the actor's limits. In this case, there are no changes according to the original definition of *i**. In the cases where the decomposition is used to associate organizational actors, then the value of the boundary property is external.

The example (Figure 4.10) shows the internal boundary of the decomposition for associate goals, tasks, and resources, and also the external limit of the decomposition for representing actors.

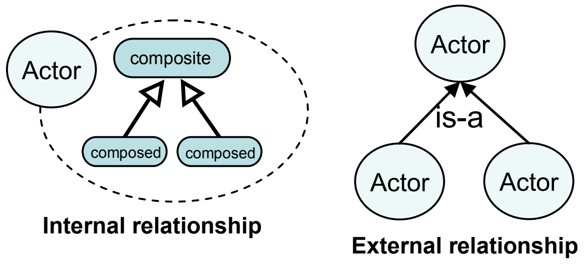


Figure 4.10 The boundary property for the decomposition link.

j) *Operators*: The original specification of this modeling construct only considers the use of *AND* logical operator. However, practical experiences have demonstrated (Sannicolo, Perini and Giunchiglia 2001) that it is also necessary to represent different alternatives for satisfying a parent goal or task. Thus, we have included the *OR* logical operator to represent the alternatives. This is the reason why the name of this modeling construct has been changed to And/Or Decomposition. Figure 4.2 presents the syntax for the AND/OR decomposition links.

4.5.4 Summary of decomposition as an aggregation relationship

The decomposition relationship, which implements the *part-of* abstraction mechanism, should be used in the following scenarios: a) there is evidence that the low-level components completely satisfy the root node, b) the elements involved in the decomposition are elements of the same type, and c) it is possible to define "a priori" the exhaustive set of *parts* that compose the *whole* (closed world assumption).

4.6 The association (member-of) relationships

The association models the existence of some kind of logical relationship between two entities. As stated above, the aggregation, which refines the association, implies stronger coupling. Therefore, in the case of the associations, their specification will be less restricted than the definition of the decomposition, which implements the aggregation mechanism.

In the association, as well as in the aggregation, it is possible to specify the role of the entities involved in order to clarify the structure (Cossentino and Sabatucci 2003). The *i** framework provides several mechanisms to associate primitive concepts in less restricted links than aggregation: means-end links, contribution links and dependency relationships. Therefore, these primitive concepts were categorized as association relationships.

4.6.1 A multi-property framework for characterizing the association relationships in *i**

The association relationships can be distinguished by using the same properties used to characterize the aggregation relationships: *multiplicity, transitivity*, *reflexivity*, *symmetry*, *homogeneity*, *world assumption, existence dependency, boundary and operators* (Elsmasri and Navathe 2004). In contrast to aggregation which restricts the values of transitivity, reflexivity and symmetry (transitive, not reflexive and not symmetric), the standard definition of association does not imply specific values for the defined properties (Albert 2006). This is because the association indicates a weaker coupling between the associated elements. For this reason, specific values for the different kinds of associations in *i** have been given in order to distinguish them.

We also consider the satisfiability as a relevant property to characterize the association relationships on *i**.

The definition of the properties for characterizing the aggregation relationship has been detailed in section 4.5.1. The definition of the satisfactibility is the following:

a) Satisfiability:

Defined over: the ends of the relationship

Meaning: This property specifies the level of fulfillment reached by a goal through the fulfillment of its associated subgoals.

Values: full satisfaction / partial satisfaction

It is important to point out that the values for the satisfiability property (full satisfaction / partial satisfaction) imply the intention of satisfaction more than satisfaction itself. In this context, full satisfaction implies full evidence that a goal would be satisfied. In the same way, partial satisfaction implies that there is a not complete evidence to indicate that the root goal will be satisfied.

Following, each relationship that implements the association in *i** is presented according to the same schema used to define the aggregation abstraction mechanism.

4.6.2 Means-End links as an association mechanism

This means-end link proceeds by refining a goal into subgoals in order to identify plans, resources and softgoals (the means) that provide ways to achieve the goal (the end). Therefore, the means-end represents the various alternatives that exist to satisfy a root element.

The means-end relationship implies full satisfaction. This indicates that each alternative solution represented by the *means* element implies the full satisfaction of the *end* element.

Usually, in the *i** approach, the modeling of the internal behavior of the organizational actors starts with the definition of a set of goals that the actor needs to fulfill. Then, once these goals have been detected, the means for achieving these goals must be elicited following a top-down strategy. However, it is also possible for the elicitation activities to start with the definition of low-level tasks of the organizational actors. In this case, the next activity would be the determination of high-level goals (*ends*) that give support to low-level goals.

The means-end link can be denoted by: me(X,Y), where *X* represents the *end* element and *Y* represents the *means* element. Figure 4.11 present the notation of this modeling concept.

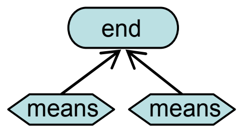


Figure 4.11 The means-end notation.

4.6.3 The characterization of the means-end link based on the proposed framework

The semantic of the *i** means-end relationship is defined by giving values to each one of the properties of the framework.

means-end definition	$\forall$ X,Y me(X,Z) $\wedge$ isa(Y,Z) $\Rightarrow$ me(X,Y)
	$\forall$ X,Y me(X,Y) $\Leftrightarrow$ $\neg$wholeof(Y,X) *
Sort set	{goal, resource, plan}

Multiplicity	(1,*), (1,*):
	A(end)$\Rightarrow$smaller_equal_that(size(r(end),1) $\wedge$ greater_equal_that(size(r(end), *)
	A(means)$\Rightarrow$smaller_equal_that(size(r(means),1) $\wedge$ greater_equal_that(size(r(means), *)

Transitivity	Transitive $\forall$ X,Y,Z me(X,Y) $\wedge$ me(Y,Z) $\Rightarrow$ me(Z,X)
Reflexivity	Non-reflexive $\forall$ X $\neg$ me(X,X)

Symmetry	Anti-symmetric: $\forall$ X,Y me(X,Y) $\wedge$ Y $\neq$ X $\Rightarrow$ $\neg$me(Y,X)
Homogeneity	Non-homogeneous: me(X,Y) $\wedge$ type(X) = type(Y) $\Rightarrow$ False
World assumption	Open world assumption: me({D1,D2,...Dn},C) $\wedge$ $\exists$Z me(Z,C) $\Rightarrow$ True
Shareability	Shareable: me(X,Y) $\wedge$ me (Z,Y) $\wedge$ X $\neq$ Z $\Rightarrow$ True
Existence dependency	Non-existence dependency : $\forall$ X,Y meR(X,Y) $\wedge$ $\neg$ $\ell(\partial$ (X)) $\Rightarrow$ $\neg$R($\ell(\partial$ (X)), $\ell(\partial$ (Y)))
Boundary	Internal
Operators	OR
Satisfiability	me(X,Y)$\wedge$ins(C, X)$\wedge$ins(D, X)$\wedge$FS(C) $\Rightarrow$ FS(D)

Once the values for the relevant properties have been established, the following step is the analysis of the means-end relationship based on the assigned values.

a) *Multiplicity*: The original means-end concept is defined in the original *i** proposal as follows:

means-end (*i**)
(end - means) (1:N, 1:1)

This cardinality indicates that an *end* can be associated with 1 or more *means*, and it also indicates that a *means* can only be related to a root node. However, in practice, we have found several cases where the same *means* helps to achieve more than one *end*. Therefore, the proposed value for the Multiplicity property is the following:

means-end (revisited concept)
(end - means) (1..N, 1..N)

Figure 4.12 presents the case where an instance of an organizational task is used to satisfy several instances of high-level goals of the enterprise.

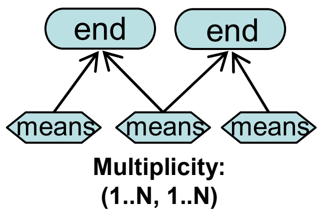


Figure 4.12 The multiplicity property for the means-end link.

b) *Transitivity*: We propose the means-end as a transitive relationship. Therefore, we can establish that if a goal is a means for satisfying a softgoal, the set of plans that implements the goal are also means for fulfilling the softgoal root.

This topic has been not analyzed before in the i* bibliography.

Figure 4.13 represents an example of a transitive means-end relationship.

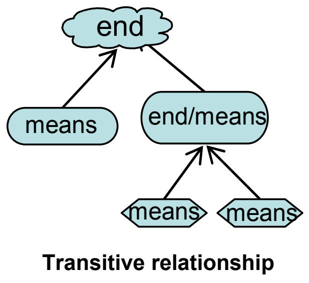
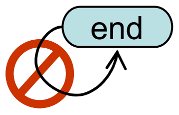


Figure 4.13 The transitivity property for the means-end link.

c) *Reflexivity*: It is not possible to define a means-end relationship that connects an instance of an intentional element to itself. This is because this relationship would indicate that an element can be *means* and *end* at the same time. For this reason, the relation is not reflexive. The value of this constraint comply with the original *i** definition. Figure 4.14 represents the means-end as a non-reflexive relationship.

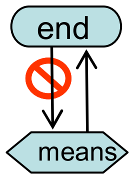


Non-reflexive relationship

Figure 4.14. The reflexivity property for the means-end link.

d) *Symmetry*: It is not possible to create a means-ends relationship between the intentional elements A and B, where B is already connected to A by another means-end relationship. Therefore, this kind of relationship is not symmetric (Figure 4.15).

The value of this constraint complies with the original *i** definition.



Non-symmetric relationship

Figure 4.15 The symmetry property for the means-end link.

e) *Homogeneity*: Following, we present the values of the homogeneity in the original *i** definition of the means-end links:

means-end (*i**)
Non-Homogeneous (end - means) Task, Goal, Resource, Softgoal – Task Goal – Goal Softgoal – Softgoal

The *i** philosophy that gives support to these values is the following: the Means of the relationship represents how to fulfill the end (which represents what to do). In this way, the natural way for expressing how to fulfill an activity in *i** is using the concept of task. This is why, in most cases tasks are used to represent the means of the relationship in *i** models. However, sometimes, in practice, more expressiveness is needed in this modeling construct in order to represent, for example, the set of alternatives goals that satisfy a softgoal.

In order to reduce the possible semantic overlapping between the means-end links and the decomposition links, we propose restricting the kind of elements involved in the relationships. We propose the means-end relationship to be a polymorphic relationship that is used to associate only elements of different types. Therefore, the means-end analysis can be the appropriate modeling concept to represent the refinement of a goal in a set of alternative tasks that allows us to satisfy it.

In our proposed language, we argue that the relations Task-Task, Goal-Goal, and Softgoal-Softgoal need to be modeled using decomposition links instead of a means-end relationship. This is because our decomposition applies to elements of the same type, which exactly correspond to the above-mentioned cases. We have eliminated the softgoal-task relationship because in our method it is not possible to define tasks that fully satisfy goals that cannot be precisely defined (softgoal). We have also eliminated the resource-task relationship because the focus in *i** is rarely placed on indicating the generation of resources as a result of organizational tasks.

The Homogeneity property for the revised concept is defined as follows:

means-end (Revisited Concept)
Non-Homogeneous (end - means) Goal –Plan Softgoal – Goal

Figure 4.16 presents examples of the use of means-links for associating elements of different types.

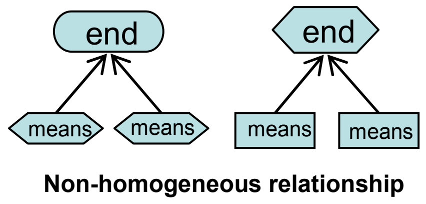


Figure 4.16 The homogeneity property for the means-end link.

f) *World Assumption*: The Means-End relationship implies and open world assumption. This means that there are no mandatory restrictions to specify the exhaustive set of end nodes that permit the root node to be fulfilled. Therefore, means could exist that are not currently specified in the means-end representation (nodes currently unknown) and that could also satisfy the end node. As a result of this, it is possible to incorporate, at any given time, new instances of means that permit to fulfill the End. One of the reasons for assigning the open world assumption to the means-end relationship is that the focus of the means-end is descriptive rather than prescriptive. The idea is to describe the different alternatives to satisfy an *end* without the restrictions for doing an exhaustive list of *means*.

It was not possible to find information about this topic in the i* bibliography.

Figure 4.17 represents the case where new means are added in execution time in order to satisfy the root goal

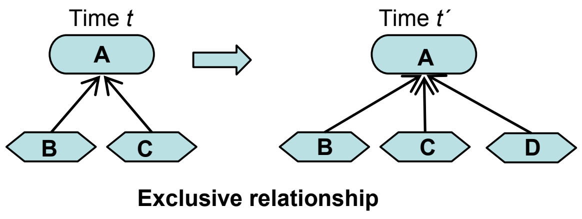


Figure 4.17 The open world assumption property for the means-end link.

g) *Existence-dependency*: The means-end relationship represents a non-existence dependency relationship, which indicates that the existence of the instances of the *means* is not subordinated to the existence of the instances of the *end* node. This is because there is not a strong coupling between *end* and *means*. One direct consequence of the existence dependency is the delete propagation schema. In the means-ends, if an instance of a root node is eliminated, only the links of the relationship must be deleted (Figure 4.18).

There is no information about this topic in the original *i** definition of the means-end relationship.

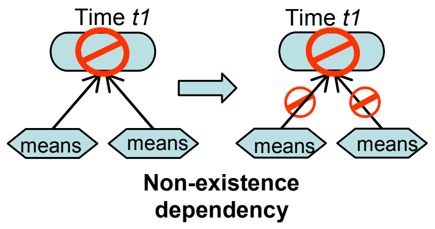


Figure 4.18 The existence dependency property for the means-end link.

h) *Boundary*: The means-end links can only be used within the actor's limits to associate goals, tasks and softgoals. The value of this constraint complies with the original *i** definition.

Figure 4.19 shows the internal boundary of the means-end relationship.

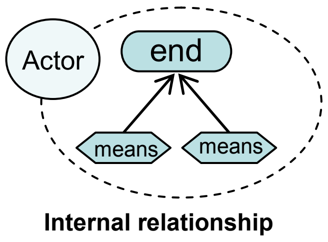


Figure 4.19 The boundary property for the means-end link.

j) *Operators*: The specification of this modeling construct only considers the use of OR logical operator in order to specify the various alternatives that could exist to satisfy the root element.

j) Satisfiability:

The means-end relationship implies fully satisfaction. This indicates that each alternative solution represented by the *means* components implies the fully satisfaction of the *end* component.

4.6.4 Summary of means-end as an association relation

We can conclude that the means-end relationship should be used when there is enough evidence to assure that the alternative subcomponents (*means*) fully satisfy the root component (*end*). The means-end links could be considered to be similar to the OR-Decomposition in the sense that both represent alternative solutions, and also in both cases, the satisfaction of the leaf components implies the full satisfaction of the root node. The difference between the means-end and the or-decomposition is that the former must be used to relate elements of different kinds and the or-decomposition must be used to associate elements of the same kind. Therefore, this kind of relationship is the appropriate selection to detail the set of plans that allow a target goal to be fulfilled, and also to represent the organizational goals that permit a quality attribute (softgoal) to be satisfied.

4.6.5 Contribution links as an association mechanism

The contribution link is a special association relationship that applies only with goals (hard goals and softgoal) and plans. Contribution analysis allows the designer to point out goals, softgoals and plans that can contribute positively or negatively towards reaching a specific goal. The contribution links must be applied following the viewpoint of a specific actor that wants to fulfill a specific objective.

The contribution link permits the analyst to represent partial and full satisfaction relationships among instances of modeling concepts. Specifically, the contribution link is the only abstraction mechanism to associate elements through partial satisfaction. The means-end and And/Or decomposition imply full satisfaction relationships. This modeling concept cannot be used to associate constructs of actor type.

The following table presents the different qualitative metrics for the contribution links.

+	positive contribution: partial satisfaction
++	positive contribution: full satisfaction
-	negative contribution: partial denial
--	negative contribution: full deny

A contribution can be annotated with a qualitative metric, denoted by +,++,-,--. In particular, if goal g1 contributes positively to goal g2 with metric ++, then if g1 is satisfied, so is g2. If goal g1 contributes positively to goal g2 with metric +, then g2 is partially satisfied if g1 is satisfied. The labels - and – represent the partial and sufficient negative contribution towards the fulfillment of a goal. The contribution link can be denoted by: cont(X,Y), where *X* represents the element "contributed by" and *Y* represents the element "contributed to".

Figure 4.20 presents the notation of this modeling concept.

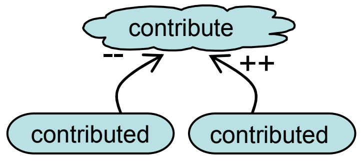


Figure 4.20 The contribution link notation.

4.6.6 The characterization of the contributions link based on the proposed framework

The semantics of positive (+) contribution links is defined giving values to each one of the properties of the framework.

contribution definition	∀ X,Y +cont(X,Z) ∧ isa(Y,Z) ⇒ +cont(X,Y)
Sort set	{goal, resource, plan}

Multiplicity	(0,*), (0,*):
	A(contribute)
	⇒smaller_equal_that(size(r(contributeto),*) ∧ greater_equal_that(size(r(contributeto), 0)
	A(contributed)
	⇒smaller_equal_that(size(r(contributedby),*) ∧ greater_equal_that(size(r(contributedby), 0)

Transitivity	Transitive ∀ X,Y,Z +cont(X,Y) ∧ +cont(X,Z) ⇒ +cont(Z,Y)
Reflexivity	Non-reflexive ∀ X ¬ +cont(X,X)
Symmetry	Symmetric: ∀ X,Y+cont(X,Y)∧+cont(Y,X) ∧ Y ≠ X ⇒ True
Homogeneity	Non-homogeneous: +cont (X,Y) ∧ type(X)=goal ⇒ type(Y)=goal ∨ type(Y)=plan ∨ type(Y)=softgoal +cont (X,Y) ∧ type(X)=plan ⇒ type(Y)=goal ∨ type(X)=plan∨ type(Y)=softgoal +cont(X,Y) ∧ type(X)=softgoal ⇒ type(Y)=goal ∨ type(Y)=softgoal
World assumption	Open world assumption: +cont({D1,D2,...Dn},C) ∧ ∃Z+cont(Z,C) ⇒ True
Shareability	Shareable: +cont(X,Y) ∧ +cont(Z,Y) ∧ X ≠ Z ⇒ True
Existence dependency	Non-existence dependency : ∀ X,Y +cont(X,Y) ∧ ¬ ℓ(∂ (X)) ⇒ ¬R(ℓ(∂ (X)), ℓ(∂ (Y)))
Boundary	Internal / External
Operators	+ positive contribution: partial satisfaction

Satisfiability	+cont(X,Y): FS(X) ⇒ PS(Y)
	PS(X) ⇒ PS(Y)

The semantics of positive (++) contribution links is defined giving values to each one of the properties of the framework.

Definition	∀ X,Y ++cont(X,Z) ∧ isa(Y,Z) ⇒ +cont(X,Y)
Sort set	{goal, resource, plan}

Multiplicity	(0,*), (0,*):
	A(contributedto)
	⇒smaller_equal_that(size(r(contributedto),*) ∧ greater_equal_that(size(r(contributedto), 0)
	A(contributedby)
	⇒smaller_equal_that(size(r(contributedby),*) ∧ greater_equal_that(size(r(contributedby), 0)

Transitivity	Transitive ∀ X,Y,Z++cont(X,Y)∧++cont(X,Z) ⇒ ++cont(Z,Y)
Reflexivity	Non-reflexive ∀ X ¬ ++cont(X,X)
Symmetry	Symmetric: ∀ X,Y ++cont(X,Y) ∧ ++cont(Y,X) ∧ Y ≠ X ⇒ True
Homogeneity	Non-homogeneous: ++cont (X,Y) ∧ type(X)=goal ⇒ type(Y)=goal ∨ type(Y)=plan ∨ type(Y)=softgoal ++cont (X,Y) ∧ type(X)=plan ⇒ type(Y)=goal ∨ type(X)=plan∨ type(Y)=softgoal ++cont(X,Y) ∧ type(X)=softgoal ⇒ type(Y)=goal ∨ type(Y)=softgoal
World assumption	Open world assumption: ++cont({D1,D2,...Dn},C) ∧ ∃Z++cont(Z,C) ⇒ True
Shareability	Shareable: ++cont(X,Y) ∧ ++cont(Z,Y) ∧ X ≠ Z ⇒ True
Existence dependency	Non-existence dependency : ∀ X,Y ++cont(X,Y) ∧ ¬ ℓ(∂ (X)) ⇒ ¬R(ℓ(∂ (X)), ℓ(∂ (Y)))

Boundary	Internal / External
Operators	++ positive contribution: full satisfaction
Satisfiability	++cont(X,Y): FS(X) $\Rightarrow$ FS(Y) PS(X) $\Rightarrow$ PS(Y)

The same value schema must be followed for defining the negative (-,--) contributions.

Once the values for the relevant properties have been established, the following step is the analysis of the contribution relationship based on these assigned values.

a) *Multiplicity*: The value for the Multiplicity in the original *i** definition is defined as follows:

contribution (*i**)
Contributed by – contributed to (0..*, 0..*)

We consider that this definition allows us to correctly represent the cases we found in practice. This value of multiplicity indicates that it is possible for some instances of the goals or task to not contribute with other instances of modeling constructs, and it also indicates that it is possible for a goal to not have influence on the satisfaction of any other instances of modeling elements.

Figure 4.21 shows a cardinality (1..*,1..*) for a specific contribution relationship.

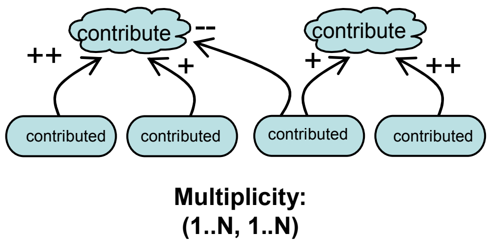


Figure 4.21 The multiplicity property for the contribution link.

b) *Transitivity*: We propose the contribution as a transitive relationship. Therefore, we can establish that if goal E contributes positively to goal C, then the achievement of goal A that is contributed by C is also influenced indirectly by goal E. Figure 4.22 presents an example of the contribution as a transitive relationship.

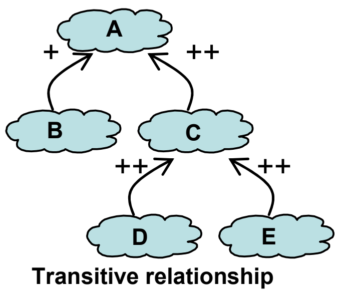


Figure 4.22 The transitivity property for the contribution link

c) *Reflexivity*: It is not allowed to define reflexive contribution relationships. Therefore, it is not possible to define a contribution relationship that connects an intentional element to itself, because this specification would indicate that the satisfaction of a goal has influence on its own satisfaction. There is no information about this topic in the *i** bibliography. Figure 4.23 shows an example of the non-reflexive contribution relationship.

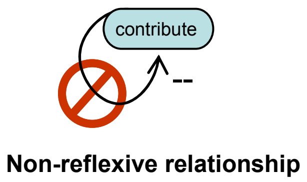


Figure 4.23 The reflexivity property for the contribution link

d) *Symmetry*: It is allowed to define symmetric contribution relationships. The reason that supports this justification is the existence, in practice, of circular-arc graphs among goals, where the satisfaction of goals influences the achievement of goals that already influence the former (Figure 4.24).

This topic has been not analyzed before in the i* bibliography.

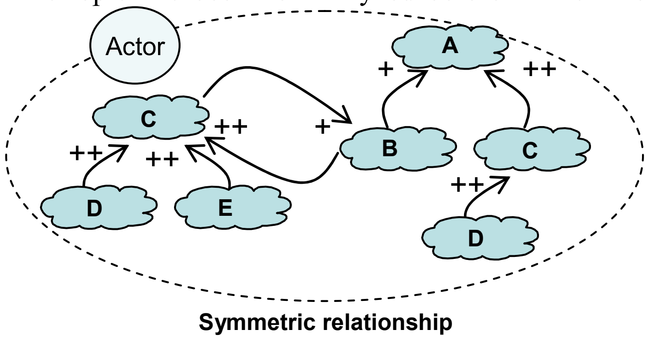


Figure 4.24 The symmetry property for the contribution link

e) *Homogeneity*: Following, we present the values of homogeneity relationship property in the original *i** definition of this modeling construct:

contribution (*i**)
Not Homogeneous (contributed by – contributed to) Softgoal – Task, Goal, Resource, Softgoal

In *i**, the softgoals are contributed by tasks, goals, resources and softgoals. The philosophy that supports these values is the following: softgoals are goals that cannot be precisely defined; therefore, only contribution analysis makes it possible to define how the instances of other modeling constructs influence the fulfillment of the softgoal. However, we consider that, in practice, not only softgoals are influenced by the environment. It is possible to find examples where tasks or goals are influenced by other instances of modeling constructs. For example, the use of a new encryption system in a bank has direct influence on the secure access to the data customers (Figure 4.26). In this case we have the case where a business plan influences a business goal.

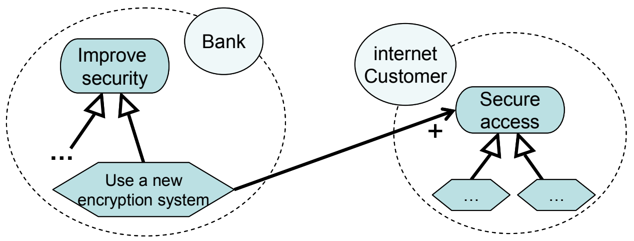


Figure 4.25 The boundary property for the contribution link

An analysis of real cases was performed in order to find the different alternatives that could be represented with the And/Or Decomposition. We have determined that plans, goals and softgoals could be influenced by other plans, goals and softgoals. The homogeneity dimension for the revised concept of the contribution was defined as follows:

contribution (revisited concept)
Non-homogeneous (contributed by – contributed to) Plan – (Plan, Goal, Softgoal) Goal - (Plan, Goal, Softgoal) Softgoal – (Goal, Softgoal)

The contribution link is the only abstraction mechanism that can be used to associate instances of elements of different and same types. The Figure 4.26 represents some examples of the use of contribution links.

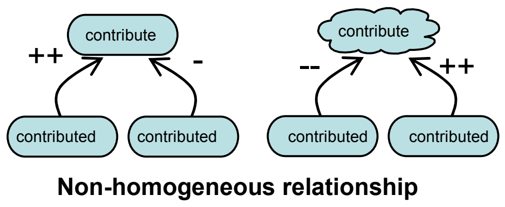


Figure 4.26 The homogeneity property for the contribution link

f) *World assumption*: The contribution link is an open world assumption relationship mechanism. This indicates that it is possible to determine, in execution time, new contributions of the existent goals with other instances of elements in the organizational model. In fact, the contributions represented in the model depend on the particular aspect being analyzed in the model. Therefore, the contributions for representing security aspects could be different from those considered for representing contributions of performance aspects.

There is no information about this topic in the original *i** definition of the contribution relationship.

g) *Existence-dependency*: The contribution link represents a non-existence dependency relationship, which indicates that the existence of the instances of the *contributed-to* element is not subordinated to the existence of the instances of the *contributed-by* element. The reason to do this is that there are no hierarchical relationships among the element participants in the contribution relationship. Therefore, it is only possible to delete the relationship that associates the delete node with the contributed node (Figure 4.27). This is because, typically, the remaining elements can continue contributing with other elements.

It was not possible to find information about this topic in the i* bibliography.

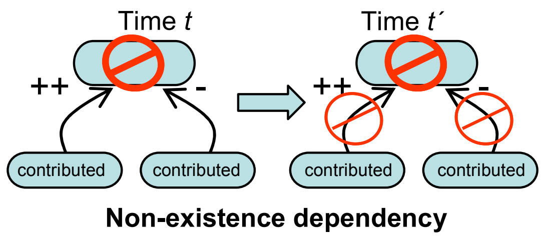


Figure 4.27 The existence dependency property for the contribution link

h) *Boundary*: In the revised version of this modeling construct, we consider that this modeling concept can be used to represent internal and external relationships. Therefore, this concept allows us to represent how an instance of a modeling construct contributes to the performance of an instance of a modeling element of another actor. In the original definition of *i**, the contribution analysis is considered as an internal relationship. However, we consider that activities executed by an actor in the enterprise environment may influence the performance of actions of other actors in the enterprise. Figure 4.25 shows an example of this situation.

j) *Operators*: The contribution link concept uses the qualitative metrics ++,+,-,-- to represent the rate of contribution between two instances of modeling elements.

j) Satisfiability:

As stated above, the contribution link could imply full or partial satisfaction depending on the qualitative metric. In the case of a [++] contribution between the constructs A and B, where B contributes to A, this indicates that the satisfaction of B implies the fulfillment of A. In the case of a [+] contribution between the constructs A and B, where B contributes to A, this indicates a positive influence on the performance of B in the fulfillment of A. In this case, there are no changes according to the original definition of *i**.

4.6.7 Summary of contribution link as an association relation

This class of relationship should be used when an instance of an intentional element contributes positively or negatively to the achievement or satisfaction of another instance of a modeling construct.

The contribution link is the only abstraction mechanism that allows us to define partial contributions among elements of the same or different types.

4.6.8 Dependency as an association mechanism

A dependency between two actors indicates that one actor depends on another actor to attain a goal, execute a plan, or deliver a resource. The former actor is called the *depender*, while the latter is called the *dependee*. The object (goal, plan resource) around which the dependency centers is called *dependum*. By depending on other actors, an actor is able to achieve goals that he would otherwise be unable to achieve on his own, or not as easily, or not as well. There are four types of dependencies: goal dependency, softgoal dependency, task dependency and resource dependency. The dependency relationship can be denoted by dep(X,Y,D), where *X* is the *depender* actor, *Y* the *dependee* actor, and *D* the *dependum* of the relationship. Figure 4.28 presents the notation of this modeling concept.

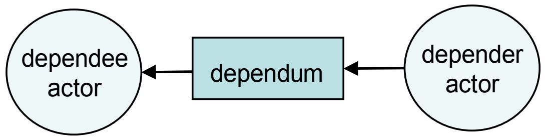


Figure 4.28 The dependency notation

4.6.9 The characterization of dependency based on the proposed framework

The semantics of the dependency relationship is defined by giving values to each one of the properties of the framework.

Standard definition of contribution	$\forall$ X,Y dep(X,Z) $\wedge$ isa(Y,Z) $\Rightarrow$ dep(X,Y)
Sort set	{actor, goal, resource, plan}
Multiplicity	(1,*), (1,*): A(dependee) $\Rightarrow$smaller_equal_that(size(r(dependee),1) $\wedge$ greater_equal_that(size(r(dependee), 1) A(depender) $\Rightarrow$smaller_equal_that(size(r(depender),1) $\wedge$ greater_equal_that(size(r(depender), 1)
Transitivity	Transitive / Non transitive $\forall$ X,Y,Z,D dep(X,Y,D) $\Rightarrow$ dep(X,Z,D_1) $\wedge$ dep(Z,Y,D_2) $\wedge$ $D_1 = D_2$

Reflexivity	Non-reflexive: $\forall$ X,Y,D dep(X,Y,D) $\wedge$ X =Y $\Rightarrow$ False
Symmetry	Symmetric: $\forall$ X,Y,D dep(X,Y,D) $\wedge$ dep(Y,X) $\wedge$ Y $\neq$ X $\Rightarrow$ True
Homogeneity	homogeneous: dep(X,Y,D)$\wedge$type(X)=actor$\wedge$type(Y)=actor $\wedge$ type(D)={goal, resource, plan} $\Rightarrow$ True
World assumption	Open world assumption: dep(X,Y,D) $\Rightarrow$ $\exists$Z dep(X,Z,D)
Shareability	Shareable: dep(X,Y,D) $\wedge$ dep(Z,Y,D) $\wedge$ X $\neq$ Z $\Rightarrow$ True
Existence dependency	Non-existence dependency : $\forall$ X,Y,D cont(X,Y,D) $\wedge$ $\neg$ $\ell(\partial$ (X)) $\Rightarrow$ $\neg$R($\ell(\partial$ (X)), $\ell(\partial$ (Y)))
Boundary	External
Operators	None
Satisfiability	Full Satisfaction

Once the values for the relevant properties have been established, the following step is the analysis of the dependency relationship based on these assigned values.

a) *Multiplicity*: A specific dependency relationship associates only two actors (*depender* and *dependee*) through one *dependum* element. However, an actor could be associated with many organizational actors through incoming dependencies. Also, an actor could participate in many dependencies playing the *depeendee* role. Therefore, the multiplicity of a dependency relationship is (1:*, 1:*).

The value of this constraint complies with the original *i** definition.

Figure 4.29 presents the case where an actor is associated with several actors through dependency relationships.

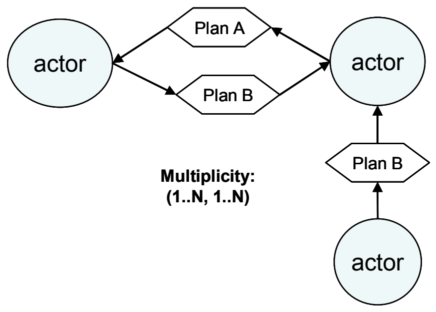


Figure 4.29 The multiplicity property for the dependency relationship.

b) *Transitivity*: The dependency implies a delegation of responsibilities between two actors. If an actor A1 delegates the element E1 to the actor A2 and this actor also delegates E1 to the Actor A3, then we can establish that A1 indirectly depends on A3 for achieving E1. The transitivity characteristic of the dependency can only be applied if the *dependum* involved in the dependencies is the same in all the relationships for the actors participating in the chain of responsibilities (Figure 4.30)

The Example shows the dependency relationship as a transitive relationship.

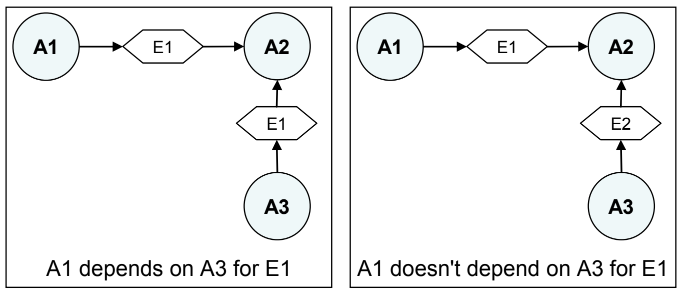


Figure 4.30 The transitivity property for the dependency relationship

c) *Reflexivity*: The dependency relationship is a non-reflexive relationship. Therefore, it is not allowed to define a dependency that connects an actor because this relationship would indicate that an actor depends on himself to do something. The correct representation of this semantics implies the use of an internal goal or task in the internal description of the actor, which indicates that the actor must fulfill a specific goal or task.

The value of this constraint complies with the original *i** definition.

The example shown in Figure 4.31 indicates that it is not possible to define a dependency in which the *depender* and the *dependum* are the same actor.

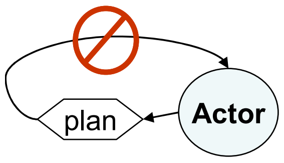


Figure 4.31The reflexivity property for the dependency relationship

d) *Symmetry*: The dependency relationship supports the specification of symmetric relationships. Thus, the dependency relationship permits the creation of a dependency between the actors A and B, where B is already connected to A by another dependency relationship.

Figure 4.32 presents the dependency as a symmetric relationship.

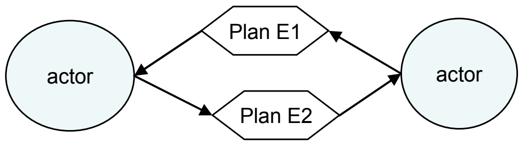


Figure 4.32 The symmetry property for the dependency relationship

e) *Homogeneity*: The dependency is the only *i** relationship that allows us to associate organizational actors. The elements of a dependency relationship are always two actors and the *dependum* object. The *dependum* could be a goal, a resource or a plan.

f) *World Assumption*: The definition of dependencies is one of the key points of the *i** modeling framework. We have found that, in practice, the definition of new dependencies in execution time is needed in order to represent complex scenarios. This is because it could be complicated to determine "a priori" all the possible dependencies that exist in the network of actor in an enterprise. For this reason, we define the dependency relationship as an open world assumption relationship.

Again, it was not possible to find information about this topic in the i* bibliography.

h) *Existence-dependency*: The dependency links represent a non-existence dependency relationship, which indicates that the actors involved in the dependencies could exist even when the dependencies disappear. As a consequence of this fact, if an instance of a dependency is eliminated from the model, the actors involved remain the same (Figure 4.33).

This topic has been not analyzed before in the i* bibliography.

Example: the example, which represents the non-existence dependency property of the dependency relationship, indicates that the existence of the actors involved in the relationship is not determined by the existence of dependencies between them.

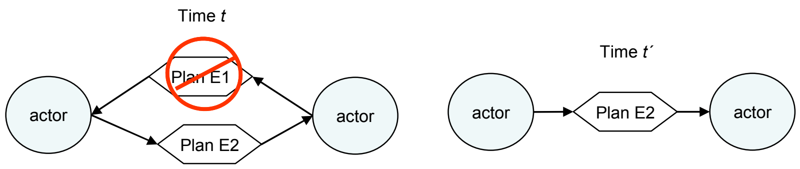


Figure 4.33 The existence dependency property for the dependency relationship

i) *Boundary*: the concept of dependency allows us to relate actors through external relationships (Figure 4.34). It is not allowed to define dependencies within the limits of individual actors. j) *Operators*: In this proposal, there are no operators for the dependency relationship.

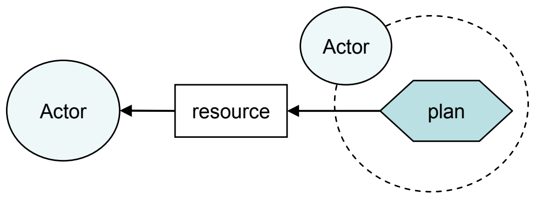


Figure 4.34 The boundary property for the dependency relationship

4.6.10 Summary of dependency as an association relationship

This kind of relationship must be used to represent the delegation of responsibilities between actors.

The dependencies represent the unique mechanisms provided by *i** to represent communication and social relationships among actors. This is why the concept of delegation of responsibilities involved in the dependency represents a powerful mechanism to represent, not only strategic interest of the actors, but also to represent low-level activities such as the delivery and request for resources among actors.

4.6.10.1 Guidelines to represent dependency relationships

The empirical evaluation of *i** has revealed that one of the main sources of inconsistent results in novel analysts is the definition of *i** dependencies. In order to give an initial solution to some of these issues, some useful guidelines for representing dependency relationships are presented below.

Guideline 1: The actor defined as *depender* in a dependency relationship must always be the actor that becomes vulnerable as the result of the dependency relationship. However, in the literature, it is possible to find several examples where the vulnerable actor has been placed as *dependee* actor of the dependency relationship. Figure 4.35 shows an example of this kind of inconsistent dependency. In this case, the actor who becomes vulnerable is the *dependee* of the dependency relationship.

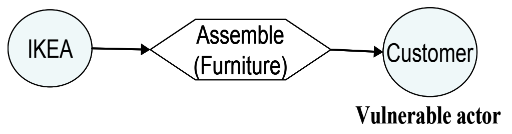


Figure 4.35 A non-consistent use of a dependency relationship

In order to solve inconsistencies of this kind we must:

1) Determine the actor that becomes vulnerable if the dependency is not fulfilled. 2) Determine the appropriate dependency according to the point of view of the vulnerable actor.

Figure 4.36 shows a correct specification of the same case shown in Figure 4.35.

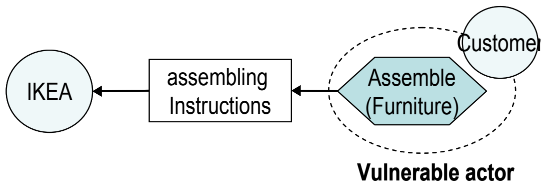


Figure 4.36 A consistent use of a dependency relationship

Guidelines 2: It is not possible to use only one dependency to represent the case where the *depender* and the *dependee* are both vulnerable in the dependency relationship. In this case, two different dependencies must be created.

Guideline 3: In plan dependency, we find the only case where the actor that decides how to fulfill the activity is the *depender*. In the case of the resource, goal and softgoal dependencies, the actor that prescribes the procedure is the *dependee* actor.

Following this criteria, it is possible to define these simple rules for deciding between a goal and a plan dependency:

1) Determine the vulnerable actor and use this as the *depender* actor. 2) If the *depender* actor defines the plan for fulfilling the activity, then a plan dependency must be used. 3) If the *dependee*

actor decides how to fulfill the activity, then a goal dependency must be used.

Guideline 4: The plan dependency provides visibility between actors because the *depender* actor decides the way in which the *dependee* must perform the activity; thus, it is possible to introduce monitoring tasks to control the performance of the plan delegated to the *dependee*. In the case of the goal dependency, it is not possible for the *depender* to monitor the fulfillment of the goal because the *dependee* must take all the decisions about the fulfillment of the goal.

4.7 The generalization (*is-a*) relationship

Generally speaking, the generalization relates *superclasses* to their specializations called *subclasses*. Subclasses inherit all properties from their superclasses. Subclasses may define new specific properties. The generalization represents the *is-a* relationship.

4.7.1 A multi-property framework for characterizing the generalization in *i**

The generalization relationship can be distinguished by the following constraints: *transitivity, symmetry, reflexivity, homogeneity, coverage*, and *mutual exclusion*. Each of these influences how the “Superclass” is related to the “subclasses”. Similarly to the aggregation and association mechanisms, in our proposal, these relevant properties are used to define the framework to define generalization. The framework defines our particular definition of generalization according to the properties of the framework.

Following, the definition of *coverage* and *mutual exclusion* properties is presented. The definitions of the *transitivity, symmetry, reflexivity,* and *homogeneity* have already been presented in section 4.5.1.

a) Coverage:

Defined over: the ends of the relationship

Meaning: The coverage is total if each member of the generic class is mapped to at least one member among the member classes. The coverage is partial if there are some member(s) of the generic class that cannot be mapped to any member among the member classes. (Elmasri 2004)

Values: total / partial

b) Mutual exclusion

Defined over: the ends of the relationship

Meaning: In an exclusive relationship, a member of the generic class is mapped to one element of, at most, one subset class. In an overlapping relationship, there are some members of the generic class that can be mapped to two or more of the subset classes (Elmasri 2004).

Values: exclusive / overlapping

Once the properties that we consider relevant to characterize the generalization have been defined, the *i* is-a* relationship is defined based on the proposed framework.

4.7.2 The *i* is-a* relationship

The *is-a* relationship is a modeling primitive that is supported by the original *i** definition. In *i**, the generalization relationship allows us to define actor hierarchies. In this sense, the *is-a* relationship is a mono-morphic construct that must only be applied to actor elements. In this thesis, we propose a polymorphic *is-a* construct that can be applied to the following sort of elements: goal, resource, plan and actor. The generalization relationship can be denoted by *is-a*(X,Y), where *X* is the sub-class component and *Y* the super-class component.

4.7.3 The characterization of generalization based on the proposed framework

Once the properties have been introduced, the semantics of the generalization is defined by giving values to each one of the properties of the framework. The first rows of the table represent the definition of the concept.

Standard *is-a* definition	is-a(C,D) $\wedge$ ins(X,C) $\Rightarrow$ ins(X,D)
	class(C) $\wedge$ class(D) $\wedge$ is-a(C,D) $\Rightarrow$True
Transitivity	Transitive: is-a(C,X) $\wedge$ is-a(X,D) $\Rightarrow$ is-a(C,D)
Symmetry	Non-Symmetric $\forall$ C,D is-a(C,D) $\Rightarrow \neg$ is-a(D,C)
Reflexivity	Reflexive $\forall$ C is-a(C,C)
Homogeneity	Homogeneous is-a(C,D) $\Rightarrow$ type(C) = type(D)
Coverage	partial
Mutual exclusion	Exclusive: is-a(C,D) $\Rightarrow$ $\neg\exists$Z is-a (C,Z)

Once the values for the relevant properties have been established, the following step is the analysis of the generalization relationship based on these assigned values.

a) *Transitivity*: The *is-a* relationship is a transitive relationship (Figure 4.37). Therefore, we can establish that if a modeling element C is associated with the element B through a *is-a* relationship and this element is also associated with A through a *is-a* relationship, then it is possible to establish that A is a generalization of C. In this case, C is a type of B, and B is a type of A, then C is a type of A.

There is no information about this topic in the original *i** definition of the *is-a* relationships.

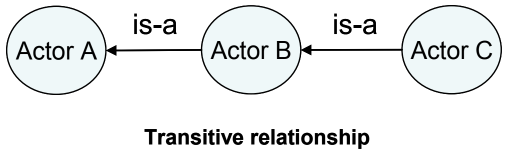


Figure 4.37 The transitivity property for the *is-a* relationship

b) *Symmetry*: The *is-a* relationship is a non-symmetric relationship. Therefore, it is not allowed to define a *is-a* relationship between the intentional elements A and B, where B is already connected to A by a *is-a* relationship (Figure 4.38).

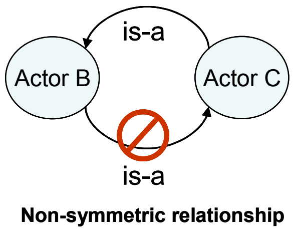


Figure 4.38 The symmetry property for the *is-a* relationship

c) *Reflexivity*: The *is-a* relationship is a reflexive relationship. The value of this property indicates that it is possible to specify that a *is-a* relationship connects an instance of a modeling concept to itself (Figure 4.39).

There is no information about this topic in the *i** bibliography.

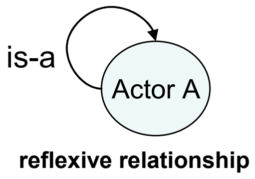


Figure 4.39 The reflexive property for the *is-a* relationship

d) *Homogeneity*: The *is-a* relationship must be applied to associate elements of the same kind. In the original *i** definition of the *is-a* relationship, it can only be applied to associate instances of actors. In our proposed definition, we apply the concept of polymorphic relationship to permit the use of actors, goals, tasks and resources as sorts of the *is-a* relationship. The *is-a* relationship applied to actors allows us to represent the actors´ hierarchies in an enterprise. The *is-a* relationship applied to goals permits the definition of categories of goals, making possible the definition of abstract goals, general goals, achievement goals, maintenance goals, executable goals, etc. The *is-a* relationship applied to resources permits the definition of hierarchies of physical or informational resources. Finally, the *is-a* relationship applied to the definition of instances of tasks allows us to represent abstract organizational procedures that are specialized into concrete organizational tasks (Figure 4.40).

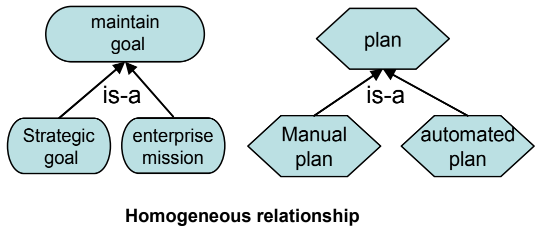


Figure 4.40 The homogeneity property for the *is-a* relationship

e) *Coverage*: The value of coverage for the generalization in *i** is partial. This indicates that not all generic classes must be mapped into specific class members.

f) *Mutual exclusion*: the value of the mutual exclusion property for the generalization is exclusive. This indicates that the multiple heritance is not allowed in our definition of the *is-a* relationship (Figure 4.41).

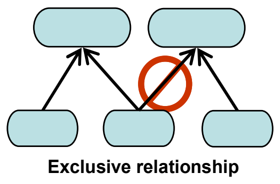


Figure 4.41 The exclusivity property for the *is-a* relationship

4.7.4 Summary of *is-a* as a classification relationship

The specialization/generalization relationship offers a powerful mechanism to create categories and hierarchies of concepts. This is why we have extent this modeling concept to associate modeling elements of different type (goal, resource, tasks and actor).

4.8 The classification (*instance-of*) relationship

The classification *relates* a class with a set of objects that share the same properties. An object must be an instance of at least one class (class ← instance). It is also known as *is-of* or *is an instance of*.

4.8.1 A multi-property framework for characterizing the classification in *i**

The classification relationship can be distinguished by the following constraints: *transitivity, symmetry, reflexivity,* and *world assumption*. Each of these influences how the class is related to its corresponding instances. Similarly that the aggregation and specialization mechanisms, in our proposal the relevant properties are used to define the framework for characterizing the classification. The framework defines our particular definition of classification according to the properties of the framework.

The definitions of *transitivity, symmetry, reflexivity,* and *world assumption* have already been presented in section 4.5.1.

4.8.2 The *instance-of* relationship as classification mechanism

At the present time, there are no definitions for the *instance-of* modeling construct in the *i** framework. In this thesis, we propose a polymorphic *instance-of* construct that can be applied to the following sort of classes: goal, resource, task and actor.

The *instance-of* relationship can be denoted by ins(X,Y), where *Y* is the class component and *X* the instance of the corresponding class.

4.8.3 The characterization of classification based on the proposed framework

Once the properties have been introduced, the semantics of the classification must be defined by giving values to the framework properties.

Standard instance-of definition	$\text{class}(C) \wedge \text{instance}(D) \wedge \text{ins}(D,C) \Rightarrow \text{True}$
Transitivity	Non-Transitive: $\text{ins}(C,X) \Rightarrow \exists Z\ \text{ins}(Z,C)$
Symmetry	Non-Symmetric $\forall\ C,D\ \text{ins}(C,D) \Rightarrow \neg\ \text{ins}(D,C)$
Reflexivity	Non-Reflexive $\forall\ C \neg\ \text{ins}(C,C)$
World assumption	Open world assumption: $\text{ins}(\{D1,D2,...Dn\},C) \wedge \exists Z\ \text{ins}(Z,C) \Rightarrow \text{True}$

Once the values for the relevant properties have been established, the following step is the analysis of the classification relationship according to the assigned values.

a) *Transitivity*: The *instance-of* relationship is a non-transitive relationship. This is because it is not possible to use an instance component to define new sub-instances. Therefore, an instance element cannot be instantiated (Figure 4.42).

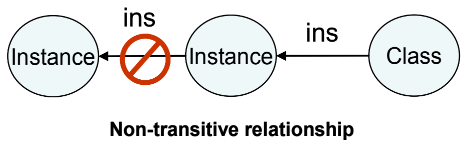


Figure 4.42 The transitivity property for the classification relationship

b) *Symmetry*: The *instance-of* relationship is a non-symmetric relationship. Therefore, it is not allowed to define a *instance-of* relationship between a class A and an instance B, where B is already connected to A by a *instance-of* relationship (Figure 4.43).

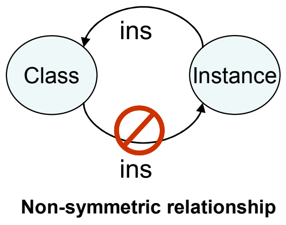


Figure 4.43 The symmetry property for the classification relationship

c) *Reflexivity*: The *instance-of* relationship is a non-reflexive relationship. The value of this property indicates that it is not possible to specify that a class element can also be an instance component (Figure 4.44).

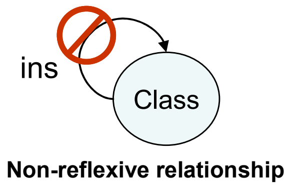


Figure 4.44 The reflexive property for the classification relationship

d) *World assumption*: The classification represents an open world assumption relationship. This indicates that it is not possible to define "a priori" an exhaustive set of instances for a specific class component. Therefore, when certain instances have been created in a time *t*, it is possible to add new instances of the same class in time *t´* (Figure 4.45).

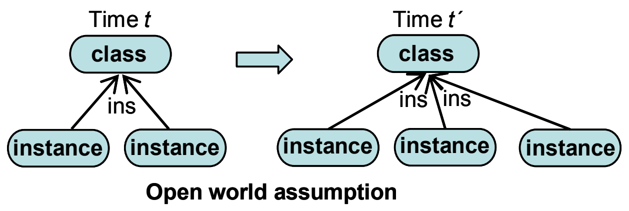


Figure 4.45 The close world assumption property for the classification link.

The classification relationship offers a powerful mechanism to create instances of the classes provided by the modeling language. The explicit indication of instances in the organizational model enables us to make specific models that precisely characterize the solution space.

4.9 Final Considerations

In this Chapter, a revised version of the *i** modeling concepts has been proposed in order to make it comply with the service orientation proposed in this thesis. Also, the revised version of the *i** modeling concepts was developed in order to propose solutions to the problems detected in the empirical evaluation. This is because practical experiences revealed that in several cases, the *i** concepts are not interpreted in the same way by different modelers. Also, an exhaustive bibliography study about *i** and its methodological derivations (GRL and Tropos) has revealed that there is not a consensus about the semantics of the modeling concepts. All this generates a feeling of semantic ambiguity that makes it difficult to ensure the repeatability and traceability of the modeling results in practical cases.

In this Chapter, we have presented a specific characterization for the modeling primitives of the *i** Framework based on a multi-property framework approach. The framework identifies a set of relevant properties (dimensions) that allows us to precisely define the modeling primitives by giving values to the dimensions of the framework.

One of the key points of this work is the determination of the appropriate set of properties to characterize each modeling primitive. An exhaustive bibliographic analysis about abstraction mechanisms has been carried out in order to avoid the selection of an arbitrary set of constraints. According to this analysis, a standard set of properties that has been used to characterize the aggregation, association, generalization and specialization were obtained. All these properties, together with others that we consider relevant for a specific modeling primitive, were used to define each one of the proposed frameworks.

The analysis of the values for the framework properties allows us to precisely justify the proposed modifications to the original *i** definitions. The proposed approach makes it possible to clearly differentiate the modeling primitives of *i** so that modelers get better guidance on which primitives to use.

Chapter 5

5. The Service-Oriented Architecture for the *i** Framework

The objective of this Chapter is to introduce the definition of the components of the proposed service-oriented architecture for the *i** framework. We present meta-models for understanding business services and the relationships between the components of the service-oriented proposal.

5.1 Introduction

As stated in Chapter 3 (empirical evaluation of *i** framework), the main conclusion of this evaluation is that, despite the advantages of the *i** modeling approach, practical experiences have revealed that there are certain issues that need to be improved to ensure their effectiveness in practice. We have concluded that *i** needs to be extended with mechanisms to define granules of information at different abstraction levels, and composition mechanisms to manage these granules (granularity mechanisms). Guidelines are also needed to start the organizational modeling process with the definition of a high-level view of the enterprise. Then, the guidelines must help the analyst to refine the high-level view into more concrete primitives until the lower abstraction level of the processes is reached (refinement mechanisms).

Our proposed solution is based on the concept of business service as a high-level concept that encapsulates fragments of an organizational model as composite business processes. We illustrate our approach using a case study in the travel agency sector. The case study (which is an extension of the rental car case study used in the empirical evaluation of the *i** framework) considers the set of services offered by a company specialized in selling travel packages and car rentals.

5.2 The proposed solution: a business service approach for the *i** framework

Our proposed solution to improve the *i** organizational modeling process is based on the hypothesis that it is possible to focus the organizational modeling activity on the values (services) offered by the enterprise to their customers. In this thesis, we will call them *business services*. Following this hypothesis, the proposed method provides mechanisms to guide the organizational modeling process based on the business service viewpoint.

Using the proposed approach, the monolithic structure of the *i** strategic rationale model can be broken down into several business services (Figure 5.1). These business services can be used as the basic granules of information that allow us to encapsulate a set of *i** business process models.

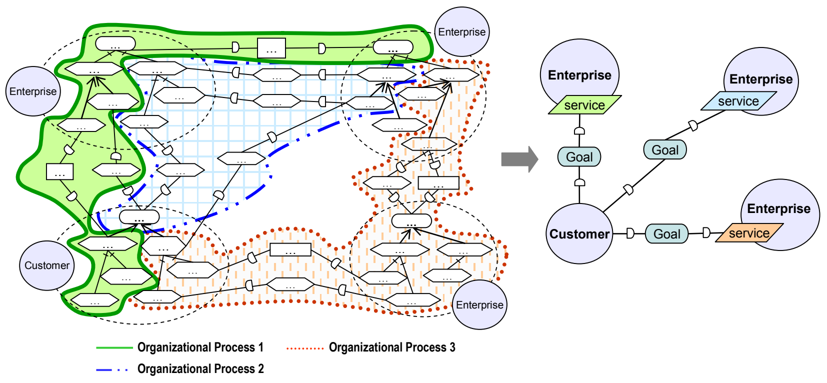


Figure 5.1 The business service strategy

One of the practical implications of this proposal is that the focus of the modeling activity has been changed from the actor's viewpoint to the service's viewpoint. In the current state of *i** and Tropos, the modeling process starts by determining the relevant actors in the organizational setting and also by determining the goals they want to fulfill. The following step consists of determining the tasks needed to satisfy the actors´ goals. As a result of this analysis of the actor's goals, the delegation of responsibilities to other actors must also be detected. These delegations are represented using the concept of strategic dependency. As a result of this modeling process based on actors, the current mechanisms for decomposition, refinement, and modularity in *i** are limited only to the actors´ boundaries.

In our business service approach, the modeling process starts by considering the enterprise as a service provider and by eliciting the services that the enterprise offers to end customers. The following step consists of determining the way in which the business services satisfy the goals of the enterprise. Once the services have been elicited, we need to refine each service in the set of business processes needed to perform it. As a result of this new approach, the mechanisms for decomposition, refinement, and modularity are focused on business services.

With this proposed approach, we can take advantage of the powerful intentional and social characteristics of *i** combined with a compositional modeling process, which could be more comfortable for non-expert analysts in *i**.

This thesis addresses the explanation of the services-oriented approach: (1) informally, by giving a set of graphical diagrams and demonstrating their use with examples. (2) formally, by defining axioms to define the rules of the service architecture.

5.2.1 What is a service?

Several service definitions have been proposed according to the application domain where the service concept is used. In this sense, most current software engineers associate the concept of service with *web services* or *e-services*. However, currently there is no consensus about the definition of either services or e-services. The Baida´s work (Baida 2006) offers a well-establish survey of definitions of services in several application domains (business research, computer science and information science). Following, a brief description of the services terminology is presented based on the definitions provided in Baida (Baida 2006)

Services in Business Research

In the business research community, there is consensus on considering services as (business) activities that result in value for the customers.

At this level, we found two different approaches for describing services. One definition deals with the economic value produced by customers and providers interchanging objects of economic value (business value perspective). Other proposals focus the service definition on the processes needed to produce values for the customers (business operation perspective).

More recently, the concept of e-service has been incorporated in the terminology of business research. This specific kind of service is considered as an extension of "traditional" services. In this sense, *e-services* have been defined as providing services over electronic networks (Rust and Kannan 2002). In Ruyter works (Ruyter et al. 2001) e-services are defined as "an interactive, content-centered and internet-based customer service, driven by the customer and integrated with related organizational customer support processes and technologies with the goal of strengthening the customer-service provider relationship". In this definition, the customers are considered as the main initiators of the activities that compose the service.

Although this application area is directly concerned with the representation of services as business activities, no explicit and precise definitions have been given for the concept of business services.

Services in Computer Science

In computer science, the concept of service is highly associated with the specific concept of web services.

One of the most accepted definitions of web services in the computer science community is the one provided by the Stencil Group, where services are defined as “loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols” (Stencil Group 2001). Functionality is one of the key concepts in this definition, and in almost all definitions about web services. However, it is important to point out that the functionalities make reference to functions of the software components more than business activities. There is no consensus on the relationship between the software functionalities (web services) and the business activities (business processes) needed to provide services in the organizational level. Nonetheless, we can still deduce that an implicit semantic relationship exists between both service specifications.

Sometimes, in the computer science community, the term *service* is used as a synonym for ‘web service’, as well as ‘e-service’. However, several authors have pointed out the differences among these three closely related concepts. The term service has been defined as organizational activities in a definition according to the business research area. The term e-service deals with organizational activities performed by a software system that works on internet but without emphasizing on how these services are operationalized on a specific platform. Finally, web-services have been defined as a low-level mechanism to implement e-services using precise implementation technologies.

In a more precise classification, services in computer science can be categorized as services to provide information to customers and providers (information-providing services), and also services that modify the "state" of the customers or providers (world-altering services).

Services in Information Science

The information science community has adopted the definitions of service provided either by business research (representing services as business activities that produce value) or services as defined in computer science (functionalities associated with internet and software systems). There is no new (re)definitions of the concepts defined in these application areas, more specifically, the use of web service is adopted from the computer science area, the service concept is adopted from the business research area, and the e-service concept is used with the same level of ambiguity presented in these previous communities.

A specific definition of business service has been developed in this thesis that defines the services at the organizational level. This definition is presented below.

5.2.2 Our conceptualization about business service

As stated above, there is no consensus on the definition of business service; even though it has been widely used in the service-oriented computing bibliography. In the cases where some explanations were given about this concept, services have been associated with organizational activities that are performed using internet. The following definition from (Amsden 2005) is an example of this business services conception: "The Business Services Model (BSM) is a dynamically created UML2 model of a service specification between business clients and IT implementers. The Business Services Model is a mediator between the business requirements expressed in process models and any implementation, including object or service-oriented implementations". As this definition states, business services are considered as a high-level specification of services that are implemented for use on internet.

Our concept of service concerns the organizational environment and organizational processes rather than the functionalities offered by software systems. In fact, the definition of services in this thesis does not imply that services need to be implemented by software systems. This is because the activities that compose the service can be executed manually by the organizational actors. The business service architecture enables the formalization of the relationship between the abstract definition of services (from the customer point of view) and its realization through a set of business processes.

We have defined a business service as a functionality that an organizational entity (an enterprise, functional area, department, or organizational actor) offers to other entities in order to fulfill its goals. To provide the functionality, the organizational unit publishes a fragment of the business process as an interface with the users of the service. The business services concept refers to the basic building blocks that act as the containers in which the internal behaviors and social relationships of a business process are encapsulated.

Our service-oriented architecture for the *i** framework provides a formal relation between an abstract representation of services and the set of processes that perform them. In this sense, the service specification is a contract that specifies the rules that determine how the providers and requester collaborate in order to achieve their objectives.

The proposed definition of business service complies with the definition of services defined in business research in the sense that is based on organizational activities and customers. However, our definition is different from those provided by current research works because it emphasizes the social and intentional perspectives on services rather than a traditional transactional perspective.

The services can be seen as an explicit agreement among customers (that want to fulfill their goals by using a service) and the providers (that want to fulfill their own goals by offering a service). By using the service, the customer extends its capabilities by using a set of services provided by an external entity. Therefore, the customer delegates the responsibility to a provider to perform the activities of the service. Although the delegation of responsibilities among requesters and providers extends the capabilities of the requester, it can also affect the requester who becomes vulnerable if the provider fails to deliver the service,

In our approach, services have a direct influence on the fulfillment of the goals of customers and providers. This makes our proposal different from the current research works which are based on describing services as transactional activities or services as business values generators.

The idea of our approach is to introduce a precise conceptual hierarchy consisting of business services that are refined in business processes, which are finally expanded in what we call business protocols. These protocols constitute the lower-level of the service description.

The proposed business architecture for the *i** framework permits the appropriate representation of the following key aspects:

- The services offered by the enterprise.
- The providers (enterprises) and requesters (final customer) involved in the service.
- The communication between providers and requesters.
- Shows the reasons for the enterprise to offer a service and the reasons for the customer to request it.
- Shows the values interchanged by the execution of the service and the reason for transferring these values.
- Indicates the reasons for the values being interchanged among the service participants.

The explicit representation of these aspects in a service model enables the analyst to improve the model before starting the development of services at the implementation level. Thus, the analyst can use the service approach to generate a view of the current situation of the enterprise as a starting point for the generation of the future situation of the enterprise.

5.2.3 Why a service orientation?

One of the fastest emerging technologies in software engineering is the separation of concerns, which is an established software engineering theory that is based on the notion that it is beneficial to break down a large problem into a series of individual problems or concerns. This allows the logic required to solve the problem to be decomposed into a collection of smaller, pieces, so that each piece can address a specific concern (Elr 2006).

The service orientation can be considered as a specific mechanism to implement the separation of concerns. This is because the philosophy of services is to isolate the abstract functionality (service) from the details of its implementation. Therefore, the service-oriented approach promotes decomposition and granularity, which are basic concepts of the separation of concerns approach. This is the reason why we argue that service technology can benefit the management of the complexity of the *i** modeling process.

Another reason to adopt a service orientation for the *i** framework, besides the well-known advantages of the SOA, is that this architecture enables the enterprise to quickly respond the changing market conditions. The dynamic environment of business today represents a challenge for the current modeling methodologies. Modeling techniques usually offer the means to model a stable application domain where changes to the specification are not frequent. However, nowadays, enterprises need to change very quickly in order to respond to very frequent market challenges. In this sense, enterprises today require more flexibility and agility in modeling techniques to add new functionalities or to modify existing ones in current enterprises.

One of the main advantages of representing services as basic units to model an enterprise is related to the reuse of high-level services. The services can be used as a key mechanism to adapt the enterprise to new market conditions by including new services or by the modification of the existing services. This is because a service represents a self-contained organizational unit with a weak coupling with other services. This makes it possible to accomplish modifications in the service structure without disturbing the structure of the other services in the same environment.

The service-oriented architecture enables the enterprise to manage the complexity of each service in an incremental process starting with a high-level description of the services, and finishing with a low-level description of the processes that compose the service. This approach enables us to make improvements or replacements of processes without altering the abstract representation of the services offered by the enterprise.

From the business modeling perspective, business services are relevant because they enable the analyst to focus on high-level descriptions rather than analyzing the entire organization in detail in the first modeling approximation. In this sense, Jones (Jones 2005) argue that "the organizations are, at the high level, first focused on the what (key functions) and only secondly on the process, the how". The proposed business service architecture adopts this philosophy by introducing a service global model that represents all the service offered by the enterprise without details about their implementation by a means of business processes.

5.2.4 The characteristics of business service orientation

Although a standard set of service-orientation principles does not exist in the current literature, there is, however, a common set of principles that are associated with service orientation (Erl 2006): services are autonomous, share a formal contract, are loosely coupled, are composable, are reusable, are discoverable and they abstract underlying logic.

Following, we discuss how the business service-oriented approach presented in this thesis complies with these set of principles of service orientation. To do this, we have adapted the definitions provided in the Erl works (Elr 2006) according to the proposed business service architecture:

*Business services are autonomous***.** All the business activities needed to satisfy the services reside within an explicit boundary. Therefore, a business service does not depend on other services to be executed. In practice, real business services offered by an enterprise are conceptualized and implemented as autonomous entities that could be offered in isolation to final customers.

*Business services share a formal contract***.** The business service can be considered as a contract between the provider and the requester. This contract describes the set of interactions and information exchanges needed to provide the service. The service contracts provide a formal definition of (Elr 2006):

- The protocol for requesting the service. This establishes the set of requirements for the customers needed to start the service.
- The set of business processes that make operational the service.
- The set of inputs and outputs of the service, which must be encapsulated in the service definition.
- The set of restrictions to perform the service. The restrictions can be imposed by internal or external entities.
- The dependencies among customers and provider.
- The protocol to finish the execution of services. This defines the conditions that must be fulfilled in order to finish the service.

The *i** models that describe business services can be considered as a service contract at the class level. In the case of business services, the contracts are commonly named as *terms and* *conditions*. The restrictions placed in the contract will be the conditions that the customers must accept in order to use the service (instance level). Since the business service tries to model the real world, the definition of the contracts is given by external entities that regulate the service offering.

Business services are loosely coupled. The set of dependencies among requesters and providers that make up the service must be limited in order to comply with the service contract. The loosely coupled characteristic of the service orientation enables the analysts to respond to unforeseen changes in the market in an agile way.

*Business services abstract underlying logic***.** The business logic beyond the abstract service definition must be invisible to the outside world. In fact, only the activities declared in the interface are visible to the service requester. In most cases, a business service can be established as a black box where the underlying logic of the service is hidden to the service requesters.

In the current literature, there is no consensus about the appropriate abstraction level to define services. In some cases, it might be interesting to define services as very general business functionalities; in other cases, it could be more convenient to design services as individual business processes that offer a value to final customers. The definition of the appropriate abstract level will depend on the granularity that can be detected in the enterprise.

Business services are composable. One of the main objectives of the thesis has been to provide a solution to the scalability issues in *i**. The source of most scalability problems is the lack of mechanisms to manage the composition. These problems are eve greater when the model grows in size and complexity. In this case, mechanisms are needed to appropriately manage the levels of granularity.

In our proposal, complex services can be decomposed into basic services, allowing the service description to be represented at different levels of granularity. The definition of a model that represents the abstract definition of the services offered by the enterprise make it possible to orchestrate the services that collaborate to satisfy the goals of the enterprise. This characteristic promotes reusability and the creation of service abstraction layers.

Business services are reusable. Regardless of whether immediate reuse opportunities exist, services are designed to support potential reuse. Service orientation promotes the reuse by generating services and service components with a minimized dependency on other service components. The definition of self-contained components enables the analyst to reuse them with minimal modifications.

In this thesis, each service needs to be defined in an isolated way in order to promote reusability. Reuse is one of the key objectives of the proposed business service approach.

Business services are discoverable. The enterprise must make it public fragments of its business process in order to allow the customer to use the business service. All the customer-provider interactions must be encapsulated in a specific business service. The explicit modeling of the protocol for requesting a service (this is the *means* to discover the service) enables the analyst to avoid the specification of redundant services. The explicit representation of business activities needed to support a service avoids the specification of services that implement redundant behaviors.

5.3 A Business Service Architecture for the *i** Framework

To make the practical application of the business service orientation possible, the business service architecture must be introduced. This architecture must provide definitions and precise alignments for the concepts used in the proposal. We have also developed meta-models to help understand business services and the relationships among the components of this model.

Some of the definitions of the business services architecture are adaptations of the concepts presented in the W3C description for web services architecture (W3C Working Group 2004). This was a conscious decision to be able to generate a definition that is compliant with the current standards for defining services.

5.3.1 The service-oriented strategy

The key idea of the service-oriented approach is to use the business services as building blocks that encapsulate internal and social behaviors. Therefore, complementary models were defined to make it possible to reify the abstract concept of service in low-level descriptions of its implementation.

The business service architecture is composed of three complementary models (Figure 5.2) that offer a view of what an enterprises offers to its environment and what enterprise obtains in return:

- *Global Model*. In the proposed method, the organizational modeling process starts with the definition of a high-level view of the services offered and used by the enterprise. The global model permits the representation of the business services and the actor that plays the role of requester and provider. Extensions to *i** conceptual primitives are used in this model.
- *Process Model*. Once business services have been elicited, they must be decomposed into a set of concrete processes that perform them. To do this, we use a process model that represents the functional abstractions of the business process for a specific service. This model provides the mechanisms required to describe the flow of

- multiple processes. Extensions to *i** conceptual primitives are used in this model.
- *Protocol Model*. Finally, the semantics of the protocols and transactions of each business process is represented in an isolated diagram using the *i** conceptual constructs. This model provides a description of a set of structured and associated activities that produce a specific result or product for a business service. This model is represented using the redefinition of the *i** modeling primitives (which is explained in Chapter 4).

The proposed approach enables the analyst to reuse the definition of protocols by isolating the description of the processes in separate diagrams. In this way, the process model represents a view of the processes needed to satisfy a service but without giving details of its implementation. Each business process is detailed through a business protocol. The detailed description of the protocols is given in the protocol model. The protocols are represented using the *i** notation.

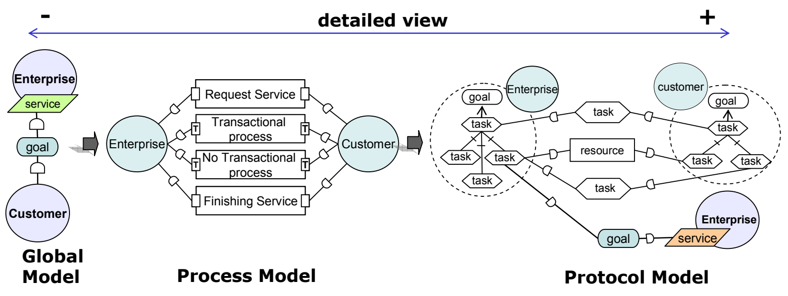


Figure 5.2 The business service proposal

The main idea of this approach is to promote the granularity of the service definition by isolating the organizational behavior of each business service in a separate business description. The meta-model that represents these elements of the business service proposal for the *i** framework is presented in Figure 5.3. The meta-model is represented using concept maps, which is an informal graphical way to illustrate concepts and relationships. The boxes represent a concept and the arrows represent relationships. The concept maps help the analysts to rapid navigate the key concepts to know how they relate to each other.

The proposed meta-model establishes that business services are the mechanism to fulfill the business goals. The business services are composed of business processes, which can be defined as transactional processes or non-transactional business processes. Business services and business processes are both represented by using the proposed extension to the *i** modeling primitives. Finally, the meta-model indicates that business protocols, which are the low-level specification of a business service, need to be represented with i* modeling primitives. This protocol model represents the organizational behavior that is needed to perform a business process.

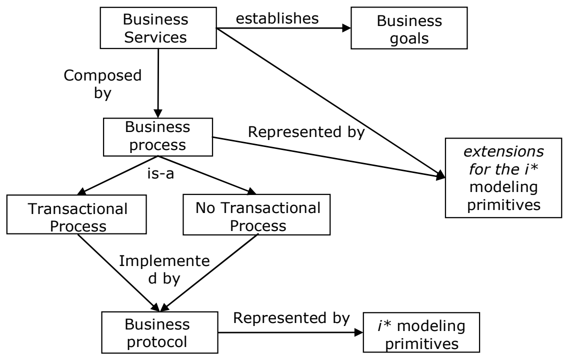


Figure 5.3 The meta-model of the service-oriented architecture components

5.3.2 Overview of engaging a business service

In practice, there are many ways to implement business services, but, generally speaking, the following steps are required. 1) The service requester requests the service following the established protocol, 2) The service provider analyzes whether or not the service requester fulfills the conditions for the service, 3) The service provider agrees or disagrees to provide the service, 4) The service semantics are performed by the requester and the provider actor, and 5) The requester and the provider actor agree to finish the service. Figure 5.4 presents a simple pattern of the steps to perform a business service.

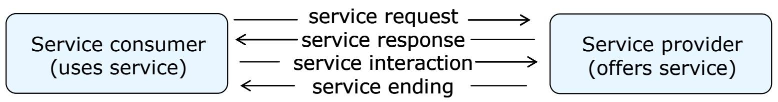

Service consumer (uses service)
service request
service response
service interaction
service ending
Service provider (offers service)

Figure 5.4 The basic pattern of the process of engaging a business service

5.3.3 Implications of the service-oriented strategy

As a result of the service orientation, the focus of the *i** modeling activity has been modified to consider services as the basic decomposition unit.

In the original *i** approach, the modeling process starts by determining the participating organizational actors in the enterprise. Once the actors are elicited, their internal activity must be represented using the *i** relationships: means-end, task decomposition, and contribution links. This is why the decomposition mechanisms in *i** are limited to the actor's boundaries.

We argue that this current *i** approach can be useful in the context of small cases studies with a limited number of involved business processes. This approach is also very useful when the modeling activity is focused on designing new enterprises, starting "from scratch". However, in cases where the model grows in size and complexity, or in cases when the modeling activity focuses on the representation of the current situation of an existing enterprise, then it could be complicated to represent the fragments of the model that correspond to several business processes that are represented in the strategic rationale model (Figure 5.5).

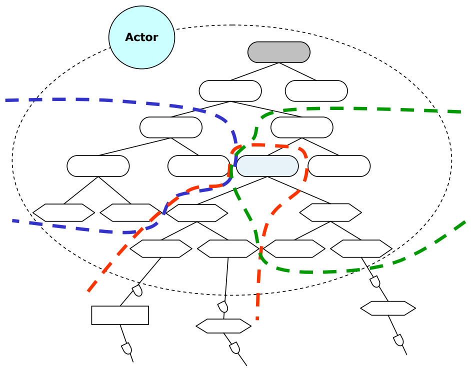


Figure 5.5 Actor as basic concept for decomposition in *i**

The service-oriented architecture proposed in this thesis changes the modeling focus from actors to service descriptions. In this sense, the decomposition applies to services (which represent business functionalities) that are reified in business processes. Therefore, the service approach breaks the modeling actor cohesion of the original *i** definition (Figure 5.6). This implies a change in the business modeling approach since the method proposed starts with the definition of services in place of starting with the identification of goals of the current *i** process. We consider that, in the initial modeling stages, it is more convenient to first answer the question *what we do* rather than the question *how we do it*.

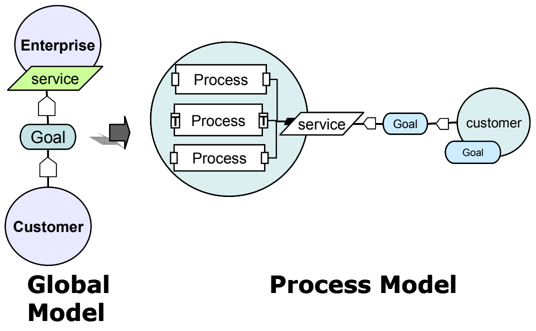


Figure 5.6 Services as basic decomposition mechanism

5.3.4 The service-oriented components

To make the application of the service-oriented architecture possible, several aspects need to be considered where modeling business processes. The set of service components are presented above:

- Intentional elements
- Actors
- Business services
- Service requester and provider
- Service request
- Service visibility
- Service delegation rules
- Business processes

Each one of these components, which influences the definition of the service architecture, is analyzed in detail below.

5.3.5 Intentional elements

In our service-oriented approach, we adopt the intentional elements of the *i** framework: goal, resource, softgoal, task and dependency. A goal represents the strategic interests of a business actor. In *i** we distinguish hard goals from softgoals. A softgoal represents a goal whose fulfillment conditions can be clearly established. Softgoals are typically used to model non-functional requirements. A task specifies a particular course of action that produces a desired effect. The execution of a task can be a *means* for satisfying a goal or a softgoal. A resource represents a physical or informational entity. We have also adopted the graphical notation of these intentional elements. Dependency between two actors indicates that one actor depends, on the other (for some reason) in order to attain a goal, execute a task, satisfy a softgoal, deliver a resource, or provide a service. The former is called *depender* and the latter is called the *dependee*. The object around which the dependency centers is called the *dependum*, which can be a goal, resource or plan.

The graphical notations associated with these concepts are represented in Figure 5.7.

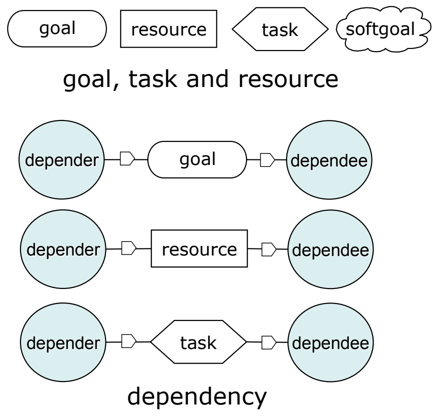


Figure 5.7 Graphical notation for intentional *i** modeling concepts

5.3.6 Actors

In this proposal, the business actor concept models an independent intentional organizational entity (person, functional area, department, or enterprise) that uses or offers services. According to the *i** philosophy, the actor has strategic goals and intentionality within the organizational setting.

A business actor is graphically represented as a circle with the name of the actor, as defined in the original *i** definition.

In the *i** framework, the actor could be specialized into agent, roles and positions (Yu 1995). The agents represent specific instances of people, machines, or software with concrete physical manifestations that occupy a specific responsibility within the enterprise. The agents can play several roles. A role is an abstract characterization of behavior within some specialized context. In a *is-a* relationship, all specialized sub-roles inherit all properties of the generalization super-role. A collection of roles describes an actor's position.

One of the advantages of this representation of actors in *i** is the possibility to represent actors at the class and instance level in the same model. Therefore, it is possible to represent classes of actors (agents), as well as specific instances of these classes (roles). This is very useful to represent, for example, existing software systems that support business activities, as well as to represent generic actors that have a clear role in the business processes. Figure 5.8 shows an example of the representation of agents and roles following the *i** notation.

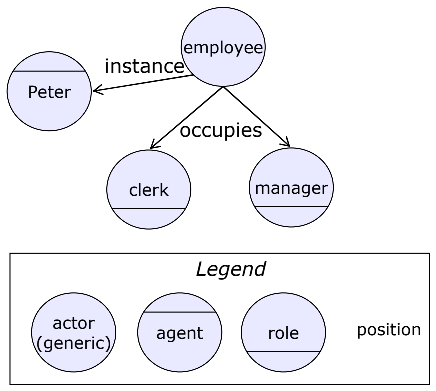


Figure 5.8 Agent, roles and positions

5.3.5.1 Actor composite structure

The *is-a* relationship has been used in actor modeling to represent the specialization of generic actors into specific agents playing roles in the organizational context. However, this relationship does not correspond to the hierarchies of organizational structures for business actors, where the key concept is the subordination of actors according to actor hierarchies. Therefore, we use the concept of composite actor structure in order to represent the hierarchical relationships between the actors.

The composite actor structure associates the actors by subordination links. The *subordinated by* link reflects the hierarchical dependencies that exist in a chain of command (line of authority and responsibility through which orders are passed within a social unit) in real enterprises. The key concept about subordination is that if an actor subordinates another actor, then the first can delegate activities to the latter. The subordination implies that if one actor subordinates to another actor, then the first one is responsible for the behavior of the second and it can implement monitoring mechanisms to control and evaluate the subordinated actor's work (Giorgini 2006).

The subordinated relationship enables the analyst to represent the capability of an actor to assign responsibilities to its subordinates (Figure 5.9). One of the implications of the actor composite model is that "the occupants of superior roles inherit all the positive access rights of their inferiors, and conversely ensures that the occupants of inferior positions inherit any prohibitions that apply to their superiors" (Moffett and Lupu 1999).

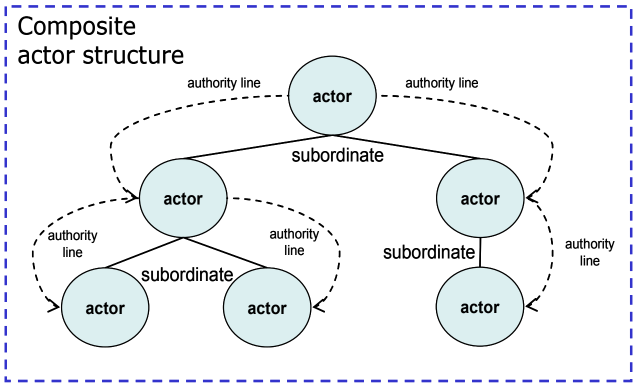


Figure 5.9 The composite actor structure

The subordination link implies the following control principles (Moffett and Lupu 1999):

- Delegation: this is one of the main concepts for activity decentralization. Delegation is based on the fact that it is

- impossible for one person to directly manage all the activities related to offer certain business functionality. The delegation of responsibilities enables the delegate actors to have full authority to carry out their delegated activities. In this thesis, we argue that delegation can be done through the definition of subordinate actors.
- Supervision: This is an activity that is carried out on someone by someone else in an immediate superior position in the organizational hierarchy. Supervision does not imply the observance of a specific activity; it implies the monitoring of subordinates to make sure that the tasks are being correctly executed. Subordination needs to be applied because even if the *delegator* is no longer responsible for the delegated tasks; this actor is responsible for the activity and is therefore responsible for ensuring the execution of the delegated tasks.
- Review: This is the opposite of supervision concept; a review is carried out on specific activities.

The actor hierarchy based on subordination links can be equivalent to hierarchical models based on actor aggregation. This is because, in hierarchies based on actor aggregation and actor subordination, the aggregated actor is responsible for a larger number of activities than the subordinated actors. In both approaches, the rule is that the activities can be delegated from the aggregated to the parts. Also, both (hierarchical models based on subordination and hierarchical models based on aggregation) share similar approaches to define supervision and to review control principles.

The composite actor structure allows us to represent the situation where several actors collaborate to provide a service to customers.

As stated above, the concept of composite actor involves the representation of organizational actor hierarchies, which allows us to explicitly analyze the delegation of responsibilities to properly provide a service. An example of a composite actor structure is shown in Figure 5.10 for the car rental case study.

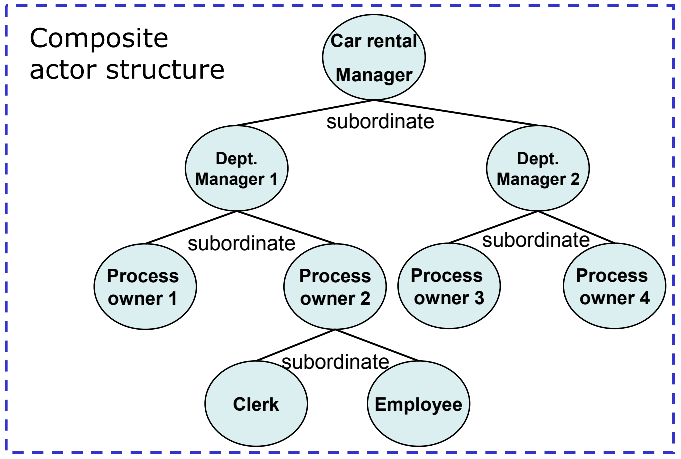


Figure 5.10 The composite actor structure

An important issue in the representation of actors and services is the determination of the responsibilities to execute the business tasks. In this context, an actor can be responsible for the service, but this actor often does not take an active role in the performance of the service, which is delegated to other actors. Based on this knowledge, it is possible to categorize the actors into *internal* and *offered* based on their visibility to service requesters.

5.3.5.2 Actor Types

We have distinguished two kinds of internal actors: those actors that are responsible for the business service (normally these actors are the department managers or the directors of the enterprise); and those that perform the business processes needed to implement the business service (normally these actors correspond to the intuitive notion of *employees*). Normally, actors of this kind do not have relationships with the final customers of the offered business service.

The external actors are those that directly interact with the customers using the interface for offering and requesting the service (corresponding to the intuitive notion of *clerks*).

Figure 5.11 shows the representation of the types of actors involved in a business service. This model uses organizational actor hierarchies to represent the actor(s) responsible for the service, the actor(s) that perform the business processes, and the actor(s) that interact with the customer to offer the service. The arrows in the model indicate the delegations of responsibilities based on the organizational hierarchy. The figure represents the decomposition of services into business processes, and the further decomposition of the processes into specific organization tasks. It is important to point out that delegation is defined by following the formal clauses of the architecture. Thus, the model shown in Figure 5.11 is not included in our service-oriented approach and is only used for explanation purposes.

The delegation of the service components to specific organizational actors is based on the subordination chain represented in the composite actor structure. The delegation rules will be explained in detail in section 5.3.11.

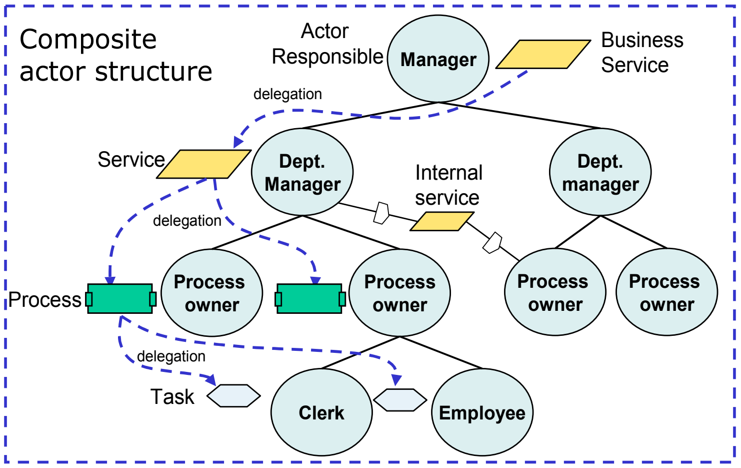


Figure 5.11 The composite actor structure as basis for delegating services

The composite actor structure enables us to clearly define how each business service is associated to a responsible actor, who will have the responsibility to execute or delegate it to subordinated actors. This model also makes it clear how the subordinated actors of the service owner are the actors that are responsible for the processes needed to perform each business service (Figure 5.12). It is important to point out that this model is not included in the service-oriented architecture and it is only used the explanation of the relationship between the composite actor structure and the organizational structure.

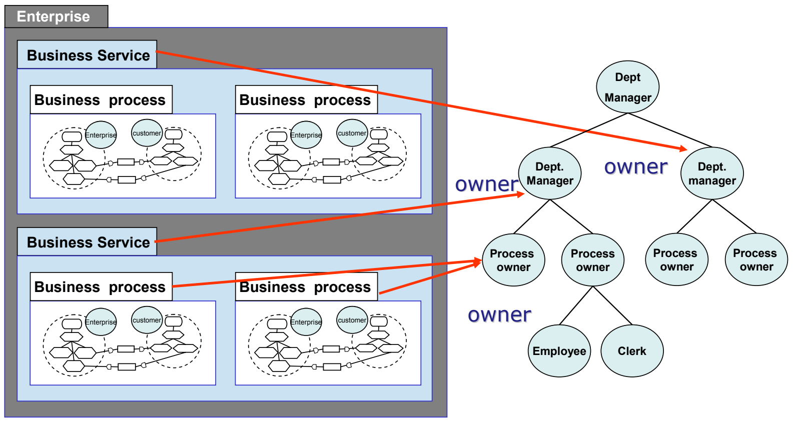


Figure 5.12 The composite actor structure as basis for ownership determination

5.3.7 Business Services

A Business Service is a self-contained, stateless business functionality that is offered to potential customers through a well-defined interface. Ideally, business services should not depend on the context or state of other services. Thus, a business service should be viewed as an abstract set of business functionalities that are provided by a specific actor.

It is important to point out that the concept of services that we use in this thesis concerns functionalities at the organization level and interactions among organizational actors and companies, rather than functionalities offered by software systems and machine-to-machine interactions (such as web services). Business service modeling is relevant to accurately determine the kind of organizational work performed by the organization, which is independent of any future, concrete implementation. This implementation could be done using web services, but it is important to avoid the potential confusion associated with the use of the term “service”. Business services are high-level descriptions of basic, cohesive and relevant activities of a given organization.

The business services have been represented using an extension of the notation of the *i** framework. The concept of dependency provided by the *i** framework has been modified to appropriately represent the social agreement between customers and providers.

In extension to the *i** notation, the goal dependency must be linked with a business service placed on the boundary of the service provider. A business service is graphically represented as a parallelogram that is located on the boundary of the business actor.

The goal dependency involved in services description indicates that the customer depends on the provider in order to satisfy a certain goal through a specific business service. In the graphical representation, the service has been placed in the boundary of the provider actor to indicate that the business service is the only interface between providers and requesters. The arrows of the dependency must always be directed toward the service provider. Each actor can provide 0…n services, and each service can be requested by 1…0 end consumers. This indicates that an organizational actor is not obligated to offer services; however, if a service is offered, then, there must be at least one potential consumer. Figure 5.13 shows the simplified view of the service notation, where only the business services are presented and the internal goals of the involved actors are hidden in order to simplify the model.

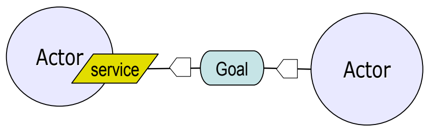


Figure 5.13 The business service notation (simplified model)

It is important to point out that this definition of service based on the concept of dependency places emphasis on the service as business functionalities offered to external customers, rather than considering services only as resources that offer values to the customer. We argue that physical or informational resources must be defined in the context of a specific business service; therefore, the resources are represented in subsequent modeling phases where the business processes that perform a service as defined.

The business service plays the role of interface between the provider and the requester. This indicates that all interactions among these actors must be contained in the definition of the service. It also indicates that the service is the only mechanism that is allowed to associate the enterprise and the customer. This characteristic enables the analysts to encapsulate all business processes associated with the service in the abstract concept of service interface.

The abstract definition of business services as interfaces makes it possible to define a high-level view that represents all the services offered by an enterprise, hiding all the details of the implementation of the services. In this sense, it is important to point out that the view presented in Figure 5.13 is focused on representing "what" a business service is about, rather than on the implementation of the offered services. An extended version of this model (global model) could be used to represent "why" the services are offered or requested. To do this, the offered services are explicitly associated with internal goals in the customers and providers (Figure 5.14).

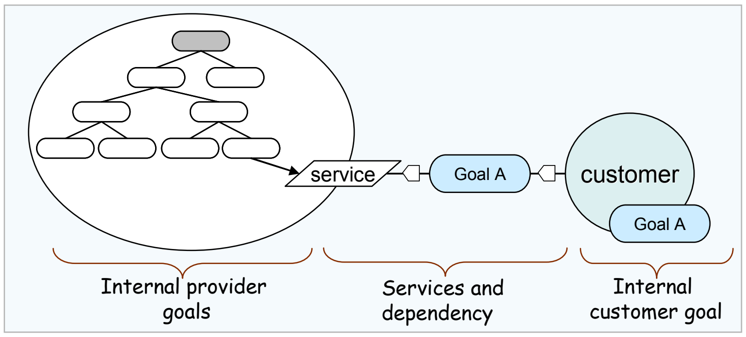


Figure 5.14 The business service notation (expanded model)

The modification of the *i** notation to represent business services also offers a solution to a well-known specification problem with original *i** notation for goals and task dependencies.

The original *i** task dependency implies the existence of a precise procedure to accomplish the activities involved in the task. In task dependency, the *depender* actor is the actor that must establish the procedure to execute the plan. In goal dependency, the *dependee* actor is the actor responsible for satisfying the goal. This is why the *dependee* must take all the needed actions and decisions in order to satisfy the goal, and the *depender* actor does not care about how the goal is fulfilled.

This semantics seems to be enough to solve the possible ambiguities in defining goal and task dependencies; however, this is very limited when modeling complex real cases. With the original *i** notation, it is not possible to define "clean" descriptions of the scenario where there is a precise procedure to accomplish an activity (task) but where the actor that prescribes the procedure is the *dependee* actor and where the vulnerable actor is the *depender* actor. Figure 5.15 shows an example of this situation. In this example, the actor that becomes vulnerable in the dependency relationship is the *depender*. Also, the actor that prescribes the activity is the *dependee*. This makes the description incorrect for the original restrictions that define tasks dependencies in *i** (where the actor that prescribes the activity must be the *depender*). However, this is not an isolated example, and we found this kind of specification to be frequent in real case studies.

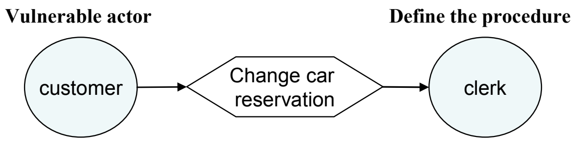


Figure 5.15 An inconsistent task dependency

We argue that there exists an overuse of the goal in *i** or Tropos models. Most non-expert analysts use goal dependencies to represent activities that could be clearly classified as task delegation. Therefore, the example in Figure 5.15 would be represented as a goal dependency, although the *dependum* clearly represents a task dependency.

In the specific case of the service orientation, the *dependee* (service provider) is the actor that establishes how the activity must be performed and the *depender* (customer) is the actor that becomes vulnerable if the service is not satisfied. This service description can't be represented by using the "pure" *i** notation. The proposed notation for business services enables the analyst to precisely describe this situation by using an extension of the goal dependency. Three clear differences could be found between the concept of *i** task dependency and the proposed business services specification:

- Task dependency does not indicate the implication of the *depender* actor (user) in the execution of the tasks. In task dependency the *depender* (customer) delegates all the responsibility to the *dependee* actor. Service representation indicates the implication of the *depender* in the execution of some activities associated with the service.
- Task dependency does not necessarily imply further decomposition. Sometimes, it is possible to decompose a task (in the boundary of the *dependee* actor) that has been

- delegated, but it is also true that sometimes the delegated task could be executed without further decomposition. Service description implies the reification of the service into more concrete behavior representations. In this case, service specification always implies that services must be decomposed into a set of business processes to perform them.
- Task dependency implies the *depender* as the actor that prescribes the activity. Service dependency implies the *dependee* as the actor that prescribes the way to perform the service.

In this proposal, two types of business services are distinguished: basic and composite business services.

5.3.6.1 Basic and composite business services

A basic business service is an atomic building block that still represents a service. Therefore, a basic service is decomposed into processes without further service decomposition.

A composite service aggregates multiple business services and implements mechanisms that coordinate the aggregated services. Therefore, a composite business service is a service that is composed of other composite or basic business services.

The feature model proposed in Czarnecki´s research works (Czarneki et al. 2000) have been used to manage the variability in service composition. The four features proposed by Czarnecki enable the analyst to represent the several possibilities that exist to combine business services: *mandatory*, *optional*, *alternative*, and *or* features. Thus, we found the feature model to be the appropriate mechanism to represent the aggregation of complex and basic services. The feature model is detailed in Chapter 6 where the service-oriented method is explained.

Figure 5.16 shows the representation of basic and composed services.

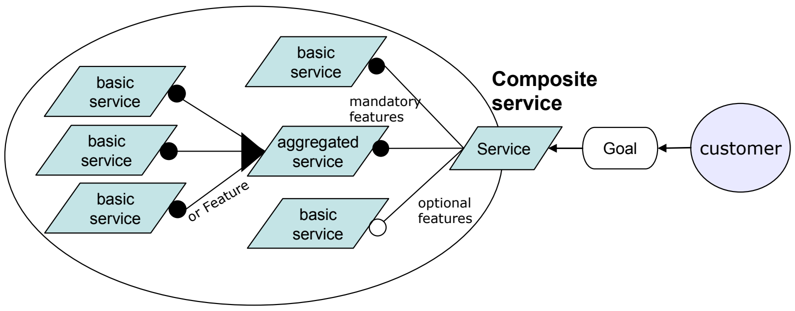


Figure 5.16 Composite and basic service configuration

An example of basic and composite business services is shown in Figure 5.17 for the running example. In the case of the *integrated travel planning* composite business service, it consists of the aggregation of services to reserve a flight, a hotel and a car for a specific trip. In the case of the *walk-in reservation* basic business service, it is directly implemented by *check car availability* and *formalize reservation* business processes that will be represented in the process model.

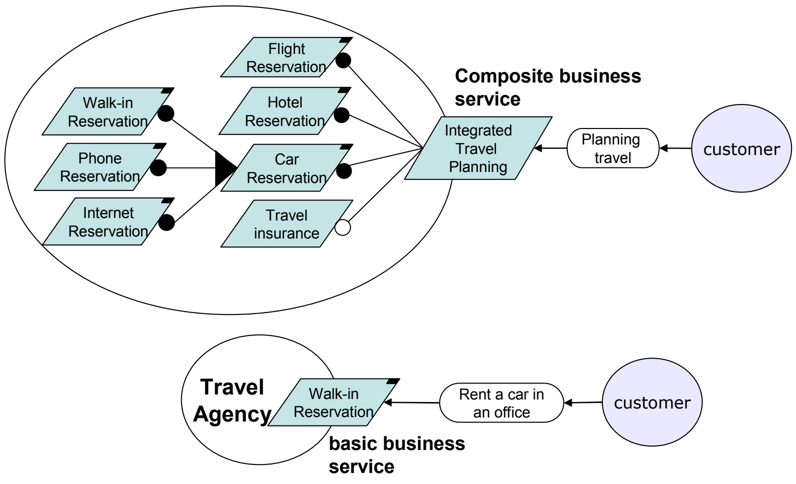


Figure 5.17 Examples of composite and basic service

In the case of a basic business service, this needs to be decomposed into a set of business processes to perform it. Figure 5.18 shows the meta-model for the basic and composed business services proposed in this thesis.

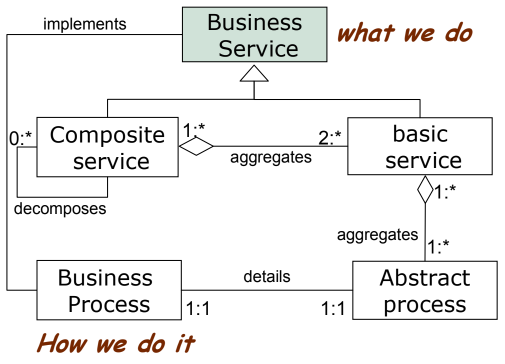


Figure 5.18 The meta-model for composite and basic services

According to this meta-model, the composite and basic business services are specializations of the class business service (inheriting the basis characteristics of services, such as stateless, loosely coupling, modularity, etc). Each basic business service is an aggregation of abstract processes that must be detailed by using a business process model. Thus, the concrete business processes are the means for implementing a specific business service. This service configuration enables us to consider the business services as the abstract representation of *what to do* while the business process model represents the representation of *how to do it*.

Another important classification of services categorizes these in *supporting* and *offered*, depending on whether the services have visibility only in the boundary of the enterprise or if they are visible to external actors in the environment.

5.3.6.2 Offered and supporting business services

An offered business service is a functionality that an enterprise offers to end customers. Therefore, services of this kind are requested by a number of external customers that use the service interface to interact with the service provider.

To provide this functionally, the enterprise publishes a fragment of a business process as the interface with the potential customers. The customers interact with business services in a manner prescribed by the restrictions that are imposed by the enterprise that offers the corresponding service or by external entities that regulate the service.

In accordance with this service classification, the offered business services are those services that offer a certain fragment of the business functionality to potential customers (persons or companies) to request and to use the service.

An example of this business service is shown in Figure 5.19. This figure shows the example of the service *Walk-in Car Rental* offered by a Car Rental Company to potential walk-in customers. We use the concept of dependency to indicate that the customers depend on the Rental Company to use a service to fulfill their strategic goals. This dependency also indicates that the Company offers the service to potential users.

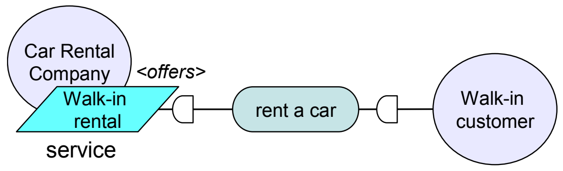


Figure 5.19 An example of an offered business service

A supporting business service is a functionality that an organizational entity (functional area, department, organizational actor) offers to internal entities of the enterprise. In this case, the supporting business service represents the means that fulfill the end (in the sense of the goal represented by the offered business service). The supporting business services provide support to business processes or other supporting business services. In the same way that offered business service can provide a supporting function to multiple external end customers, supporting services can be requested and consumed by multiple business services or business processes.

It is important to point out that; in most cases there exists a natural coexistence of offered and supporting business services. Following the car rental example, we found that one of the services that is needed to satisfy the service *check rates* is the service *check car availability*. This service that cannot be offered to end customers and it can only be managed by the clerk of the renting company, who is an internal actor of the enterprise.

Figure 5.20 shows an example of supporting business services associated with the offered service *Walk-in Rental*. In this example, the organizational unit responsible for offering the service uses the services offered by other organizational units. A branch could request a car from another associated branch (of the same company) if a car is not available in the first one. The branch can also request an analysis of the customer in order to approve or deny the rental. In this proposal, an offered business service could be executed by using a set of supporting business services. As stated above, only the offered business services are visible for external customers.

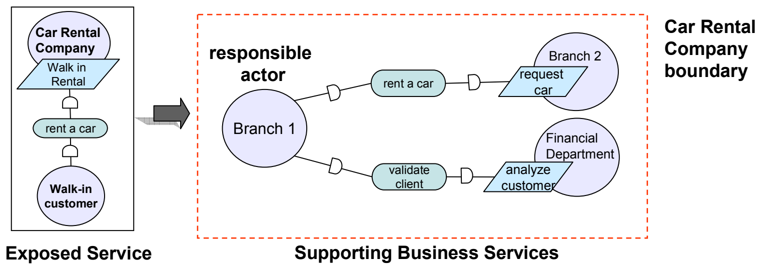


Figure 5.20 An example of supporting business services

Figure 5.21 shows the meta-model services for offering and supporting services proposed in this research work.

The distinction between basic, composite, supporting and offered services makes it possible to create a consistent organizational model made up of the set of business services. This allows us to encapsulate organizational behaviors in cohesive building blocks.

The meta-model indicates the following: a) the business services can be specialized into supporting and offered business services, b) both supporting and offered business services are composed by a set of abstract processes, and finally, the abstract processes are refined into business process.

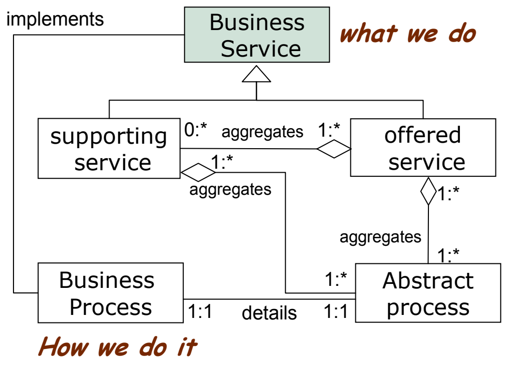


Figure 5.21 The meta-model for supporting and offered services

5.3.8 Requester(s) and Provider(s)

The objective of an enterprise is to offer services to customers in order to fulfill its strategic goals and to provide added value to its customers.

The service provider is the person or organization that offers the business service to potential customers. The service requester is the person or organization that wants to use the service in order to fulfill her/his goals. According to the *i** modeling approach, the requester depends on the service provider to increase its capabilities.

In our business service-oriented proposal, as in the case of service at the implementation level, the requester is usually the one that initiates the service activity.

An enterprise can provide services to other enterprises as well as consume services from other external entities. Thus, a service provider can also play the role of service requester in the same business configuration. Figure 5.1 shows an example of an enterprise playing the role of requester and provider.

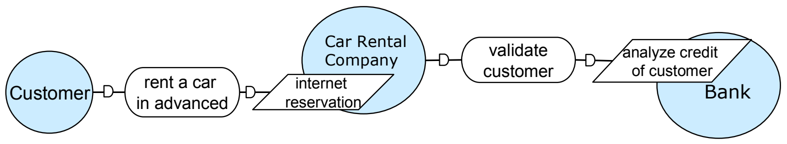


Figure 5.22 Example of an actor playing the role of requester and provider

In this example, the Car Rental Company plays the role of provider for the offered business service *Internet Reservation*; it also plays the role of requester of the service *Analyze credit of customer* offered by the entity Bank. It is important to point out that this kind of double role can also be found in the specification of supporting business services.

An advantage of the proposed service-oriented approach is the support of the social and intentional analysis. Some concepts must be considered in order to represent the social and intentional aspects that support the relationship between requester and provider: ownership, provisioning, request, trust and dependency. We have adopted these concepts from the works of Serenity project (Asnar et al. 2006). However, in this thesis, these concepts have been adapted to the proposed service orientation for the *i** framework:

- Ownership: ownership indicates that the provider is the legitimate owner of the business service. The service owner has full authority to perform the service or to delegate this authority to another subordinated actor. In our proposed models, ownership is presented graphically by placing the service on the boundary of the actor representation.
- Provisioning: provisioning indicates that the provider has the necessary capabilities to provide the service. It is

possible to develop an extended version of the global model to represent the internal behaviors needed to perform each business service that is provided by the enterprise. The extended version of the global model represents the manner in which the abstract goals are refined into low-level actor's activities that are required to satisfy the service.

- Request: This indicates that the requester intends to achieve its goals by using an offered business service. The service request is graphically represented by joining a goal dependency (directed from the requester to the provider) with the service that is located on the boundary of the provider.
- Trust: From the requester's point of view, trust between two actors indicates the belief that one actor does not misuse the resource (informational o physical) involved in the business service execution. The former actor is called the *truster*, while the latter is called the *trustee*. From the provider's point of view, trust indicates the confidence of the provider on the requester to have visibility about the internal processes needed to perform a business service. In this thesis, the notation proposed by (Giorgini et al. 2006) has been adopted to graphically represent the trust in service models. The graphical notation corresponding to the Trust notion is represented in Figure 5.23.
- Delegation: this indicates that one actor delegates the permissions to carry out a specific activity (a business service) to another actor. By delegating the provider the responsibility to perform a business service, the requester extends its capabilities.

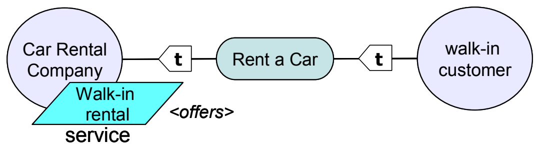


Figure 5.23 The graphical notation for trust

Delegation and Trust are two closely related concepts. Both concepts have a direct influence on the vulnerability of the requester actor. Thus, it is important to point out that requesting a service increases the capabilities of the requester actor, but it is also true that this actor becomes vulnerable if the service provider fails to deliver the service.

Figure 5.24 presents the meta-model that represents the requester and provider concepts.

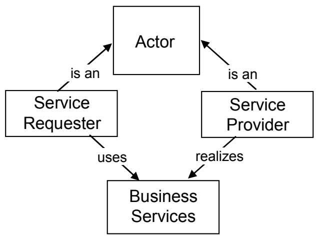


Figure 5.24 Meta-model for requester and providers

5.3.9 Requesting a service

In order to provide the functionality associated with business services, the enterprise must offer certain fragments of its business processes as an interface with potential customers. One of the fragments that needs to be offered is the mechanism for requesting the service.

It is necessary to point out that the services of similar contexts and domains have similar processes for requesting the service. This is because these services normally share the same regulations (of external entities) to offer services to customers, and also because these services share similar internal regulations and restrictions to perform their business processes. The similarities in the definition of protocols make the definition of organizational patterns for the basic protocols of the enterprise in similar domains possible. For example, most of the car rental companies shared the same "terms and conditions" for renting a car (the users need to have the permitted age, have a valid driver license and have a valid credit card). The terms and conditions enable the clerks to validate the authorized users.

As stated above, each business process is implemented through a business protocol that is represented using the *i** notation. Figure 5.25 shows the generic schema we propose to represent the protocol for requesting business services.

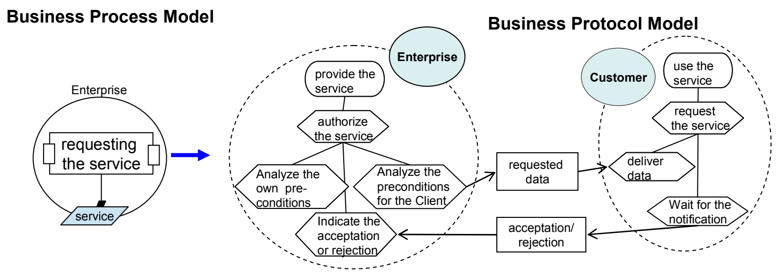


Figure 5.25 The generic schema for requesting services

An example of the processes needed for requesting a *Walk-in Rental* is shown in Figure 5.26 for the Car Rental Management case study. In this example, the following tasks have been represented in the protocol model to perform a walk-in rental: Request data customer, analyze customer information, check the rental preconditions, check the bank references of the customer (to do this, the car rental company uses an external business service provided by a banking institution), and finally, accept or deny the rental of the car to the walk-in customer. Note that, the protocol model is represented using the pure *i** notation. The reason for this is that *i** is well equipped to represent the behavior of the organizational actors and the rationalities of this behavior.

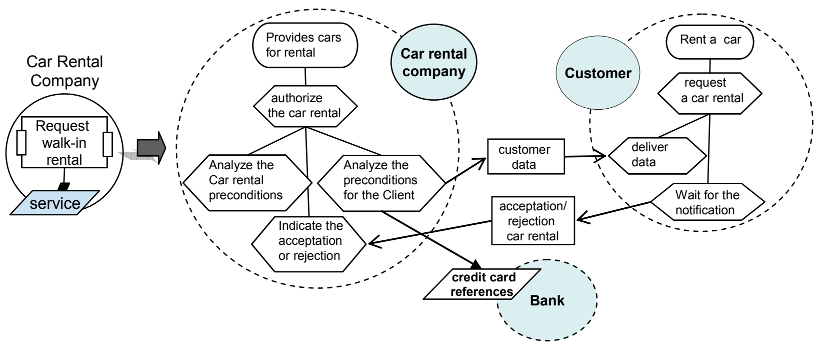


Figure 5.26 The requesting process for a Walk-in Car Rental

5.3.10 Business processes

The main idea of the proposed service approach is the reification of abstract representation of services into concrete business processes. A model that provides an abstract representation of the business processes is proposed (business process model) as an extension of the *i** framework. In this model, a concrete business process is represented as an abstract building block.

The processes represented in the model can be categorized into transactional and non-transactional. The definition given in (OASIS 2007) for transactional processes based on WS-transactions has been used to characterize processes in our service-oriented approach. To be considered transactional, a business service must fulfill the following ACID conditions:

Atomicity: The participants of the business service (provider and requester) must confirm or cancel the agreement about the service. The entire sequence of actions of the process must be either completed or aborted. The transaction cannot be partially successfully. In the case where some of the participants reject the service conditions, all operations of the business services must be cancelled.

Consistency: A consistent result of the business service must be obtained every time the service is executed. More specifically, the business process takes the resources (physical or informational) from one consistent state to another.

Isolation: the effects of the business service are not visible until all participants confirm or cancel. Intermediate stages of the process are not visible to the external world.

Durability: The effects of the transactions must be stored.

An example of a transactional process for the running example is the following:

The transaction begins:

- Customer selects a car to rent, and makes a rent request.
- Car rental company quotes price.
- Customer agrees to the price, and gives money to provider in exchange for object.
- Car rental company delivers receipt and acknowledges sale. The transaction is committed.

The transaction ends

In the case of non-transactional business processes, not all the processes that make up the business process need to be executed in order to perform it. In the case of a non-transactional process, some of the participants of the service can cancel some of the activities associated with the service without affecting the final result of this service. Therefore, no rollback activities are needed to solve the interruption of the service in an intermediate step. Also, in a process of this kind, the effects or intermediate stage of the business service can be visible during the service execution.

The generic schema of the process model is shown in Figure 5.27. The concept of *i** dependency has been adopted in order to represent the processes associated with a specific business service. The process dependency indicates that the requester delegates to the provider with the responsibility to perform the process. As in the same case as the service dependency, if the provider fails to provide the process, then the requester becomes vulnerable. It is necessary to point out that the main difference between task and process dependency is the granularity of their specifications. In the case of task dependency, we refer to a specific piece of work considered as a basic unit of work. Pre- and post-conditions can be defined as completion criteria for task dependencies. In the case of process dependency we refer to abstract activities that encapsulate a collection of related, structured activities that produces a specific service or product for a particular customer. Thus, a process can be broken down into specific tasks. This is the reason why a new primitive has been proposed to represent business processes.

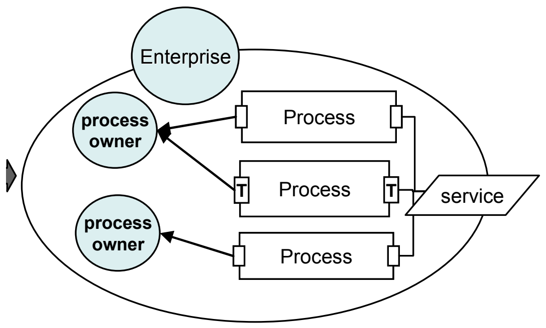


Figure 5.27 The generic schema for the process model

5.3.11 Visibility rules

One of the advantages of modularity is the possibility of using mechanisms to control visibility between the service requester and the service provider.

Two different kinds of visibility aspects have been considered in this work: actor and service visibility.

5.3.10.1 Rules for service visibility

In the most general case, the customers only have visibility of the offered business services. In this scenario, the customer does not have visibility of the supporting services needed to perform the offered services. Only the internal actors of the organization have visibility of the supporting business services. Three different scenarios were proposed to represent the different service visibility schemas: black box, grey box, and white box visibility.

In the case of the black box schema, the requester does not have visibility of the internal processes needed to perform the service. In the case of a grey box, the requester can introduce certain monitoring tasks to control the service, and finally, in the white box schema, the requester has total visibility of the composite processes of the service. The visibility schemas are explained in detail in Chapter 6 where the service-oriented method is presented.

5.3.10.2 Rules for actor visibility

In almost all cases, the customer of offered services does not have visibility of the internal actors that execute the task of the business service. This is because the customers of the services usually interact with the external actors of the business service (those playing what we have called the *clerk* role). However, in certain cases, the customer does have visibility of other internal actors of the enterprise.

Figure 5.28 presents an example of the schema for visibility of services and actors. This figure represents the standard schema for actor visibility, where the customer has visibility of the following elements: a) the offered business services, and b) the actors of the provider that interact with external customers (clerks). In this standard schema, the customer does not have visibility of the following business elements: a) the internal processes and the supporting business services that are needed to perform the offered business services. b) The customer does not have visibility on all the actors involved in service execution. c)

The delegation of services and tasks inside the enterprise boundary is also hided to the customers.

Based on these characteristics of the schema for visibility, the business service represents the appropriate interface between the providers and the service customers.

Figure 5.28 also presents the propagation of visibility of the actors within the enterprise boundary. In visibility of this kind, an actor can supervise the services and processes of the actor in the subordination chain that is defined in the composite actor structure. In this context, an actor has visibility of the services and processes of other subordinated actors.

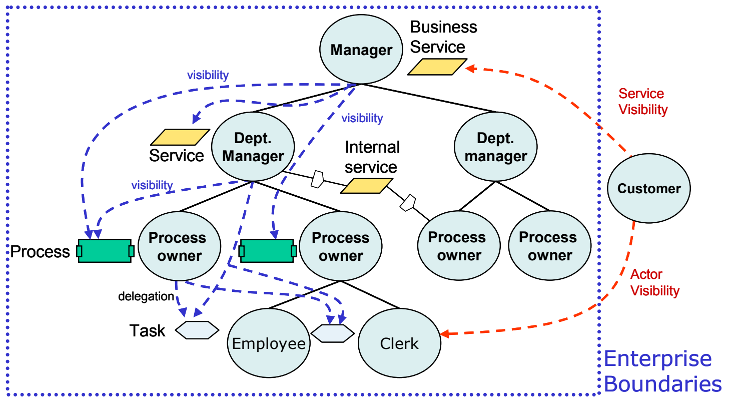


Figure 5.28 Visibility of services and processes

The visibility of services and processes is defined by using formulas explained in section 5.5. It is important to point out that the model in Figure 5.28 is not included in our service-oriented approach and is only used for explanation purposes.

5.3.12 Delegation rules

Based on the hierarchical model defined in the composite actor structure, the actor responsible for a business service can delegate it to its subordinate actors based on the hierarchical model defined in the composite actor structure. In this context, only the actor responsible for a business service can delegate the responsibility to perform the services or part of them (processes, tasks) to subordinated organizational entities (functional areas, departments, or internal actors).

The delegation of services allows us to define intentional relationships between the different entities that make up an organization in accordance with the *i** proposal. The reason is that delegation describes and identifies the situations where the actor responsible for the business service becomes vulnerable if the delegated actor fails to perform the service. The explicit delegation of responsibilities allows us to make an analysis of business process reengineering.

Figure 5.29 shows the schema for the delegation of services and processes based on the composite actor structure and the organizational structure of the enterprise. This model is used to exemplify that the Director of the company can delegate the responsibility to perform a service to his two subordinated managers. The managers can also delegate the service responsibility to their department managers. In the normal delegation schema, that needs to be defined in the composite actor structure, a manager can not delegate a service to another actor that is not in its subordination chain.

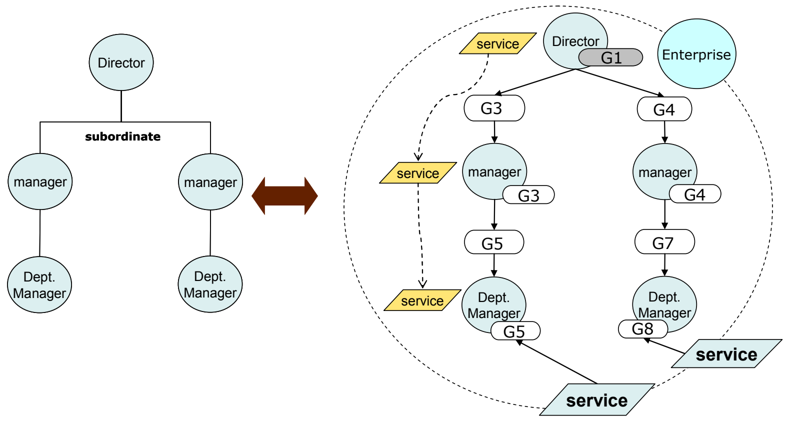


Figure 5.29 Delegation based on the composite actor structure

It is important to point out that delegation schema is defined by using formulas explained in next sections. In this sense, the model in right side of Figure 5.29 is not included in our service-oriented approach and is only used to exemplify the relation between the composite actor structure and the service delegation schema.

5.4 Architectural models

The business services approach architecture proposed in this work is composed of three complementary models: The global model, which represents the high-level view of the services; the process model, which represents the processes that compose each service; and finally, the protocol model, which represents the behavior of each business process.

5.4.1 The Global Model

The global model represents an abstract view of the services offered by the enterprise to potential customers (offered services). In this model, the business services are associated with the enterprise goals. Thus, it is possible to shows how the services are used to satisfy the enterprise goals and also to represent how a business service provides a solution to fulfill the goals of the service customers.

Two different views of the global model can be used according to the information that needs to be represented: the abstract view and the concrete view.

5.4.1.1 Abstract view of the global model

It is possible to provide a simplified representation of the global model that only shows the actors and their offered business services. This model hides the internal behaviors needed to provide the service. Thus, this model only indicates the services as black boxes and the goals dependencies associated to the business services.

Figure 5.30 shows the abstract representation of some external business services of the Car Rental Management case study. The abstract representation of the services in the global model enables analysts to use the global model to create the first agreements with the enterprise stakeholders.

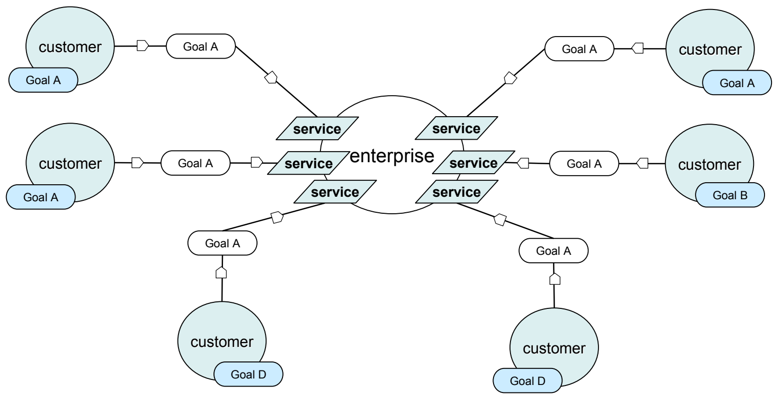


Figure 5.30 The abstract view of the global mode

5.4.1.2 Concrete view of the global model

In the concrete global model, the offered business services are linked with the internal goals of the provider actor. To do this, a goal-refinement tree must be defined to captures the existing reasons for each business service provided by the enterprise. Therefore, it is possible to show how the services are the mechanisms to satisfy the enterprise goals. This characteristic represents a novel approach for representing services because most of the current service-oriented approaches only consider the procedural aspects of the processes involved in the service, without considering the rationalities that exist behind these processes. Figure 5.31 presents an example of the concrete view of the global model.

The concrete view of the global model expands the abstract view of the global model by defining the goals of the actors and associating these goals with the services that are offered by the enterprise to potential customers.

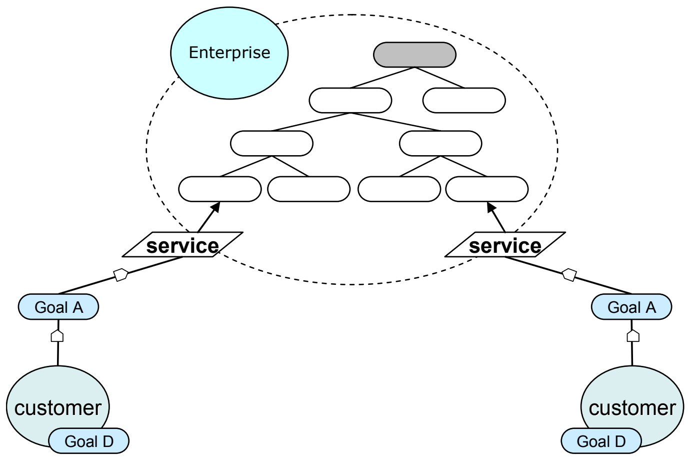


Figure 5.31 The concrete view of the global model

The explicit representation of the association between services and goals enables analysts to use goal analysis mechanisms to improve the performance of the model. To carry out this process, we propose the use the formal framework for goal reasoning proposed by Giorgini (Giorgini et al. 2002).

The formal framework for goal reasoning proposed by Giorgini allows us to evaluate qualitative and quantitative goal relationships, and also to detect and solve contradictory situations in the satisfaction of goals. Contradictory situations can be found, for example, when we want to allow for multiple decompositions of a goal G into sets of sub-goals, where some decompositions suggest satisfaction of G while other suggest denial. Therefore, different business services generated for goal decomposition could lead to contradictory situations.

The global model could be used not only for modeling the choreography of isolated enterprises, but also as a powerful mechanism to represent offered business services from different companies. The global model can be used to define the choreography between services of different enterprises. The choreography concerns describe externally observable interactions between the service provider and requester through a business service. In this case, the business services will define the description and semantics of the contracts among different enterprises, which must be accepted as use conditions.

The service global model is also used to define the relevant conditions that define a business service, visibility relationships, and trust policies.

As stated above, it is possible to define three different visibility schemas (black box, grey box, and white box) depending of the possibility of the service customer to monitor the activities associated with the business services.

In order to specify security and trust policies, which constrain the behavior of the requesters and the providers, we use a semantic extension of the *i** modeling construct proposed in Giorgini´s works (Giorgini et al. 2006).

The semantic richness of this proposal allows us to use the service global model to clearly represent the notions of delegation and trust for execution, as well as the delegation and trust for permission. The work of (Giorgini et al. 2006) also offers a well-founded solution to represent ownership relationships. Thus, it is possible to explicitly represent the actors that have permission to execute the business services. It is important to point out that graphical extensions to the *i** modeling concepts have been proposed in Giorgini´s work to fit the semantics of the security and trust policy concepts.

The values defined for the global model can be propagated through the processes that compose the service. In this way, the visibility rules defined for offered services can be propagated to provisioning services and business processes.

5.4.2 The Process Model

Once the external business services have been represented in the global model, each business service is refined into more concrete processes that are required to perform it using the business process model.

The process model uses the process dependency to represent the processes that compose each business service elicited in the previous modeling step. In this thesis, two types of processes are defined: a) the external processes which are those in which the customer plays a role by performing certain actions of the process. The processes for requesting and finishing the service are examples of external processes, b) internal processes which are those that involve only actions of the organizational actors of the enterprise.

Practical experiences revealed that there are processes that always need to be represented in a business service specification: the process for requesting the service and the process for finishing the service. In the former, it is necessary to represent the group of activities for initiating the services. In the latter, several actions must be performed in order to create an agreement about the finalization of the service. This is the reason why the requesting and finishing services are always specified in the proposed service-oriented architecture for the *i** framework.

There are several aspects that need to be considered in the definition of business processes: transactional properties of the processes, definition of authorized actors, execution order of the involved processes, delegation, and the visibility policies.

5.4.2.1 Transactional and non-transactional processes

The representation of the business processes in an isolated model makes it possible to represent transactional processes. Based on practical experiences, we have determined that there are processes that fulfill the conditions to be considered as transactions. As stated in 5.3.9, we adopt the concept of transactional process based on WS-transactions (OASIS 2007) to identify transactions in our service-oriented approach. We have adopted this specific proposal because the values for atomicity, consistency, isolation, and durability are more flexible than values defined in the original definition of transactions for software execution.

In our approach, a specific notation has been developed in order to represent the set of processes that compose a business service. The concept of milestone is also proposed to represent the order of execution of the business processes.

One of the key differences of our method and current *i** approach is the definition of processes. In our proposal, the processes are always associated to a specific business service. In current *i** approach, there is no a higher level that business process.

5.4.2.2 Authorized actors

The process model has the appropriate abstraction level to define the authorized actors to perform the processes of the business service.

Due to the current characteristics of the *i** framework, it is only possible to represent the authorized actors that fulfill the conditions to request the services. It is not possible to present those actors without authorization to use the service. There is also no mechanism to graphically represent the requirements for the authorized actors. For this reason, these requirements need to be represented in the actor's formal model specification in Formal Tropos specification (Fuxman et al. 2001).

In the Car Rental case study, the Rental Company establishes the policies regarding the actors that are authorized to rent a car (the minimum permitted age, a credit card, and a valid driver license). These requirements are shown in Figure 5.32. The use of the service is denied to customers that do not comply with these service regulations. This is why the regulations are usually used to specify the process for requesting the business service.

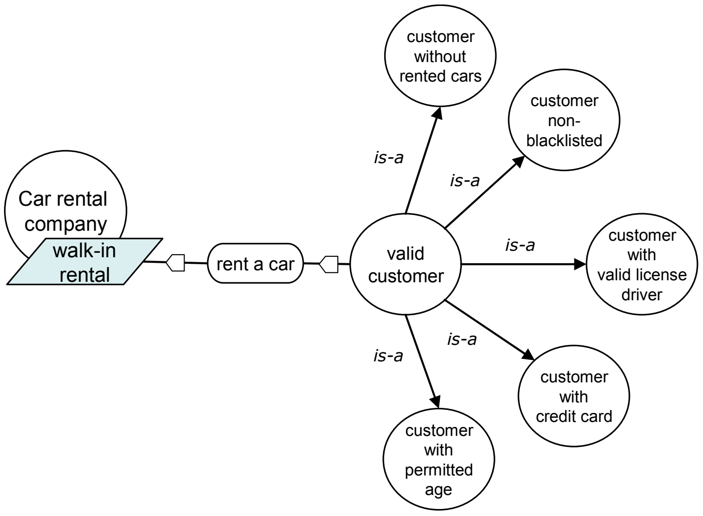

Car rental company
walk-in rental
rent a car
valid customer
is-a
customer without rented cars
customer non-blacklisted
customer with valid license driver
customer with credit card
customer with permitted age

Figure 5.32 An authorized actor to rent a car

5.4.2.3 Process execution order

The lack of mechanisms for representing the execution order of the organizational task is one of the more relevant issues of the current *i** framework. We consider that it is necessary to provide flexible mechanisms to represent the execution order but without breaking the intentional focus of the *i** process model. To do this, we propose using the concept of milestones to indicate the execution order. A milestone, which associates two processes, represents that the execution of a process depends on the executions of a previous process. The concept of milestone complies with the concept of dependency of the *i** framework.

Figure 5.33 presents the generic schema for the process model. In this model, the process for requesting and finishing the service has been represented as a default option for business services.

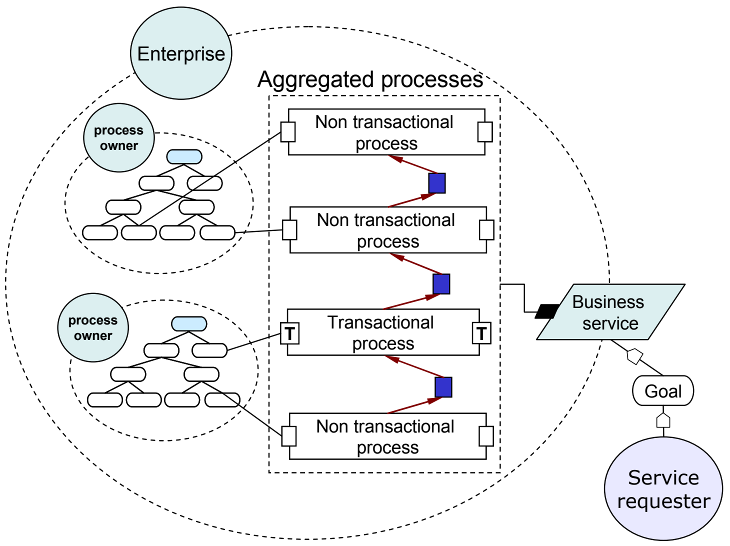


Figure 5.33 The generic schema for the process model

It is important to point out that the delegation and visibility properties of the process model should be inherited from the global model policies.

5.4.3 The Protocol Model

The protocol model focuses on describing the organizational behavior of each business process elicited in the previous step. The protocol model will constitute the lower level *i** specification of the proposed business services architecture. In order to specify the organizational behavior, the protocol model is represented using the revise version of the *i** modeling constructs, that were presented in Chapter 4.

The definition of supporting business services in the protocol model allows us to generate a simplified view of the semantics of each business process. The use of business services makes it possible to reduce the number of modeling elements that compose the protocol model, facilitating its creation and reuse.

The description of the protocols in an isolated model permits the definition of patterns for the recurrent cases; this is the case of the protocol for requesting a service, which represents the act of the requester and provider to define an agreement for using the service. In this specific case, we have detected that there is a generic pattern for representing the group of tasks and resources needed to request a service.

It is important to point out that the definition of protocols is constrained by the standards defined by the industry. This situation also enables the definition of protocol patterns for specific application domains.

Another advantage of the proposed approach is that the specification of fragments of processes with precise semantics opens the possibility to reuse its specification in other similar projects.

The generic schema for the protocol model is shown in Figure 5.34. This Figure is useful to indicate that each business process must be refined into a concrete model by using the *i** notation. Business services of external business actors can be represented to indicate the need of the service provider to use external business services. As commented before, this model is represented by using the revisited version of the *i** concepts proposed in Chapter 4.

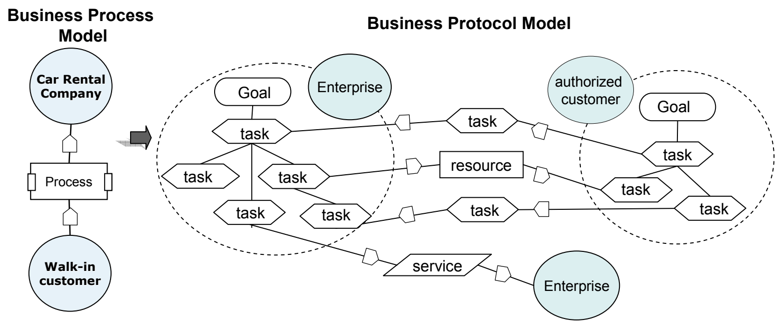


Figure 5.34 The generic schema for the protocol model

5.5 The formalization of the components of the proposed service-oriented approach

A formalization of the graphical diagrams of the business service architecture is proposed in order to fix the semantics of the components and also to permit the automatic analysis of organizational requirements. The formalization of the concept was made in Datalog, a language of logical facts and rules, which is a subset of Prolog language. The basic concepts of Datalog are the following: predicate, term, constant, variable, clause, rule, fact. It is important to point out that several definitions of the components of the proposed service-oriented architecture have been adapted from those presented in the work of Zanonne (Zannone, 2007) for modeling security and privacy models.

5.5.1 Predicates

These predicates enable us to identify the main concepts of the service-oriented architecture proposed in this thesis: service, goal, task, resource, process. The predicate service establishes the type of the business service (st ∈ {offered, supporting}). The predicate process also indicates the associated type. (pt ∈ { transactional, nontransactional}). The predicates regarding actor, agent and roles enable the analyst to define generic actors and instances of actors (agent) playing specific roles. The unary predicate actor is introduced for those cases when it is not necessary to distinguish among agents, roles and positions.

Type Predicates
service (Type: st, Service: s)
goal (Goal: g)
task (Task: t)
resource (Resource: r)
process (Type: pt, Process: p)
actor (Actor: x)
agent (Agent: a)
role (Role: l)

Following, the formalization of the actor properties is presented. The properties have been used to establish that an actor can play the role of provider or requester for a specific service (predicates *provide* and *request*). The actor that provides the service is the legitimate service owner, who is responsible for managing the relationship with the customer (predicate *own*). The service owner can delegate the responsibility to perform the service to another subordinated actor (predicate *delegate*), So that, the new owner of the service will be the subordinated actor. Predicate *service_delegatechain* represents chains of delegation of service among the chains of subordinated actors in the enterprise. The delegation of services to subordinated actors is only permitted if the service owner trusts in the delegated actor to provide the service (Predicate *trust*). These predicates can also be used to indicate the trust of the requester in the provider to provide a certain service. Predicate *should_perfom* is used to indicate actors who should directly fulfill the service. The service owner must implement the necessary mechanism to ensure that it can satisfy the business service (predicate *can_satisfy*). If the service owner delegates the responsibility to perform the service, then it is necessary to indicate whether the new service owner has authorization to delegate the service (predicate *per_delegate*); if not, this actor must directly provide the service. Predicate *monitoring* indicates that an actor can execute supervision activities over the actor that provides the service.

Actor Properties

provide(Actor: x, Service: s)
request(Actor: x, Service: s)
own(Actor: a, Service: s)
delegate(Actor: x , Actor: y, Service: s)
service_delegatechain(Actor: x , Actor: y, Service: s)
trust(x: actor, y: actor, s: service)
trustchain(x: actor, y: actor, s: service)
should_perform(Actor: x, Service: s)
can_satisfy(Actor: x, Service: s)
per_delegate_serv(Actor: x, Service: s)

monitoring(Actor: x, Actor y, Service: s)
monitoringchain(Actor: x, Actor y, Service: s)

The association relations predicates presented below define the relationships among business actors. Predicate *play* identifies the role played by an agent. Predicate *is-a* is used to model generalization and specialization hierarchies of actors. Predicates *specialize* and *instance* are used to link the social level of *i** with the individual level. The *part-of* relationship is useful to describe how an organizational unit is part of an enterprise, and how a specific actor is part of a department, which is, in turn, part of an organizational unit. This flexibility is possible due to the power of the *i** notion of actor for representing abstract concepts and specific individuals (agent, roles and position). Predicate *subordinates* enable us to define the hierarchical relationships of a chain of command in which an actor dominates another actor. Predicate *subordinatedchain* represents chains of subordination among members of the enterprise.

Actors relations
play(a: agent, p: role)
is_a(p: role, q: role)
specialize(Role:p, Role s)
instance (Agent: a, Role: p)
is_part_of(x: actor, y: actor)
subordinate(x: actor, y: actor)
subordinatedchain (x: actor, y: actor)

As stated above, the feature model has been used to define the service variability. The variability in defining services enables the definition of aggregation of high-level services into sub-services. The formalization of the service properties makes reference to the four possible relationships among services and sub-services: mandatory, optional, alternative, and or-decomposition.

Service relations

mandatory_decomposition(s:service, s1:service, . . . , sn : service)
optional_decomposition(s:service, s1 : service, . . . , sn : service)
alternative_decomposition(s:service, s1: service, . . . , sn : service)
or_decomposition(s : service, s1 : service, . . . , sn : service)

The objective of the provider to offer a business service is to satisfy its own goals and the goals of the service requester. This is why the formal definition of the predicate *satisfy_ex* indicates that several organizational goals of the requester can be satisfied by using a specific service. The predicate *satisfy_in* indicates that goals of the provider can be satisfied by providing a business service.

Service Properties

satisfy_ex(s: service, a: actor, g1:goal, ….gn:goal)
satisfy_in(s: service, a: actor, g1:goal, ….gn:goal)

One of the basic mechanisms to associate services with the strategic enterprise objectives is the decomposition relationship. In our proposal, a goal can be refined using AND and OR decomposition. *AND_decomposition* applies if goal g is decomposed into sub-goals g1 and g2, whereas *OR_decomposition* applies when the goal g is decomposed into sub-goals g1 or g2.

Goal refinement

AND_decomposition(g : goal, g1 : goal, . . . , gn : goal)
OR_decomposition(g : goal, g1 : goal, . . . , gn : goal)

5.5.2 Axioms for predicates

The axiomatization of predicates that define the semantics underlying the service-oriented architecture for the *i** framework is presented below. It is important to point out that the letters S, G, T and R are used to indicate, respectively, Service, Goal, Task and

Resource. When referring actors we use letters X, Y and Z. Agents are identified by using A, B and C, and roles are identified by P,Q and V.

Following, we present the predicates that map the social level with the individual level.

From social level to individual level
specialize(P,Q) ← is-a(P,Q)
specialize(P,Q) ← specialize (P,V) ∧ is-a(V,Q)
instance(A, P) ← play (A,P)
instance(A,P) ← instance(A,Q) ∧ specialize(Q,P)
provide(A,S) ← provide(P,S) ∧ instance (A,P)
request(A,S) ← request (P,S) ∧ instance(A,P)
own ← own (P,S) ∧ instance(A,P)
delegate(A, B, S) ← delegate (P,Q,S) ∧ instance (A,P) ∧ instance(B,Q)
service_delegatechain (A, B, S) ← service_delegatechain (P,Q,S) ∧ instance (A,P) ∧ instance(B,Q)
trust(A, B, S) ← trust (P,Q,S) ∧ instance (A,P) ∧ instance(B,Q)
should_perform(A, S) ← should_perform(P,S) ∧ instance(A,P)
can_satisfy(A, S) ← can_satisfy(P,S) ∧ instance(A,P)
per_delegate_serv(A, S) ← per_delegate(P,S) ∧ instance(A,P)

One of the main reasons to formalize the modeling concepts presented in this thesis is to attempt to eliminate the ambiguity that could exist by only providing definitions and examples of the components of the proposed service-oriented architecture.

5.6 Our business service approach as starting point for services in the implementation level

One of the main concerns for representing services in the organizational level is the incompatibility of current approaches to map services in the implementation level (web services) with services in the business model. At the present time, no solutions have been given to enable the analyst to softly translate an enterprise model into a set of web services that implement the business logic. This is because current enterprise models are mainly focused on representing the semantics of transactional business processes rather than being focused on considering the business processes as means to provide services or values to customers.

Recently, research has been done to attempt to make the transformational process systematic; however, the main issue of these efforts is the non-correspondence between business processes and web services. In this thesis, we have analyzed two main approaches that consider the transitions between services at the organizational level and services at the implementation level: a) The works where *i**/Tropos have been used in a service context, and b) The works that offer solutions to represent services at the organizational level (e^{3}value, BPEL, Business State machines).

In the first approach, the *i** and Tropos notations have been used in the organizational context as a starting point for the definitions of services in the implementation level, mainly web services (Lau and Mylopoulos 2004), (Colombo, Mylopoulos and Spoletini 2005), Kazhamiakin (Kazhamiakin, Pistore and Roveri 2004). The main issue of this approach is that, while the resultant specification must be a set of services that fulfill the properties of being a loosely coupled set of components with coarse-grained, well-defined, self-contained and stateless functionalities, the *i** model is the representation of closely-coupled business processes with non coarse-grained functionalities. This is because in *i** models, all information about business processes is represented in the same diagram at the same abstraction level, without granularity among the concepts. Therefore, there is a natural non-correspondence between these two models (business and implementation) that makes the transformational process difficult (Figure 5.35).

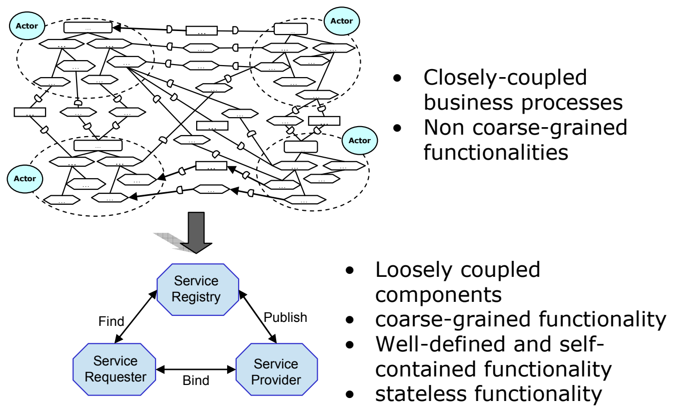

Actor
Actor
Actor
Actor
• Closely-coupled business processes
• Non coarse-grained functionalities
Service Registry
Find
Publish
Service Requester
Bind
Service Provider
• Loosely coupled components
• coarse-grained functionality
• Well-defined and self-contained functionality
• stateless functionality

Figure 5.35 The correspondence between a pure i* model and a web service description

In the second approach, we can find several notations for representing services at the organizational levels (Cherbakov et al. 2005), (Baida 2006). The main disadvantage of these techniques is that they are basically transactional descriptions, where the focus is placed on the representation of the processes and the choreography needed to model the set of services. In this case, the specification in the organizational model appropriately matches the description of services in the low level. The main issue of this approach is that the transactional description of the business services lacks support for strategic descriptions of business processes and also lacks mechanisms to determine the conformance between business processes and strategic objectives. Therefore, it is not possible to perform organizational improvement tasks before generating services from business processes (Figure 5.36).

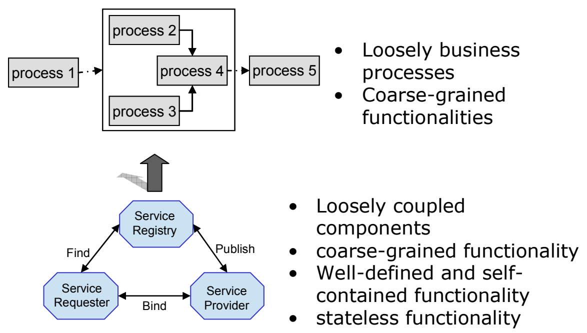

process 1
process 2
process 3
process 4
process 5
Loosely business processes
Coarse-grained functionalities
Service Registry
Find
Publish
Service Requester
Bind
Service Provider
Loosely coupled components
coarse-grained functionality
Well-defined and self-contained functionality
stateless functionality

Figure 5.36 The correspondence web services and ad-hoc business models

In our proposal, the *i** framework has been adapted in order to represent services at the organizational level. The abstract representation of the business service enables the analyst to manage the complexity of the service implementation in an incremental way. Therefore, the business services represented in the model represent well-defined functionalities that encapsulate a set of self-contained business activities needed to implement the service. Thus, it is possible to suggest a light transition between the services represented in the organizational level and the services represented in the implementation level (Figure 5.37).

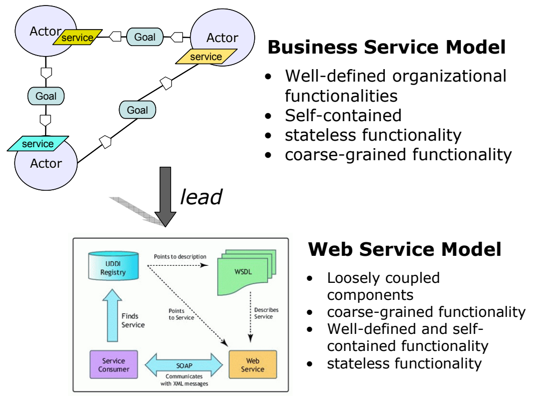


Figure 5.37 The correspondence between services in business and implementation levels

5.7 Final Considerations

In this Chapter, we propose extending *i** in order to address the weaknesses reported in the empirical evaluation. Specifically, we offer a solution for the problems of refinement, modularity, complexity management, reusability and scalability. Our solution is based on the concept of a business service architecture where encapsulated organizational units can only participate in actor dependency networks through well-defined interfaces.

The Chapter presents the main components of the proposed service-oriented architecture in terms of modeling primitives and modeling diagrams.

Some formalization is given in order to establish the semantics of the service components. To do this, Datalog, a language of logical facts and rules, which is a subset of Prolog language has been used to represent the formal properties of the service-oriented architecture.

With the proposed modifications, our intention is to overcome the current limitations that practitioners face when using *i** in its current state. In fact, these modifications are intended to both, solve the problems that were detected and to make facilitate practical application of the method.

Chapter 6

6. The Service-Oriented Method for the *i** Framework

This section introduces our method to enhance the process for creating and representing an organizational model using a service-oriented approach.

6.1 Introduction

One of the main contributions of the *i** framework is the use of social and intentional concepts in order to represent the complex semantics of enterprises. These modeling concepts distinguish *i** from rest of modeling techniques that offer process-based organizational descriptions. The *i** concepts offer powerful semantics for representing the complex structures of enterprises today. However, the use of *i** is limited due to the lack of mechanisms to incrementally construct an organizational model. There are several research works that offer well-founded methods for starting the elicitation process with more conventional mechanisms, such as business goals. Nonetheless, in almost all goal-oriented requirement techniques, the low-level goals are directly translated into requirements for the information system. Although it is true that this approach is closer to the final users; it is also true that this approach does not allow us to carry out analyses (business process reengineering, dependency analysis, workflow analysis, tasks analysis) that are fundamental for obtaining a set of requirements that reflect the expected functionality by the users of the information system.

In this thesis, we propose joining taking advantage of both worlds by performing the early elicitation process with a service-oriented method, which uses the well-founded social and intentional characteristics of the *i** framework to appropriately represent the enterprise situation.

The proposed method will enable us to describe an enterprise as a composite of business services that encapsulate a specific organizational behavior. We introduce the concept of refinement though the decomposition of the business services into a set of business processes that represent the detailed view of the activities needed to perform the service.

We illustrate our approach using the same case study used in the previous Chapter. The case study (which is an extension of the rental car case study used in the empirical evaluation of the *i** framework) considers the set of services offered by a company that is specialized in selling travel packages and car rentals.

The steps of the proposed method for representing an organizational model using the business service architecture are presented in detail in the following sections.

6.2 Overview of the proposed method

The main objective of the proposed service-oriented method is to produce a description of the current way in which the enterprise offers/uses services in order to fulfill its current needs. The objective of this stage is to create a simple view of the services that are used and offered by the enterprise being analyzed. The details about the reification of business services into business processes are also presented in this modeling stage.

Most current business modeling techniques have neglected the representation of the current situation of the enterprise, providing top-down approaches where the manager's point of view is used to obtain the high-level goals of the enterprise. This manager's point of view is also used to refine the goals of the enterprise until the level of tasks needed to satisfy the goals is reached. However, this approach which is useful in the design of new enterprise models usually produces models that are too ideal to correspond with the current way business processes in an existing enterprise are performed. Therefore, it is not the appropriate source for improving the enterprise.

The description of the current situation of the enterprise is important because it allows us to understand the current goals of the enterprise and how these goals are achieved through the involvement of organizational actors' processes of the enterprise (Kavakli and Loucopoulos 1999).

Loucopoulos also establishes that "any organizational reform requires, prior to designing new business processes and support information systems, a clear understanding of the current enterprise situation in terms: (a) what are the current enterprise processes; and (b) what is the purpose that current enterprise processes aim to fulfill." (Loucopoulos and Kavakli 1997).

The idea of this thesis is to represent the current enterprise situation based on the service-oriented architecture. To do this, four aspects need to be represented:

- What: Definition of the scope of services, this is about determining what the service actually is.
- Who: Definition of who the external actors that drive the service are.
- Why: Identification of the reasons to offer services.
- How: Representation of the details about the processes that coordinate the services and as well as the details on how a services itself is implemented.

We use the proposed architectural models (defined in Chapter 5) in order to represent these four key aspects. The global model is the appropriate means to represent the *what*, *who* and *why* aspects. The process model gives a high-level view of the processes that compose each service. Finally, the protocol model offers a detailed view of the implementation details of each business process.

The steps to construct the global, process and protocol models are presented below.

- The first step is to represent the current enterprise situation which consists of defining a service global model. The objective of this phase is to define a model that represents the business services offered and used by the enterprise to fulfill its current goals. Once the external business services have been represented in the service global model, the delegation structure for each business service must be identified and represented using the composite actor structure.
- Once the delegation model has been defined, a business process model for each business service must be created. The objective of this phase is the identification and representation of the set of business processes that make up each one of the business services.
- The last step in the representation of the current enterprise situation is the definition of a business protocol model for each business process defined in the previous steps. Once the process model has been represented at a high abstraction level using the business process model, the behavior of each one of the processes that compose the business service must be identified and represented. A protocol model, which uses the reviewed version of the *i** modeling language, is generated for each process as a result of this step (Figure 6.1). The proposed approach provides the necessary support for managing the complexity of the modeling activity, allowing the analyst to represent each fragment of a business service in isolated diagrams.

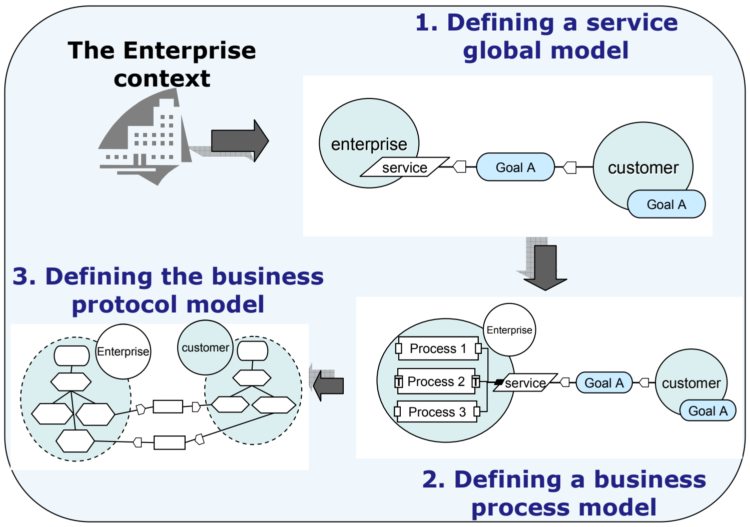


Figure 6.1The overview of the representation of the current enterprise situation

- Once the current enterprise situation has been defined, it is possible to use the generated diagrams to produce a description of the alternative solutions for offering/implementing business services in order to satisfy the desired goals of the enterprise. The objective of this modeling stage is to generate new descriptions of the business services that enable the enterprise to adapt to new external conditions. To do this, softgoals need to be used to evaluate the new services according to the quality factors desired in the enterprise.

It is important to point out that the description of future enterprise situation is not always needed. This description is only needed in situations where business process reengineering is required to solve enterprise problems or in situations where new market conditions obligate the enterprise to modify its services or to offer new services.

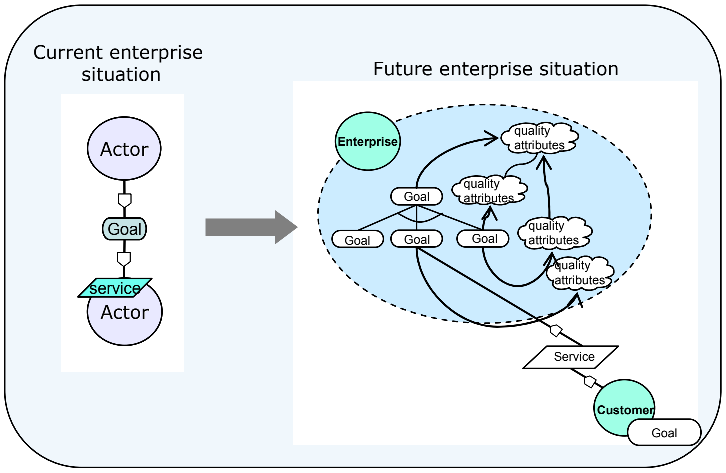


Figure 6.2 The overview of the representation of the future enterprise situation

6.3 The strategy of the service-oriented method

One of the key points in the service-oriented method is to use an intermediate model between the information elicited from the organizational setting and the modeling diagrams that are proposed in this thesis. The intermediate model uses goals structures (goal-refinement trees) and composite actor structures to represent all details of the complex organizational setting. Thus, the intermediate model is used to generate the diagrams of our proposal (global, process and protocol models) that offer a clear and simple view of the organization context. Algorithms were developed to automatically translate the intermediate model into the service-oriented diagrams (Figure 6.3).

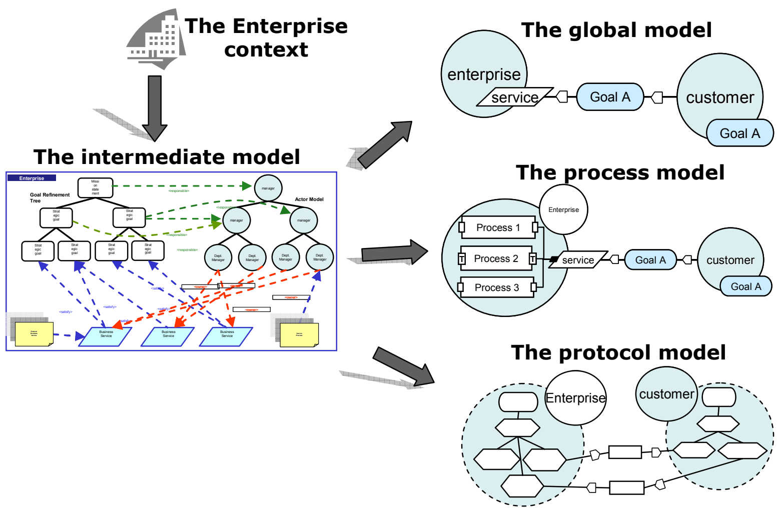


Figure 6.3 The strategy of the service-oriented method

In this sense, we can establish that intermediate model represents the "working area" of our proposal and the service-oriented diagrams represent the "clean" models to be reviewed by the final customers.

The intermediate model allows the analyst to clearly demonstrate the relationships among goals, business actors, services and the chain of command of actors in the enterprise. This relationship is established at three different levels: service level, process level and protocol level. As stated above, the intermediate model contains a goal-refinement tree that describes the decomposition of the goals that support the services/processes/protocols depending on the level of description of the model. This goal structure is represented in the left side of the model. The intermediate model also describes the composite actor structure that establishes the chain of subordination of the actors in the enterprise. This structure is represented in the right side of the model. Finally at the bottom of the model the services/processes/protocols are represented (Figure 6.4).

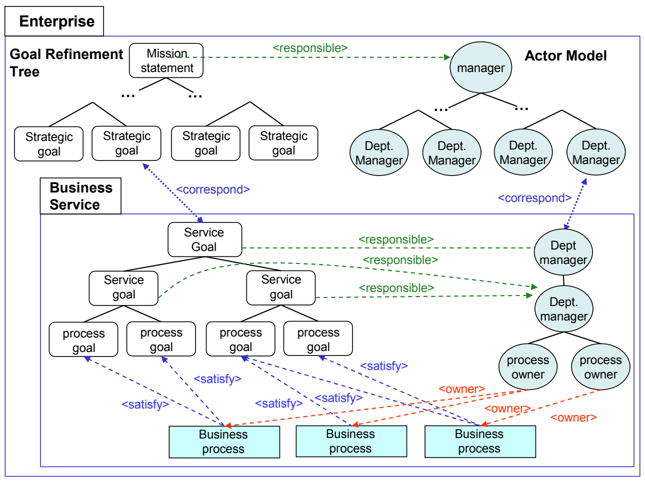


Figure 6.4 An example of the intermediate model

It is important to point out that only the service diagram, process diagram and protocol diagram are visible for final users.

The steps of the proposed method for representing an organizational model using the business service architecture are presented in detail in the following sections.

6.4 Defining the service global model

The objective of this phase consists of defining a model that represents the services that the enterprise offers to fulfill its strategic objectives as well as the services that the enterprise use in order to fulfill its current goals. To do this, the offered services are explicitly associated with the enterprise goals of the provider actor. This characteristic enables the analyst to align the enterprise goals with the offered business services (Figure 6.5). As stated in Chapter 5, business services are represented as an extended *i** goal dependency that associates the requesters´ needs with the offered service. The service is placed in the boundary of the actor description in order to indicate that all further organizational behaviors must be encapsulated in the service as an interface between enterprise and customers.

The global model allows us to represent a high-level view of the business services, hiding the details of the service implementation. Therefore, the global model is a simple and easy to understand model and it could be used to create the first agreements among the business participants.

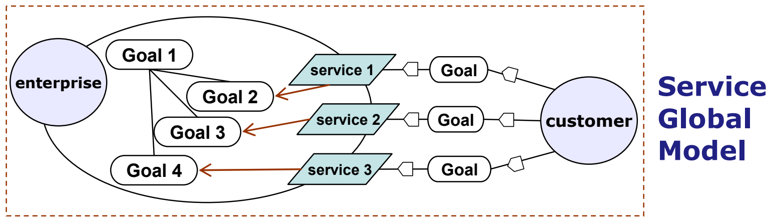


Figure 6.5 Global model to associate goals and services

Two complementary views of the service global model have been proposed in this research work.

- *Abstract view of the global model*. This model is focused on representing a simple view of the offered business services.
- *Detailed view of the global model*. This model is focused on detailing the goals that are satisfied by the offered business services.

Subsections 6.3.1.1 and 6.3.1.2 respectively detail the set of steps to create the abstract and detailed view of the global model.

6.4.1 Defining the abstract view of the global model.

The abstract view of the global model provides a high-level description of the offered enterprise service. Therefore, the model only represents what is offered without indicating the reasons why enterprise delivers the services, and without explaining the current implementations of the services.

The following algorithm must be applied in order to create the abstract view of the global model:

1. Elicit the services offered by the enterprise.
2. An *i** actor that represents the entire enterprise is created.
3. For each business service elicited, it is necessary to detect the potential customers for the service. An *i** actor is created for each customer of the business services.
4. A service dependency is created between the enterprise and the customers.
5. The basic and composite business services are represented in the model

The steps of the proposed algorithm are presented in detail below.

Step 1: Eliciting business services

In the first step to define the global model, the enterprise managers must identify the services offered by the enterprise to potential customers. In this step, the business managers view the enterprise as a service provider, where the enterprise customers are viewed as service requesters. As defined in Chapter 5, not all the business behaviors can be considered as a business service. These are processes where the enterprise offers a specific functionality to external customers. From the analyst's perspective, it is true that functions are something the organization cannot exist without; however, not all business functions offer values to external customers. This obligates the enterprise to offer business functionalities as service interfaces. Therefore, those enterprises that do not offer values to customer through business functionalities cannot be modeled using the concept of business services.

The appropriate sources for eliciting business services are personal interviews with enterprise managers, who have a broader view of the business capabilities.

Step 2: Representing the service provider

Once business services have been elicited, an *i** actor that represents the enterprise is created in the business model. As stated in previous Chapters, the *i** framework offers the possibility of representing generic actors as well as individuals (agents). This characteristic enables us to represent the complete enterprise as a generic actor. It is important to point out that this is an intermediate representation of the enterprise actor. In the following steps, we will determine the sub-actors that compose it. Figure 6.6 shows the services offered by the enterprise that is analyzed in the running example.

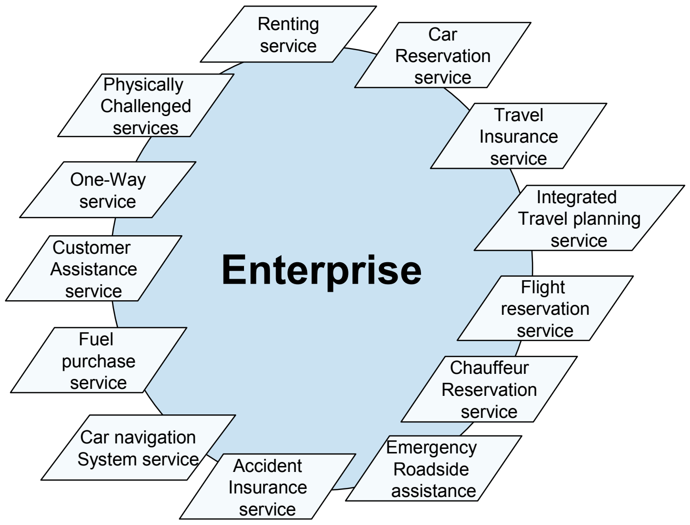

Renting service
Car Reservation service
Physically Challenged services
Travel Insurance service
One-Way service
Integrated Travel planning service
Customer Assistance service
Enterprise
Flight reservation service
Fuel purchase service
Chauffeur Reservation service
Car navigation System service
Emergency Roadside assistance
Accident Insurance service

Figure 6.6 Business services for the running example

Step 3: Representing the service requesters

The service requester is the person or organization that wants to use the service in order to satisfy its needs. According to our proposed modeling approach, the requester depends on the service provider to increase its capabilities by using the functionalities that the business service offers.

The potential customers of the offered services must be detected for each one of the elicited business services. In this case, the normal sources for this information are also the personal interviews with the enterprise managers.

Once the potential customers for the business services have been detected, they have to be represented as actors in the global model. To do this, a generic actor must be created in the business model using the *i** notation.

Step 4: Representing the service dependency

The service provider (enterprise) and requester must be associated through a goal dependency that indicates that the customer depends on the provider in order to satisfy a certain goal through a specific business service. In the graphical representation, the service has been placed in the boundary of the provider actor to indicate that the business service is the only interface between the providers and the requesters. The arrows of the dependency must always be directed toward the service provider. Figure 6.7 shows the basic schema for the abstract view of the global model.

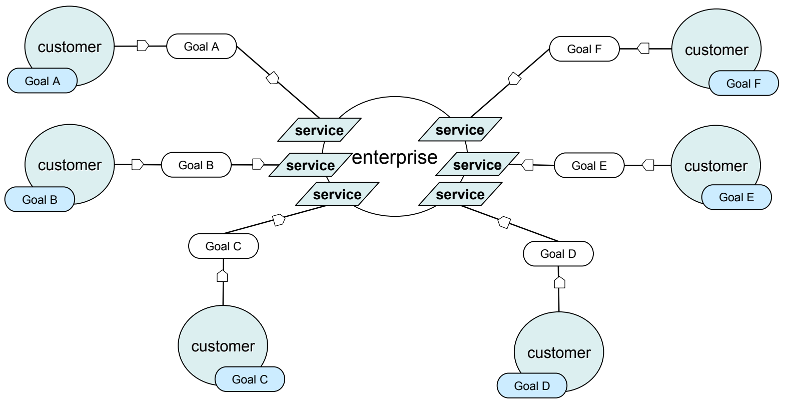
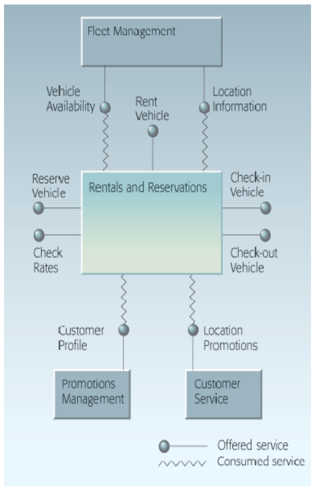
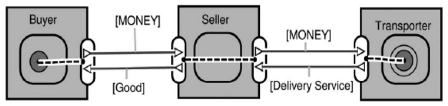


Figure 6.7 The abstract view of the global model

The graphical representation of services has advantages ones with current service representations, where offered and consumed services are represented as a functional description without indicating which goals are satisfied by the services. Figure 6.8 shows two examples of the situation explained above. These representations correspond to the proposals of IBM (Cherbakov et al. 2005) and **e^3** value (Baida 2006), in which the services are directly associated with each other without indicating the needs that are satisfied by the service execution.



IBM proposal



e^3 value proposal

Figure 6.8 Representation of services at the organizational level

In our proposed syntax, the goal of a customer is delegated towards the service provider (this is the reason why the name of the internal goal is the same as the goal dependency associated with the business service). This indicates that the requester depends on the provider to satisfy its needs through a specific business service.

The objective of the abstract view of global model is the creation of initial agreements between the analyst and the customers.

Step 5. Defining composite and basic business services.

One of the main issues in the definition of the global model is the definition of the composite and basic services. As was stated in Chapter 5, a composite service aggregates multiple business services and implements mechanisms that coordinate the aggregated services. A basic service is decomposed in processes without further decomposition.

The definition of business service composition raises some issues related to how to manage the variability of the aggregated services. There are several possibilities for combining obligatory and optional services that need to be considered in this modeling stage. The possibility to represent alternative business services must also be represented in this stage.

The issues about to combining composed business services have been managed by using the feature model proposed in Czarnecki research works (Czarneki et al. 2000). Czarnecki proposed four features to represent the several possibilities that exist to combine business services: *mandatory*, *optional*, *alternative*, and *or* features. Table 6.1 shows the four Czarnecki features.

Mandatory: the child feature in this relation is always present when its parent feature is included. For example, if an integrated travel planning is requested, the flight reservation must be included.

Optional: the child feature in an optional relation may or may not be present when its parent feature is present. For example, it is possible to include a chauffeur in the car reservation if the customer requests it.

Alternative: a child feature in an alternative relation may be included if its parent feature is included. In this case, only one feature of the set of children is included. For example, when an integrated travel planning is organized, only one means of transportation must be defined (flight, bus, or car).

Or–relation: the child feature in an or–relation may be included if its parent feature is included. Then, at least one feature of the set of children may be selected. For example, to rent a car, a customer can hire a GPS localization system or a chauffeur or both at the same time.

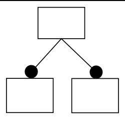
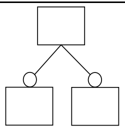
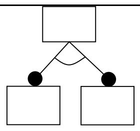
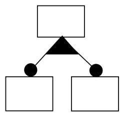
 **mandatory**	A mandatory feature is included in the description of a concepts if its parent is included.	 **optional**	An optional feature may be included in the description of a concept if and only if its parent is included in the description.
 **alternative**	If the parent of a set of alternative features is included, then exactly one feature from this set of alternative features must be included.	 **or**	If the parent of a set of alternative features is included, then any nonempty subset from the set of or-features is included

Table 6.1 The feature model alternatives

In this research work, the feature model has been used in order to represent the complex configurations that are found in combining business services.

Figure 6.9 shows the variability schema in the composition of business services by using the feature model. This model indicates that a composite business services is composed by two mandatory business services (that need to be necessarily performed if the composite business services is activated) and by two optional services. One of the aggregated business services is also refined in four business services where at least (and only) one of these needs to be selected to accomplish the composite business service.

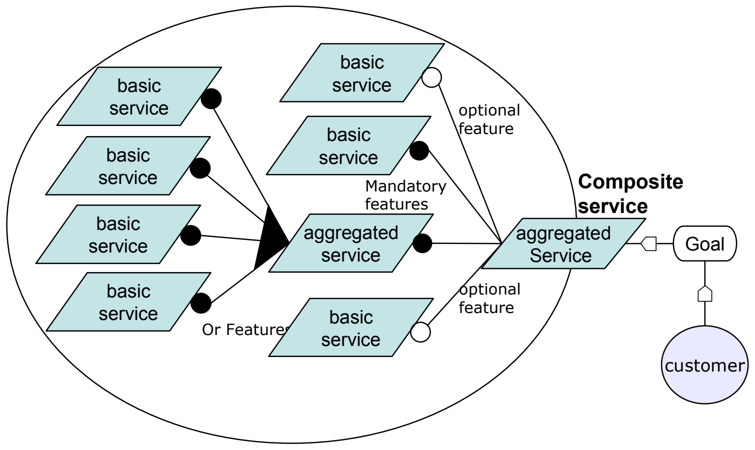


Figure 6.9 Service variability through the feature model

The variability model represents a very powerful mechanism to represent the several possibilities to decompose a business service in low-level descriptions.

It is important to point out that only basic business services can be further decomposed into business processes. This is because the refinement of a composite service in a set of business processes would generate a process model that joins all the processes of the services (basic and composite) that compose the composite service that is being refined. This situation would break the concept of modularity that is the key concept of the proposed method.

Figure 6.10 shows an example of the use of the feature model to service composition. In this example, the *integrated travel planning* composite business service is decomposed into three mandatory services: flight reservation, hotel reservation, car reservation. This indicates that wherever an *integrated travel planning* business service is defined, these composite services must be included. The *integrated travel planning* service is also decomposed into optional service travel insurance, which indicates that it is optional for the customer to select this service along with the other obligatory services. Following this example, three alternatives have been defined for the service *car reservation*, services: walk-in reservation, phone reservation and internet reservation. The alternative feature obligates that exactly one of the alternatives must be selected; therefore, the customer must select one of these alternatives to make the car reservation. The feature model represents a powerful mechanism to manage the variability of composed business services.

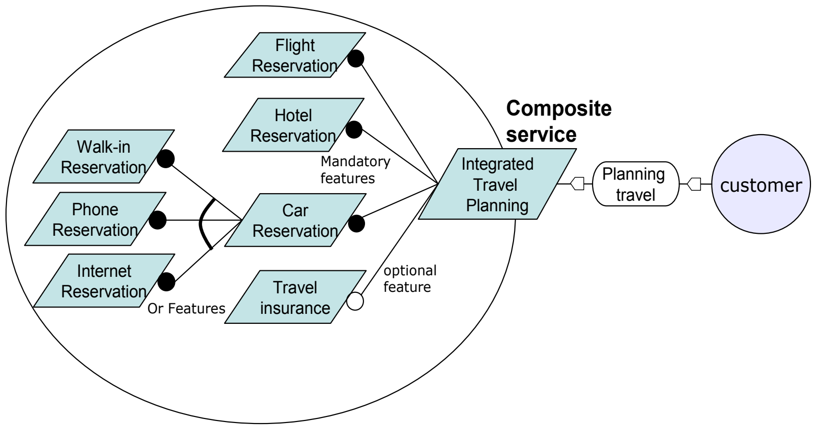


Figure 6.10 An example of feature model to service composition

The global model offers the flexibility to represent a) all the service decompositions or b) a specific business service. However, as stated above, only the basic business services can be decomposed into business processes. Therefore, the processes for a composite business service will be those resulting from the sum of all the processes of the sub-services.

6.4.2 Defining the detailed view of the global model.

The specific view of the global model focuses on representing the reasons for delivering business services to potential customers.

In order to create the concrete view of the global model, the following algorithm must be applied:

1. Detect the provider's needs and goals and represent these in a goal-refinement tree in the intermediate model
2. Define the actor that is responsible for the service's goals and represent these in the intermediate model.
3. Detect social dependencies from the intermediate model.
4. Review the schema for service delegation.
5. Define the schema of visibility for services-

The steps of the proposed algorithm are presented in detail below.

Step 1. Defining the provider's needs and goals

The first step for the creation of the detailed global model is the identification of the goals that support each one of the offered services and their representation in the intermediate model. The objective of this step consists of describing how the business services contribute to the satisfaction of the strategic goals of the enterprise. To do this, an abstraction process must be carried out for each offered service elicited in order to determine the goals that are satisfied by the service execution.

The enterprise managers identify the strategic enterprise goals using a goal-refinement tree (GRT) which was proposed in Estrada works (Estrada, Martinez, and Pastor 2003). In this goal structure, the general goal represents the mission statement of the organization, the internal nodes represent the groups of low-level goals for the satisfaction of a general goals, and finally, the leaf nodes of the goal-refinement tree represent the strategic goals (long term goals) of the enterprise that are satisfied by the offered business services. The main idea of this process is to determine the reasons (in terms of organizational objectives) that exist in the enterprise to offer a certain service to potential customers. Figure 6.11 represents the intermediate model that associates enterprise goals and business services). We consider that an enterprise goal must exist for each offered service.

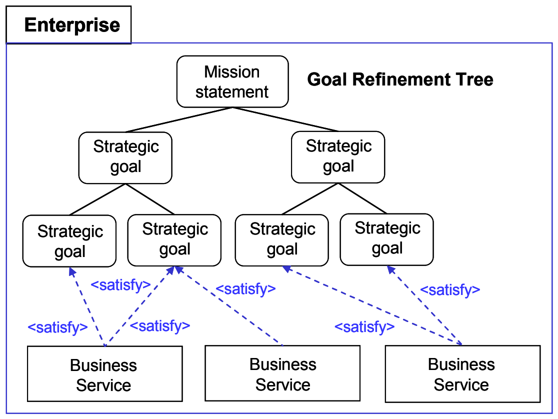


Figure 6.11 Associating strategic goals and business services in the intermediate model

It is important to point out that in this modeling level, the enterprise is being represented as a generic actor without identifying the internal actors that compose the enterprise.

Two complementary approaches can be followed to create the goal-refinement tree (GRT): refinement strategy and abstraction strategy. The refinement strategy is useful in the cases where the analyst elicits the goal from the point of view of the organizational managers, who tend to express high-level goals.

In the *refinement strategy*, it is necessary to select some of the general goals of the organization and determine the subset of subgoals that permit us to satisfy them until the level of business services is reached. This information must be elicited using the mission statement or by interviews with the enterprise managers. This information is used to construct the high levels of the goal-refinement tree.

The abstraction strategy is useful in situations where the analyst elicits the goal from the organizational actors who tend to express low-level goals or operations.

In the *abstraction strategy*, it is necessary to take the actors that make the business services operable as a source for the low-level goals of the goal-refinement tree. Later, the general goals that are satisfied by the specific goals of the enterprise actors must be determined.

The goal decomposition process ends when all the services offered by the enterprise have been mapped with a strategic goal of the goal-refinement tree. However, it is important point out that not all the strategic goals of the enterprise must be satisfied by external services. Some goals could be satisfied through internal process without interaction with final customers.

The next step consists of determining the inconsistencies between the goals elicited from the manager's point of view and the goals detected from the actor that makes the service operable. Once the inconsistencies have been detected, it is necessary to create a model that reconciles the different points of view about the goals satisfied by each business service. Figure 6.12 shows the representation of the detailed view of the global model that is generated from the intermediate model.

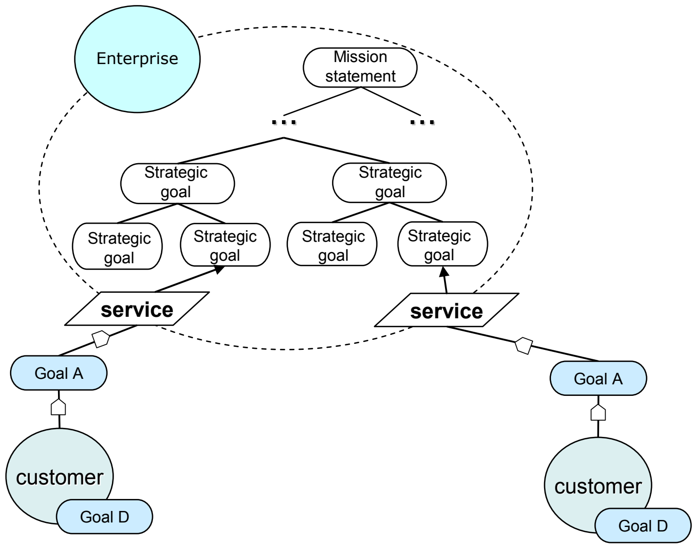


Figure 6.12 The general schema of the detailed global model

As a final result of this first process, a model is produced that presents a simple view of the services offered and the reasons why these services are provided. Figure 6.13 shows a fragment of the detailed view of the business service global model for the running example. In this case, the goals of the services offered by the enterprise (flight reservation, car reservation, hotel reservation and integrated travel planning business services) have been mapped with the strategic interests of the enterprise

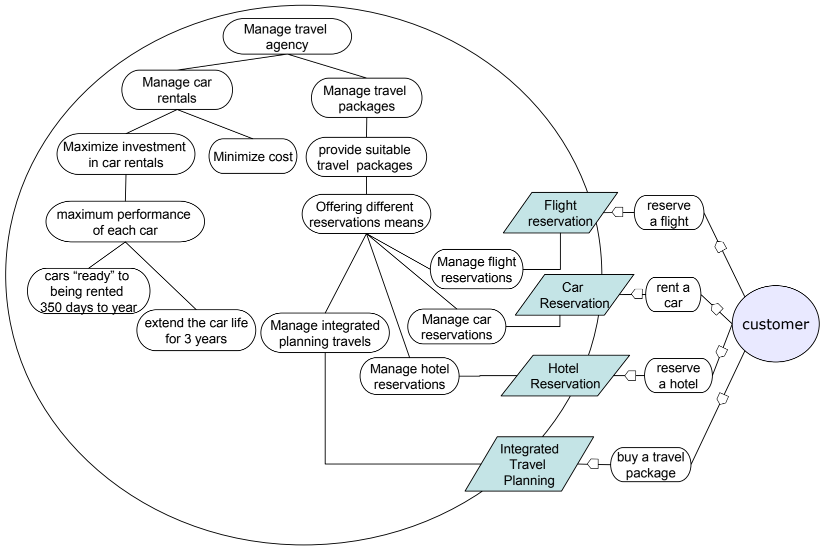

Manage travel agency
Manage car rentals
Manage travel packages
Maximize investment in car rentals
Minimize cost
provide suitable travel packages
maximum performance of each car
Offering different reservations means
Flight reservation
reserve a flight
cars "ready" to being rented 350 days to year
extend the car life for 3 years
Manage flight reservations
Car Reservation
rent a car
customer
Manage integrated planning travels
Manage car reservations
Manage hotel reservations
Hotel Reservation
reserve a hotel
Integrated Travel Planning
buy a travel package

Figure 6.13 Fragment of the detailed view of the global model for the running example

Following this goal decomposition schema, it is also necessary to detail the goals of the composed services. Thus, we need to represent the composition hierarchy to represent it in the global model. Once the basic and composed services have been represented, a goal-refinement tree must be constructed for each elicited service. Following with the running example, Figure 6.14 shows a fragment of the detailed global model for the composed service *car reservation*. In this model, the refinement process started taking the goal from the service (*Manage car reservation* goal) as the root of the sub-goal-refinement tree.

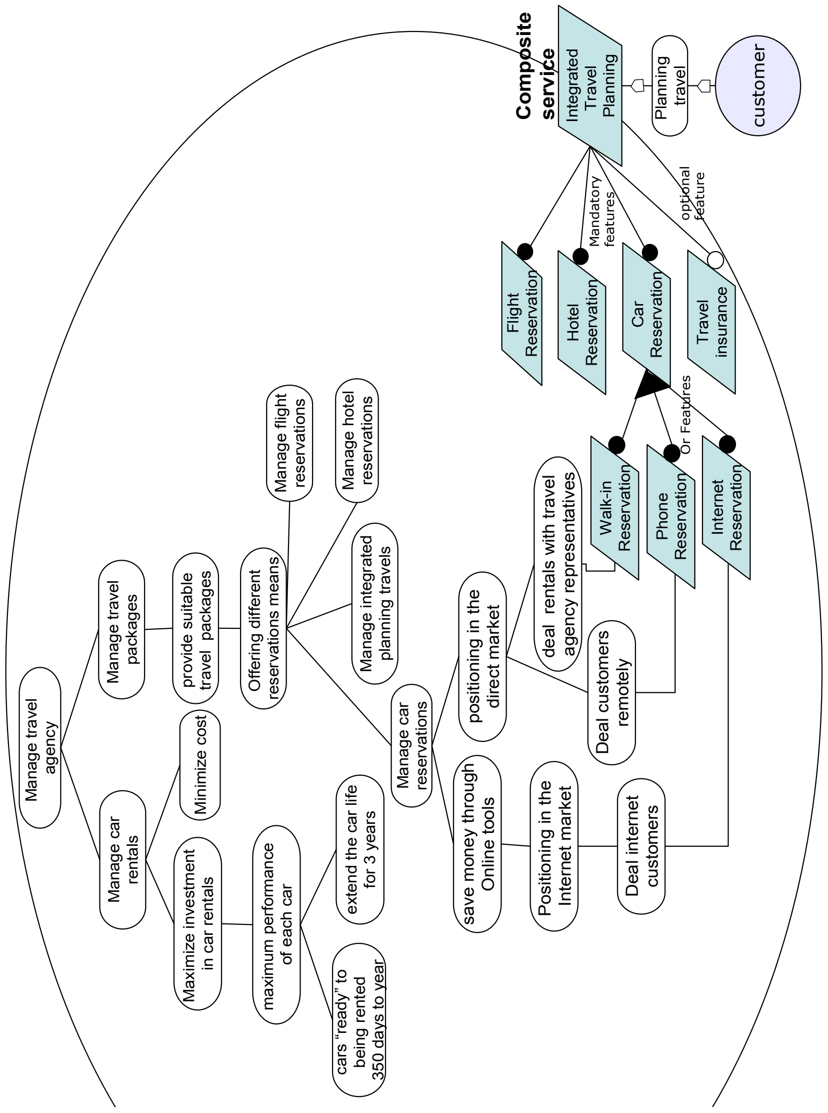

Composite service
Integrated Travel Planning
Planning travel
customer
Mandatory features
optional feature
Flight Reservation
Hotel Reservation
Car Reservation
Travel insurance
Or Features
Walk-in Reservation
Phone Reservation
Internet Reservation
Manage travel agency
Manage travel packages
Manage car rentals
provide suitable travel packages
Minimize cost
Maximize investment in car rentals
Offering different reservations means
Manage flight reservations
Manage hotel reservations
Manage integrated planning travels
Manage car reservations
maximum performance of each car
extend the car life for 3 years
cars "ready" to being rented 350 days to year
positioning in the direct market
save money through Online tools
deal rentals with travel agency representatives
Deal customers remotely
Positioning in the Internet market
Deal internet customers

Figure 6.14 Example of a detailed view of a global model for composed services

Figure 6.15 shows a simplified view of the detailed view of the global model that only represents the goal for the *Manage car reservation* service.

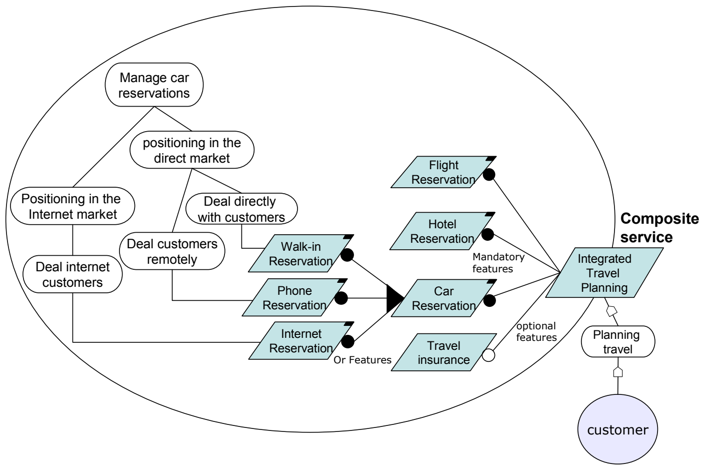


Figure 6.15 Example of the detailed view of the global model for composed services

As commented above, not all elicited goals need to have associated services as a satisfaction mechanism. In many cases, the enterprise goals are satisfied by executing internal processes or by requesting services from external entities. Thus, the service global model enables analysts to represent the enterprise as a requester and provider of business services.

Step 2. Defining the actors that are responsible for the service's goals.

In this step, the enterprise managers identify the stakeholders that are responsible (goal owners) for the strategic goals detected in step 1, which support the elicited business services. Until now, the enterprise has been represented as a generic actor without the details of internal actors and behaviors that are needed to provide the business services. The main objective of this step is the definition of the organizational actors that participate in the implementation of each business service. This requires considering the enterprise as a network of actors with social relationships whose aim is to provide value to customers.

One of the key points in this proposal is the explicit representation of the actor that is responsible for satisfying the goals in the goal-refinement tree. In most of the current goal-based approaches, the elicited goals are not mapped with specific actors. This is done in order to provide a more abstract description of the enterprise goals. However, we consider that in order to provide a more complete description of the current enterprise situation, the actor with responsibilities must also be elicited and represented in the business model.

In this process of identifying of the actors responsible for achieving the elicited goals, it is possible to find potential dependency relationships among actors. The dependency relationship can be detected when the actor responsible for a goal is different from the actor that is responsible for satisfying some of its subgoals. This situation indicates some kind of dependency (to be determined) among the actors for fulfilling their goals.

In this phase of analysis of strategic goals, the actors identified as being responsible for goals are usually department managers. The detected stakeholders are represented in the composite actor structure, which is an organizational chart that represents the lines of authority of the organizational actors using subordination links (see section 5.3.5.1). As stated above, the key concept about subordination is that if a role subordinates another role, then the first can delegate activities to the latter. The hierarchical goal structure defined for each service must comply with the hierarchical structure of the composite actor structure. This is because in the definition of the internal behavior of the enterprise, an actor can delegate a goal to another actor only if the first subordinates to the latter.

The explicit relationship among the goal model and the actor model allows us to detect the following:

a) The situations where a goal has been assigned to several organizational actors, who share the responsibility to satisfy the strategic goal,

b) The situations where there is not an actor that is clearly identified as being responsible for the goal.

Based on the relationship between the goal model and the actor model represented in the intermediate model, it is also possible to determine the assignations of the responsible stakeholder (service owner) to assure the correct performance of the business services (Figure 6.16). In figure, we exemplify the relations among business goals, the composite actor structure and the business services in the intermediate model. The managers and department manager are the actors that are responsible to satisfy the goals of the enterprise. The business services are satisfied by the offered business services. This model express that the business actor are the owners of the business services. This complex configuration

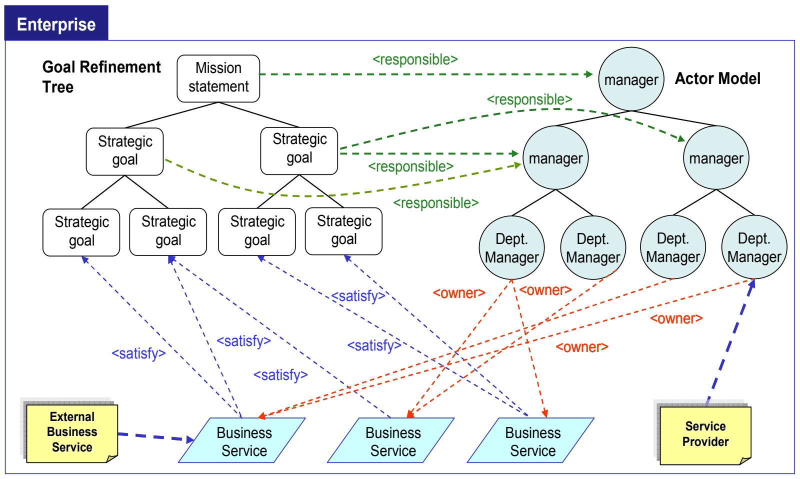


Figure 6.16 The relationship between goals, actors and business services in the intermediate model

Step 3. Detecting dependencies from the goal-refinement tree

As stated above, at the moment, the enterprise has been represented as a generic actor without identifying the internal actors that compose the enterprise. In this step, the goal-refinement tree of the intermediate model is used to refine this monolithic actor and to detail the actors participating in the services and their dependency relationships. The main idea of this process is to use the relationships among the actors that satisfy the goals and their position in the composite actor structure in order to automatically generate their corresponding global model. Figure 6.17 shows the approach followed in this modeling stage.

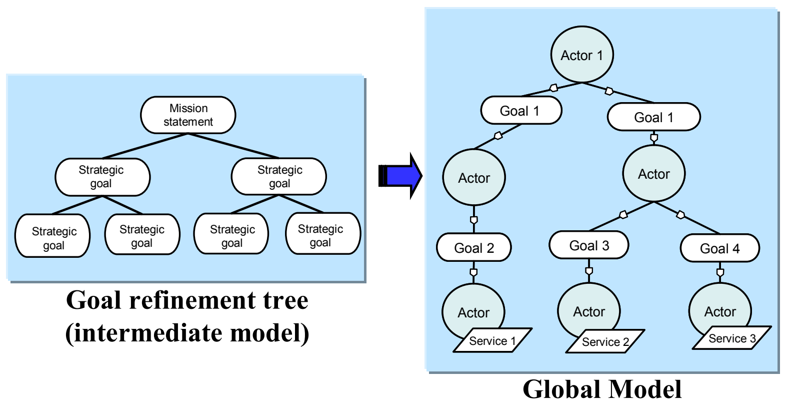


Figure 6.17 From goal structures to dependency models

To make the model transformation automatic, the following goal classification and dependency generation process must be applied.

Step 3.1 Goal classifications in the Goal-Refinement Tree (GRT).

A detailed goal classification was proposed to structure the GRT in order to represent the potential dependencies among the organizational actors that are responsible for goals in the refinement tree. Once the organizational context has been elicited by using the goal-refinement tree, the goals must be classified according to the proposed classification. This is a critical step to ensure the correct translation of the goal structure into the *i** dependency model.

Achievement goals: In this case, the actor responsible for fulfilling the goal does not depend on another actor to satisfy its own needs. When analyzing a goal decomposition (goal–subgoals), goals of this kind can be detected if the actor responsible for both goals is the same organizational actor. This indicates that the first actor will execute the main goals as well as the sub-goals associated with this goal.

Achievement-dependency goals: In this case, the actor responsible for fulfilling the goal depends on another actor to fulfill the goal. Goals of this kind can be detected when the actors needed to satisfy a goal decomposition (goal - subgoals) are different. This indicates that the actor responsible for a parent goal has delegated the goal or fragments of this goal to another actor.

The goals represented in the goal-refinement tree must be classified according to the proposed classification in order to perform the generation of the goal dependencies.

Step 3.2 Dependency generation process based on the goal-refinement tree.

The global model to be generated is focused on representing the relationships between the enterprise actors involved in the execution of business services, where the goal dependency is the focal point of the modeling activity. Therefore, to create this model, it is necessary to take a subset of the goal-refinement tree where a dependency between actors exists (Achievement-dependency goals). It is important to point out that there are goals in the GRT that could be satisfied by the actor itself without dependencies with other actors (achievement goals). Goals of this kind are not represented in the strategic dependency model. Following, we present the steps to achieve the translation of the GRT into the SD Model.

The first step consists of using the actors responsible for the goals of the goal-refinement tree to create the organizational actors of the dependency model using the *i** syntax to define generic actors in the intermediate model (Figure 6.18).

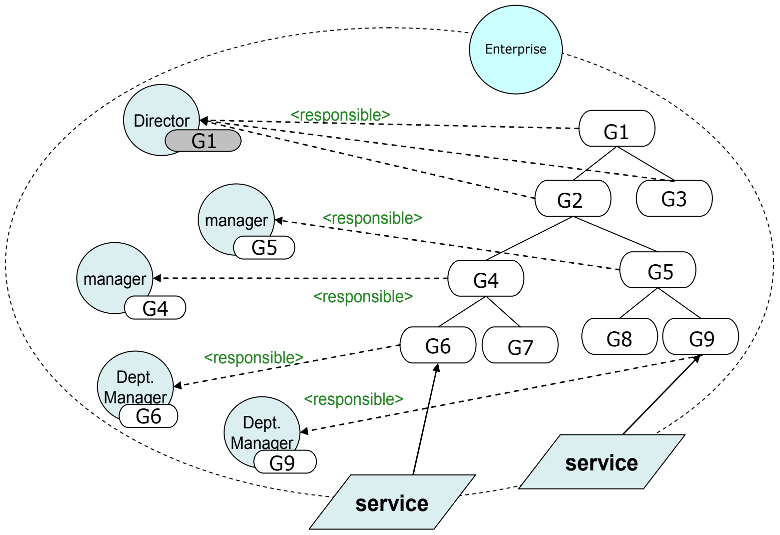


Figure 6.18 Generation of the *i** actor from the actors responsible for the service goals

In the process of creating goal dependencies, the actors responsible for achieving the achievement-dependency goals must be analyzed in order to determine the actor that plays the role of *depender* (the actor responsible for the parent goal) and the actor playing the role of *dependee* (the actor responsible for the delegated goals that appear as leaf goals). The achievement-dependency goal must be translated into a goal dependency with the same name between the *depender* actor and *dependee* actor (

Figure 6.19).

The identified actors with the responsibility to satisfy goals or achieve operations in our case study are: Director, reservation manager, internet reservation manager, manager of representative agencies, internet reservation system, phone reservation agency, and travel agency.

The second step is to use the achievement-dependency goals of the goal-refinement tree to create the goal dependencies in the strategic dependency model. As it was previously mentioned, the achievement-dependency goals are goals that represent dependency relationships between actors.

In this modeling stage, it is possible to determine that some of the delegated goals can be better satisfied by specifying internal business services. In this case, the analyst must determine if the business activities associated with the fulfillment of the delegated goal have the behavioral conditions to be considered as a business service. If so, the internal business service must be associated with the elicited business goal.

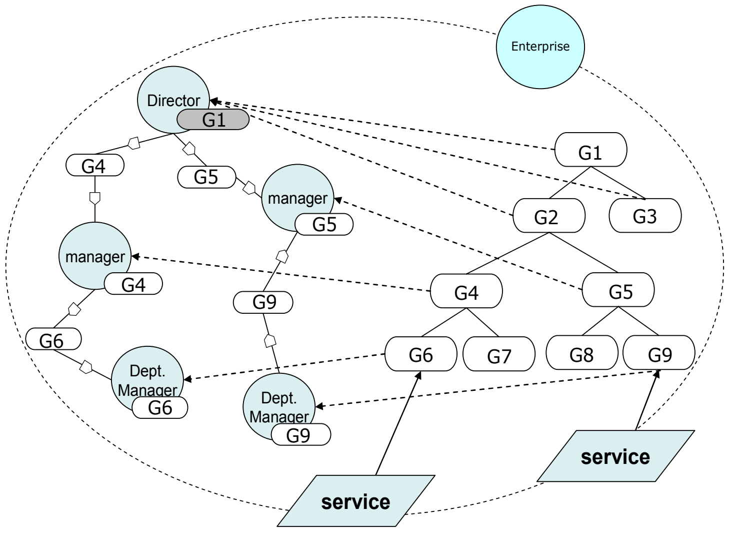


Figure 6.19 Generation of dependencies from goal-refinement tree

In Figure 6.19 the Director of the company is the actor that is responsible for the mission statement and the high-level enterprise goals (G1,G2,G3). Goal G4 and G5 of the goal-refinement tree have been assigned to actors that are different from the Director;

therefore, we can determine a dependency among the Director and the business managers. In the case of goals G4 and G7, the same actor is responsible for fulfilling the goals; this is why no dependency relationship needs to be created in this case.

Figure 6.20 shows an example of the dependency generation process for the car rental business services of the running example. This figure represents three offered business services: internet reservation, phone reservation and walk-in reservation. The goals that are supported by the business services are also represented in the model. Based on this goal structure, a chain of dependencies is created to represent how the goals have been delegated. In the model, the Director depends on the reservation manager to manage the reservations. The reservation manager depends on the manager on internet reservation to save money by using online reservations. The reservation manager depends on the representative agency manager to manage the direct market. The internet reservation manager depends on the internet reservation system to provide the internet reservation service. The representative agency manager depends on the phone reservation agency to provide the phone reservation and finally, the representative agency manager depends of the travel agency to provide the walk-in reservation service.

It is important to point out that this generation process is an intermediate step to define a final model that represents the future situation of the enterprise.

Step 4. Reviewing the delegation schema

One of the main advantages of the explicit representation of the actor responsible for performing the service goals is the possibility to analyze the goal delegation chain among these actors.

The following formula defines the delegation chains of the services inside the organization.

goal_delegatechain (X,Y,S) ← delegate(X,Y,S)
goal_delegatechain (X,Y,S) ← delegate (X,Z,S) ∧ goal_delegatechain (Z,Y,S)

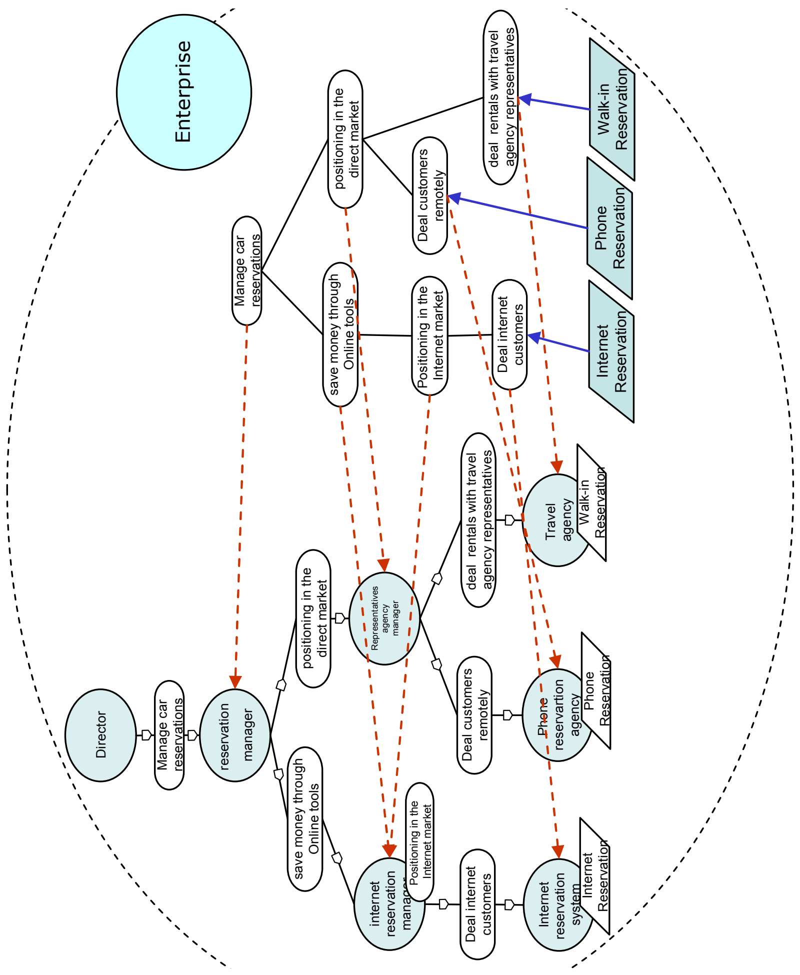

Enterprise
Manage car reservations
positioning in the direct market
deal rentals with travel agency representatives
Deal customers remotely
save money through Online tools
Positioning in the Internet market
Deal internet customers
Walk-in Reservation
Phone Reservation
Internet Reservation
Director
Manage car reservations
reservation manager
positioning in the direct market
save money through Online tools
Representatives agency manager
deal rentals with travel agency representatives
Deal customers remotely
Travel agency
Walk-in Reservation
Phone reservartion agency
Phone Reservation
internet reservation manager
Positioning in the Internet market
Deal internet customers
Internet reservation system
Internet Reservation

Figure 6.20 Example of dependencies generated for the running example

We consider two different scenarios for the satisfaction of the enterprise goals.

In the first scenario, an actor must satisfy the goal directly if he/she is the service owner or if another actor has delegated the goal to him/her. The actor has to satisfy the goal without further delegation. The formula that describes this scenario is the following:

should_perform_goal(X,S) ← own(X,S) ∧ satisfy (X,S)
should_perform_goal(X,S) ← goal_delegatechain (Y,X,S) ∧ own(X,S) ∧ satisfy (X,S)

In the second scenario, an actor can satisfy the goal by itself or by delegating the goal to another actor. The formula that describes this scenario is the following:

can_satisfy_goal(X,S) ← should_perform_goal(X,S)
can_satisfy_goal(X,S) ← goal_delegatechain(X,Y,S) ∧ can_satisfy_goal(Y,S)

An actor can delegate a goal to another actor if the actor is the owner of the goal. The following formula indicates direct and indirect subordination.

per_delegate_goal(X,S) ← own(X,S)
per_delegate_goal(X,S) ← goal_delegatechain (X,Y,S) ∧ can_satisfy(Y,S)

The following formula indicates no further goal delegation.

Per_delegate_goal(X,S) ← own(X,S)
per_delegate_goal(X,S) ← goal_delegatechain (X,Y,S) ∧ should_perform (Y,S)

In goal decomposition, the role of the composite actor structure is not relevant in defining the delegation schema. Goal delegation can apply between two actors, although a subordination relationship does not exist between them. Thus, a requester depends on the provider to use a certain service, and the provider depends on the provider to pay for the service. In this case, there is a cyclic dependency where no subordination exists among requester and provider.

As a result of this set of steps, we obtain a global model that represents the current enterprise situation, where the modeling activity is focused on representing all the services that the enterprise offers and the services that the service uses at a high abstraction level, hiding the details to perform this. This model represents the enterprise actor involved in satisfying the organizational objectives through the offered business services (Figure 6.21).

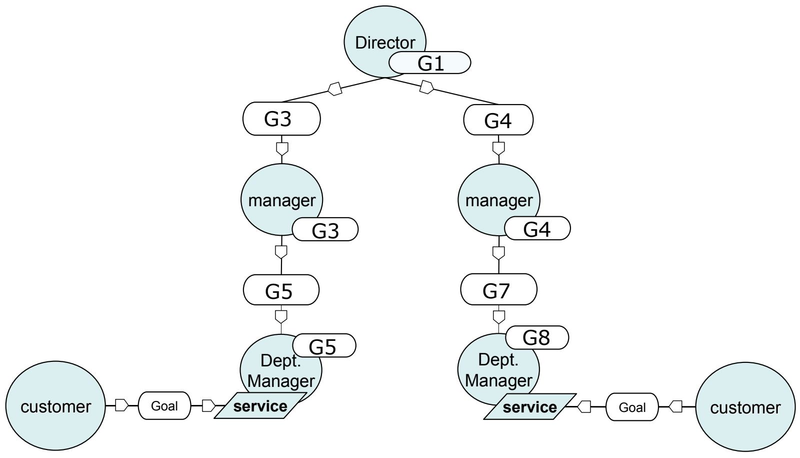


Figure 6.21 The global model resulting from the generation process

Step 5. Defining the visibility schema.

Once a global model that represents the offered business services has been created, it is necessary to define the visibility schema that answers the following questions: can the customer see the internal services? What about task monitoring? What about trust between the service provider and the requester?. In the proposed service-oriented architecture, the business services represent the appropriate interface between the requester and provider, so that all interactions among both actors must be done through the service. Therefore, we consider the offered services as the appropriate source to define the visibility schemas that could be propagated to the process that make up each business service.

The visibility schema associates two correlated concepts: monitoring and trust. Monitoring can be defined as the set of tasks needed for an actor to carry out supervision over the actor that executes tasks. In our case, monitoring refers to activities that enable the requester to supervise the activities of the business services providers. Trust indicates the belief that one actor does not misuse the resource (informational o physical) involved in the business service execution. The former actor is called the *truster*, while the latter is called the *trustee*. From the provider's point of view, trust indicates the confidence of the requester to have visibility about the internal processes that are needed to perform a business service.

Three different scenarios were proposed to represent the different service visibility schemas: black box, grey box and white box visibility. We define monitoring and trust values for each one of the proposed visibility schemas. We adopt the notation proposed in (Zannone 2007) for indicating the white, grey and black box visibility schemas (WB, GB and BB respectively), monitoring (ME) and Trust (TE). The indication of the kind of visibility schema is placed on the arrow of the goal dependency that is linked with the business service.

White box is the less restrictive visibility schema where the provider trusts the requester. In the white box schema there are intermediate checks (public view of the private production process) during service composition, and the enterprise trusts in the customer (Figure 6.22). In this schema, the customer can introduce controls at different times during the service execution to monitor the service. Regarding with trust, the enterprise trusts in well-known customers.

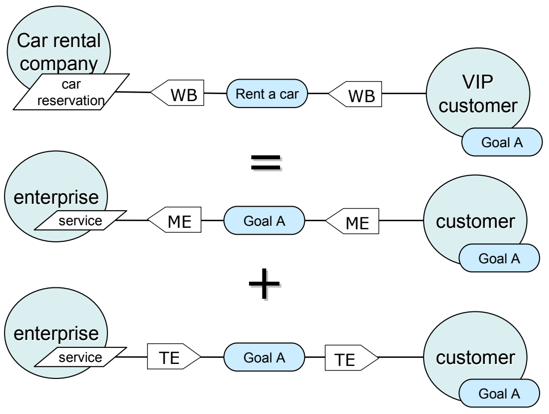


Figure 6.22 White box visibility schema

In the grey box visibility schema there are intermediate checks during service composition, but the enterprise does not trust the customer.

In this schema, the customer can introduce controls at different times during service execution to monitor the service. About trust, the enterprise does not trust in infrequent customers (Figure 6.23).

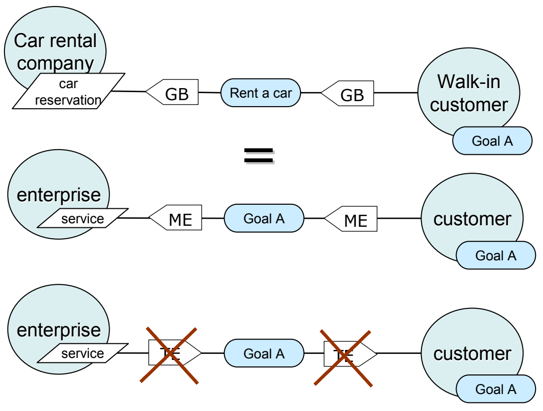


Figure 6.23 Grey box visibility schema

Finally, in the black box schema, there are no intermediate checks during service composition, and the enterprise does not trust in the customer. In this schema, the customer can only see the service's outputs. Regarding with trust, this schema implies no trust in the requester actor. It must be used for unknown customers (Figure 6.24).

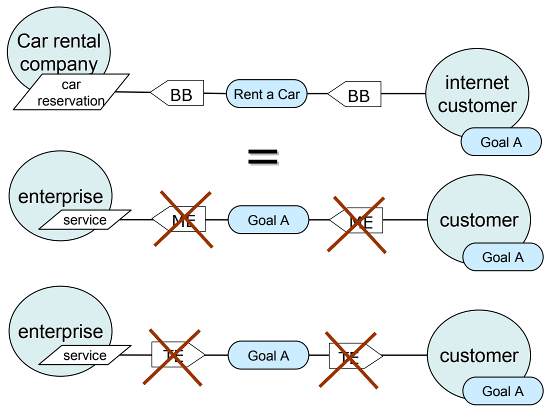


Figure 6.24 Black box visibility schema

The axiom for general monitoring was presented in Chapter 5. The specific formulas for the visibility schemas are presented below.

Whitebox visibility:

- Blackbox_visibility(X,S) ← depend(X, Y,S) ∧ monitoring (X,S) ∧ trust (Y,X)

Greybox visibility:

- Blackbox_visibility(X,S) ← depend(X, Y,S) ∧monitoring (X,S) ∧ ¬trust (Y,X)

Blackbox visibility:

- Blackbox_visibility(X,S) ← depend(X, Y,S) ∧ ¬ monitoring (X,S) ∧ ¬trust (Y,X)

Figure 6.25 shows an example of the simplified view of the global model for the services *Internet reservation*, *Phone reservation* and *Walking-in reservation*. In this model, The Director of the enterprise depends on the Reservation Manager to manage all of the reservations. The reservation manager depends on the Internet reservation manager to save money using internet resources. The reservation manager also depends on other actors to manage the direct reservation. Finally, the goals *deal with internet customers*, *deal with customers remotely* and *deal with rental with travel agency representatives* have been delegated to the Internet reservation system, the Phone reservation agency and the Travel agency, respectively. This model could be used to create the first agreements with the stakeholders.

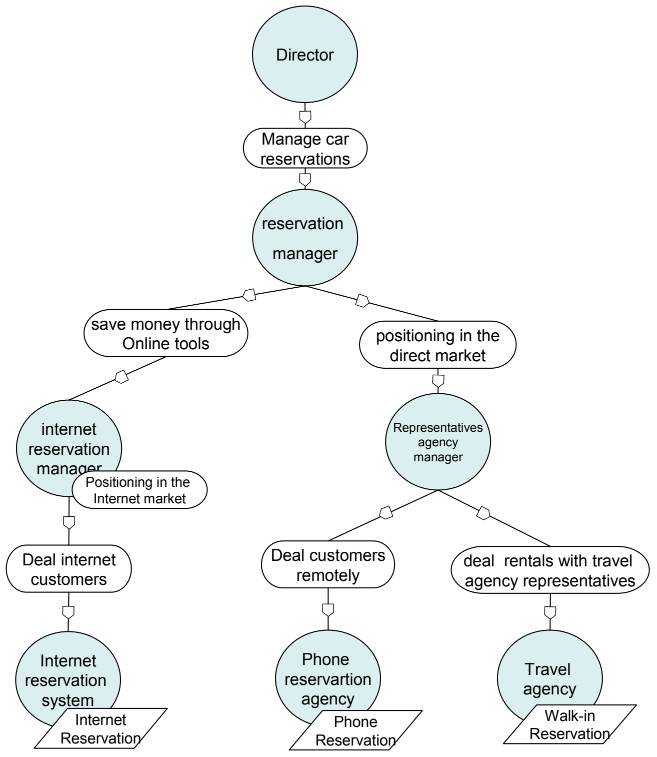

Director
Manage car reservations
reservation manager
save money through Online tools
positioning in the direct market
internet reservation manager
Positioning in the Internet market
Representatives agency manager
Deal internet customers
Deal customers remotely
deal rentals with travel agency representatives
Internet reservation system
Internet Reservation
Phone reservartion agency
Phone Reservation
Travel agency
Walk-in Reservation

Figure 6.25 An fragment of the simplified view of the global model

6.5 Defining the process model

Once the business services have been represented at a high abstraction level using the service global model, then it is necessary to identify and to represent the business processes that compose each one of the business services. To do this, the intermediate model, that is used to generate the global model, must be extended to represent the goals of the processes and the actors responsible for performing them. The approach followed to carry out this stage is the same one used to generate the global model, but applying this method to the description of each business service as follow: a) obtain the service's goals, b) refine its until the level of concrete processes is reached, and c) identify the actors that are responsible to perform them. The main objective of this process is to align the goals that support the business services with the goals that support the business processes (Figure 6.26). We consider that each business process must be justified in terms of business services to be offered to potential customers.

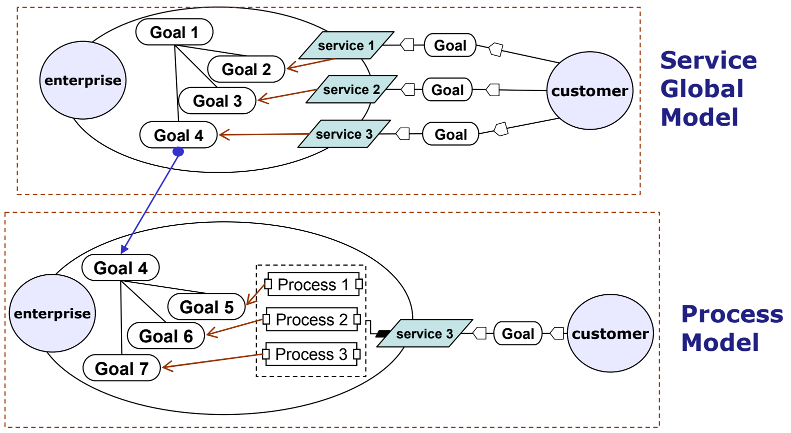


Figure 6.26 Aligning service goals and process goals

Once the offered business services, the goals that support the services and the responsible actors for goals have been elicited, the delegation of these services among the organizational actors must be analyzed. To do this, we need to determine how the original service owner (the actor responsible for the service) delegates it to another subordinate actor in the composite actor structure. This propagation of service delegations ends when the level of goals that are satisfied by specific processes is reached (Figure 6.26).

Step 1. Determining business processes by refinement

The first step of this phase consists in representing the internal goals associated to each business service using the goal-refinement tree in the intermediate model. To do this, the main goal of each business service is placed as the root of the goal-refinement structure. Then, the subgoals needed to satisfy this goal must be elicited and placed in the goal structure. This refinement process ends when the elicited goals can be satisfied by the business processes of each business service.

Step 2. Determining process goals by abstraction

In this step, the service owners identify the business processes needed to satisfy the business services, and then relate these processes to the goals detected in the first phase. In the practical application of this approach, we have detected that there are certain processes that recurrently appear in the specification of business services, e.g. the processes for requesting and ending the services. Therefore, we consider that these processes can be defined as default business processes in the process model.

Step 3. Linking goals and processes with organization actors

The next step consists of mapping the goal structure elicited in Step 1 with the composite actor structure. This is done to assign the responsibilities for satisfying the process goals to the responsible actors. It is also necessary to relate the elicited processes with the composite actor structure. This is done to determine the process owners, who are the actors responsible for performing some processes of the business service

As a result of these phases, an expanded intermediate model is created (Figure 6.27).

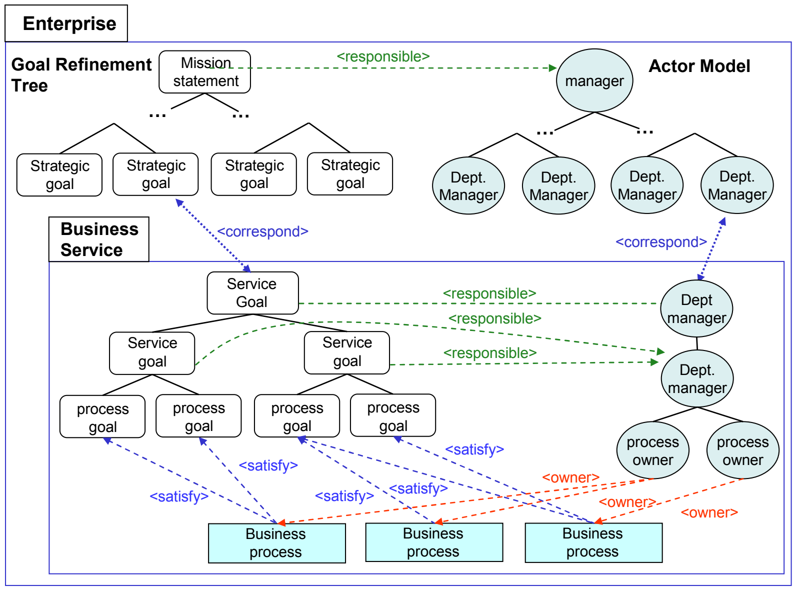


Figure 6.27 The expanded intermediate model

Step 4. Reviewing the delegation schema

With regard to the delegation of services, it implies that the *delegatee* actor (the actor on whom the services is delegated) is the new service owner and it can directly provide the service or delegate it to another subordinated actor. The delegation also implies the possibility of the *delegater* actor (the actor that delegates the service) to monitor the performance of the delegated service.

As mentioned in Chapter 5, the delegation is a concept that is closely correlated with the hierarchical structure of actors in the enterprise. In the service decomposition context, an actor can delegate the responsibility to perform a service to a subordinated actor. Figure 6.28 represents the needs to consider the composite actor structure for delegating goals. This model is used to exemplify that the Director of the company can delegate the responsibility to satisfy goals to his two subordinated managers. The managers can also delegate the goal responsibility to their department managers. In the normal delegation schema, that needs to be defined in the composite actor structure, a manager can not delegate a goal to another actor that is not in its subordination chain.

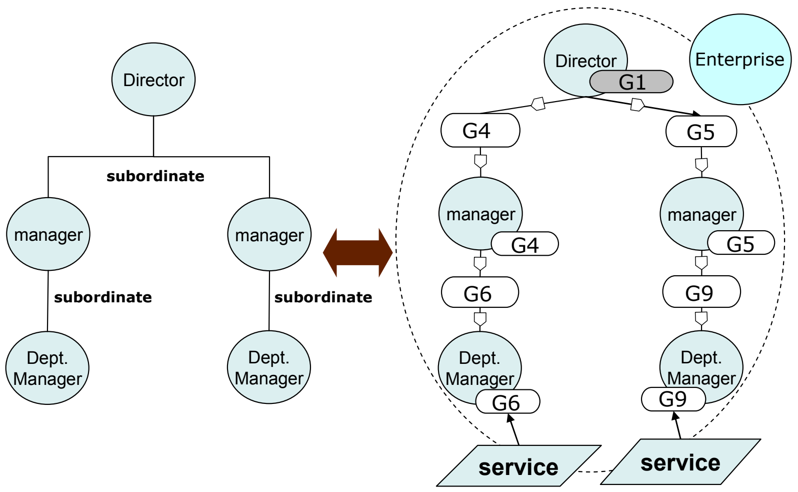


Figure 6.28 The composite actor structure as the basis for delegation

We consider two different scenarios for the satisfaction of the enterprise services.

In the first scenario, an actor must perform the service directly if he/she provides the services and if either actor has delegated the service to him/her, and if the services have been delegated by an actor in an upper level in the line of authority. The actor has to execute the service without further delegation. The predicate *should_perform* has been defined to describe this scenario.

should_perform(X,S) $\leftarrow$ provide(X,S)
should_perform(X,S) $\leftarrow$ provide(X,S) $\wedge$ service_delegatechain (Y,X,S) $\wedge$ subordinatedchain(Y,X)

In the second scenario, an actor can satisfy the service by itself or by delegating the service to another actor in the authority line. The predicate *can_satisfy* specifies this delegation situation.

can_satisfy(X,S) $\leftarrow$ should_perform(X,S)
can_satisfy(X,S) $\leftarrow$ servicedelegatechain(X,Y,S) $\wedge$ can_satisfy(Y,S) $\wedge$ subordinatedchain(X,Y)

The predicate *per_delegate_serv* indicates that an actor can delegate the service to another actor if the actor is the owner of the service and if the delegate actor is a subordinate.

per_delegate_serv(X,S) $\leftarrow$ own(X,S)
per_delegate_serv(X,S) $\leftarrow$ service_delegatechain(Y,X,S) $\wedge$ subordinatedchain(Y, X,) $\wedge$ per_delegate_serv (Y,S)

The delegation of services can be performed in two basic schemas according to the service execution predicates: the chain delegation schema and delegation without further delegation. In both schemas, it is possible for the *delegater* to monitor the performance of the service delegated to the *delegatee*.

delegate(A,B,S) $\leftarrow$ own(B,S) $\wedge$ monitoring(A,B,S) $\wedge$ can_satisfy (B,S) $\wedge$ subordinate(A,B)
delegate(A,B,S) $\leftarrow$ own(B,S) $\wedge$ monitoring(A,B,S) $\wedge$ should_perform(B,S) $\wedge$ subordinate(A,B)

The predicate *service_delegatechain* defines the delegation chains of the services in organizational actor structure. One of the key issues for delegation of services is to determine the chain of subordination because it is the structure that restricts the delegation.

Predicate *trustchain* defines the chain of trust in the delegation schema. As stated above, the visibility schema defined in the global model (which defines the level of trust between the requester and the provider) is propagated towards the process and protocol models.

```
service_delegatechain(A,B,S) ← delegate(A,B,S)
service_delegatechain(A,C,S) ← delegate(A,B,S) ∧ service_delegatechain(B,C,S)

trustchain(A,B,S) ← trust(A,B,S)
trustchain(A,C,S) ← trust(A,B,S) ∧ trustchain(B,C,S)
```

The predicate *Subordinatedchain*, specifies the chain of subordination that defines the composite actor structure. As stated above, the composite actor structure is based on the subordination relationships among organizational actors.

```
subordinatedchain(A,B) ← subordinate(A,B)
subordinatedchain(A,C) ← subordinate(A,B) ∧ subordinatedchain(B,C)
```

The proposed formalization captures the alternative paths that exist to perform a service: either directly perform it or delegate it to another subordinated actor.

Figure 6.28 represents the scenario where the Director of the company can not provide the service directly, thus delegating it to the subordinated manager (predicate *can_satisfy*). The manager can not provide the service and delegate it to the department manager. In this case, the department manager can not delegate the service and must execute it without further delegation (predicate *should_perform*).

Step 5. Detecting dependencies from the goal-refinement tree

In this step, the goal-refinement tree is used to create the dependencies among the elicited organizational actors. The main idea of this process is to use the relationship among the actors that satisfy the goals and their position in the composite actor structure in order to automatically generate their corresponding process model. To do this, we need to apply the same goal classification and dependency generation process proposed in the global model in order to obtain the dependencies among the organizational actors to fulfill the process associated with each business service.

To create the graphical representation of the process model, the following algorithm must be applied:

- For each one of the processes needed to provide the business service, an internal process that is linked with the business service must be created. It represents the abstract functionality of the service. We use the notation explained in Chapter 5 to represent the business processes.
- The actors responsible for implementing the services are used to create *i** sub-actors inside the organizational actor that represents the whole enterprise.

Figure 6.29 shows the schema that enables us to associate business processes and business goals regarding a specific business service. The main objective of this schema is to align the goals of the process with the goals that the enterprise wants to fulfill in order to provide a certain business service.

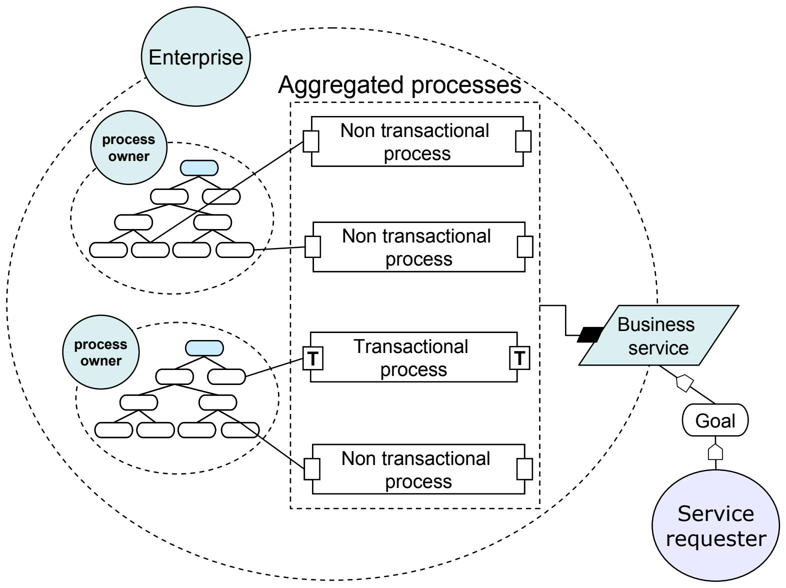


Figure 6.29 Linking business goals and business process

A detailed view can be also obtained that show the dependences among actor to provide the processes associated to the business service Figure 6.30. In this model, the processes have been assigned to their responsible actors and the dependencies have been also generated from the intermediate model.

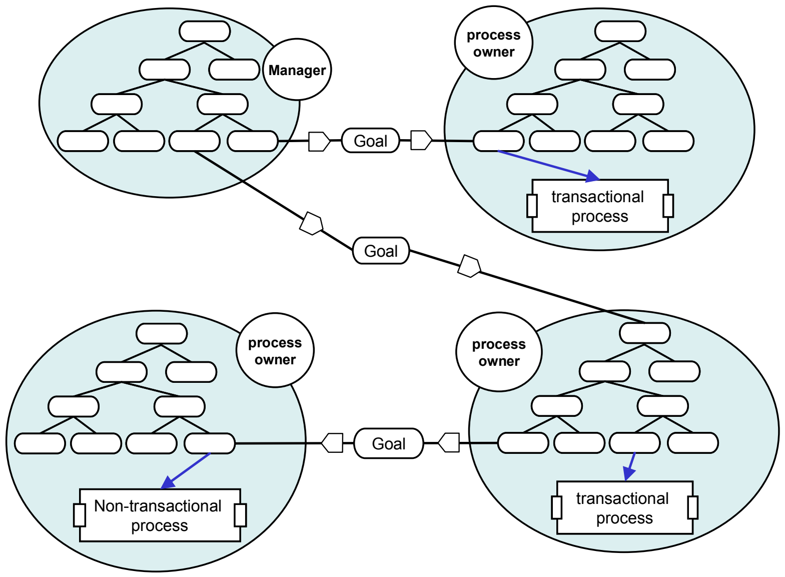


Figure 6.30 The detailed view of the process model

Step 6. Defining the visibility schema.

The visibility schema defined in the global model is propagated towards the process model. In the typical case (Figure 6.31), the service requester has visibility of the offered service. It can only see the processes that are offered by the organizational actors that interact with the final customer without visibility of the internal business processes that compose the service (black box visibility schema). However, scenarios where the customers have visibility of the internal process (white and grey schemas) can also occur in practice.

With respect to actor visibility, the service requesters can usually see the actor playing the role of basic employee, as in the case of clerks. In a more general schema, the service customer does not have visibility of enterprise managers.

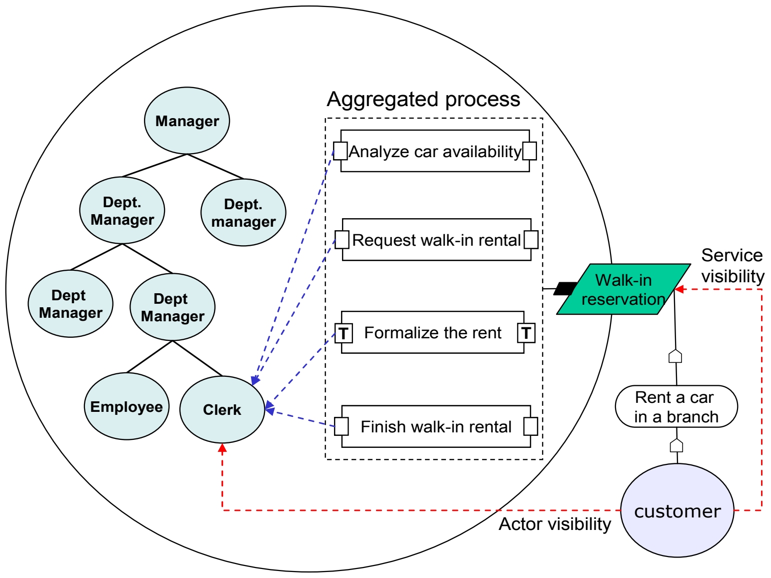


Figure 6.31 Service and Actor visibility

Step 7. Specifying process execution order

The next step of this phase consists of representing the process execution order. To do this, we use a flexible definition of the execution order based on milestone concept. Milestones denote logical checkpoints in the normal flow of the process. Thus, a milestone denotes the completion of a phase of work within the business process. In this thesis, we use the concept of milestone to indicate the dependency to execute the processes. A milestone associates two processes where the arrow indicates the process dependency. Figure 6.32 shows the syntax of milestones.

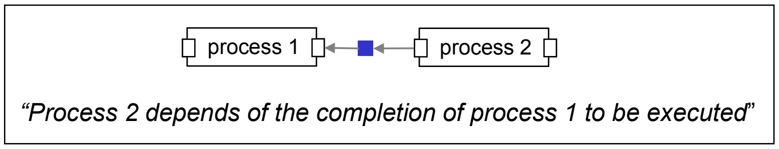

process 1
process 2
"Process 2 depends of the completion of process 1 to be executed"

Figure 6.32 Execution order based on milestones

A simplified process model (one that does not present the business goals associated with business processes and the actors responsible for performing the processes) can be represented, showing only the abstract definition of the processes that compose a business service (Figure 6.33).

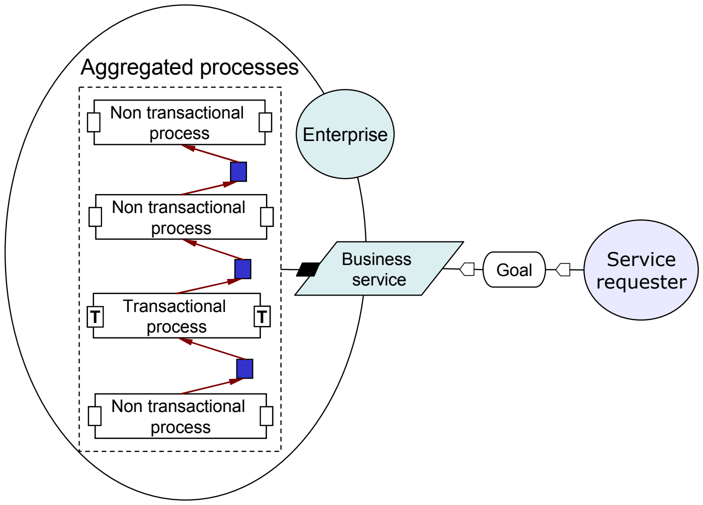


Figure 6.33 Simplified view of the process model

Figure 6.34 shows an example of the simplified view of the process model for the *walk-in rental car* case study. In this model, milestones indicate the execution order as follows: to request a walk-in rental it is necessary to first analyze the car availability. To formalize the car rental, the request is needed, and finally, to finish the business service, the service should be formalized in a previous time.

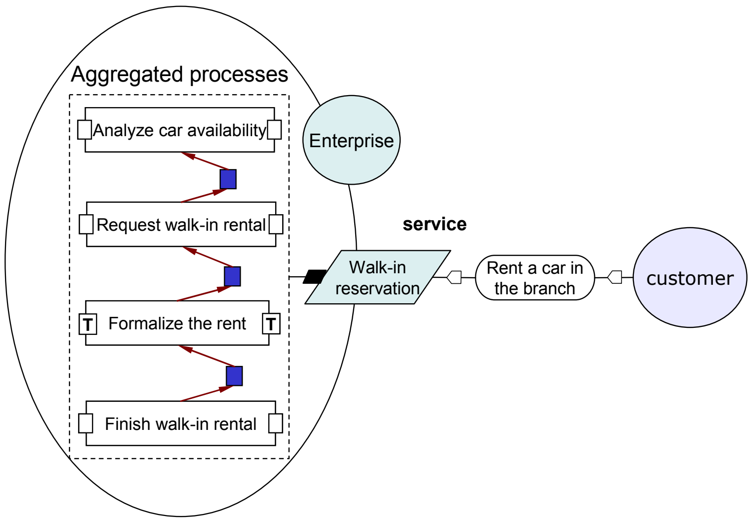


Figure 6.34 Example of the process model for the running example

The business process model provides a high-level view of the process that makes up each one of the Business services offered by the enterprise.

6.6 Defining the protocol model

Once the processes that compose a business service have been represented at a high abstraction level using the business process model, then it is necessary to identify and represent the behavior of each one of the processes that compose the business service. With this approach, the business modeling process can be defining by reducing the complexity of the enterprise analyzing specific fragments of the service (those that represent a specific business process). The main objective of the protocol model is to align the goals that support the business processes with the goals of the organizational actors that perform the process (Figure 6.35). The protocol model, which is the lower-level description of the business service architecture, is represented using the revisited concepts of the *i** framework.

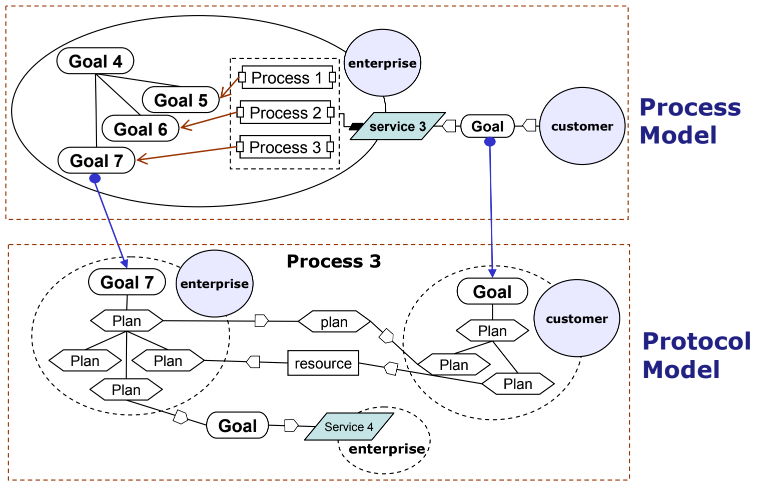


Figure 6.35 Aligning process goals and actor goals

It is important to point out that the protocol model is specified using the revisited version of the *i** modeling concepts; therefore, no new primitives have been added in this modeling stage.

The method required to create the protocol model is similar to the process to create the detailed view of the global model and the process model: use the intermediate model as elicitation mechanism and then, translating this model in a protocol model that represents the behaviors needed to perform each business process. However, a more complete goal classification was proposed to capture the organizational setting.

We propose using the goal-refinement tree to create an organizational model that allows us to carry out these business analyses (business process reengineering analysis, dependency analysis, and task analysis*)* before taking decisions on the functionality of the information system. This allows us to have an improved organizational model that could be used to take design decisions. We consider that the *i** framework provides enough semantic richness to represent the complex social structures in the enterprise. The transformation process to generate *i** models from goal structures has been presented in detail in (Estrada, Martinez, and Pastor 2003). Following, the steps to create the protocol model are defined:

Step 1. Determining business tasks by refinement

The first step of this phase consists of representing the internal goals associated with each business process using the goal-refinement tree in the intermediate model. To do this, the main goal of each business process is place as a root of the goal-refinement structure. Then, the subgoals needed to satisfy this goal must be elicited and placed in the goal structure. The intermediate nodes represent the groups of low-level goals for the satisfaction of a more general goal. Finally, all the leaves represent operational goals that satisfy the low-level goals. This refinement process ends when the level of specific business tasks is reached.

Step 2. Determining process goals by abstraction

The process owners identify the business tasks needed to satisfy each business process, and then associate these tasks with the goals elicited by goal-refinement.

Step 3. Linking the goals and process with organization actors

This step consists of associating the elicited goals and tasks with the composite actor structure. This is done to assign the responsibilities to achieve the process goals and tasks.

As a result of these phases, an expanded intermediate model is created (Figure 6.36) that associates the business service and the processes. As stated above, the goal of each process detected in the previous steps must the root of the goal-refinement tree.

Step 4. Detecting dependencies from the goal-refinement tree

In this step, the goal-refinement tree in the intermediate model is used to create the dependencies among actors that are needed to perform the processes that make up each business service.

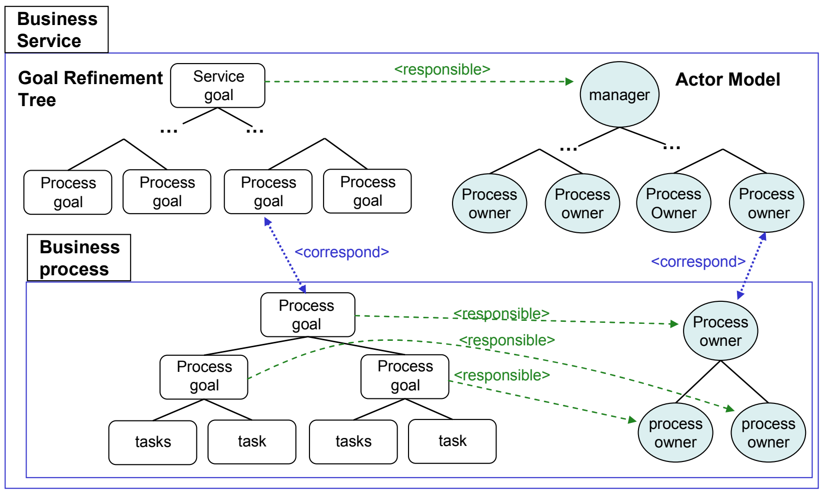


Figure 6.36 The expanded intermediate model

In the goal transformations proposed to generate the global and the process model, we only consider two types of elements: goals and services. However in the generation of the protocol model, task and resource dependencies need to be also represented to capture the organizational behavior for each process. To do this, a more complex goal classification was developed to represent not only the internal goals or operations of the business actors, but also to represent the cases where there are relationships among actors. Relations of this kind imply that the actors depend on other actors to satisfy their goals or perform their operations. These relations are fundamental for creating the strategic models of the *i** Framework that represent each protocol model. For this reason, the goal classification is not exhaustive; we classify only the goals necessary to create an *i** business model.

Operational Goals: They are performed by the correct state transition of one of the business actors and change the state of one or more objects (Dardene et al. 1993). They are characterized by pre-, post- and trigger- conditions. There are two types of Operational Goals:

Operation-Dependency. In this case, the actor responsible for completing the operation depends on another actor to provide a resource or execute another operation. This kind of Operational Goal is represented in the GRT as OP-Dep.

Operation Without-Dependency. In this case, the actor responsible for completing the operation does not depend on another actor to complete the operational goal. This kind of Operational Goal is represented in the GRT as OP-WDep.

Achievement Goals: These goals are refined in Operations Without-Dependency or in other Achievement Goals. They are represented in the GRT as *AG*.

Achievement-Dependency Goals: These goals are refined in Operational Goals, where at least one of these is an Operations-Dependency or in they are defined in another Achievement-Dependency Goal. They are represented in the GRT as *ADG*.

General Goals: These are high-level goals that are used to express the business manager's point of view. Goals of this type lead directly to General Goals, Achievement Goals or Achievement-Dependency Goals.

This classification (Figure 6.37) will be the basis to transform the goal-refinement tree into the strategic dependency and rational model for the protocol model.

Step 5. Generating dependencies from the goal structure

The goal-refinement tree is the starting point for the generation of a protocol model represented in the i* framework. The process begins with the creation of a strategic dependency model (SD model). The SD model is focused on representing the dependency relationships that exist among the organizational actors. For this reason, this model must be constructed using a subset of the GRT (the goals in which a dependency exists between the actors).

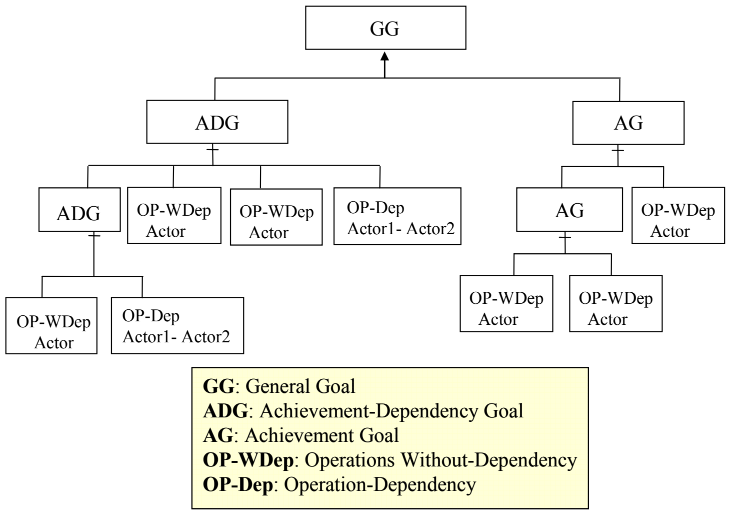


Figure 6.37 The classification in the goal-refinement tree

The first step is to use the organizational actors of the GRT to create the actors of the SD model.

The second step is to use the Achievement-Dependency Goals of the GRT to create the goal dependencies in the strategic dependency model. As mentioned above, the Achievement-Dependency Goals are goals that are refined Operational Goals where at least one of these is an Operation-Dependency. Therefore, these kinds of goals represent dependency relationships between actors.

The third step is to use the Operation-Dependency of the GRT to create the resource and task dependencies of the strategic dependency model. As mentioned above, the Operation-Dependencies are goals that involve more than one actor for their execution. The Operational Goals performed by a single actor represent the internal actions of each actor in the strategic rationale model. An Operation-Dependency must be translated into a task dependency if the actor that depends on the execution of the operation specifies a particular way of doing it. An Operation-Dependency must be translated into a resource dependency if the *depender* actor depends on the delivery of a resource to complete the operation.

The SD model is useful for detecting potential problems with the performance of the business model for finding: actors with a large number of dependencies, actors that represent bottlenecks, redundant dependency relationships, etc. This information can be used to improve the business model.

Once the SD model is created, the strategic rationale model must be created in order to detail the internal tasks that accomplish the dependencies.

Step 6. Generating a rationale model from goal structure

The construction of the strategic rationale model (SR model) consists in defining the internal operations that all actors carry out in order to reach their dependencies. To do this, the Achievement Goals of the goal-refinement tree are translated into internal goals or internal tasks in the strategic rationale model. This is done using task decomposition to create internal task-refinement trees in each business actor. Some of these internal goals or tasks will be connected with the task dependencies or resource dependencies defined in the strategic dependency model.

In the case of operations of the GRT that have been derived in resource dependencies, it is necessary to indicate the delivery of the resource in the *depender* actor. To do this, an internal task must be created in the *depender* actor to indicate the delivery of the resource and link it to the resource dependency. As result of the process, a business model is generated that represents the rationalities behind the business processes (Figure 6.38).

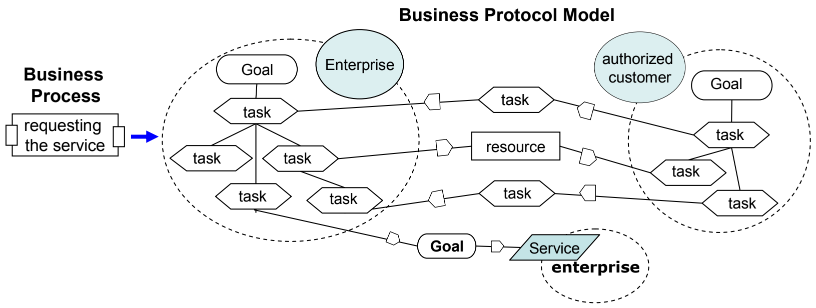


Figure 6.38 Strategic model generated for business processes

Step 7. Determining transactional business processes

The next step consists of the determination of the transactional process. To do this, each one of the processes is analyzed in order to determine the processes that comply with the ACID properties of the business transactions: a) **A**tomicity - The entire sequence of actions must be either completed or aborted. The transaction cannot be partially successful, b) **C**onsistency - The transaction takes the resources from one consistent state to another, c) **I**solation - A transaction's effect is not visible to other transactions until the transaction is committed, d) **D**urability - Changes made by the committed transaction are permanent and must survive system failure. The transactional process must be indicated in the process model by placing an indication in the box that represents the process (Figure 6.34). In our case study, the *formalize the rent of a car* process can be considered as a transactional process. When the rent is formalized, all operations that compose the process must be executed; therefore it is not possible to only perform the payment without receiving the car rental contract, or it is not possible to receive the car without the previous payment. All operations must be successfully performed to have a car rented. This is why this process is marked as transactional in Figure 6.34.

Figure 6.39 presents an example of the protocol model for the running example. It is important to point out that the definition of business protocol enables the analysts to represent the enterprise as a service requester. In this model, the enterprise depends on the customer to obtain her/his personal data and the customer depends on the car rental company to obtain acceptance of rejection of the request to rent a car. In our case study, the analyzed enterprise uses an external business service provided by a bank entity to validate the credit of the customer that requested a walk-in rental.

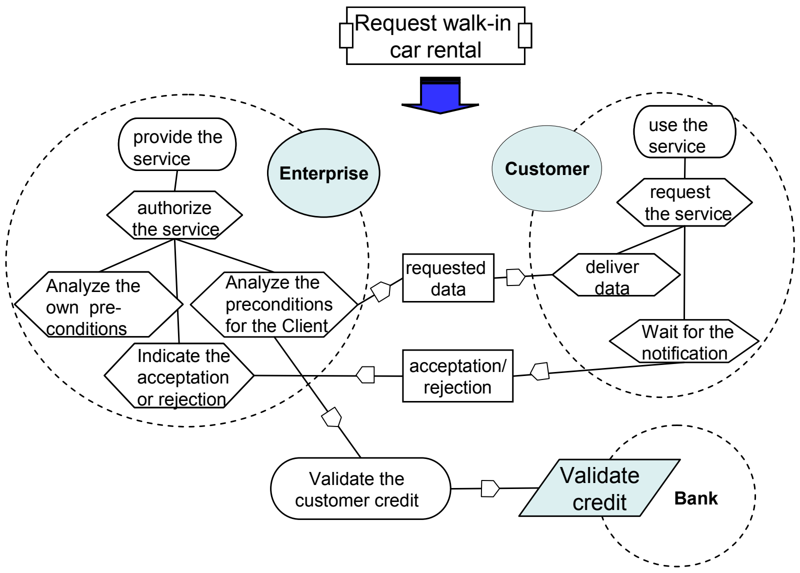

Request walk-in car rental
provide the service
Enterprise
authorize the service
Analyze the own pre-conditions
Analyze the preconditions for the Client
Indicate the acceptation or rejection
requested data
acceptation/ rejection
Customer
use the service
request the service
deliver data
Wait for the notification
Validate the customer credit
Validate credit
Bank

Figure 6.39 The protocol model for requesting walk-in rental

6.7 The service-oriented method as a mechanism to align business goals

One of the advantages of the service-oriented method proposed in this thesis is the explicit capability of the method to align the goals among the service refinement levels. In the global model, the objective is to align the goals of customers with business services and then, associate the enterprise objective with the offered business services (Figure 6.40). In the process model, the objective is to align the service goal with process goals by abstraction and refinement. Finally, in the protocol model, the process goals are aligned with the goals of the actors that are involved in performing the business processes. This approach enables the analysts to trace a specific goal through service decomposition, and it also permits justifying each business activity with the elicited enterprise goals.

We consider that although exhaustive goal analyses are needed to accomplish the proposed method, the result of the process is a consistent business model

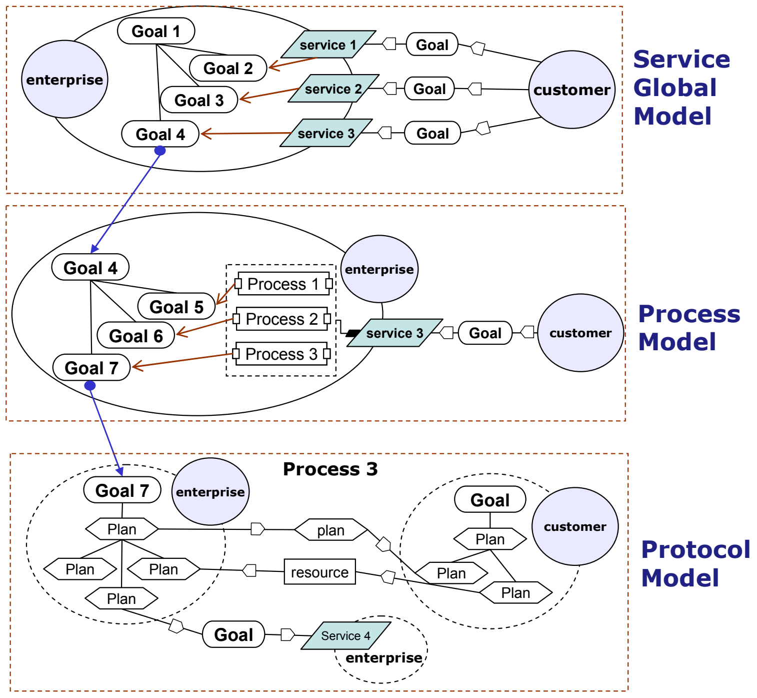


Figure 6.40 The strategy for aligning service, process and protocol goals

6.8 Analyzing the future enterprise situation

The objective of analyzing the future enterprise situation is to produce a description of the alternative solutions for offering/implementing business services in order to satisfy the desired goals of the enterprise. To do this, two different approaches can be selected: a) propose new business services that enable the enterprise to adapt to new external conditions, and b) adapt the existing business services to fit new market conditions.

6.8.1 Analyzing the market conditions

An analysis of the current business process model with the new market conditions must be performed in order to detect the needed modifications.

Several conditions can affect business services, such as policies, new market restrictions, the economic situation of the enterprise, etc. These conditions obligate the enterprise to consider eliminating or changing the current business services, or these conditions can obligate the enterprise to define new business services.

6.8.2 Defining objectives to be satisfied

The definition of the goals to be achieved by the inclusion or modification of current business services is the first stage in representing the future enterprise situation. The global model offers appropriate means to represent these goals to be satisfied.

The alternative solutions to resolve the challenges of the enterprise are represented using the concept of goal. However, it is also necessary to determine how the proposed alternative solutions influence the quality attributes predefined in the enterprise. The quality attributes are represented as non-functional attributes (softgoals) that qualify the set of alternative solutions to the new market conditions. Using softgoals, it is possible to evaluate the positive and negative contributions of the alternative solutions with the desired goals to be achieved. Figure 6.41 shows the schema for using softgoals in the context of the proposed service-oriented approach.

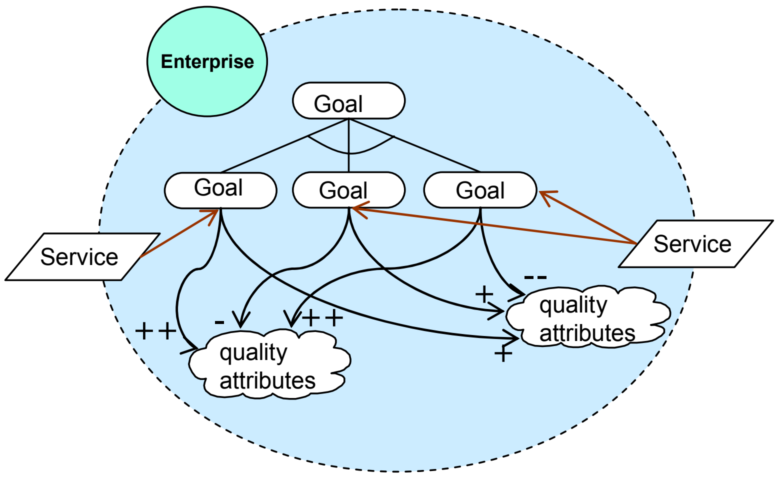


Figure 6.41 Softgoals to represent the desired goals in the future enterprise situation

Softgoal analysis applies to the different refinement levels of the proposed service-oriented approach: global, process and protocol models.

In our case study, the goal defined to represent the desired future enterprise situation in car rentals was to increase the customer satisfaction in checking out the rented car. Alternative solutions must be proposed in order to achieve this objective. The alternatives evaluated are: a) increasing the associated branches b) providing express check out and c) delivering the car directly to the customers. The analysis of contributions is applied to evaluate the impact of the proposed solution over the following attributes: cost, security, and effort.

The contribution analysis permits us to determine that even express check out can be convenient for customers, this solution increases the risk in security due to fraudulent users. In the case of increasing the number of associated branches, it implies increasing cost. Finally, the option of delivering the car directly to the customer seems to be best option because the costs are smaller than the option of opening new branches (Figure 6.42).

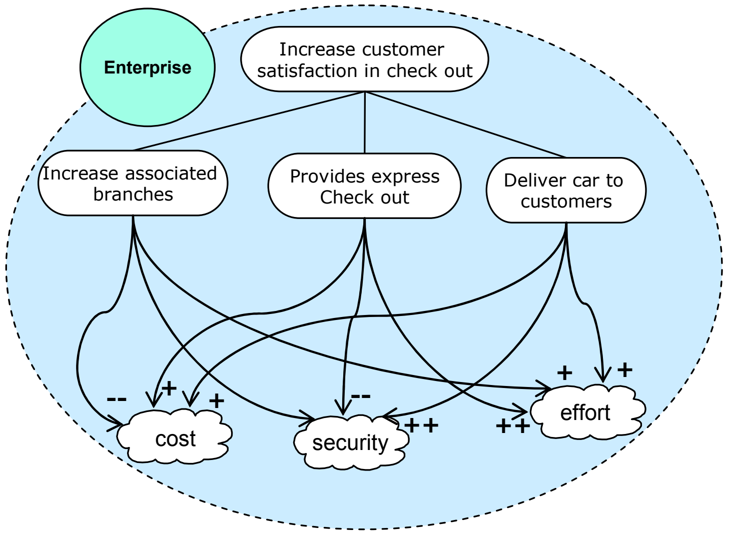

Enterprise
Increase customer satisfaction in check out
Increase associated branches
Provides express Check out
Deliver car to customers
--
+
+
--
++
++
+
+
cost
security
effort

Figure 6.42The softgoal as mechanism to evaluate alternative solutions

6.8.3 Adapting the enterprise to the selected alternative

Consider the situation where the enterprise has decided to modify the way it currently delivers the rented car based on the analysis of contribution and softgoals. In this case, the current process for delivering cars to customers must change: instead of performing the checkout of the car in the branch, the car rental company decides to deliver the car to the customer. This modification implies changes in the *car rental* business service in order to add the process of delivering the car. In the modified service, when the reservation is formalized, the company assigns a driver and a vehicle as required in the service contract. Therefore, the process *select driver* and *deliver process* are included in the process model of the *car rental* business service (Figure 6.43).

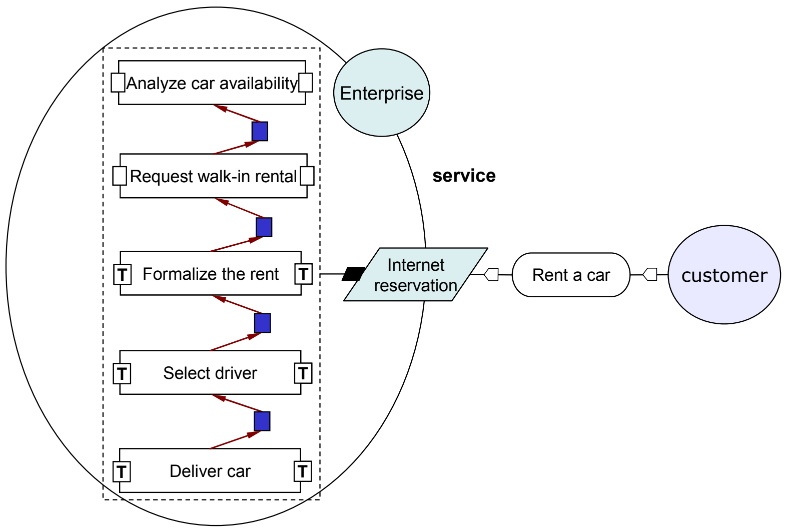


Figure 6.43 New process model for the car rental business service

The following step is to generate new processes to select car drivers and to deliver cars to customers, the rest of the processes remain the same. With the proposed business service architecture, it is possible to add or remove processes or services without affecting the entire business model. The approach enables the analyst to reuse entire processes to modify current services or to add new services to the business model.

The powerful semantics of softgoals and contributions analysis enables the analyst to carry out several performance analyses to be sure that the business model appropriately fits the market conditions.

6.9 Summary

As a solution to the issues detected in the experimental evaluation, a method to represent an organizational model as a composition of business services has been proposed. In this method, the services represent the functionalities that the enterprise offers to potential customers. Thus, the business services are the building blocks that allow us to represent a business model in a three-tier architecture: business services, business processes and business protocols. The organizational modeling process starts with the definition of a high-level view of the services offered and used by the enterprise. Later, each business service is refined into more concrete process models, according to the business service method proposed in this thesis. Finally, business protocols are represented using the revised version of the modeling concepts of the *i** framework proposed in Chapter 4.

The proposed method is composed of two main steps: a) define the current enterprise situation, which reflects the current business processes required to satisfy the current business goals, and b) define goals to be satisfied in the future situation of the enterprise, and adapt the current enterprise model to fit the desired goals.

This method to elicit the current situation of the enterprise group several techniques such as goal modeling and organizational chart modeling to construct a business service model that is represented using the proposed architectural diagrams. The main advantage of this proposal is that it provides a solution for the problems of refinement and granularity. It is important to point out that many of the negative results in the evaluation of *i** are related to the lack of mechanisms for controlling the refinement and the granularity of the information represented in the organizational model.

The proposed guidelines for determining the future enterprise situation are based on the analysis on the softgoals and contributions links in order to represent how the alternative solutions influence the quality attributes of the enterprise. The analysis of contributions represents the appropriate mechanism to help managers in taking decisions about the future of the enterprise.

Chapter 7

7. The Service-Oriented Method: a case study

In this Chapter, we illustrate the service-oriented method using a real project as a case study in the domain of education institutions.

7.1 Introduction

We validate the method proposed in this thesis by developing a case study in the domain of education institutions. The case study consists of a real project to model the processes of a postgraduate institution (www.cenidet.edu.mx) that offers Master and PhD programs in the following areas: computer science, mechanics and electronics

The objective of the analysis was to determine if the business processes of this education institution meet the requirements needed for the certification in the quality management standard ISO 9001:2000. This standard helps to evaluate if a service-oriented organization achieves standards of quality that are recognized throughout the world.

The objective of the case study was to use the service-oriented method to model the specific process to register students in the academic semesters of the postgraduate programs. The case study was implemented by students of a Master program in computer science of the institution being analyzed.

7.2 Applying the service-oriented method

The first step in achieving the objective of the proposed method is to develop a strategic dependency model using the "pure" *i** notation (Yu, 2003). This model represents the dependencies among the organizational actors, making it explicit the social behaviors of the actors in the business model.

The actors involved in the process to register students in the educational company are the following: vigilance agent, students, professors, faculty advisors, student control department, studies control department, department chair, finance department, and planning department. This information was elicited by using the manuals of processes of the institution and by personal interviews with Directors and department managers.

Figure 7.1 presents the *i** dependency model for the *registering student's* case study. The model represents the dependencies of the actors that are needed to accomplish the student registration. In this model, the student is the actor that is responsible for a large number of actor dependencies with all the actors in the model. Thus, if the student fails to perform some goal, the entire process will fail. The dependency model makes explicit the issues in the model; however, this model becomes to be difficult to manage if it grows in size and complexity.

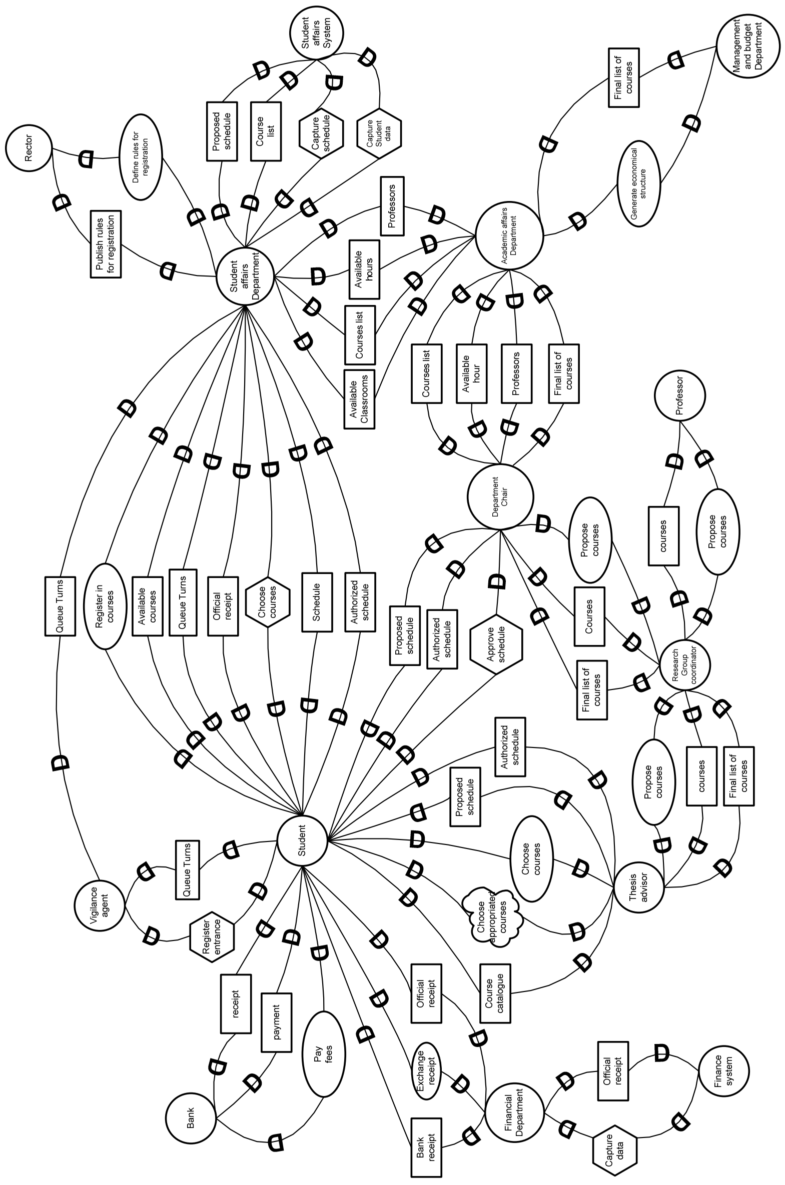


Figure 7.1 Strategic dependency model for the register students' case study

Figure 7.2 presents a fragment of the strategic dependency model that shows the dependencies of the student with the business actors in the model.

In this model, the student is involved on the following dependencies as the *depender* actor (The student depending on other business actors):

- The student depends on the bank to pay the fees of the registration.
- The student depends on the Vigilance Agent to obtain a queue turn to register (the queue turn establishes the order to register students).
- The student depends on the Financial Department to obtain the official payment receipt.
- The student depends on the Student Affairs Department to make the registration
- The student depends on Student Affairs Department to obtain the list of available courses.
- The student depends on the Student Affairs Department to obtain the authorized schedule.
- The student depends on the Department Chair to authorize the schedule.
- The student depends on the Thesis Advisor to make the selection of courses.
- The student depends on the Thesis Advisor to obtain the course catalogue.

In all this dependencies the student becomes vulnerable if the other actors fail to deliver a resource or satisfy a goal.

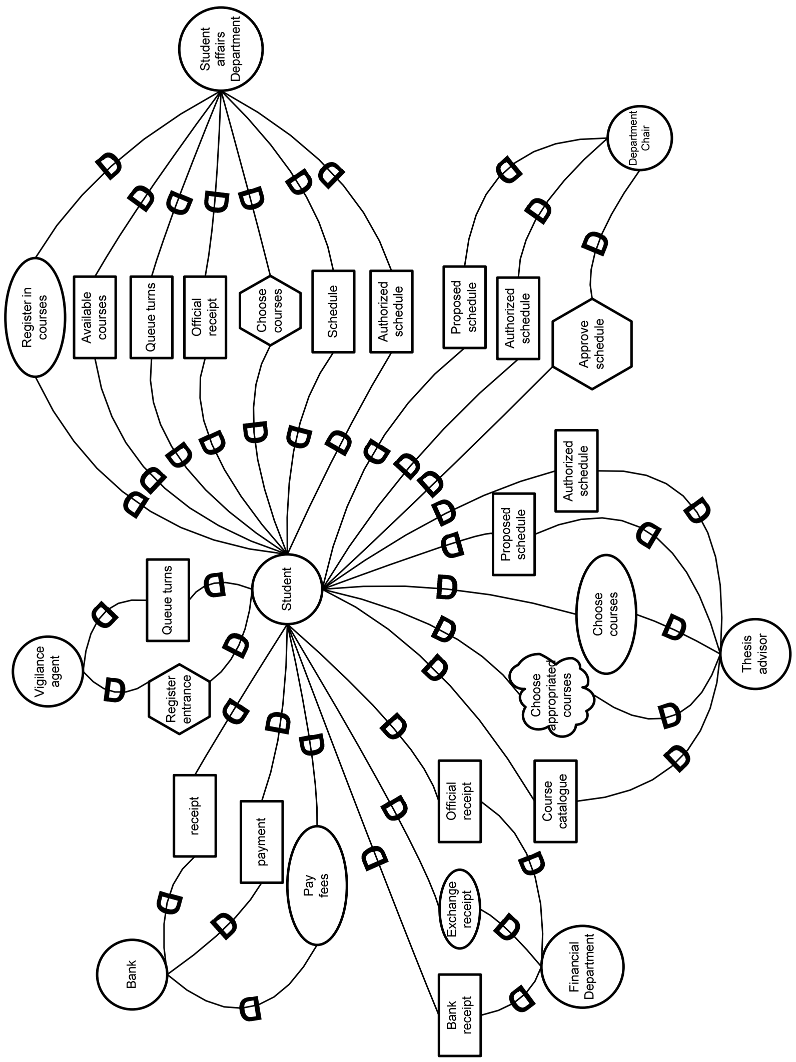


Figure 7.2 Fragment of the strategic dependency model

Once the dependencies among actors have been detected in the previous stage, a rationale model needs to be created that represents the rationalities of the organizational actors.

The rationale model is focused on describing the internal behaviors needed for the actor to fulfill its dependencies with other actors in the enterprise process. To determine this information, personal interviews with organizational actors were done in order to specify the role of each actor in the processes that are needed to register students in the Master and PhD programs.

Figure 7.3 illustrates the rationale model for *registering student's* case study. In this model, the analyst must represent the internal goals and tasks that are needed to satisfy the actor dependencies. One of the issues the strategic rationale model is that all the elements in the model are represented in the same abstraction level, without indications of the hierarchy of objectives and tasks. In this current modeling scenario, the model generated as a result of this process is a composite of a large number of elements. This situation makes it difficult to determine which fragments of the business model correspond to the business processes that help to fulfill the organizational objective of the educational institution. It is also very complicated to try to follow the chain of events and objective associated to the satisfaction of a specific actor goal.

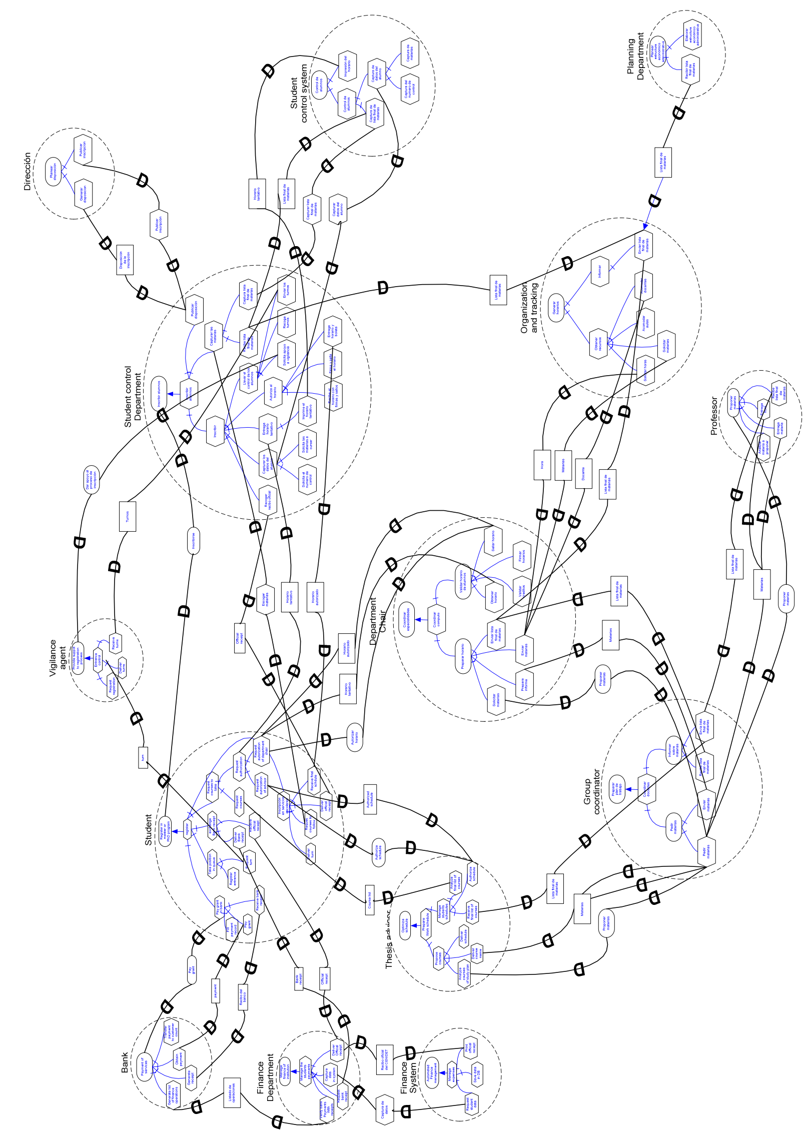


Figure 7.3 The strategic rationale model for the case study

Figure 7.4 illustrates a fragment of the strategic dependency model which is focuses on the dependencies of the students. In this model, the student performs the following actions to register in the master or PhD program: a) Pay fees in the bank, b) Take position in queue, c) Exchange bank receipt, d) Request courses to take, and e) Register in the Student affairs Department.

The task decomposition tree for each high-level goal is presented below.

- Pay fees

  * pay fees in the bank
  * receive bank receipt
- Take position in queue

  * register entrance
  * request turn
- Exchange bank receipt

  * deliver bank receipt
  * receive official receipt
- Request courses to take

  * request courses
  * request authorization
- Register in the Student Control Department

  * deliver turn
  * request courses to follow
  * deliver official receipt
  * receive final schedule

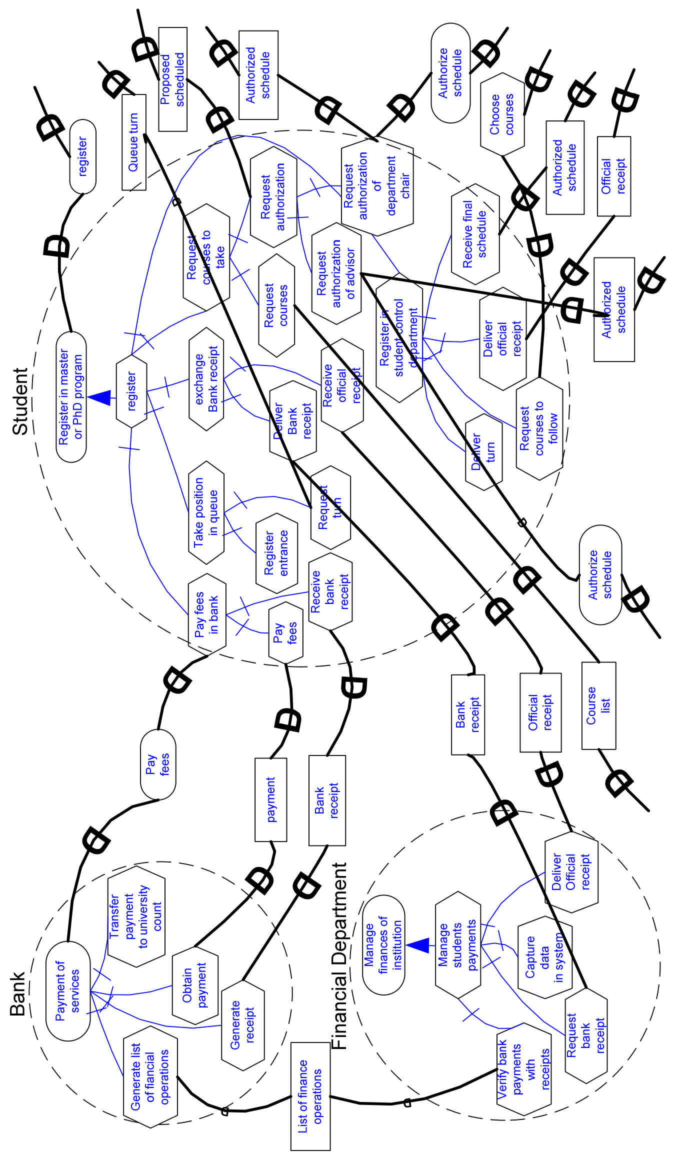


Figure 7.4 Fragment of the strategic rational model for the case study

The conventional dependency and rationale models were generated in order to demonstrate the differences between the current *i** approach and the service-oriented approach proposed in this thesis.

The first step of our service-oriented method is to define a global model that represents the enterprise being analyzed as a service provider (Figure 7.5). As a result of the analysis of the academic institution, a set of offered services have been detected: innovative products development, development projects, industrial courses, PhD programs, Master programs, and research projects development. It is important to point out that this model represents a more abstract level that that dependency model shown in Figure 7.1 because the service model represents all the services of the enterprise in a simple and clear view while the dependency model represents only a process of the enterprise being analyzed.

The association between the offered business services and the potential customers is represented below:

- The innovative industry requests to develop research projects.
- The student requests to follow a master degree program.
- The student request to follow a PhD degree program.
- The industry requests to develop innovative products.
- The industry requests to develop application projects.
- The industry requests to take industrial courses.

Each one of the potential customer has specific objectives to request a service. As stated above, the global model has the appropriate abstraction level to make the first agreements with the final customers.

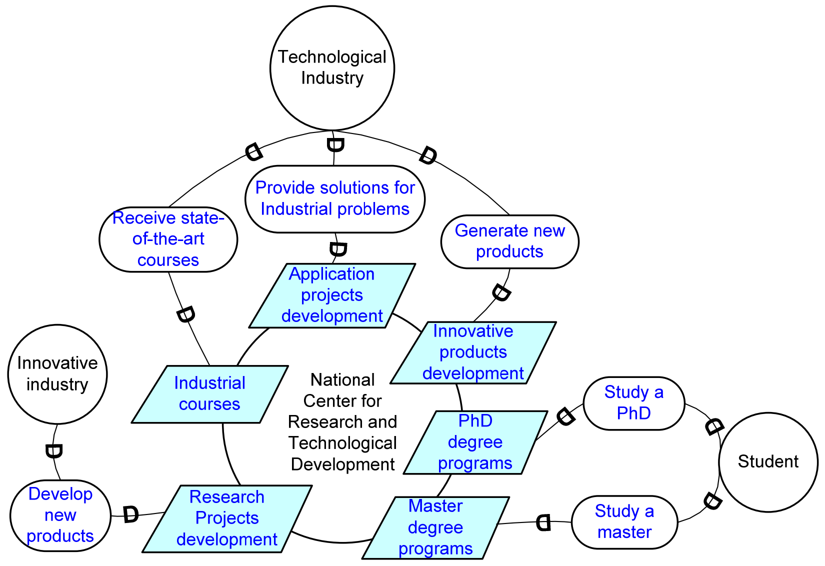

Technological Industry
Provide solutions for Industrial problems
Receive state-of-the-art courses
Generate new products
Application projects development
Innovative products development
Innovative industry
Industrial courses
National Center for Research and Technological Development
Study a PhD
PhD degree programs
Student
Develop new products
Research Projects development
Master degree programs
Study a master

Figure 7.5 the global model for the scholar institution being analyzed

The formulas that represent the global model for the case study are presented below:

Type Predicates
actor(innovative_industry) actor(student) actor(industry) agent (CENIDET) service (offered, research_projects_development) service (offered, master_degree_programs) service (offered, PhD_degree_programs) service (offered, innovative_products_development) service (offered, application_projects_development) service (offered, industrial_courses) own(CENIDET, research_projects_development) own(CENIDET, master_degree_programs)

```
own(CENIDET, PhD_degree_programs)
own(CENIDET, innovative_products_development)
own(CENIDET, application_projects_development)
own(CENIDET, industrial_courses)

provide(CENIDET, research_projects_development)
provide (CENIDET, master_degree_programs)
provide (CENIDET, PhD_degree_programs)
provide (CENIDET, innovative_products_development)
provide (CENIDET, application_projects_development)
provide (CENIDET, industrial_courses)

satisfy_ex(research_projects_development,   innovative_industry,
develop_new_products)
satisfy_ex(master_degree_programs, student, study_a_master)
satisfy_ex(PhD degree programs, student, study_a_PhD)
satisfy_ex(innovative_products_development,              industry,
generation_new_products)
satisfy_ex(application_projects_development,             industry,
solution_for_industrial_projects )
satisfy_ex(industrial_courses,    industry,    receive_state-of-the-
art_courses)

request(innovative industry, research_projects_development)
request(student, master_degree programs)
request(student, PhD_degree_programs)
request(industry, innovative_products_development)
request(industry, application_projects_development)
request(industry, industrial_courses)
```

Once a high level view of the enterprise has been generated through the global model, the next step is the definition of the internal structure of each of the services elicited. We select the *Master degree program* business service offered by the academic institution as the example to be illustrated in this thesis. This service is the composite of *register student*, *teaching course* and *student advisory* business services (Figure 7.6). Thus, the *Master degree program* business service is graphically associated to its corresponding services using the global model. The services *teaching course* and *student advisory* are considered as supporting services because they can only be requested by students of the master program. In the case of the service *register student*, it can be requested by new students of the program.

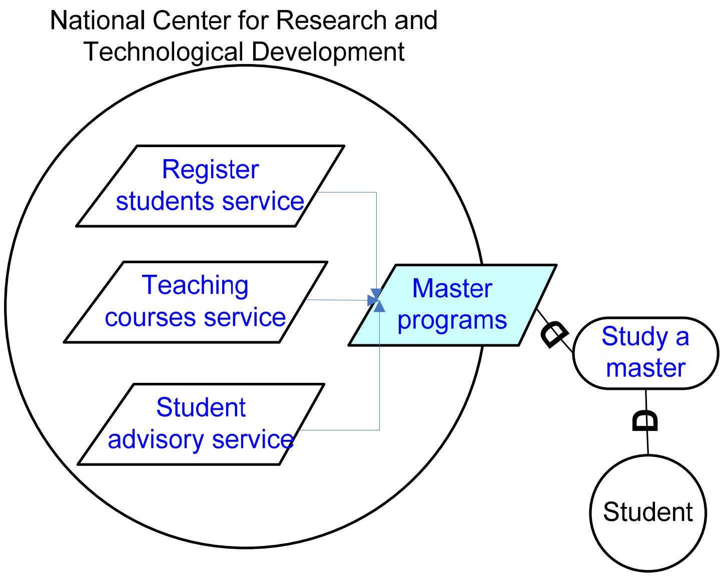


Figure 7.6 Service decomposition for business service

The clauses that formally represent the decomposition of the offered service into aggregated business services are presented below:

Service relations

service(offered, master_program)
service(offered, register_students)
service(supporting, teaching_courses)
service(supporting, student_advisory)
mandatory_decomposition(master_program, register_students, teaching_courses, student_advisory)

The next step in the proposed method is to determine the relation among the high-level goals of the enterprise and the offered business services. This is done by determining how the offered services help to fulfill the general goals of the enterprise (Figure 7.7). We have selected the *register students* business services for an in-depth analysis.

The goals that have been elicited need to be represented in the expanded view of the service global model. It is important to point out that all services represented in this model have influence on the satisfaction of the goals of the enterprise.

The global model in Figure 7.7 indicates that in order to provide the service *register students* business services, the following goals need to be fulfilled: a) offer appropriate courses, b) register students in the master/PhD program, c) manage fee payments, and d) manage the courses of the program.

Each of the elicited goals has been refined in a goal-refinement tree in order to determine how the goals need to be satisfied. For example, to satisfy the goal *offer appropriate courses*, the academic institution must fulfill the following sub-goals: a) obtain a list of courses from the professors, b) select courses for a specific semester, and c) authorize the courses.

As stated above, the objective of this phase is the determination of the map between the strategic objectives of the enterprise and the offered business services. This is one of the contributions of this work because other proposals to consider services at the business level (Cherbakov et al. 2005) (Baida 2006) do not consider the association between goals and services.

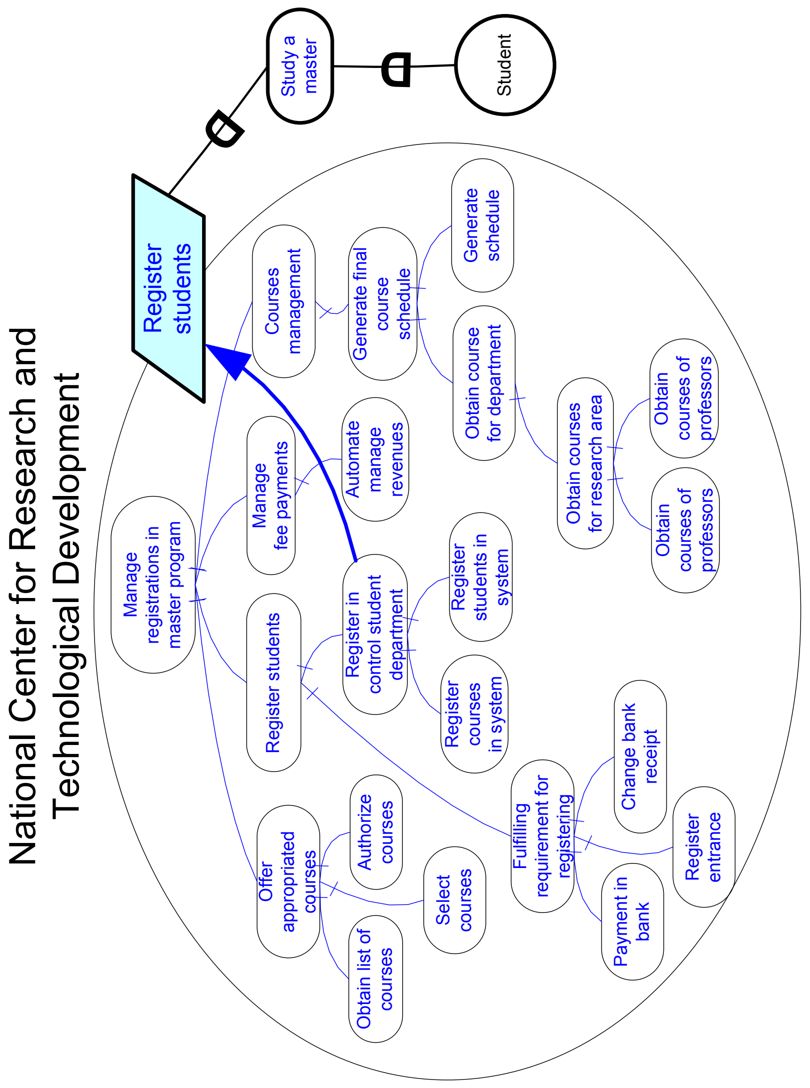

National Center for Research and Technological Development
Register students
Study a master
Student
Courses management
Generate final course schedule
Generate schedule
Obtain course for department
Obtain courses for research area
Obtain courses of professors
Obtain courses of professors
Manage fee payments
Automate manage revenues
Manage registrations in master program
Register students
Register in control student department
Register students in system
Register courses in system
Offer appropriated courses
Authorize courses
Select courses
Obtain list of courses
Fulfilling requirement for registering
Change bank receipt
Register entrance
Payment in bank

Figure 7.7 Goal model associate to register students business service

The clauses that represent the goal decomposition of the actors involved in our case study are presented below:

Goal refinement

AND_decomposition(manage_registration_in_master_program, offer_appropriated_courses, register_students, manage_fee_payments, courses_management)
AND_decomposition(offer_appropriated_courses, obtain_list_of_courses, select_courses, authorize_courses)
AND_decomposition(register_students, fulfilling_requirements_for_registering, register_in_control_student_department)
AND_decomposition(Manage_fee_payments, automate_manage_revenues)
AND_decomposition(courses_management, generate_final_course_schedule)

Once the relationship between business services and enterprise goals are detected, the actors that are responsible for each business goal must be detected to create the expanded global model. It represents how the offered business services are decomposed in a set of goal delegations, where some of these delegations are satisfied through internal business services (Figure 7.8).

The next step of the proposed method is the determination of the actors that are responsible to satisfy the elicited business goals. To do this, the name of the actor that is responsible for the goal must be associated with the elicited goal.

This is one of the main differences of our approach with other goal-based techniques where no actors are identified to be responsible to satisfy the business goals. In our approach, we need to detect goals owners in order to create the associated business service model.

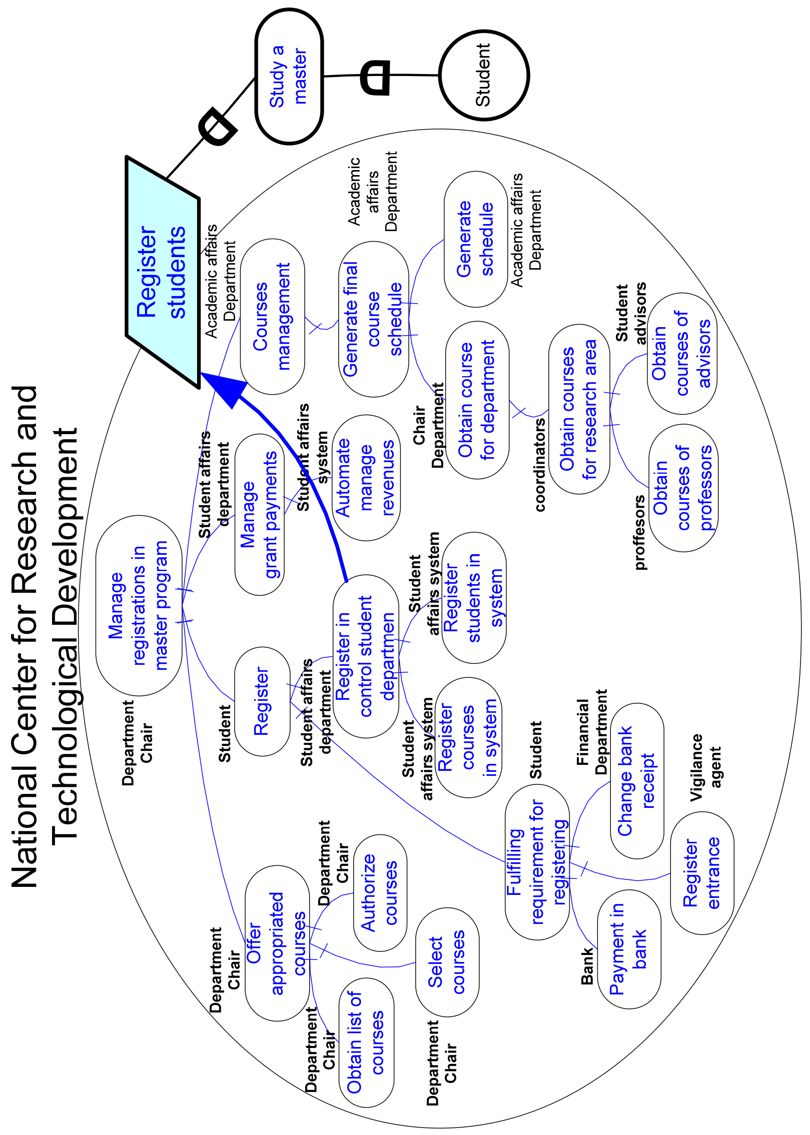

National Center for Research and Technological Development
Register students
Study a master
Student
Academic affairs Department
Courses management
Academic affairs Department
Generate final course schedule
Generate schedule
Academic affairs Department
Chair Department
Obtain course for department
coordinators
Obtain courses for research area
Student advisors
Obtain courses of advisors
proffesors
Obtain courses of professors
Student affairs system
Automate manage revenues
Student affairs department
Manage grant payments
Department Chair
Manage registrations in master program
Student
Register
Student affairs department
Register in control student departmen
Student affairs system
Register students in system
Student affairs system
Register courses in system
Student
Fulfilling requirement for registering
Financial Department
Change bank receipt
Vigilance agent
Register entrance
Bank
Payment in bank
Department Chair
Offer appropriated courses
Department Chair
Authorize courses
Department Chair
Select courses
Department Chair
Obtain list of courses

Figure 7.8 The identification of responsible for elicited goals

Once the actors with responsibilities to satisfy goals have been identified, the algorithm proposed in this thesis must be used in order to obtain a global model that represents the social structure of the actors involved in the analyzed business service (Figure 7.9).

The proposed method uses the goal structure defined in Figure 7.8 in order to detect the potential dependencies among actors. To do this, each element of the goal structure is used to create an element in the global model.

As a result of this process, a global model is created that represents the internal and supporting business services that are needed to satisfy the offered service *register student*. In this model the complete list of stakeholders is generated and the dependencies among these actors can be correctly elicited.

Figure 7.9 represents the global model for the services associated to the registration of students in a master/PhD program. The Student uses the services offered by the Vigilance Agent, Bank, Financial Department, Student affairs Department, Department Chair and Thesis Advisor. As stated above, these services will be further refined into process and protocols.

In our case study, the students do not offer services to other actors participating in the business model. This indicates that student only plays the role of requester of business services.

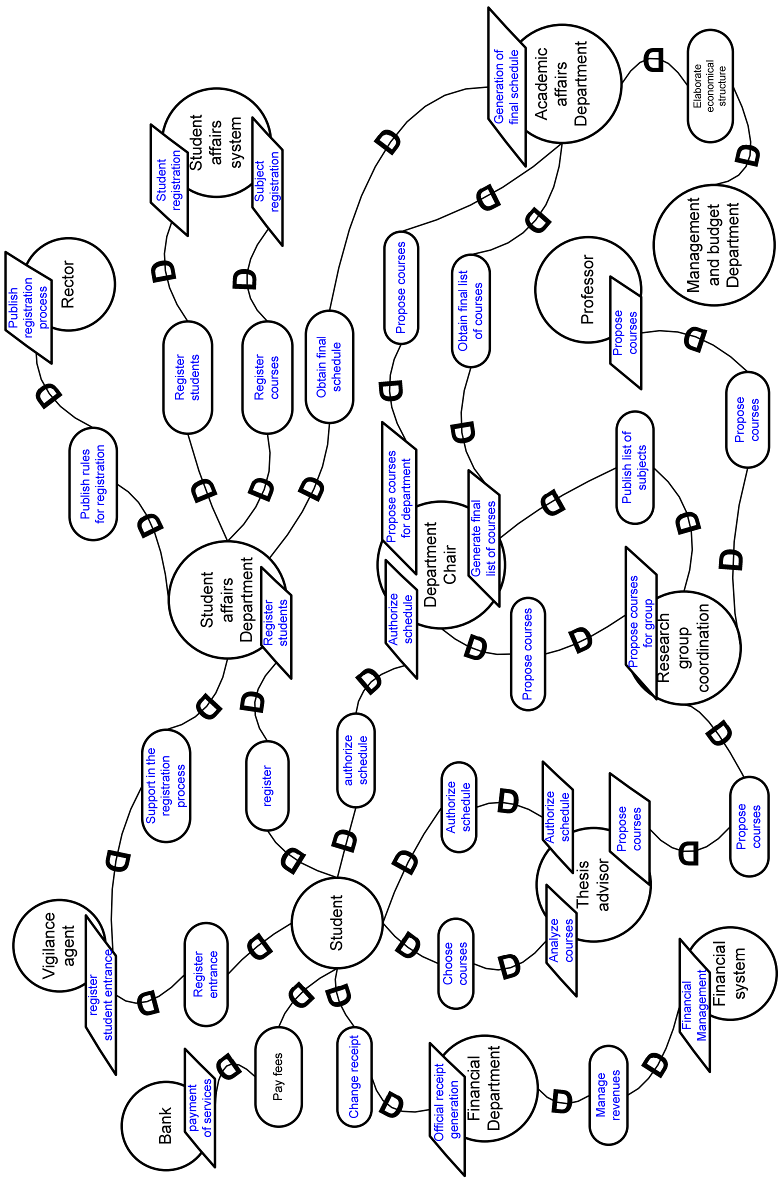


Figure 7.9 The service global model for the analyzed case study

As stated in previous Chapter, one of the basic analyses in this modeling stage is the determination of hierarchical relationships of subordination among actors.

The subordination, which is a key factor in business delegation process, can only be applied to actors that work in the same functional area. In our case study, the department chair, the student's advisor and the professor actors are part of the same academic department. This is why the service *propose courses* can only be delegated following the command chain defined by this hierarchical structure (Figure 7.10).

The analysis of the chain of subordination in the enterprise is useful to validate the chain of dependencies among the actors. In this sense, within a organization unit, an actor can delegate a service to another actor if they are associated through a subordination relationship

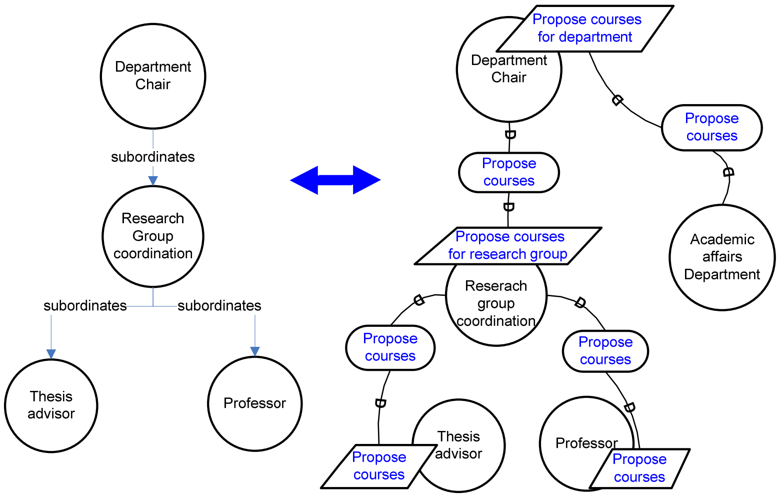


Figure 7.10 Service delegation based on actor subordination relationships

The clauses for defining the service delegation are the following:

```
Service delegation
-----------------------------------------------------------------
actor(Department_Chair)
actor(Group_Coordination)
actor(Thesis_Advisor)
actor(Professor)

service (supporting, propose_courses_for_department)
service (offered, propose_courses)
own(Department_Chair, propose_courses_for_department)
provide(Department_Chair, propose_courses_for_department)
request(Research_Department,  propose_courses)

subordinate(Department_Chair, Group_Coordination)
subordinate(Group_Coordination, Thesis_Advisor)
subordinate(Group_Coordination, Professor)

delegate(Department_Chair, Group_Coordination,
propose_courses_department) ← own (Group_Coordination,
propose_courses_department) ∧ monitoring (Department_Chair,
Group_Coordination, propose_courses_department) ∧
can_satisfy (Group_Coordination, propose_courses_department)
∧ subordinatedchain(Department_Chair, Group_Coordination)

can_satisfy(X,S) ← servicedelegatechain(Department_Chair,
Group_Coordination, propose_courses_department) ∧
can_satisfy(Group_Coordination, propose_courses_department)
∧ subordinatedchain(Department_Chair, Group_Coordination)

delegate(Group_Coordination, Thesis_Advisor, propose_courses)
← own (Thesis_Advisor, propose_courses) ∧ monitoring
(Group_Coordination, Thesis_Advisor, propose_courses) ∧
should_satisfy (Thesis_Advisor, propose_courses) ∧
subordinatedchain (Group_Coordination, Thesis_Advisor)
```

should_perform(Thesis_Advisor, propose_courses) $\leftarrow$ provide (Thesis_Advisor, Propose_courses) $\wedge$ service_delegatechain (Group_Coordination, Thesis_Advisor, propose_courses) $\wedge$ subordinatedchain(Group_Coordination, Thesis_Advisor)

delegate(Group_Coordination, Professor, propose_courses) $\leftarrow$ own (Professor, propose_courses) $\wedge$ monitoring (Group_Coordination, Professor, propose_courses) $\wedge$ should_satisfy (Professor, propose_courses) $\wedge$ subordinatedchain (Group_Coordination, Professor)

should_perform(Professor, Propose_courses) $\leftarrow$ provide (Professor, propose_courses) $\wedge$ service_delegatechain (Group_Coordination, Professor, propose_courses) $\wedge$ subordinatedchain(Group_Coordination, Professor)

The next step in the service-oriented method is the determination of the goals that are supported by the business services offered by the enterprise. The objective of this step is to refine the service's goals until the level where goals can be satisfied by business processes is reached. To do this, the refinement process initiates with the analysis of the general goal of the business service. Then, it is necessary to refine this goal until business processes can be detected.

As stated above, a goal-refinement tree must be constructed where the root of the tree is linked with a specific service that is offered by the organizational actor. Figure 7.11 shows the association of enterprise goals with the *register students* business service of the actor *Student affairs Department*.

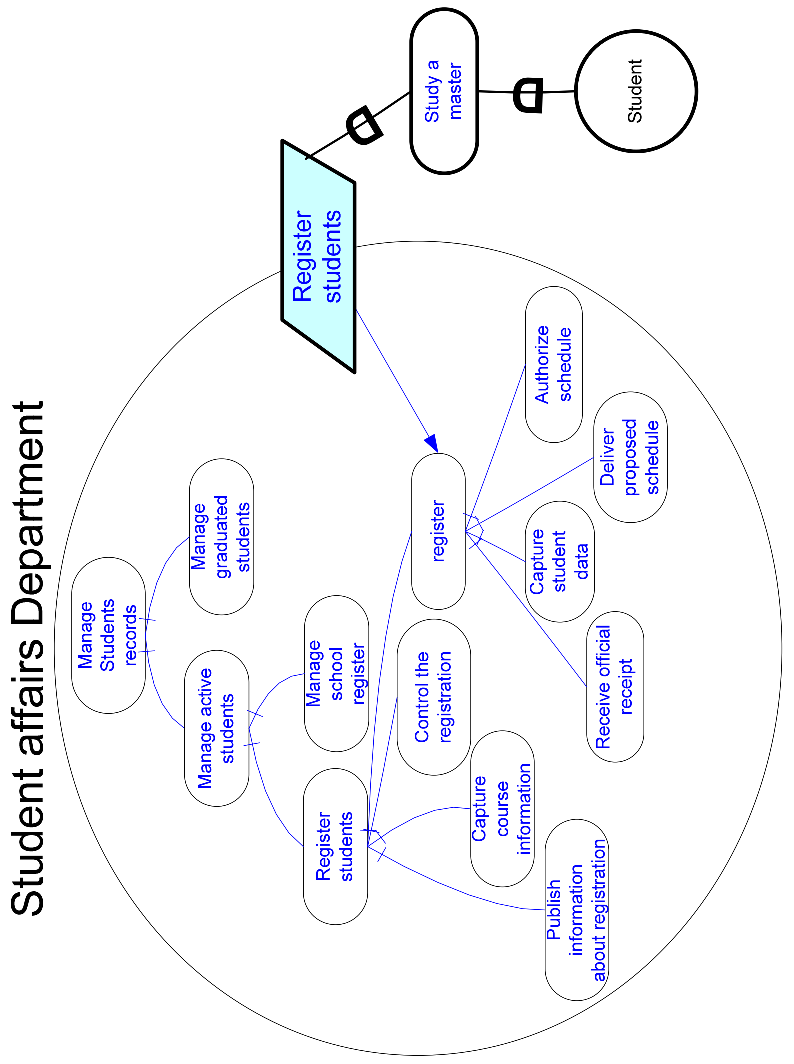


Figure 7.11 The association of enterprise goals and business services

The clauses for defining the decomposition of the service's goal are the following:

Goal refinement

AND_decomposition(manage_student_records, manage_active_student, manage_graduated_student)
AND_decomposition(manage_active_student, register_students, manage_school_register)
AND_decomposition(register_students, publish_infomation_about_registration, capture_courses_information, Control_registration, register)
AND_decomposition(register, receive_official_receipt, capture_student_data, deliver_proposed_schedule, authorize_schedule)
 	
satisfy_in(register_student, register)

The models that are generated from these steps enable the analyst to explicitly represent the reasons of the enterprise to offer a specific business service in a specific manner. The next step of the proposed method consists of defining the businesses processes that permit to implement the business services and also permit to satisfy the business goals.

As commented above, the objective of this modeling stage is the determination of high-level processes that implement the business services. To do this, we propose to represent the processes that make operational the goals identified in Figure 7.11. It is important to point out that only the leaves nodes of the goal structure need to be refined into business processes.

The relationship among business goals and their corresponding business process is presented below. The order of execution of the processes represented in the model is also indicated using a consecutive number.

Goal: Publish information about registration

Process: Obtain information about registration (1)

Goal: Capture course information

Process: Register courses schedule of the professors (2)

Process: Obtain final list of courses (3)

Goal: Control the registration

Process: Request support to vigilance agent (4)

Process: Deliver queue turns to vigilance (5)

Process: Obtain queue turns from students (6)

Goal: Capture student data

Process: Request control number (7)

Process: Request courses to be taken (8)

Goal: Authorize schedule

Process: Receive signed schedule (9)

Process: Seal schedule (10)

Process: Deliver final schedule (11)

The model generated from this step uses the proposed notation to indicate the relation among goals and business processes. This model also enables the analyst to graphically represent the business processes that are needed to provide a business service to potential customers.

The concept of milestone is used in this model to represent the execution order of the processes. The model represents that, for example, in order to provide a signed schedule to the student, the student must determine first the courses to follow..

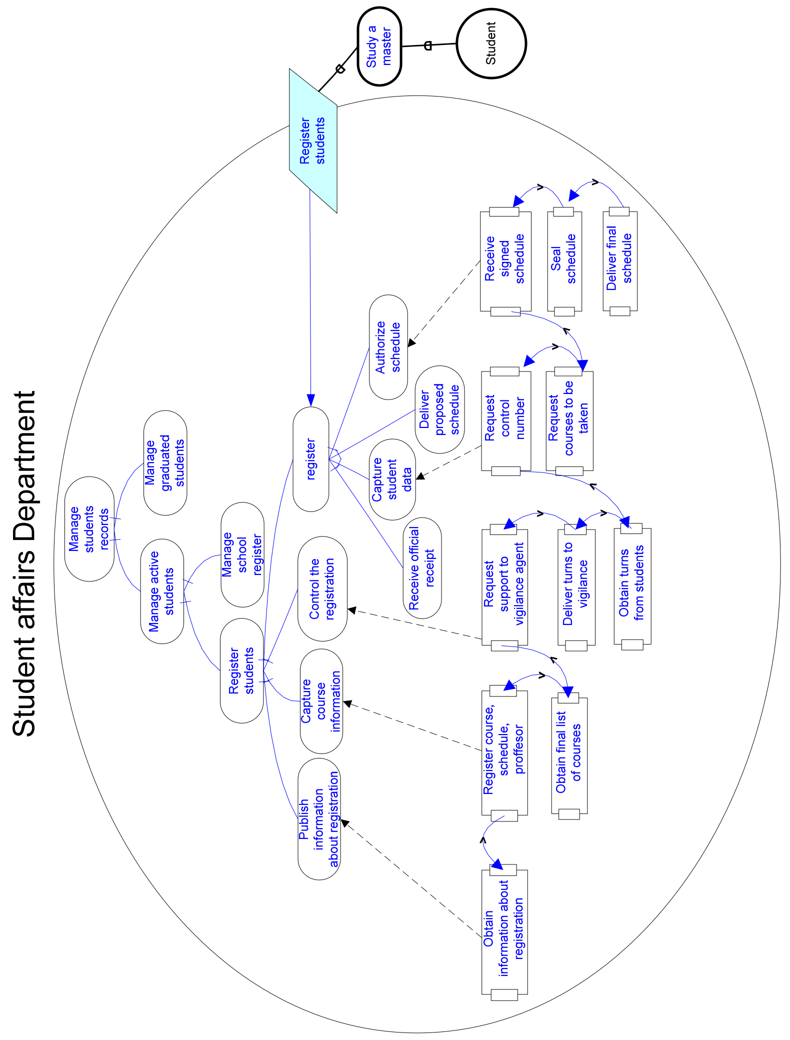

Student affairs Department
Student
Study a master
Register students
register
Authorize schedule
Deliver proposed schedule
Capture student data
Receive official receipt
Manage students records
Manage graduated students
Manage active students
Manage school register
Register students
Control the registration
Capture course information
Publish information about registration
Receive signed schedule
Seal schedule
Deliver final schedule
Request control number
Request courses to be taken
Request support to vigilance agent
Deliver turns to vigilance
Obtain turns from students
Register course, schedule, proffesor
Obtain final list of courses
Obtain information about registration

Figure 7.12 The process model for the *register students* business service

Finally, the last step of the method is the definition of the protocol model for the elicited processes. To do this, the actors responsible for the process must be selected and represented in an isolated business model. As stated above, the protocol model represents the low-level specification in our service-oriented approach. This model is used to present the organization behavior needed to satisfy a specific business process.

As stated above, the protocol model is represented by using the "pure" *i** notation. Thus, the interaction among business actors is based on the concept of dependency. The advantage of this model is the use of *i** concepts to represent a very specific organizational unit. Similarly to previous models, the source for the generation of the protocol model is the goal structure that associates goals and processes.

Figure 7.13 represents the protocol model for the *request support to registration* process. In this model the following business behavior is represented.

The main objective of the student is to make the registration. To do this, the student must: a) register his/her entrance in the institution, b) request a turn to make the registration, and c) deliver the turn in the Student affairs Department. The student depends on the Student Control Department to deliver the turn, and also depends on the vigilance agent to make the entrance register and to obtain the queue turn to make the registration. The Student affairs Department depends on the vigilance agent to provide support to the registration process and to deliver the turns to the students to be registered. To achieve its dependencies, the Student affairs Department must perform the following tasks: request support to vigilance, obtain queue turns from students and send queue turns to vigilance agent. The student depends on the vigilance agent to register the entrance in the institution and to obtain the turns. The Vigilance Agent must request registration, delivers turns and receives turns from the Student affairs Department.

The model has been generated using a goal-refinement tree to capture the goals of the process and then, translating the goal-structure into a strategic rationale model using the proposed algorithm.

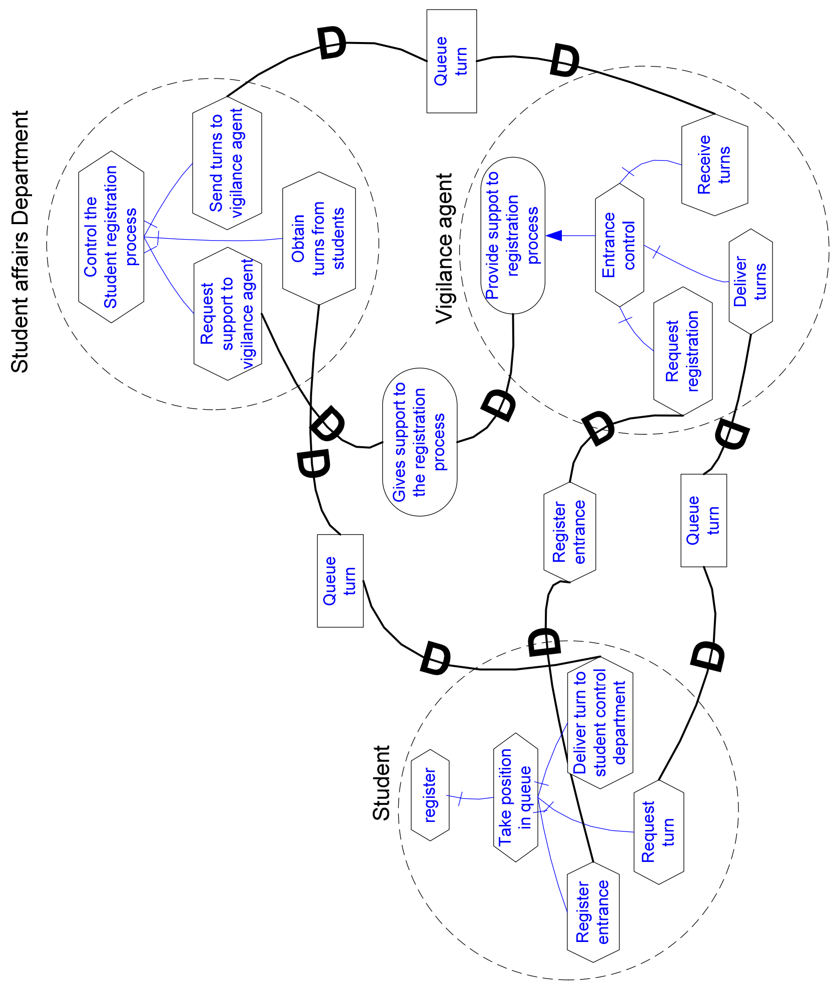

Student affairs Department
Control the Student registration process
Send turns to vigilance agent
Obtain turns from students
Request support to vigilance agent
Queue turn
Vigilance agent
Provide suppot to registration process
Entrance control
Receive turns
Deliver turns
Request registration
Gives support to the registration process
Register entrance
Queue turn
Queue turn
Student
register
Take position in queue
Deliver turn to student control department
Request turn
Register entrance
D

Figure 7.13 Protocol model for process *request support to registration process*

7.3 Analyzing the proposed service-oriented method

Some of the conclusions obtained from the application of the service-oriented method in a real case study are the following:

The proposed method enables the analyst to manage the business modeling process in an incremental way, where the business services are the building blocks. To give preliminary results about the evaluation of the proposed method we use the same evaluation features used in the empirical evaluation (Chapter 3). The students involved in developing the case study gave preliminary analysis about the evaluated features.

Refinement: This is one of the features that were improved over the original definition of the *i** framework. The method permits to start the modeling process with a high level view of the business model. This model, which is made up with a few modeling elements is simpler to be analyzed than current strategic rational model.

Each fragment of the model (represented as a business service) is refined into abstract processes that are detailed in a more concrete protocol model. However, in spite of the advantages of the proposed approach, novel analysts might find it difficult to use the several refinement steps of the method.

Modularity: In the method, the business services play the role of building blocks that encapsulate internal structures of the model. A similar approach is used to define abstract processes that encapsulate a specific organizational behavior represented with the *i** framework. One of the advantages of the service-oriented strategy is that changes for building blocks can be done without affecting other parts of the business model

Repeatability: In this approach, two solutions have been proposed to manage repeatability: revisiting the *i** modeling concepts and providing a method to incrementally develop business models. However, despite the methodological improvements, the repeatability rate is still low. Thus, it could sometimes be difficult for novel analysts to determine whether to consider a specific set of activities as a business process or as a business service. With the proposed method, the definition of the modeling primitives has been improved; however, problems of ensuring repeatability still exist.

Complexity Management: This feature measures the capability of the modeling method to provide a hierarchical structure for its models, its primitives and its concepts. Complexity management has been improved by using a refinement approach. The service-oriented method obligates the analyst to hierarchically construct a business model. The process starts with an abstract model that hides the implementation details and the process ends with the definition of low-level descriptions of business activities.

Expressiveness: This approach adopts the well-founded capability of the *i** framework to represent the social and intentional aspects of a business model.

Traceability: One of the advantages of the service-oriented approach is the possibility to trace a specific business goal through the refinement chain. Our approach enables the analyst to move back and forth between refinement models corresponding to different development stages. The alignment of goals along the different modeling stages is one of the contributions of this work.

Reusability: the possibility to isolate specific business activities in building blocks permits the analyst to reuse building blocks to construct or redefine an existing business model.

Scalability: the problems of scalability have been reduced with the proposed approach; however, it does not offer a definite solution to the scalability problems. One of the scalability problems was detected in the definition of the global model, where the services are represented in the frontier of the actor. This is useful to indicate that services are the interface between the requester and the providers, however, this approach has limitations when a business model contains organizational actor that offers a large number of business services. A new service representation is needed to solve this scalability problem.

Domain Applicability: Our approach adopts the well-founded capability of the *i** framework as an appropriate representation means to represent several application domains.

7.4 Final Considerations

Preliminary results of the application of the service-oriented method permit to determine its advantages to represent a business model in an increasing way. However, more research efforts are needed in order to provide more concrete solutions to repeatability and scalability problems. Intensive and exhaustive empirical evaluations of the proposed approach are needed to precisely detect the weak points of the proposal.

Chapter 8

8. Conclusions and further work

The *i** modeling framework is widely used for organizational modeling. The framework focuses on strategic relationships between actors in order to capture the social and intentional context of an enterprise. This thesis presents our work on improving *i** as a business modeling technique based on a service-oriented approach.

Nowadays, there are many research projects that use the *i** framework in different application domains. In all these applications, *i** concepts have been used to capture the social and intentional elements of each specific domain, thereby supporting software development. However, despite the well-known theoretical advantages of the *i** modeling approach, there are certain issues that still need to be improved to assure their effectiveness in practice. The first part of the thesis discusses an in-depth analysis of the *i** framework that confirms its usefulness and identifies potential weak spots.

8.1 The empirical evaluation of the *i** Framework

One of the main contribution of this first thesis section consists of an empirical evaluation of *i**, using a feature-based evaluation framework and three industrial case studies. The case studies were conducted in collaboration with a software company that has adopted the OO-Method for software development. This is a model transformation method that relies on a CASE tool ([7]) to automatically generate complete information systems from object-oriented conceptual models. The OO-Method can be viewed as a computer-aided requirements engineering (CARE) method where the focus is on properly capturing system requirements in order to manage the complete software production process. Thus, the evaluation framework has been designed keeping in mind that it is to be used within model-based software development environments.

The features that were evaluated (refinement, modularity, repeatability, complexity management, expressiveness, traceability, reusability, scalability and domain applicability) were selected from well-established research works where agent-oriented techniques have been evaluated. In the evaluation framework, the analysts assigned a qualification to indicate how well or badly each feature was supported by *i** framework.

The evaluation has demonstrated that there is a set of issues that needs to be addressed by the *i** modeling framework to ensure its successful application within industrial software development projects. These issues boil down to a lack of modularization mechanisms for creating and structuring organizational models.

Finally, the last contribution of the first thesis section is the definition of a set of conclusions to be considered in the definition of future versions of *i**.

8.2 The definition of the modeling language

In the second part of the thesis, we extended *i** in order to address the weaknesses reported in this paper. Specifically, two complementary solutions were proposed in this thesis to solve the detected issues of refinement, modularity, complexity management, reusability and scalability: the first one is revisiting the *i** modeling concepts and the second one is proposing a business service architecture as an extension of the *i** framework.

With respect to the analysis of the *i** modeling concepts, we present a proposal to formally characterize *i** modeling primitives based on a multi-property framework.

The main idea of using a multi-property framework to characterize the conceptual primitives of *i** was to define a set of properties that defines each modeling element. Our research was focused on the characterization of the *i** relationships. This is because the empirical evaluation demonstrated that *i** relationships (decomposition, means-end, contribution, dependencies and is-a) are the source of most of the repeatability problems found in using *i** in industrial case studies. Therefore, one of the contributions of second section of this thesis was to review the semantics of *i** relationships to ensure that they fit the analysis's needs in practical case studies. This was done instead of following the criteria of using the semantics of the modeling primitives according to a specific methodological technique (Tropos, Grl or *i**).

To perform the characterization, we have classified the *i** relationships according to the standard abstraction mechanisms found in literature: association, aggregation, generalization, and classification. Once each *i** concept was mapped with a specific abstraction mechanism, we defined a multi-property for each modeling category. We selected a set of properties that clearly restricted the way in which the elements associated through the relationship could be associated. Therefore, the framework captured relevant constraints that are expressive enough to ensure that modeling concepts can be properly distinguished.

Formalisms to define the constraints were imposed in order to reduce the possible ambiguity of giving only plain explanations for the proposed constraints. Several meetings were held with designers and users of *i** and Tropos in order to make a consensual judgment about the relevant properties for the modeling concepts. In these meetings, the ambiguities that were detected in practical case studies were presented and discussed. Additionally, we carried out an exhaustive review and analysis of the *i**/Tropos bibliography. By doing this, we were able to reach a consensus about the values for the proposed properties.

As result of revisiting the *i** concepts, we have obtained a stable version of the modeling elements for our business service approach. This version makes possible to clearly differentiate the modeling primitives of *i** so that modelers will have better guidance on what primitives to use in different situations.

8.3 The service-oriented architecture

The last section of the thesis concerns the definition of the service-oriented architecture and the service-oriented method associated with this architecture. Our solution is based on the concept of a Business Service Architecture, where encapsulated organizational units can only participate in actor dependency networks through well-defined interfaces. Our research work is based on the hypothesis that it is possible to focus the organizational modeling activity on the business services offered by the enterprise to their customers. As a consequence of this hypothesis, the proposed method provides mechanisms to guide the organizational modeling process from the business service viewpoint. The proposed service-oriented architecture for the *i** framework allows that the monolithic structure of the *i** strategic rationale model to be broken down into several business services.

Another contribution of this research work is that the focus of the modeling activity has been changed from the actor's viewpoint to the service's viewpoint. In the current state of *i** and Tropos, the modeling process is focused on how to discover the actors´ tasks that are needed to satisfy the actors´ goals and objectives. As a result of this analysis, the delegation of responsibilities to other actors must also be detected. The current mechanisms for decomposition, refinement, and modularity in *i** are limited only to the actors´ boundaries. In our business service approach, the services are considered the focal point for the modeling process. Therefore, the proposed approach provides methods to determine how the business services are implemented through a specific organizational behavior. As a result of this new approach, the mechanisms for decomposition, refinement, and modularity are focused on business services.

The proposed architecture distinguishes three abstractions levels (services, process and protocols) and describes a methodological approach to align the business models produced at these abstraction levels. This enables the analyst to trace a specific business goal through the modeling process. The architecture, which is composed by three modeling diagrams, captures the relevant aspects in service modeling: the service composition, service variability, service objectives, services resources and service behaviors.

All modeling elements and modeling diagrams introduced in the service-oriented architecture for the *i** framework have been detailed in the third thesis section.

8.4 The service-oriented method

The modeling method associated with the service-oriented architecture is also presented, which enables the analyst to construct a business model in incrementally way. The proposed method will enable the analyst to describe an enterprise as a composition of business services that encapsulate a specific organizational behavior. We introduce the concept of refinement through the decomposition of business services into a set of business processes that represent the detailed view of the activities needed to perform the service.

One of the contributions of the last thesis section is the definition of an elicitation process that combines the advantages of goal-refinement structures with service-oriented diagrams that use the well-founded social and intentional characteristics of the *i** framework to appropriately represent the enterprise situation. One of the objectives of eliciting the organizational context using goal-refinement structures is to attempt to hide the intentional concepts of the *i** framework for the analysts. To do this, a specific goal category has been proposed in this research work.

In the current goal-based elicitation methods, the low-level goals are used to obtain the requirements of the information system. However, in this approach, the design decisions are taken too early, and the requirements are generated without knowledge of the performance of the organization. This approach focuses on generating software specifications rather than on supporting reasoning and analysis about the performance of the business process.

Using only a goal-based structure for elicitation, it is not possible to show: the order of execution of the operations, the work product flow, the workflow, the summarization of responsibilities of each business actor, etc. Therefore, it is not possible to improve the organization before generating the requirements of the information system.

We propose using the goal structure to automatically create a business model that allows us to carry out this business analysis (business process reengineering analysis, dependency analysis, and task analysis*)* before making decisions on the future situation of the enterprise. We have also proposed a method to automatically transform the elements of goal-refinement structure into the diagrams of the proposed service-oriented architecture.

We define a set of steps to generate organizational models that reflect how the goals of each actor as well as the general goals of the organization can be satisfied through the offered business services. The method generates a high level view of the services requested and offered by the enterprise. Then, each business service is analyzed in-depth in order to determine its associated business processes. Finally, each process is detailed using the revisited version of the *i** framework.

8.5 Summary of main contributions

Several contributions have been made in this thesis:

- **An empirical evaluation of the *i** Framework** in a real software development environment was carried out that provides information about the strengths and weaknesses of the *i** framework in practice.
- **A modeling language for the service-oriented method** was proposed. The modeling language is the result of revisiting the *i** modeling constructs and proposing new semantics for these constructs.
- **A service-oriented architecture** was developed that considers the modeling diagrams and the analysis needed to represent services at the organizational level. New modeling diagrams based on *i** have been proposed that overcome some of the problems detected in the empirical evaluation.
- **A service-oriented method** that provides a procedure to elicit the organizational setting using a service orientation is presented as a relevant contribution in this thesis. The proposed method uses new modeling elements based on the social dependencies of *i**.
- **A transformation method to translate goal structures into *i** models** is presented that isolates the intentional elements of the i* framework to novel *i** analyst.

8.6 Related Publications

The contributions of this thesis are supported by the set of publications carried out throughout this research work. These have been published in several international journals, book chapters, conferences and workshops.

8.6.1 International Journals

- Alicia Martínez, Oscar Pastor, Hugo Estrada. "A pattern language to join early and late requirements". *Journal of Computer Science and Technology (JCS&T), special issue on Software Requirements Engineering*. Vol. 5, No. 2. July 2005. ISSN 1666-6038.
- Alicia Martínez, Hugo Estrada, Oscar Pastor. "Generation of requirements model from business models: a pattern-based approach", *Informatics Technology Management Journal* Num. 7, Vol. 2. December 2004. ISSN 1657-8236 pp. 11-21, (published in Spanish).
- Oscar Pastor, Alicia Martinez Rebollar, Hugo Estrada. "Generation of Software Requirements Specifications from Business Models", *Informatics Technology Management Journal,* Num. 1, Vol. 1. 2002. ISSN 657-82364. pp. 53-65, (published in Spanish).

8.6.2 Book Chapters

- Oscar Pastor, Hugo Estrada, Alicia Martínez. i*, its applications, variations, and extensions. The strengths and weaknesses of the i* Framework: an experimental evaluation. Editors: (Accepted for its publications by MIT Press)
- J. Sanchez Diaz, O. Pastor Lopez, H. Estrada Esquivel, A. Martinez Rebollar**,** J. Belenguer Fáguas, "9. Semi Automatic Generation of User Interface Prototypes from Early Requirements Model", Perspectives on Software Requirements Editors: Julio Cesar Sampaio do Prado Leite, Jorge Horacio Doorn. Kluwer Academic Publishers, Boston Hardbound, ISBN 1-4020-7625-8. USA 2004.

8.6.3 International Conferences and Workshops

- Hugo Estrada, Alicia Martínez, Oscar Pastor, John Mylopoulos, "An experimental evaluation of the *i** Framework in a Model-based Software Generation Environment", in *18th Conference on Advanced Information Systems Engineering (CAISE 06)*. Luxembourg, Grand-Duchy of Luxembourg. June 2006. Lecture Notes in Computer Science, Vol. 4001, ISSN: 0302-9743. 2006. Pp. 513-527.
- Alicia Martínez, Oscar Pastor, Hugo Estrada. "A pattern language to join early and late requirements", in *VII Workshop on Requirements Engineering (WER 04)*, Tandil Argentina 2004. pp 51-64.
- Alicia Martinez, Oscar Pastor, Hugo Estrada. "Isolating and specifying the relevant information of an organizational model: a process oriented toward information system generation", in *International Conference on Computational Science and its Applications (ICSSA 2004)*. Perugia, Italy Springer LNCS 3046, pp. 783-790.
- Hugo Estrada, Oscar Pastor, Alicia Martinez and Jose Torres-Jimenez. "Using a Goal-Refinement Trees to obtain and refine organizational requirements", in *International Conference on Computational Science and its Applications (ICSSA 2004)*. Perugia, Italy, Springer LNCS 3046, pp. 506-513.
- Hugo Estrada, Alicia Martinez, Oscar Pastor. "Goal-based business modeling oriented towards late requirements generation", in *22nd International Conference on Conceptual Modeling (ER 2003)* October 2003, Chicago, Illinois, USA. ISBN 3-540-20-299-4, Springer LNCS 2813, pp. 277-290, 2003.
- Alicia Martinez, Jaelson Castro, Oscar Pastor, Hugo Estrada. "Closing the gap between Organizational

Modeling and Information System Modeling", in *VI Workshop on Requirements Engineering (WER 2003)*. Piracicaba SP, Brazil, 2003. pp 93-108.

- Hugo Estrada, Jaelson Castro, Oscar Pastor, Alicia Martínez. "Goal-based organizational modeling oriented towards late requirements generation", in *17th Brazilian Symposium on Software Engineering - SBES'2003*.
- Hugo Estrada, Alicia Martinez, Oscar Pastor, Juan Sanchez. "Generation of Software Requirements Specifications from Business Models: a goal-based approach", in *V Workshop on Requirements Engineering (WER 2002).* Valencia, Spain November 11-12, 2002, pp. 177-193, (published in Spanish).
- Alicia Martinez, Hugo Estrada, Oscar Pastor. "The Business Model as starting point of the software requirements: a methodological approach", in *9° International Congress on Computer Science Research (CIICC´02)*. Puebla, Mexico. October 2002, pp. 197-208, (published in Spanish).
- Alicia Martinez, Hugo Estrada, Juan Sanchez, Oscar Pastor. "From Early Requirements to User Interface Prototyping: A methodological approach", in *17th IEEE International Conference Automated Software Engineering (ASE2002).* Edinburgh, UK. September 2002, pp. 257-260.
- Hugo Estrada, Alicia Martinez, Oscar Pastor, Javier Ortiz, Erika Nieto. "Automatic generation of an Executable Conceptual Schema from a organizational model", in *V Iberoamerican Workshop Requirements Engineering and Software Environments (Ideas2002),* La Habana, Cuba, April 2002, pp. 281-292, (published in Spanish).
- Hugo Estrada E., Alicia Martinez R., Oscar Pastor L., Javier Ortiz H., Octavio A. Rios T. "Automatic generation of a OO Conceptual Schema from a Work flow product model", in *IV Workshop on Requirements Engineering*

- *(WER2001)*. National Technological University, Buenos Aires Argentina, November 2001, pp. 223-245, (published in Spanish).

8.7 Future research directions

With the modifications proposed in this thesis, our intention is to overcome the current limitations that practitioners face when using *i** in its current state. In fact, these modifications are intended to both, solve the problems that were detected, and to make the practical application of the method easier. Our future work will be dedicated to evaluating whether these conclusions can be generalized in practice.

9. References

Aart, C., Wielinga, B., Schreiber G. (2004). Organizational building blocks for design of distributed intelligent system, *International of Journal Human-Computer Studies*, 61(5):567-599

Albert, M., Pelechado, V., Fons, J., Ruiz, M., Pastor, O. (2003). Implementing UML Association, Aggregation, and Composition: A Particular Interpretation Based on a Multidimensional Framework, *Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAISE 03)*, Klagenfurt, Austria: Springer Verlag: 143-158.

Albert, M. (2006). Tratamiento de Relaciones de Asociación en Entornos de Producción Automática de Código, *PhD Thesis, Valencia University of Technology*, Valencia, Spain.

Ambler, S. (2005). The elements UML 2.0 Style, New York, NY, USA Cambridge University Press.

Amsden J. (2005), Business services modeling. From: http://www-128.ibm.com/developerworks/rational/library/05/1227_amsden/

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Trickovic, I. and Weerawarana, S. (2005). Business Process Execution Language for Web Services Version 1.1. From http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

Anton, Annie (1996). Goal Based Requirements Analysis, Proceedings of the Second International Conference on Requirements Engineering (ICRE 1996), Colorado, USA: 136-44.

Asnar Y., Bonato R., Bryl V., Compagna L., Dolinar K., Giorgini P., Holtmanns S., Klobucar T., Lanzi P., Latanicki J., Massacci F., MeduriV., Porekar J., Riccucci C., Saidane A., Seguran M., YautsiukhinA. and Zannone N. (2006). Security and privacy requirements at organizational level. From: http://www.serenity-forum.org/IMG/pdf/A1.D2.1__Security_and_privacy_requirements_at_organizational_level_v1.9_final.pdf

Baida Ziv (2006), Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling, *PhD thesis*, Vrije Universiteit Amsterdam, The Netherlands.

Bastos, L. & Castro J. (2003). Enhancing Requirements to derive Multi-Agent Architectures. *Proceedings of the VII Workshop on Requirements Engineering (WER 2004)*, Tandil, Argentina: University of Buenos Aires Press: 127-139.

Bergenti, F., Gleizes., & Zambonelli, F. (2004). Methodologies and Software Engineering for Agent Systems, New York: Kluwer Academic Publishing.

Bolchini, D. and Paolini P. (2002). Capturing Web Application Requirements through Goal-Oriented Analysis, *Proceedings of the Workshop on Requirements Engineering (WER 02)*, Valencia, Spain: 16-28.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. (2004). TROPOS: an agent-oriented software development methodology. *Journal of Autonomous agents and Multiagent Systems*, 8: 203-236.

Bubenko, J. A., Jr and M. Kirikova (1995), Worlds in Requirements Acquisition and Modelling, Information Modelling and Knowledge Bases VI. H.Kangassalo et al. (Eds.), IOS Press: 159– 174.

Carvallo, J., Franch, X., Quer, C. & Rodriguez, N. (2004). A Framework for Selecting Workflow Tools in the Context of Composite Information Systems, *Proceedings of the 15th International Conference on Database and Expert Systems Applications (DEXA 2004)*, Zaragoza, Spain: Springer Verlag: 109-119.

Castro, J., Alencar, F., Filho, G. and Mylopoulos J. (2001). Integrating organizational requirements and object oriented modeling, *Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE 2001)*, Toronto, Canada: IEEE Press: 146-153.

Cherbakov, L., G. Galambos, R. Harishankar, S. Kalyana, and G. Rackham (2005). Impact of service orientation at the business level. *IBM Systems Journal*, Volume 44, Issue 4: pp 653 – 668.

Chung, L., Nixon, B., Yu, E. & Mylopoulos, J. (2000). Non-Functional Requirements in Software Engineering. Boston, Hardbound: Kluwer Academic Publishers.

Colombo, E., Mylopoulos, J., and Spoletini, P. (2005). Modeling and Analyzing Context-aware Composition of Services, *Proceedings of Third International Conference on Service-Oriented Computing (ICSOC 2005)*, Amsterdam, The Netherlands: Springer Verlag: 198-213.

Cossentino, M., Sabatucci L. (2003). Modeling Notation Source – PASSI, Version 28. From: http://auml.org/auml/documents/PASSI.doc

Czarnecki K., Eisenecker U. (2000). Generative Programming. Addison Wesley Eds.

Dam, K. (2003). Evaluating and Comparing Agent-Oriented Software Engineering Methodologies. *Published master thesis*, RMIT University, Austria.

Dam, K., Winikoff, M. (2003). Comparing Agent-Oriented Methodologies, *Proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information System (AOIS 2003)*, Melbourne, Australia: Springer Verlag: 78-93.

Dardenne, A. Van Lamsweerde and S. Fickas (1993). Goal Directed Requirements Acquisition. *Science of Computer Programming*, vol. 20, North Holland: 3-50

Decker, S.; Erdman, M.; Studer, R. (1996). A Unifying View on Business Process Modeling And Knowledge Engineering. *Proceedings of the 10th Kew (Kaw96)*, Banff, Canada: 1-16.

Estrada, H., Martinez, A., Pastor, O. (2003). Goal-based business modeling oriented towards late requirements generation, *Proceedings of the 22nd International Conference on Conceptual Modeling (ER 2003)*, Chicago, USA: Springer Verlag: 277-290.

Elsmasri, R. and Navathe, S. (2004), Fundamentals on Database Systems, Addison-Wesley Publisher.

Erl Thomas (2006), Service-oriented architecture: concepts, technology and design. Prentice Hall, first edition.

Fuxman A., Pistore M., Mylopoulos J., and P. Traverso (2001). Model checking early requirements specifications in Tropos, *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering (RE'01)*, Toronto, Canada, IEEE Press: 174–181.

Giorgini, P., Mylopoulos, J., Nicchiarelli E., and Sebastiani R. (2002). Reasoning with goal models. *Technical report, Department of Information and Communication Technologies*, University of Trento, Italy.

Giorgini, P., Massacci, F., Mylopoulos, J. & Zannone, N. (2005). Modelling Social and Individual Trust in Requirements Engineering Methodologies, *Proceedings of the 3rd International Conference on Trust Management (iTrust 2005)*, Paris, France: Springer Verlag: 161-176.

Giorgini P., Massacci F., Mylopoulos J., Zannone N. (2006) Requirements engineering for trust management: model, methodology and reasoning. *The International Journal of Information Security,* 5(4): 257-274.

Gordijn, J. and Akkermans, H. (2001). E3-value: Design and evaluation of e-business models. *IEEE Intelligent Systems*, 16(4):11–17.

Gordijn J. and Akkermans, H. (2003). Value based requirements engineering: Exploring innovative e-commerce idea. *Requirements Engineering Journal*, 8(2):114–134.

Grau G., Horkoff J., Yu E. (2006) IStarQuickGuide. From: http://istar.rwth-aachen.de/tiki-index.php?page=iStarQuickGuide.

Jones, Steve (2005), A methodology for service oriented architectures. From: http://www.oasis-open.org/committees/download.php/15071/A%20methodology%20for%20Service%20Architectures%201%202%204%20-%20OASIS%20Contribution.pdf

Kavakli V.and Loucopoulos P. (1999) Modelling of Organizational Change using the EKD Framework, *Communications of Association for Information Systmes (CAIS)*, Volume 2 Article 6.

Kazhamiakin, R., Pistore, M., and Roveri, M. (2004). A Framework for Integrating Business Processes and Business Requirements, *Proceeding of the Enterprise Distributed Object Computing Conference*, California, USA: IEEE Computer Society Press: 9-20.

Keith, Mantell (2005). From UML to BPEL. Model Driven Architecture in a Web services world. From http://www-128.ibm.com/developerworks/webservices/library/ws-uml2bpel/

Kolp, M., Giorgini, P. & Mylopoulos, J. (2003). Organizational Patterns for Early Requirements Analysis, *Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAiSE'03)*, Velden, Austria: Springer Verlag: 617-632

Lau, D., Mylopoulos, J. (2004). Designing Web Services with Tropos, *Proceedings of the IEEE International Conference on Web Services (ICSWS´04)*, San Diego, USA: IEEE Computer Society Press: 306-314.

Liu, L., and Yu, E. (2003). Designing Information Systems in Social Context: A Goal and Scenario Modelling Approach. *Information Systems Journal*, 29(2): 87-203.

Loucopoulos, P. and Kavakli, E. (1995). Enterprise Modelling and the Teleological Approach to Requirements Engineering, *International Journal of Intelligent and Cooperative Information Systems*, Vol. 4, No. 1: 45-79

Loucopoullos P. and Kavakli E. (1997) Enterprise Knowledge Management and Conceptual Modeling, Conceptual Modeling, Current Issues and Future Directions, Selected Papers from the *International Symposium on Conceptual Modeling (ER 97)*, Los Angeles, USA: Springer Verlag: 123-143.

Maiden, N., Jones, S., Manning, S., Greenwood, J. & Renou, L. (2004). Model-Driven Requirements Engineering: Synchronising Models in an Air Traffic Management Case Study, *16th International Conference on Advanced Information Systems Engineering (CAiSE'04)*. Riga, Latvia: Springer Verlag: 368-383.

Martinez, A., Castro, *J.,* Pastor, O. and Estrada. H. (2003). Closing the gap between Organizational Modeling and Information System Modeling, Proceedings of the VI Workshop on Requirements Engineering (WER 2003), Piracicaba, Brazil: University Piracicaba Press: 93-108.

McDermid, J..A, (1994). Software Engineer´s Reference Book. Edit. Butterworth-Heinenmann,

Moffett J., Lupu E. (1999). The use of Role Hierarchies in Access Control, *Proceedings of the ACM Workshop on Role-Based Access Control*, Fairfax, USA: 153-160.

Mylopoulos J., (1998). Information Modeling in the Time of the Revolution. *Information System*, 23(3-4): 127-155.

OASIS (2007). Web Services Atomic Transactions. From: http://docs.oasis-open.org

Padgham, L., Shehory, O., Sterling, L. & Sturm, *A.* (2005). Methodologies for Agent-Oriented Software Engineering, *Proceedings of the Seventh European Agent System Summer School (EASSS 2005)*, Utrecht, the Netherlands: Springer Verlag.

Pastor, O., Gómez, J., Infrán, E. &. Pelechado, V. (2001). The OO-Method approach for information systems modeling: from object-oriented conceptual modeling to automated programming. *Information Systems*, 26(7): 507-534.

Pardedel, E., Wenny R., and David T. (2004). Preserving Constraints for Aggregation Relationship Type Update in XML Documents, Proceedings of the 2004 International Conference on Intelligent Agents, Web Technology and Internet Commerce (*IAWTIC 2004*), Gold Coast, Australia: 494-501.

Rust, R. T. & Kannan, P. K. (2002). e-Service: New Direction in Theory and Practice. Armonk, N.Y.M.E. Sharpe Inc.

Ruyter, K., M. Wetzels, and M. Kleijnen (2001), Customer Adoption of E-Service: An Experimental Study, *International Journal of Service Industry Management*, Vol. 12, No. 2: 184-207.

Sannicolo, F., Perini, A., and Giunchiglia, F. (2001). The Tropos modeling language, a User Guide. *Technical report*, ITC-irst.

Saksena, M., France, R., Larrondo-Petrie, M. (1999). A characterization of Aggregation. *International Journal on Computer Systems Science & Eng., 14(6):* 363-371.

Sanjiva, W., and Curbera, F. (2002). Business processes: Understanding BPEL4WS, Part 1. Concepts in business processes. From https://www.cs.tcd.ie/research_groups/kdeg/docs/presentations/IBM%20BPELWS_2423067fb7f9ef3a3e13aad29e20b85c.pdf#search=%22bpel4ws%2C%20description%22

Shehory, O. & Sturm, A. (2001). Evaluation of modeling techniques for agent-based systems, *Proceedings of the Fifth International Conference on Autonomous Agents*, Montreal, Canada: ACM Press: 624-631.

Sinan, S. (2003). Understanding the Model Driven Architecture (MDA). From http://home.comcast.net/~salhir/UnderstandingTheMDA.PDF

Stencil Group (2001). Defining Web Services. The Stencil Group. From: http://www.stencilgroup.com/ideas_scope_ 200106wsdefined.pdf

Sturm, A. & Shehory, O. (2003). A Framework for Evaluating Agent-Oriented Methodologies, *Proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information System (AOIS 2003*), Melbourne, Australia: Springer Verlag: 94-109.

Sturm, A., Dori, D. & Shehory, O. (2005). A Comparative Evaluation of Agent-Oriented Methodologies, to appear in Methodologies and Software Engineering for Agent Systems, Federico Bergenti, Marie-Pierre Gleizes, Franco Zambonelli (eds): Kluwer Academic Publishers.

Sudeikat, J., and Braubach, L., and Pokahr, A and Lamersdorf, W. (2004). Evaluation of Agent-Oriented Software Methodologies Examination of the Gap between Modeling and Platform. *Proceedings of the Workshop on Agent-Oriented Software Engineering (AOSE-2004)*. New York, USA: Springer Verlag: 126-141.

Terry, H. (1998). UML data models from an ORM perspective, Journal of Conceptual Modeling (*http://www.inconcept.com/jcm*).

Van Welie, M., Van der Veer, G.C., Eliëns, A., An Ontology for Task World Models. *Proceedings of DSV-IS'98*. Abingdon, UK: Springer-Verlag: 57-70.

W3C Working Group (2004). Web Services Architecture. From http://www.w3.org/TR/ws-arch/

Yu, Eric (1995). Modelling Strategic Relationships for Process Reengineering, *Published Doctoral dissertation*, University of Toronto, Canada.

Yu, E. & Liu, L. (2001). Modelling Trust for System Design Using the i* Strategic Actors Framework, *Proceedings of the Workshop on Deception, Fraud, and Trust in Agent Societies held during the Autonomous Agents Conference: Trust in Cyber-societies, Integrating the Human and Artificial Perspectives*. London, UK: Springer Verlag: 175-194.

Zannone N. (2007). A Requirements Engineering Methodology for Trust, Security, and Privacy, *PhD Thesis*, Department of Information and Communication Technology, University of Trento, Italia-