



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

**Tesis de Máster en Ingeniería de Software
Métodos Formales y Sistemas de Información**

**DEPARTAMENTO DE SISTEMAS
INFORMÁTICOS Y COMPUTACIÓN**

**A model-driven framework to integrate
Communication Analysis and OO-Method**



Luz Marcela Ruiz Carmona

Advisors: Óscar Pastor López
Sergio España Cubillo



Centro de Investigación en Métodos
de Producción de Software



UNIVERSIDAD
POLITECNICA
DE VALENCIA



Master thesis

A model-driven framework to integrate Communication Analysis and OO-Method

Luz Marcela Ruiz Carmona

Advisors: Óscar Pastor López

Sergio España Cubillo

September 2011, Valencia Spain

*Hubo una época en que todo era más fácil.
Tu mamá decía que ropa te ponías, y vos la
usabas sin ningún tipo de problema.
Te peinaba, y te encantaba como lo hacía.
Te cuidaba, como nadie podía hacerlo.
Y cuando tenías hambre solo llorabas.
Ibas a ser abogado, tal vez ingeniero o presi-
dente, como muchos niños más;
pero un día, sin que te dieras cuenta, creciste,
y aprendiste a decir que no.
No te conformaste.
Y sentiste que querías cometer tus propios
errores, saltar los muros que te defendían en
tu niñez.
Entonces tomaste el camino más difícil.
Te dedicaste a lo que realmente querías.
Te animaste a ser distinto.
Y por primera vez sentiste que podías.
Era tu lucha, tu convicción.
Y sin dudar, arriesgaste todo lo que tenías.
Porque en el fondo, sabías que había algo
mucho peor que fracasar.
No haberlo intentado!!*

Anónimo

*Dedicated to my grandmother Lucero.
Thank you for your support, your happiness,
your smiles and your songs from the heaven.*

Agradecimientos

Marco Tulio Cicerón, en su tratado sobre la amistad relata: "si alguien hubiese subido al cielo y hubiese contemplado la naturaleza del mundo y la hermosura de los astros; esta habría sido para él agradabilísima, si hubiera tenido a alguien al que contarle". De verdad, me siento alegre y afortunada de poder decir, que en estos dos años que llevo en el mundo de la investigación he contemplado cosas hermosas, y además, que siempre he tenido con quien compartirlas.

Deseo expresar mi más sincero agradecimiento y cariño a mi amigo y director Óscar Pastor, gracias por darme la oportunidad de hacer parte de tu equipo, porque desde el primer día me hiciste sentir como en casa, gracias por tus consejos y compañía, por darme la oportunidad de estar en el mundo de la investigación y disfrutar de el de la mejor forma. Gracias por haberme escogido, y por tu infinita fe y confianza en mí.

De forma muy especial quiero dar las gracias mi compañero, codirector y jefe Sergio España, gracias Sergio por tanto que haces por mí, por guiarme, animarme, por tu infinita paciencia, porque me has explicado cosas mil veces y siempre tienes fe de que podré hacerlo bien©. Gracias por "apretarme las tuercas" cuando más lo necesitaba y por hacerme tomar pausas cuando exageraba en las cosas. Gracias por todo lo que me has enseñado, gracias por permitirme ser parte de tu equipo y aprender de tu experiencia, gracias porque estos dos años de trabajo contigo me siento muy contenta y feliz con lo que hago. Gracias por tu positivismo y tu alegría para hacer las cosas, tu ánimo proactivo y emprendedor es admirable, me contagia de un ambiente especial que incrementa mi sentido de pertenencia y compromiso por nuestros proyectos y trabajos. Gracias porque has hecho parte de mi crecimiento profesional y personal, y aunque aún me falta mucho camino por recorrer, tus consejos y sonrisas han hecho parte de mis

cimientos como investigadora, ojalá algún día yo pueda hacer por ti aunque sea un poco de lo que tú has hecho por mí.

Quiero dar las gracias al profesor Arturo González por permitirme desarrollar mis temas de investigación en torno a su propuesta de Análisis de Comunicaciones.

Quiero dar las gracias a mi profesor Carlos Mario Zapata por sus consejos de padre, por su compañía y cariño, gracias por esos años de trabajo juntos, gracias por que como cualquier padre siempre has deseado lo mejor para mí y me dejaste partir de tu grupo seguro de que iba a crecer profesional y personalmente con el profesor Oscar Pastor.

Quiero dar las gracias a Ana Ciudad y a Pele por sus esfuerzos constantes para que yo pueda estar acá. Muchas gracias Ana, sin tu ayuda este trabajo no hubiera sido posible.

En Medellín tengo un par de profesores muy especiales que han estado muy pendientes de mí durante estos dos años, ellos son Demetrio Arturo Ovalle y María Teresa Berdugo, gracias por sus ánimos e impulsarme tanto al momento de tomar la decisión de viajar, gracias por ser un soporte y compañía desde la lejanía.

Quiero agradecer al profesor Jean Vanderdonckt por su amabilidad y disposición cuando desarrollamos las pruebas de usabilidad de la herramienta de modelado propuesta en esta tesis. Jean, muchas gracias por tus consejos y habernos enseñado tus conocimientos ante el reto que teníamos por resolver, gracias por concedernos tu tiempo y tus espacios. Trabajar junto a ti fue todo un honor.

Quiero agradecer a Mario Cervera por las tardes de dedicación y esfuerzos enseñándome a manejar la herramienta MOSKitt y los consejos dados sobre el desarrollo en Eclipse. Mario, gracias por tu ayuda desinteresada y amabilidad.

En el laboratorio 104 he pasado unas largas jornadas de trabajo, las cuales han sido especiales y alegres gracias a que tengo la suerte de

contar con unos compañeros de laboratorio excepcionales, como Ignacio, gracias Doctor Panach por tus consejos y chistes malos, gracias por enseñarme a usar olivanova y estar siempre tan pendiente de mi horario de trabajo, gracias a ti llegaba siempre temprano, gracias por tus consejos. Muchas gracias Paco por haberme hecho reír tanto en el laboratorio, gracias por tu ayuda cuando me enrollaba en lo que no hacía falta. Mil gracias a Nathalie por sus consejos y estar siempre ahí, gracias por los cafecitos para despejar un poco la mente y defenderme ante los chicos ☺. Nelly, muchas gracias por el tiempo de charlas y consejos. Jose Luis, muchas gracias por siempre querer corregir mis trabajos, por las bromas y planes en el laboratorio. Gracias Raul por tu ánimo alegre y estar siempre muy pendiente de cómo están todos en el laboratorio. Gracias a Urko y al resto de “pastorcillos” e integrantes del PROS.

Quiero expresar un especial agradecimiento a Bea y Giovanni, gracias por las tardes de charlas y risas, por sus consejos, por estar en las buenas y en las malas, gracias por emocionarse con mis historias y permitirme disfrutar de su compañía y alegría.

Muchas gracias a Daniel y Nathalie, gracias por hacer parte de mis empelicules, gracias porque han sido parte del equipo diseñador de la portada de la tesis, gracias por estos años de amistad, gracias por sus consejos y tardes de amigos, gracias por su compañía, en conjunto con Laura, Carlino y el equipo paraguayo, han hecho de este par de años un tiempo lleno de experiencias muy especiales.

Quiero agradecer a mis compañeros del máster Mariajo, Matthijs, Anicia y Lore, gracias por las tardes que pasamos realizando trabajos y estudiando para exámenes. Su compañía fue muy importante para poder finalizar mis estudios.

Lore, porque en muchos momentos de angustia estuviste ahí para hacerme ver que no valía la pena estar triste o preocupado, gracias por los momentos de “yapiró” todo lo que pasa, ¡era verdad!, muchas veces nos preocupamos de cosas que no tienen sentido y dejamos de vivir lo que realmente es importante. ¡Gracias amiga paraguayita!

Quiero darle las gracias a mis amigos de Medellín Aleja, Sandra, Indira, Yosel, Diana, Gloria, Margarita, Jhon Edison, Jose Fabio, Andrés y Lili. Gracias a todos mis amigos de la Universidad y de Sura. Gracias amigos por animarme a tomar la decisión de viajar y estar siempre tan pendiente de mi, gracias porque aunque estamos muy lejos y han pasado dos años, yo se que nuestra amistad sigue intacta, los extraño y los quiero mucho.

Gracias Erica por vivir esta experiencia conmigo, gracias por estar ahí a cualquier hora del día o de la noche, gracias por vivir cada aventura y momento, gracias por hacerme parte de tu vida aunque kilómetros de distancia nos separen, gracias por Isabellita y permitirme vivir tu embarazo como si hubiéramos estado juntas, gracias por orar por mí, por tus ánimos y alegría, gracias porque siempre has sido un rayo de Dios en mi vida.

Esther, muchas gracias por tu cariño y comprensión, gracias por ser mi compañera de piso y haber vivido una parte de mi aventura. Gracias por tus oraciones, tu ánimo y compañía. Gracias por tantas noches que me recibiste con la comida caliente porque llegaba muy tarde, gracias por ayudarme con tantas cosas que por tiempo no alcanzaba a hacer y por tu paciencia conmigo.

Estando en Medellín no alcancé a imaginar que al otro lado del mundo tenía una mejor amiga: Paqui, muchas gracias por tus consejos en el desarrollo de esta tesis. Gracias por mostrarme lo más lindo de Valencia, y por maravillarte de mi ciudad. Gracias por explicarme tantas cosas, gracias por ser como mi hermana, por tantas tardes de sonrisas, planes y empelicules. Gracias por reír y llorar conmigo, y estar conmigo en las buenas y en las malas. Gracias porque en los momentos más tristes estuviste, y te inventabas cosas para hacerme sonreír para que se me olvidaran las cosas aunque fuera por un instante. Gracias por que hoy me siento muy afortunada de seas mi amiga ☺

Doy gracias a mi familia, a mis tíos y primos, especialmente a mi tía favorita Vicky por sus consejos y sus regaños amorosos, gracias a Tere por su amor y su energía positiva, gracias Tere por ser tan linda y tan buena conmigo, gracias a Tavo por estar siempre ahí y por ser

una gran compañía para mi mamá, y claro mil gracias a mi ayudante favorito Juan José, gracias por ser un niño tan especial y haber llegado a nuestras vidas a llenarla de sonrisas y felicidad, aunque no soy capaz de explicarte porque estoy tan lejos y no voy a jugar contigo, en un futuro leerás esto y entenderás porqué.

Vero, gracias por ser mi apoyo por haber estado toda la vida junto a mí, gracias hermana porque sé que siempre estarás ahí para mí y porque juntas vamos a poder lograr muchas cosas. Te quiero mucho hermanita, y este trabajo que cierra un trozo de la meta a la que quiero llegar está dedicado a ti, gracias por vivir conmigo cada instante, porque cuidas de mis papás y los conscientes. Gracias hermanita por existir en mi vida, porque sos el regalo más lindo, gracias por tu paciencia, porque sé que te hago falta, pero sabes? Me siento muy orgullosa de ti, porque ahora ya eres grande, tomas tus propias decisiones y eres exitosa. Te quiero mucho!

Quiero finalizar agradeciendo a las personas más importantes, a quienes va dedicado todo este trabajo, gracias Papi y Mami, sin su apoyo, su cariño y sus esfuerzos yo no hubiera sido capaz de salir adelante ante tantos obstáculos. Gracias por dejarme volar, por que su compañía me ha seguido a todas partes, gracias porque desde niña siempre han creído que podía llegar muy lejos, porque a Vero y a mí nos han dado todo lo que han tenido por amor. Papi y mami, gracias por tantos sacrificios, porque de lejos he descubierto que ustedes son mis mejores amigos y mi apoyo incondicional. Gracias por sus oraciones, gracias por el enorme esfuerzo que han hecho en aprender a usar el computador y el internet para estar conectados conmigo, de verdad, que tus padres aprendan a manejar tecnologías que no habían tocado nunca solo por estar contigo no tiene precio. Gracias Papi y Mami por dejar a un lado sus sueños por vivir los de mi hermanita y los míos, gracias Papi y Mami por mostrarme que es más fuerte el deseo que querer lograr algo que los medios que tienes a tu alrededor para conseguirlo, gracias por mostrarme que si tengo sueños, puedo encontrar un camino para alcanzarlos, gracias por permitirme crecer en un ambiente lleno de Dios y de amor, esa fuerza especial que llena

mi espíritu y me impulsa para ser cada día mejor y poder llegar a ser una gran persona y una gran profesional, los amo infinitamente.

Marcela Ruiz

Overview

Organisational systems require elicitation methods and requirements specification to identify needs and inherent characteristics of their business process. Analyse System Information from a communicative perspective is necessary, because this perspective involves processes and system actors; all of this having into account how is the information communicated and how the information interacts between the system and the environment.

Communication Analysis is a requirements engineering method that proposes to specify business processes from a communicational perspective. Model-driven development (MDD) is a paradigm that provides to the requirement models of some advantages: as the potential to derive conceptual models for generating software in an automatic way. The automatic generation of software products allows an easy adoption of requirements engineering methods in an industrial environment. OO-Method is a model-driven software development method, whose conceptual models are supported by *OLIVANOVA*, a models compiler that allows the code automatic generation. This master thesis presents an integration framework to link Communication Analysis techniques and OO-Method. This integration framework follows a model-driven development approach. We propose a development framework that involves tools (e.g. Eclipse) to support modelling tasks, proposes modelling techniques and model-driven development practices that are advised by academy. We present application prototypes to analyse the advantages and challenges, the MDD community should confront the support of the MDD paradigm, and what kind of strategies the MDD community should propose to involve the MDD paradigm into industrial environments.

Resumen

Los sistemas organizacionales requieren métodos de captura y especificación de requisitos para identificar las necesidades y características intrínsecas de sus procesos de negocio. Analizar el Sistema de Información desde una perspectiva comunicativa es necesario, debido a que involucra los procesos y actores del sistema teniendo en cuenta cómo se comunica la información y como ésta interactúa entre el sistema y el entorno. Análisis de Comunicaciones es un método de ingeniería de requisitos que propone describir los procesos de negocio desde una perspectiva comunicacional. El desarrollo dirigido por modelos (MDD) es un paradigma que puede dotar a los modelos de requisitos de un valor agregado: el potencial para derivar de ellos los modelos conceptuales que servirán para la generación automática de software, permitiendo así una fácil adopción de metodologías de requisitos en un entorno industrial. OO-Method es un método de desarrollo de software dirigido por modelos, cuyos modelos conceptuales se encuentran soportados por *OLIVANOVA*, un compilador de modelos que permite la generación automática de código ejecutable. Esta tesis de Máster presenta una propuesta de integración de las técnicas de Análisis de Comunicaciones y OO-Method desde un enfoque dirigido por modelos. Se propone un marco de desarrollo que aprovecha herramientas para el soporte de modelado como Eclipse y se emplean técnicas de metamodelado y desarrollo dirigido por modelos que en la actualidad es promulgado por la academia. Se presentan prototipos de aplicaciones, donde se pueden analizar las ventajas y retos futuros que la comunidad de MDD debe afrontar para dar soporte al auge del paradigma y ésta cómo puede ser involucrada en entornos industriales.

Contents

1	<i>Introduction</i>	7
1.1	Motivation	8
1.2	Context.....	9
1.3	Objectives	10
1.4	The proposed solution	11
1.5	Research methodology	27
1.6	Thesis structure	31
2	<i>State of the art</i>	33
2.1	Frameworks for method integration	34
2.2	Integration process of RE specification with OO conceptual models	36
2.3	Technological support for business process modelling.	37
2.4	Communication Analysis method	38
2.5	OO-Method.....	45
2.6	Analysis and discussion	46
3	<i>MDD approach: a metamodel to support the techniques of Communicative Event Diagrams and Message Structures</i>	49
3.1	Initial state	50
3.2	The metamodeling strategy	51
3.3	PIM metamodel specification	52
3.4	PSM metamodel specification	54
3.5	Metamodel validation.....	138
3.6	Analysis and discussion	139
4	<i>A modelling tool for Communication Analysis requires of models</i>	141
4.1	Technological support.....	142
4.2	PSM metamodel implementation.....	144
4.3	Design of the graphical editor for communicative event diagrams and message structures.....	150
4.4	How to use the modelling tool	152

4.5	Modelling tool validation	169
4.6	Analysis and discussion	191
5	<i>Supporting the model transformation: From Communication Analysis requirements models to OO-Method object model</i>	193
5.2	Technological support.....	198
5.3	Rules implementation.....	199
5.4	Transformation example.....	200
5.5	Validation of the transformation proposal	206
5.6	Traceability support	207
5.7	Analysis and discussion	211
6	<i>Conclusions</i>	213
6.1	Contributions	214
6.2	Publications.....	215
6.3	Future work	217
7	<i>References.....</i>	219
7.1	References of this thesis	219
8	<i>Append 1.....</i>	227
8.1	Development of the modelling tool: step by step	227
9	<i>Append 2.....</i>	249
9.1	ATL rules to transform Communication Analysis requirements models to OO-Method class diagram	249

List of figures

Figure 1. General framework of integration	13
Figure 2. Stage 1 of the general framework of integration.....	16
Figure 3. Stage 2 of the general framework of integration.....	21
Figure 4. Integration framework to involve Communication Analysis method into a MDD environment	26
Figure 5. Research method followed in the thesis.....	28
Figure 6. Communicative event diagram of SuperStationery Co. <i>Sales management</i> business process (Sale)	41
Figure 7. Phase 1 explanation	53
Figure 8. PIM metamodel for communicative event diagrams.....	54
Figure 9. Phase 2 explanation	55
Figure 10. PSM metamodel for communicative event diagrams.....	57
Figure 11. Portion of the metamodel including the metaclass AGGREGATION	59
Figure 12. Example of the different AND cases	61
Figure 13. Portion of the metamodel including the metaclass AND.....	61
Figure 14. Graphical primitive of AND metaclass.....	62
Figure 15. Portion of the metamodel including the metaclass BUSINESS_OBJECT_CLASS	63
.....	63
Figure 16. Portion of the metamodel including the metaclass BUSINESS_OBJECT_FIELD	64
.....	64
Figure 17. Portion of the metamodel including the metaclass COMMUNICATION_CHANNEL.....	66
Figure 18. Example of use of COMMUNICATIVE_EVENT a in a CED	67
Figure 19. Portion of the metamodel including the metaclass COMMUNICATIVE_EVENT	68
.....	68
Figure 20. Graphical primitive of the COMMUNICATIVE_EVENT metaclass	69
Figure 21. Portion of the metamodel including the metaclass COMMUNICATIVE_INTERACTION	70
Table 14. Relationships of the COMMUNICATIVE_INTERACTION metaclass.....	71
Figure 22. Portion of the metamodel including the metaclass COMMUNICATIVE_ROLE.....	72
Figure 23. Portion of the metamodel including the metaclass COMPLEX_SUBSTRUCTURE	74
.....	74
Figure 25. Portion of the metamodel including the metaclass DATA_FIELD	76
Figure 26. Portion of the metamodel including the metaclass ELEMENT.....	78
Figure 27. Portion of the metamodel including the metaclass ENCAPSULATION	79
Figure 28. Portion of the metamodel including the metaclass END	81
Figure 29. Graphical primitive of the END metaclass.....	82
Figure 30. Example of EVENT_VARIANTS in a communicative event diagram	82
Figure 31. Portion of the metamodel including the metaclass EVENT_VARIANT	83
Figure 32. Portion of the metamodel including the metaclass FIELD	85
Table 27. Relationships of the FIELD metaclass	86

Figure 33. Portion of the metamodel including the metaclass GOAL	87
Figure 34. Portion of the metamodel including the metaclass INDICATOR	88
Figure 35. Portion of the metamodel including the metaclass INGOING	90
Figure 36. Graphical primitive of the PRIMARY metaclass.....	91
Figure 37. Portion of the metamodel including the metaclass ITERATION.....	92
Figure 38. Portion of the metamodel including the metaclass LOGICAL_NODE.....	94
Figure 39. Portion of the metamodel including the metaclass MESSAGE_STRUCTURE ..	95
Figure 40. Portion of the metamodel including the metaclass MODEL.....	97
Table 40. Relationships of the MODEL metaclass.....	97
Figure 41. Portion of the metamodel including the metaclass NODE.....	98
Figure 42. Portion of the metamodel including the metaclass OPERATIONALISATION ..	99
Figure 43. Example of OR in a CED.....	101
Figure 44. Portion of the metamodel including the metaclass OR	101
Figure 45. Graphical primitive for the OR metaclass	102
Figure 46. Portion of the metamodel including the metaclass ORGANISATION.....	102
Figure 47. Portion of the metamodel including the metaclass ORGANIZATIONAL_ACTOR	
.....	104
Table 48. Relationships of the ORGANIZATIONAL_ACTOR metaclass.....	105
Figure 48. Portion of the metamodel including the metaclass	
ORGANISATIONAL_LOCATION	106
Figure 49. Portion of the metamodel including the metaclass	
ORGANISATIONAL_MODULE	107
Figure 50. Portion of the metamodel including the metaclass ORGANISATIONAL_ROLE	
.....	109
Figure 51. Portion of the metamodel including the metaclass	
ORGANISATIONAL_ROLE_SET	111
Figure 52. Portion of the metamodel including the metaclass ORGANISATIONAL_UNIT	
.....	112
Figure 53. Example of OUTGOING in a communicative event diagram	114
Figure 54. Portion of the metamodel including the metaclass OUTGOING	114
Table 58. Relationships of the OUTGOING metaclass.....	115
Figure 55. Graphical primitive of the OUTGOING metaclass	115
Figure 56. Example of PRECEDENCE in a communicative event diagram	116
Figure 57. Portion of the metamodel including the metaclass PRECEDENCE	116
Figure 58. Graphical primitive of the PRECEDENCE metaclass	117
Figure 59. Portion of the metamodel including the metaclass PRIMARY	117
Figure 60. Graphical primitive for the PRIMARY metaclass	118
Figure 61. Portion of the metamodel including the metaclass PROCESS.....	119
Figure 62. Portion of the metamodel including the metaclass RECEIVER.....	121
Figure 63. Graphical primitive for the RECEIVER metaclass.....	122
Figure 64. Portion of the metamodel including the metaclass REFERENCE_FIELD.....	123
Figure 65. Portion of the metamodel including the metaclass SPECIALISATION	125
Figure 66. Portion of the metamodel including the metaclass START	127
Table 70. Relationships of the START metaclass.....	127
Figure 67. Graphical primitive of the START metaclass	127
Figure 68. Portion of the metamodel including the metaclass STRATEGY.....	128
Figure 69. Portion of the metamodel including the metaclass SUBSTRUCTURE.....	130

Figure 70. Portion of the metamodel including the metaclass SUPPORT	131
Table 76. Relationships of the SUPPORT metaclass	132
Figure 71. Portion of the metamodel including the metaclass SUPPORT_ROLE_SET	133
Figure 72. Portion of the metamodel including the metaclass TEXTUAL_REQUIREMENT	134
.....	
Figure 73. Portion of the metamodel including the metaclass DOMAIN	135
Figure 74. Portion of the metamodel including the metaclass OPERATION	136
Figure 75. Portion of the metamodel including the metaclass REQUIREMENT_TYPE	137
Figure 76. General view of metamodel validation	138
Figure 77. EMF defines Java, XML and UML	143
Figure 78. Subset of the Ecore model	143
Figure 79. Stage 1 phase 2 explanation	144
Figure 80. Creation of a UML class diagram Project	145
Figure 81. Creation of the PSM metamodel with UML 2.1 tools	146
Figure 82. Complete description of the PSM metamodel in a tree way	147
Figure 83. Ecore model creation	148
Figure 84. PSM metamodel in the ECORE specification	149
Figure 85. Workflow to create the modelling environment for CED	151
Figure 86. SuperStationery Co. organization chart	154
Figure 87. Communicative event diagram of SuperStationery Co. Sales manager business process (Sale)	155
Figure 88. Composition of the CED graphical editor	160
Figure 89. Specifying the communicative event SALE 1	161
Figure 90. Specifying a primary actor for SALE 1	162
Figure 91. Specifying a receiver actor for SALE 1	163
Figure 92. Specifying ingoing and outgoing interactions for SALE 1	164
Figure 93. Communicative Event diagram for Sale business process modelled in the graphical editor	165
Figure 94. Support to Message Structures with the Xtext environment	167
Figure 95. Support to Message Structures with EMF	168
Figure 96. Example of message structure supported by the EMF environment	169
Figure 97. Activity of evaluation of diagramming tool	170
Figure 98. Description of the profile 1: Ana	175
Figure 99. Description of the profile 2: Jhon	175
Figure 100. Demographic questionnaire	178
Figure 101. Percentage of male and female.	179
Figure 102. Experience with general purpose diagramming tools	179
Figure 103. Experience with CASE modelling tools	180
Figure 104. Experience with modelling methods	180
Figure 105. Experience with Communication Analysis requirements models	181
Figure 106. Task done in the modelling tool	182
Figure 107. Computer System Usability Questionnaire (CSUQ)	183
Figure 108. Mean of the answers of the CSUQ questionnaire	184
Figure 109. Mean of the answers of the SYSUSE factor	184
Figure 110. Mean of the answers of the INFOQUAL factor	185
Figure 111. Mean of the answers of the INTERQUAL factor	185
Figure 112. Additional free-response questions	186

Figure 113. Percentage of the task successfully performed	189
Figure 114. Focus group participants.....	191
Figure 115. Stage 2 explanation.....	194
Figure 116. Phase 3 explanation	195
Figure 117. Operational context of ATL and QVT	196
Figure 118. Phase 4, step of implementation of transformation rules	199
Figure 119. Workspace ATL.....	202
Figure 120. Run configuration for ATL transformation.....	203
Figure 121. SuperStationery Co. Class diagram tree view	204
Figure 122. SuperStationery Co. Class diagram in UML 2.0	205
Figure 123. Activity of evaluation of the transformation module	206
Figure 124. Traceability metamodel	208
Figure 125. Inheritance relationship between the ELEMENT metaclass and EMODELELEMENT metaclass.....	209
Figure 126. Inheritance relationship between the NAMEDELEMENT metaclass and EMODELELEMENT metaclass.....	210
Figure 127. ATL rule to create class from aggregation substructure	211
Figure 128. Example of traceability information for The SuperStationery Co.	211
Figure 129. Workflow to create the modelling environment for CED	228
Figure 130. Genmodel model for CED	229
Figure 131. How to run the textual editor for CED	230
Figure 132. Creation of a new empty EMF project	231
Figure 133. Selection of the CED creation wizard	232
Figure 134. Creation of the new CED model.....	233
Figure 135. Selection of the model object to create.....	234
Figure 136. Initial state of the SuperStationery model	235
Figure 137. Add elements to the Model	236
Figure 138. Example of creation of a Communicative Event	237
Figure 139. SuperStationery example in a text editor	238
Figure 140. gmfggraph for CED.....	240
Figure 141. gmftool for CED.....	241
Figure 142. gmfmap for CED	242
Figure 143. gmfgen for CED	243
Figure 144. Launch of the Eclipse application with the plug-in to model CED.....	244
Figure 145. Interface of CED Modelling tool.....	245

List of tables

Table 1. EBNF grammar of Message Structures.....	44
Table 2. Example of a message structure	44
Table 3. Attributes of the AGGREGATION metaclass.....	59
Table 4 Relationships of the AGGREGATION metaclass.....	60
Table 5. Attributes of the BUSINESS_OBJECT_CLASS metaclass	63
Table 6. Associations of the BUSINESS_OBJECT_CLASS metaclass	63
Table 7. Attributes of the BUSINESS_OBJECT_FIELD metaclass.....	64
Table 8. Relationships of the BUSINESS_OBJECT_FIELD metaclass.....	65
Table 9. Attributes of the COMMUNICATION_CHANNEL metaclass.....	66
Table 10. Relationships of the COMMUNICATION_CHANNEL metaclass	66
Table 11. Attributes of the COMMUNICATIVE_EVENT metaclass.....	68
Table 12. Relationships of the COMMUNICATIVE_EVENT metaclass	69
Table 13. Attributes of the COMMUNICATIVE_INTERACTION metaclass	70
Table 15. Attributes of the COMMUNICATIVE_ROLE metaclass.....	72
Table 16. Relationships of the COMMUNICATIVE_ROLE metaclass.....	72
Figure 24. Attributes of the COMPLEX_SUBSTRUCTURE metaclass.....	74
Table 17. Relationships of the COMPLEX_SUBSTRUCTURE metaclass.....	75
Table 18. Attributes of the DATA_FIELD metaclass	77
Table 19. Relationships of the DATA_FIELD metaclass	77
Table 20. Relationships of the ELEMENT metaclass.....	79
Table 21. Attributes of the ENCAPSULATION metaclass.....	80
Table 22. Relationships of the ENCAPSULATION metaclass.....	80
Table 23. Relationships of the END metaclass.....	81
Table 24. Attributes of the EVENT_VARIANT metaclass	84
Table 25. Relationship of the EVENT_VARIANTS metaclass.....	84
Table 26. Attributes of the FIELD metaclass	86
Table 28. Attributes of the GOAL metaclass	87
Table 29. Relationships of the GOAL metaclass.....	88
Table 30. Attributes of the INDICATOR metaclass	89
Table 31. Relationships of the INDICATOR metaclass.....	89
Table 32. Attributes of the INGOING metaclass	90
Table 33. Relationships of the INGOING metaclass	91
Table 34. Attributes of the ITERATION metaclass.....	92
Table 35 Relationships of the ITERATION metaclass.....	93
Table 36. Relationships of the LOGICAL_NODE metaclass.....	94
Table 37. Attributes of the MESSAGE_STRUCTURE metaclass	95
Table 38. Relationships of the MESSAGE_STRUCTURE metaclass	96
Table 39. Attributes of the MODEL metaclass	97
Table 41. Relationships of the NODE metaclass	98
Table 42. Attributes of the OPERATIONALISATION metaclass.....	99
Table 43. Relationships of the OPERATIONALISATION metaclass.....	100
Table 44. Relationships of the OR metaclass	102

Table 45. Attributes of the ORGANISATION metaclass	103
Table 46. Relationships of the ORGANISATION metaclass	103
Table 47. Attributes of the ORGANIZATIONAL_ACTOR metaclass	104
Table 49. Attributes of the ORGANISATIONAL_LOCATION metaclass	106
Table 50. Relationships of the ORGANISATIONAL_LOCATION metaclass	106
Table 51. Attributes of the ORGANISATIONAL_MODULE metaclass	107
Table 52. Relationships of the ORGANISATIONAL_MODULE metaclass	108
Table 53. Attributes of the ORGANISATIONAL_ROLE metaclass	109
Table 54. Relationships of the ORGANISATIONAL_ROLE metaclass	110
Table 55. Relationships of the ORGANISATIONAL_ROLE_SET metaclass	111
Table 56. Relationships of the ORGANISATIONAL_UNIT metaclass	113
Table 57. Attributes of the OUTGOING metaclass	114
Table 59. Relationships of the PRECEDENCE metaclass	116
Table 60. Attributes of the PRIMARY metaclass	118
Table 61. Relationships of the PRIMARY metaclass	118
Table 62. Attributes of the PROCESS metaclass	119
Table 63 Relationships of the PROCESS metaclass	120
Table 64. Attributes of the RECEIVER metaclass	121
Table 65. Relationships of the RECEIVER metaclass	122
Table 66. Relationships of the REFERENCE_FIELD metaclass	124
Table 67. Relationships of the REFERENCE_FIELD metaclass	124
Table 68. Attributes of the SPECIALISATION metaclass	125
Table 69 Relationships of the SPECIALISATION metaclass	126
Table 71. Attributes of the STRATEGY metaclass	128
Table 72. Relationships of the STRATEGY metaclass	129
Table 73. Attributes of the SUBSTRUCTURE metaclass	130
Table 74. Relationships of the SUBSTRUCTURE metaclass	130
Table 75. Attributes of the SUPPORT metaclass	132
Table 77. Relationships of the SUPPORT_ROLE_SET metaclass	133
Table 78. Attributes of the TEXTUAL_REQUIREMENT metaclass	134
Table 79. Relationships of the TEXTUAL_REQUIREMENT metaclass	134
Table 80. Elements of the DOMAIN metaclass	135
Table 81. Elements of the OPERATION metaclass	136
Table 82. Elements of the REQUIREMENT_TYPE metaclass	137
Table 83. SuperStationery Co. business processes	154
Table 84. Order form	157
Table 85. Message structure specification of communicative event Sale1	158
Table 86. Heuristics rules	172
Table 87. Description of the adapted classification of usability problems (CUP) scheme	173
Table 88. Planning of the usability evaluation	177
Table 89. Positive features	187

1 Introduction

The organisational systems require specification methods to specify requirements and for identifying needs and characteristics of business process.

On the other hand, Model Driven Development (MDD) is a paradigm that provides to the requirement models of some advantages: as the capacity to derive conceptual models for generating software in an automatic way. The automatic generation of software products allows an easy adoption of requirement methods in an industrial environment.

This chapter presents an integration approach that involves requirement specification methods into MDD environments as a first approach to confront the integration of Communication Analysis method and OO Method. This approach will be presented in a general way. Each phase and stage will be explained and the application of this approach is presented through this memory. Bellow, we will present the motivation of this thesis, context, objectives that we want to achieve, the solution approach, research method and the thesis structure.

1.1 Motivation

The Information System development demands requirement methods to establish the organisation needs. The model driven development (MDD) paradigm provides to the requirement models of advantages: as the potential to derive from it the conceptual models for the automatic software generation. Although there are a successful evolution of software projects, the last CHAOS report to show a 68% of failed projects or threatened projects [1]. Academy and Industry are agree in to designate the lack of user participation as a more influent risk factor that threat software projects [2]. To involve the user in the development process allow the early correctness of mistakes and it increments the acceptance of the final software product [3]. An effective solution to involve the user into the software development process is providing requirement engineering practices, but there are some difficulties in the industry at the moment to apply requirement engineering methods [4].

The MDD paradigm allows solving some difficulties of the requirements engineering usage. The requirements specifications are models in the practical industry; these models are the specification of the software support. This model perspective corresponds with MDD paradigm [5]. This paradigm allows us to use the models for deriving conceptual models. This conceptual model derivation highlights the importance of requirements models and intends the industrial adoption because the requirements models acquire an important role into the organisation.

We think that is important to provide requirements engineering methods that are compatible with MDD environments. There are several requirements methods in the literature, and these requirement methods are not designed according to MDD paradigm. For this reason, we propose integration framework that involve the existing requirements engineering methods according the MDD paradigm.

Communication Analysis is a requirement engineering method that propose the analysis of information systems from a communicational perspective [6]. This method is currently applied in projects in industrial environments. Due to the fact that this method is used in prac-

tice, the use of Communication Analysis by industry presents an interesting challenge and research objective: offer a technological support for modelling the requirements engineering models, and also facilitate the conceptual model derivation and model transformation. A technological platform for the requirement method can increase the adoption of the method by big and medium enterprises; also, the integration of this method into a MDD environment will allow the derivation of the conceptual models and the later software product.

This call of the industry and the academy motivate us to design a framework to integrate Communication Analysis method and OO-Method. We have to develop several steps that intend to achieve the main objective: the integration of Communication Analysis method and OO-Method. The development of each step of the framework guide the research and engineer activity of this thesis, and present the resulting products of each task. As a result of this thesis, we propose a general integration framework, and in addition, the integration of the Communication Analysis requirement method and OO-Method; this integration solves the lack of technological support of the method techniques, and increments the use among software analyst and requirements engineers.

The integration framework has been created as a previous step before to carry out the integration of Communication Analysis method and OO-Method. The integration framework could be used to integrate other requirements engineering method t and conceptual model method.

1.2 Context

This Master Thesis was developed in the context of the research center *Centro de Investigación en Métodos de Producción de Software* of the *Universitat Politècnica de València*. The work that has made the development of this thesis possible is in the context of the following research government and industrial projects:

- **MORE-MOSKITT:** *“extensión de la oferta metodológica asociada a la plataforma Moskitt, incorporando a sus componentes actuales métodos de captura y modelado de requisitos que den respuesta al problema de modelar adecuadamente los requisitos de un Sistema de Información, y su correcta incorporación en un proceso completo de producción de software robusto, eficiente y efectivo”*. This Project was collaboration between PROS research center and TECCON Ingenieros Consultores, which had a special interest in to finance the project.
- **PROS-REQ:** Requirement-based production of service-oriented software TIN2010-19130-C02-02.

1.3 Objectives

The main research goal of this thesis is *to propose an approach to integrate Communication Analysis method and OO-Method*. This proposal presents a general framework that guides the integration activity and provides software artefacts that support the integration between the requirement method and the conceptual method. This thesis proposes to use the MDD paradigm to develop technological environments for requirements methods, which provide several advantages as: agile development of software, availability of open source platforms for MDD developments (e. g, Eclipse) and possibility of software automatic generation, etc. In order to obtain the main objective, we are in front of a set of research questions that we will explain below:

- RQ1: What are the stages and phases that guide the integration of requirement methods into a MDD environment?
 - RQ1.1: What are the results of each task? The answer of this question allows knowing the resulting products after carry out each task.
- RQ2: Are there requirements engineering methods integrated with MDD environments?

- RQ3: It is possible to integrate Communication Analysis with OO-Method?
 - RQ2.2: What is necessary to know about Communication Analysis method?
 - What is necessary to know about OO-Method?
 - What are the activities in the model driven development process?
 - What is the most adequate technological architecture to support the integration of Communication Analysis method and OO-Method?
- RQ4: How can the results of the integration framework are validated?
 - What kinds of validations are necessary to validate the technological support?
- RQ5: How is the technological support that results of the framework application?
 - How can be used the technological support?

To solve these research questions and to achieve the main research goal, we have followed a research methodology which will be explained at section 1.5.

1.4 The proposed solution

Academy and Industry have agreed to point out the lack of users participation into software projects as the most influential risk that in the success of final software product [1] [2].

To involve the user into software development process allows an early detection and correction of software mistakes, furthermore, the acceptance of the final product is increased [3]. Adoption of requirements engineering techniques is an effective solution to involve the users into the software development process. Nevertheless, the adoption of requirements methods into the industry do not correspond with the prospect of the academy [4]. Among factors that hinder a industrial adoptions of requirements methods, we could to highlight:

complexity of requirements engineering, lack of training in new methods and reluctant attitude of the users [7].

Model Driven Development (MDD) paradigm, allows solving some problems of the requirements engineering. The MDD is a solution of the software engineering problems because the requirements specification are an abstract model of the industrial practice that should to support the software product, thus these concepts are compatible with MDD [5]. The MDD paradigm allows to use the models for documenting and to communicate the organisational needs according to the Information System (IS). In addition, MDD techniques allows to derive the conceptual models of the IS from the requirements models. This derivation increases the use of the requirements models and the adoption by industrial environments. Thus, the requirements engineering have a challenge: to provide requirements methods that can be used into MDD environments and to provide strategies for model transformations. The challenge could be confronted in two ways: (i) the requirements engineering community should to provide new methods that are compliant with MDD environments (for instance [8]), or (ii) integrating existing methods into MDD environments (for instance [9]).

This thesis proposes to follow the integration way. To achieve this objective, we have propose a general framework that define stages and phases that involve some task to integrate requirement methods into MDD environments. Objective is to provide a complete development environment of IS. This development environment could be established in a general framework, which takes into account model transformations to achieve a software product through models compilation.

Bellow, we will present a detailed description of the stages, phases and task of the general framework. This thesis provides an application of this framework. This example of application was carried out integrating the requirements models of the Communication Analysis method into OO-Method, an object oriented method supported by OLIVANOVA, an MDD environment that support OO Method.

The generic framework

We are aiming to integrate requirements engineering methods into MDD environments, for this reason we have conceived a general framework that involves several stages, phases and tasks in order to achieve our objective (see Figure 1).

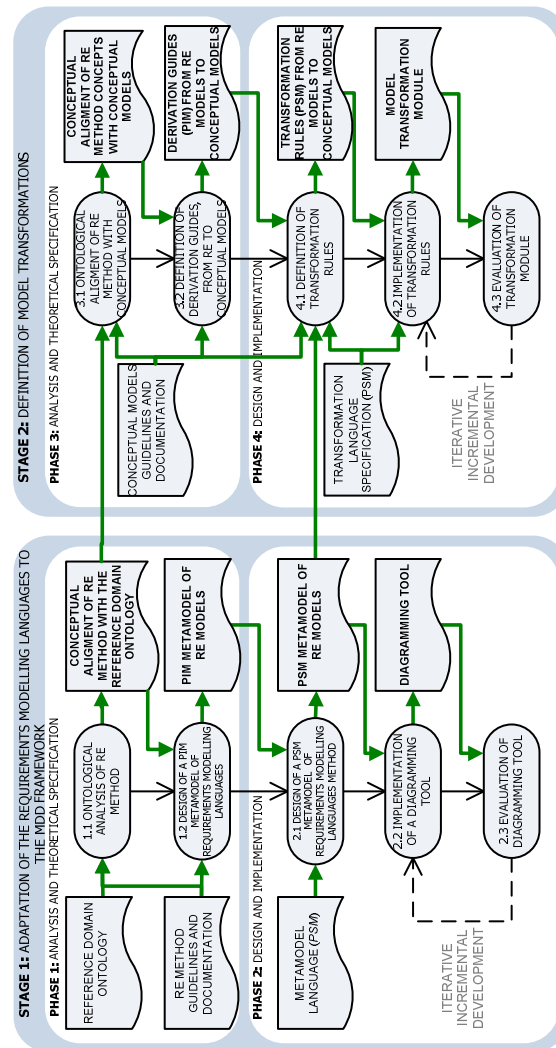


Figure 1. General framework of integration

Each stage is divided in two phases, and each phase has several tasks. We have decided to differentiate each stage according the tasks involved into the MDD process. We distinguish the stage concerning modelling tasks (stage 1) and the stage concerning model transformation tasks (stage 2).

Each phase corresponds to activities of analysis and theoretical specification and design and implementation.

Each task uses existing inputs. For instance, the input *method guidelines* correspond to the guides of the requirement method that will be integrated into a MDD framework and it is input for the tasks: *ontological analysis of RE method* and *PIM metamodel design of RE method*.

Each task has related an objective, for instance, the task *diagramming tool implementation* intends the construction of a *diagramming tool*. Each task has associated an output product

We present the specification of each stage below.

Stages of the generic framework

We are aiming to integrate requirements methods into MDD environments. Thus the general framework has two stages:

- *Adaptation of the requirements modelling languages to the MDD framework*
- *Definition of model transformations*

The *Adaptation of the requirements modelling languages to the MDD framework* involves some activities that are related to metamodeling and to the construction of diagramming tools.

The *Definition of model transformations* involves some activities related to model transformation from requirements models to conceptual models. These activities are aimed at software generation code in an automatic way.

We present a description of each phases and activities for each stage below.

Stage 1: Adaptation of requirements modelling languages to the MDD framework

Description and objectives

The objective of this stage is to adapt the modelling language of the requirements method into a MDD environment.

This stage has two phases (see Figure 2): (i) *Analysis and theoretical specification*. Which has two task: an *ontological analysis of RE¹ method* that allows to establish correspondences among the elements of the requirements models and the concepts of a reference ontology [10]. The other task is the *design of a metamodel of the requirements modelling languages*, which allows us to establish the method in a metamodel specification without considering technology restrictions such as the platform that will support the modelling tool. (ii) *Design and implementation*. This phase aims to develop tools to support the requirements engineering method. For this reason a modelling tool is designed and implemented to offer a graphical environment for representing requirements models in conformance to the requirements engineering method. We describe each phase and its corresponding tasks below.

¹ RE is Requirements Engineering in sort.

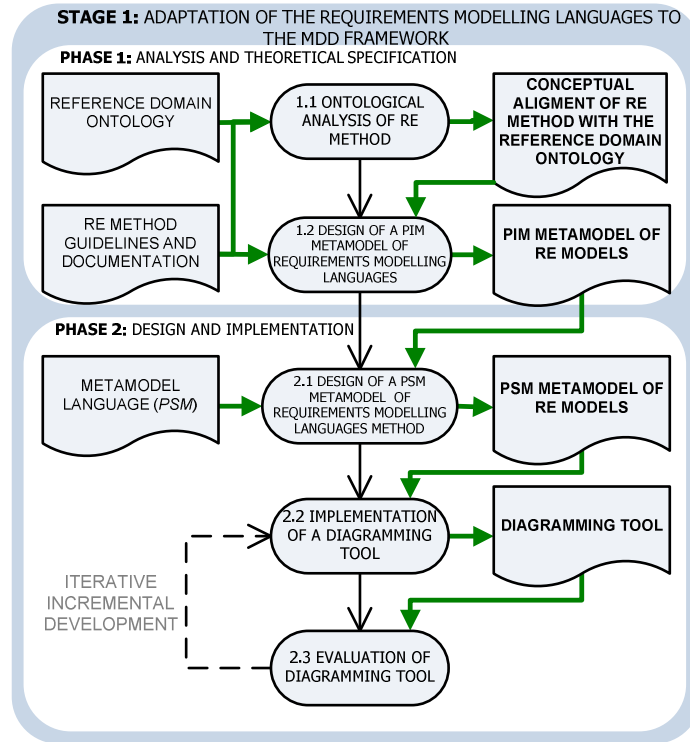


Figure 2. Stage 1 of the general framework of integration

Phase 1: Analysis and theoretical specification

This phase has an objective: to carry out an analysis of the requirements engineering method to specify the method in a high abstraction level (for instance, a metamodel). The tasks of this phase are related to a complete study of the requirements engineering method. The entries of this phase are: documentation, models, and whole information about the requirements engineering method that will be integrate.

TASK		ENTRIES	OUTPUTS	PARTICIPANT ROLES
Task 1.1	Ontological analysis of requirements engineering	- Reference domain ontology. - Requirements engineering	- Conceptual alignment of requirements engineering method	Method engineer.

	method.	method guidelines and documentation.	whit the reference domain ontology.	
Task 1.2	Design of a PIM metamodel of requirements modelling languages.	- Requirements engineering method guidelines and documentation.	- PIM metamodel of requirements engineering models.	Analyst.

Task 1.1: Ontological analysis of requirement method

The objective of this task is to establish correspondences among the elements of reference ontology and the modelled primitives of the requirements models. This task is carried out in order to analyse each primitive of the requirements model and the behaviour of it.

Therefore, these correspondences allow us to think in a high abstraction level about the requirements models.

The method engineer needs information about the method and concepts about the domain, then this task uses the reference domain ontology and the method guidelines and documentation. The method guidelines and the documentation are often in a format that is computation independent, this documentation may be reports, models, templates and whole kind of information that describe the elements of the method. However, this documentation may be in a computation format represented at a technological platform, for instance Microsoft Visio.

Products

Entries

- Reference domain ontology.
- Requirements engineering method guidelines and documentation.

Outputs

- Conceptual alignment of requirements engineering method with the reference domain ontology.

Task 1.2: Design of a PIM metamodel of requirements modelling languages

The objective of this task is to represent the requirements models of the method in a high level of abstraction (a metamodel). The metamodel help us to establish the elements, constraints and the information of the requirements models.

During this task, the metamodel is described regardless of concepts about the technological platform: it is, therefore, at the Platform Independent Model layer (PIM). This level of abstraction allows us to create the metamodel without considering technological constraints of a specific target platform; nonetheless, later on, the developers can choose the most suitable platform for implementing the CASE tool. This metamodel can be represented in a piece of paper or using any general-purpose diagramming tool. This metamodel should be specified according the desires of the analysts and the method experts, for instance, if analysts consider that is necessary to indicate the data type of the attributes, this metamodel should have this information.

Products

Entries

- Requirements engineering method guidelines and documentation.

Outputs

- PIM metamodel of requirements engineering models.

Phase 2: Design and implementation

The objective of this phase is to offer a technological environment to provide a support to requirements models of the requirements engineering method. Thus, a modelling tool is necessary for representing instances of the requirements metamodel. In order to define a modelling tool, it is possible to define a metamodel that includes information about technological platform. Thus, platform specific metamodel is appropriate to include both method-related and technology-related information. The PIM metamodel designed in the task 1.2 is the entry to build the PSM metamodel. Then, with the appropriate technology it is possible to obtain a modelling tool that provides a modelling environment.

Finally, an evaluation exercise is proposed to evaluate the usability of the diagramming tool. This evaluation can be carried out several times, according to the analyst criterion.

TASK		ENTRIES	OUTPUTS	PARTICIPANT ROLES
Task 2.1	Design of a PSM metamodel or requirements modelling languages method.	- Metamodel language (PSM). - PIM metamodel of requirements engineering models.	-PSM metamodel of requirements engineering models.	Analyst. Developer.
Task 2.2	Implementation of a diagramming tool.	-PSM metamodel of requirements engineering models.	- Diagramming tool.	Developer.
Task 2.3	Evaluation of diagramming tool.	- Diagramming tool.	--Diagramming tool.	Developer.

Task 2.1: Design of a PSM metamodel of requirements modelling languages

The objective of this task is to involve both technical and implementation characteristics into a metamodel that gather method information and technological platform information. In order to achieve this objective, it is necessary to take into account the characteristics of the target platform, and how to specify these characteristics into a metamodel. Thus, the PIM metamodel of the requirement models could be represented in the chosen technological platform. As a result, a PSM metamodel is obtained.

Products

Entries

- Metamodel language (PSM).
- PIM metamodel of requirements engineering models.

Outputs

- PSM metamodel of requirements engineering models.

Task 2.2: Implementation of a diagramming tool

The analyst should decide about the technological implementation of the diagramming tool for the requirements models. The PSM meta-model created at task 2.1 contains the information about the method and the implementation constraints, thus, developer builds a modelling tool using the MDD strategy that they consider more appropriate. For instance, the developer could use Eclipse technologies, or Microsoft technologies.

Products**Entries**

-PSM metamodel of requirements engineering models.

Outputs

- Diagramming tool.

Task 2.3: Evaluation of diagramming tool

In order to provide a usable modelling tool, the analyst should design testing environments. The analyst takes into account the feedback of the final users, or the feedback provided by the experts. In any case, it is important carry out evaluation activities of the technological support of the method.

Products**Entries**

- Diagramming tool.

Outputs

- An Improved version of the diagramming tool.

Stage 2: Definition of model transformation**Description and objectives**

The objective of this stage is to define model transformation from requirements models to conceptual models in a MDD environment.

A support to the integration of requirements engineering method and conceptual models is intended, for this reason we propose two phases: (i) Analysis and theoretical specification of the derivation guidelines. Which is has two tasks: an ontological alignment of the PIM metamodel of the method and design and definition of the rules

derivation. (ii) Design and implementation. This phase aims to develop tools support for the models transformation. For this reason, a transformation module is proposed to offer an environment to carry out model transformation activities. Figure 3 presents the general framework focused on stage 2. We describe each phase with their corresponding tasks below.

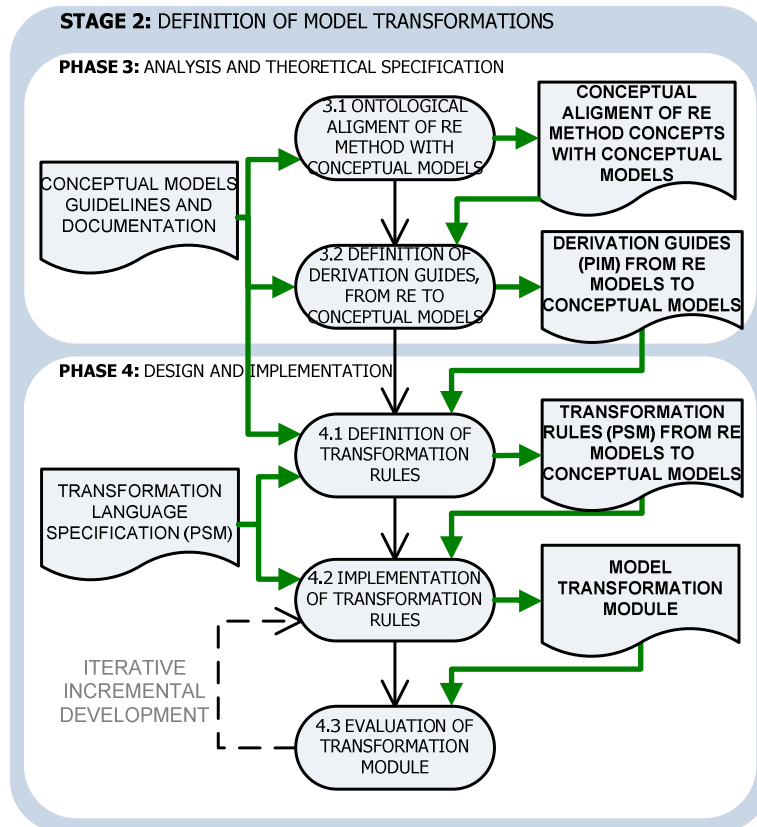


Figure 3. Stage 2 of the general framework of integration

Phase 3: Analysis and theoretical specification

TASK	ENTRIES	OUTPUTS	PARTICIPANT ROLES
------	---------	---------	-------------------

Task 3.1	Ontological alignment of requirements engineering method with conceptual models.	<ul style="list-style-type: none"> - Conceptual alignment of requirements engineering method concepts with reference domain ontology. - Conceptual models guidelines and documentation. 	<ul style="list-style-type: none"> - Conceptual alignment of the requirements engineering method concepts with conceptual models. 	Method engineering.
Task 3.2	Definition of derivation guides, from requirements engineering to conceptual models.	<ul style="list-style-type: none"> - Conceptual models guidelines and documentation. - Conceptual alignment of the requirements engineering method concepts with conceptual models. 	<ul style="list-style-type: none"> - Derivation guides (PIM) from requirements engineering models to conceptual models. 	Method engineering.

Task 3.1: Ontological alignment of RE method with conceptual models

The objective of this task is to align the concepts of both methods. Thus, it is necessary to analyse the concepts of the conceptual models and the concepts of the requirements models. This is the previous step for build the derivation guides. A complete description of this task is specified in [11].

Products

Entries

- Conceptual alignment of requirements engineering method concepts with reference domain ontology.
- Conceptual models guidelines and documentation.

Outputs

- Conceptual alignment of the requirements engineering models with conceptual models.

Task 3.2: Definition of derivation guides from RE to conceptual models

The derivation guides are the principal product of this phase. In this task, the guides should be represented in natural language or pseudocode, aiming a human reader (e.g. an analyst that intends to apply them in real projects). This guides allow us to represent the derivation guides in the transformation language more appropriate according the chosen technology.

Products

Entries

- Conceptual models guidelines and documentation.
- Conceptual alignment of the requirements engineering method concepts with conceptual models.

Outputs

- Derivation guides (PIM) from requirements engineering models to conceptual models.

Phase 4: Design and implementation

The objective of this phase is the definition of transformation rules that will be specified in a technological platform. In order to provide a transformation module, the analyst uses the PSM metamodel of the requirements models and conceptual models to define the transformation rules. Finally, the analyst should design software evaluation activities for the transformation module. These activities should involve the experts and the final users.

TASK		ENTRIES	OUTPUTS	PARTICIPANT ROLES
Task 4.1	Definition of rules transformation	<ul style="list-style-type: none"> - Conceptual models guidelines and documentation. - PSM metamodel of requirements engineering models. - Transformation language specifi- 	<ul style="list-style-type: none"> - Transformation rules (PSM) from requirements engineering models to conceptual models. 	Method engineering.

		cation (PSM). - Derivation guides (PIM) from requirements engineering models to conceptual models.		
Task 4.2	Implementation of transformation rules	- Transformation language specification (PSM). - Transformation rules (PSM) from requirements engineering models to conceptual models.	- Model transformation module.	Analyst. Developer.
Task 4.3	Evaluation of transformation module	- Model transformation module.	- Model transformation module.	Analyst.

Task 4.1: Definition of transformation rules

The objective of this task is to define the transformation rules taking into account the metamodels (requirements and conceptual models), the transformation language chosen, and the derivation guides previously defined. The transformation language specification is according to the technology chosen (e.g. ATL or QVT).

Products

Entries

- Conceptual models guidelines and documentation.
- PSM metamodel of requirements engineering models.
- Transformation language specification (PSM).
- Derivation guides (PIM) from requirements engineering models to conceptual models.

Outputs

- Transformation rules (PSM) from requirements engineering models to conceptual models.

Task 4.2: Implementation of transformation rules

The objective of this task is to implement the transformation rules in the chosen technological platform. A model transformation module is the principal product of this phase.

Products**Entries**

- Transformation language specification (PSM).
- Transformation rules (PSM) from requirements engineering models to conceptual models.

Outputs

- Model transformation module.

Task 4.3: Evaluation of transformation module

The transformation module should be evaluated in order to offer a transformation support appropriate to the users and experts of the method. In addition, it is important the usability of the modelling tool and transformation module to improve the tool acceptance and industrial distribution.

Products**Entries**

- Model transformation module.

Outputs

- Model transformation module.

Example of the generic framework application

In order to show the application of the proposal, we have carried out a probe of concept that consists to integrate Communication Analysis method and OO-Method (the main objective of this thesis). Figure 4 presents an example of how the general framework could be used in order to integrate a requirements method into a MDD environment.

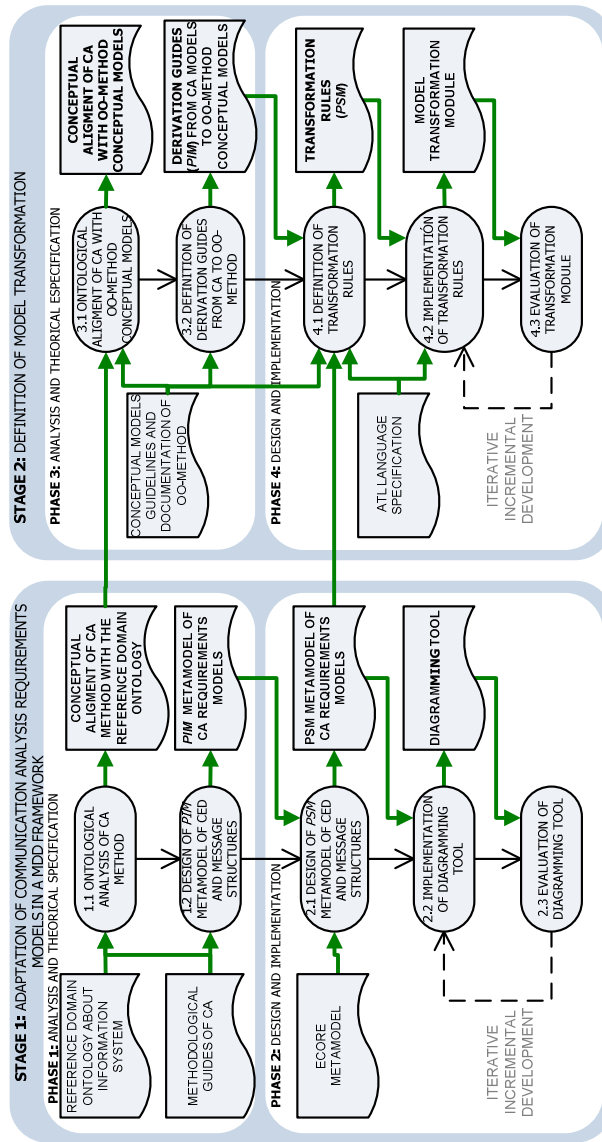


Figure 4. Integration framework to involve Communication Analysis method into a MDD environment

Communication Analysis is a requirements engineering method that proposes to analyse the IS from a communicational perspective [6]. The Communication Analysis proposal and its structure is presented at [12]. In complex projects, the system is refined into subsystems, each process is modelled through a Communicative Event Diagram (CED) and its corresponding messages are modelled through Message Structures. The principal business objects are identified and each communicative event is described by mean of templates.

The integration framework intends to involve the Communication Analysis techniques into a MDD environment. The Conceptual alignment of Communication Analysis method, methodological guides of the method and the derivation guides from the OO-Method conceptual models to Communication Analysis requirements models are proposals of PhD thesis of Sergio España [11]. Thus, taking this proposal we can to offer a technological support into a MDD environment.

Chapters 4, 5 and 6 present the development of each phase of the integration framework. We have described the decisions and resultant products of each stage.

1.5 Research methodology

The thesis follow the general guides of the scientific method [13] and the advices specified at [14]. The Figure 5 shows the structure of the work method.

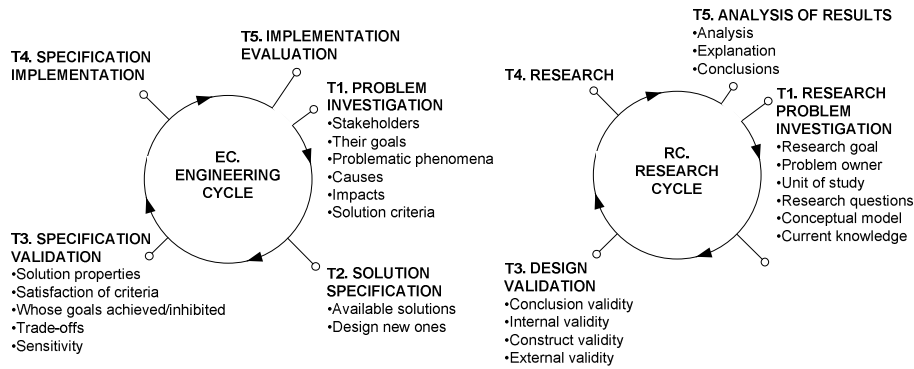


Figure 5. Research method followed in the thesis

EC1: Propose an integration framework to involve requirements methods into MDD environments

T1: Problem investigation

T1.1: Define motivation, main goal and research questions (please see sections 0 and 1.3).

T2: Solution specification

T2.1: Study of state of the art in integration methods approaches (please see Chapter 2).

T2.2: Design a general integration framework to involve requirement methods into MDD environments (please see section 1.4).

T3: Specification validation

Specification validation consists in to apply the proposed framework, where the objective is to provide a technological support for Communication Analysis and OO Method integration. To follow the stages and phases proposed at the integration framework induces to follow two engineering cycles: EC2 and EC3, which induce to follow two research cycles: RC1 and RC2 respectively.

EC2: Provide a technological support for Communication Analysis method.

T4: Problem investigation

T4.1: Define a motivation for proposing a technological support (please see Chapter 5).

T5: Solution specification

T5.1: Study of available technological support for building modelling tools (please see section 4.1).

T5.2: Select the most adequate development tool for modelling support (please see section 4.1).

T6: Specification validation

T6.1: Provide a test case to be implemented in the selected development tool (please see section 4.4).

T7: Specification implementation

T7.1: Develop of the modelling tool for Communication Analysis method (please see Append 1)

T7.2: Provide an example about how to use the modelling tool (please see section 4.4).

T8: Implementation evaluation

T8.1: Validate the created modelling tool for Communication Analysis method. (T8 requires an evaluation exercise of the modelling tool, which implies RC1).

RC1: Validate the technological support for Communication Analysis method

T9: Research problem investigation

T9.1: Define the goal of the modelling tool validation (please see section 4.5).

T9.2: State of the art about modelling tool validations.

T9.3: Are there a method about how to validate modelling tools?

T9.4: What kinds of evaluations are more adequate?

T10: Research design

T10.1: Define the research team, which include involving to Jean Vanderdonckt (an expert about usability evaluations of software products) and experts about Communication Analysis method (Sergio España) and an expert in the modelling tool.

T10.2: Design the material for the heuristic evaluation and expressiveness (please see subsection 4.5.1).

T10.3: Design the material for the usability evaluation with users (User testing) (please see subsection 4.5.2)

T10.2: Design the environment where will be carried out the evaluation.

T10.3: Install the modelling tool.

T11: Design validation

T11.1: Evaluate the material for the heuristic evaluation.

T11.2: Evaluate the material for the usability evaluation with users.

T11.3: Carry out a test of the usability evaluation with users with a set of people to identify mistakes on the material.

T11.4: Check the correct installation of the modelling tool.

T12: Research

Not applicable

T13: Analysis of results

T13.1: Specify the results of the evaluations into an Excel form.

T13.2: Analyse the results through the SPSS software for checking the mean and other interesting information (please see section 4.6).

EC3: Provide a technological support for the integration of Communication Analysis method and OO-Method.**T14: Problem investigation**

T14.1: Define a motivation for proposing a technological support (please see Chapter 6).

T15: Solution specification

T15.1: Study of available technological support for model transformation (please see section 5.2).

T15.2: Select the most adequate technological support for model transformation (please see section 5.2).

T16: Specification validation

T16.1: Provide a test case to be implementing in the selected development tool (please see section 5.4).

T17: Specification implementation

T17.1: Develop of the transformation module (please see section 5.3 and Append 2)

T17.2: Provide a traceability support (please see section 5.6)

T18: Implementation evaluation

T18.1: Validate the integration module. (T18 requires an evaluation exercise of the integration module, which implies RC2).

RC2: Validate the technological support for the integration of Communication Analysis method and OO-Method**T19: Research problem investigation**

T19.1: Define the goal of the transformation module validation (please see section 5.5).

T20: Research design

T20.1: Define the research team.

T20.2: Design the material for the evaluation.

T20.3: Design the environment where will be carried out the evaluation.

T20.4: Install the transformation module.

T21: Design validation

T21.1: Evaluate the material for the heuristic evaluation.

T21.2: Evaluate the material for the usability evaluation with users.

T21.3: Carry out a test of the usability evaluation with users with a set of people to identify mistakes on the material.

T21.4: Check the correct installation of the transformation module.

T22: Research

Not applicable

T23: Analysis of results

T23.1: Specify the results of the evaluations into an Excel form.

T23.2: Analyse the results through the SPSS software for checking the mean, and other interesting information.

T24: Specification implementation

Not applicable.

T25: Implementation evaluation

Not applicable.

1.6 Thesis structure

The approach followed in this work involves raising the abstraction level of the model-driven framework integration proposed. The work has been structured to reflect this abstraction process. First, Chapter 2 gives an overview of some relevant concepts related to frameworks for method integration, Integration process of RE specification with OO conceptual models, Technological support for business process

modelling, Communication Analysis Method and OO-Method. Chapter 3 specifies the MDD approach to support the Communication Analysis techniques. In order to do so, the techniques are formalized in a metamodels, and abstract and concrete syntax are specified. Chapter 4 presents the modelling tool for Communication Analysis requirements models. This chapter present the chosen technological support, the design of the graphical editor, an example of how to use the modelling environment and the modelling tool validation. Chapter 5 presents the transformation support. This chapter presents the rules implementation, a transformation example, transformation validation and a support for the traceability, a plus that let us to follow the transformation process. Chapter 6 summarizes the contributions, present the publications and provides some insight about further work. Finally the section with references is presented and two appends. The first one is about the development steps of the modelling tool and the second one is about the source code of transformation rules.

2 State of the art

This chapter reviews the state of the art related to this thesis. It embraces different fields, disciplines and techniques that are related to frameworks of method integration, integration process of requirements specification with object oriented conceptual models and technological support for business process modelling.

We present an overview about Communication Analysis method and OO-Method. Finally, we present a discussion about the works analysed, and how these works have been influent and important to draw up this thesis.

2.1 Frameworks for method integration

The diversity of development and change situations gives rise to a need to combine and integrate different methods [15]. There are several cases about method integration: to integrate several methods, to combine several methods or parallel execution of methods. Many current methods seem to be the result of integration of different methods or method fragments [16].

Bellow, we will present some approach about method integration frameworks; these frameworks propose different ways to confront method integration.

A metamodel approach

The use of methods can provide some assurance toward the quality of the resultant systems. Actually, many well-known development methods, such as structured analysis and design, object oriented analysis and object oriented design are supported by CASE tools with facilitate the development. A metamodel propose is presented at [17], the proposal shows an approach to integrate multiple methods through the use of metamodel. The approach lies in two aspects: the use of semantic equivalence between method components to establish uniformity between individual methods and the incorporation of procedural information, task and their order into the metamodel. The proposal presents a demonstration using a CASE tool to integrate object diagrams, state transition diagrams and data flow diagrams.

The core of the proposal lies on the metamodel strategy, the experience of this work indicates that the metamodel captures the common concepts of individual methods, and the metamodel can also facilitate the creation of a method database, and later it is possible to create a standard environment to support a set of methods. The use of metamodels allows the creation of modelling tools for supporting diagramming according to integrated method.

The experience presented in this article highlight some ideas having into account in the integration framework proposed in this thesis. The first task of our proposal brings the methods to will be integrated to

metamodel specification; a high abstraction level for relating the two methods to integrate (Communication Analysis and OO-Method).

Interoperability in MDD approach

MDD approaches propose the automatic generation of software products by means of the transformation of the defined models into the final program code. In the MDD context, interoperability can be considered a trend of model-driven technologies, where different developing and modelling approaches, tools and standards can be integrated and coordinated to reduce the quality of the final software products. An interoperability proposal is presented by [Giachetti 2011] [18], which proposes an approach to achieve the interoperability in MDD processes. This interoperability approach is based on current metamodeling standards, modelling language customization mechanisms, and model-to-model transformation technologies. The proposal presents the integration of modelling languages, to obtain a suitable interchange of modelling information and to perform automatic interoperability verification.

A multi-perspective framework for method integration

The development of large and complex systems necessarily involves many people, each one with their own perspective on the system. This is particularly true for composite systems, that is, systems which deploy multiple component technologies. The intersections between perspectives however are far from obvious because the knowledge within each is represented in different ways. Furthermore, because development may be carried out concurrently by those involved, different perspectives may be at different stages of elaboration, and may be subject to different development strategies. The approach proposed at [19] presents the perspective problem, in particular, the problem of method integration in this direction. The approach presents a model of ViewPoint, a concept that encapsulate partial representation process and specification knowledge about a system and its domain; thus, a model of ViewPoint interaction in which inter-ViewPoint rules are defined during method design. The approach in-

volves process modelling to guide ViewPoint interaction. The multi-perspective development is managed by prescribing rules represented at the ViewPoints.

2.2 Integration process of RE specification with OO conceptual models

There are several approaches to link requirements specifications with object oriented conceptual models. To bellow are presented works that confront the integration process between requirements specification with object oriented conceptual models in different ways. The experience of these works has influent the development of the integration framework and guides the thesis proposal.

The proposal presented by [de la Vara 2011] [20] lies of a methodological approach for business process-based requirements specification and object oriented conceptual modelling of information systems. The approach consists of four stages: organisational modelling, purpose analysis, specification of system requirements and derivation object-oriented diagrams.

The four stage of this proposal integrates system requirements into OO conceptual modelling. As a result of the integration, system requirements can be useful for an OO perspective, IS modelling and development. For performing the stage, first ETDs (Extended Task Description) are analysed to specify several details that are necessary for derivation of the OO conceptual schema of an IS. Next, a class diagram and state transition diagrams are modelled by following two sets of rules that determine the correspondence between the system requirements of an IS and its OO conceptual schema. The rules transformations are presented in natural language and these are fully automatable. The analyst criteria can be taking into account during rules automation.

The further link with OO-Method is possible, characteristics and details distinctive between OO-Method and the OO conceptual schema

generated need the analyst criteria to establish the correspondence between ETDs and OO-Method conceptual schema.

2.3 Technological support for business process modelling.

Actually there are different tools for supporting business process modelling; bellow we will present an overview of some tools for supporting business process modelling. Study the existent tools allow us to know the support for BPM and the current technology to develop business process modelling environments.

AuraPortal BPM [21] is a software to support enterprise management. The BPM module support project management to model level, orchestration, monitoring and business rules specification. Distribution of this tool is commercial.

MOSKitt is a modelling framework supported by Eclipse, which includes plug-ins to modelling support of BPMN and UML [22]. MOSKitt is a FREE CASE tool, built on Eclipse which is being developed by the Conselleria de Infraestructuras, Territorio y Medio Ambiente to support the gvMétrica methodology (adapting Métrica III to its specific needs). gvMétrica uses techniques based on the UML modeling language. MOSKitt's plug-in architecture makes it not only a CASE Tool but also a Free Modelling Platform to develop this kind of tools. MOSKitt is being developed within the gvCASE project framework. This is one of the projects integrated ingvPontis, the Conselleria global project for the migration of its entire technological environment to free Software.

Recently, a plug-in for Microsoft Visio have been developed; Interfacing Technologies Corporation is the company that carry out the plug-in development [23]. This plug-in supports BPM diagrams. Distribution of this plug-in is commercial.

2.4 Communication Analysis method

Communication Analysis is a method for the development and computerisation of enterprise Information Systems. This method focuses on communicative interactions that occur between the IS and its environment. Communication Analysis is currently being used by important Spanish enterprises and governmental institutions. The communicational perspective of the method has been overviewed in a previous publication [12].

From a systemic point of view, the kind of problem that Communication Analysis confronting involves at least three systems. The Organisational System (OS) is a social system that is interested in observing, controlling and/or influencing a portion of the world. We refer as Subject System (SS) to the portion of the world in which the OS is interested (a.k.a universe of discourse). An Information System (IS) is a socio-technical system, a set of agents of different nature that collaborate in order to support communication between the OS and its environment.

Communication Analysis proposes a requirements structure that allows a stepwise refinements approach to ISs description. Also, the proposed method allows tackling with static and dynamic perceptions of reality (by giving support to discovering and describing that duality). The structure and the method flow are divided in five levels:

L1.System/subsystems level refers to an overall description of the organisation and its environment (OS and SS, respectively) and also involves decomposing the problem in order to reduce its complexity.

L2.Process level refers to business process description both from the dynamic viewpoint (by identifying flows of communicative interactions, a.k.a. communicative events) and the static viewpoint (by identifying business objects).

L3.Communicative interaction level refers to the detailed description of each communicative event (e.g. the description of its associated message) and each business object.

L4.Usage environment level refers to capturing requirements related to the usage of the CIS, the design of user interfaces, and the modelling of object classes that will support IS memory.

L5.Operational environment level refers to the design and implementation of CIS software components and architecture.

Levels L1, L2 and L3 belong to the problem space, since they do not presuppose the computerisation of the IS and they aim to discover and describe the communicational needs of users. Levels L4 and L5 belong to the solution space, since they specify how the communicational needs are going to be supported.

On the system/subsystem requirements level, the analyst describes the OS from the strategic point of view. On the one hand, when the organisation is complex, it is advisable to decompose the problem into subsystems or organisational areas. On the other hand, the analyst elicits requirements related to strategic-level business indicators.

On the process requirements level, Communication Analysis proposes describing business processes from a communicational perspective. The aim is to discover communicative interactions between the IS and its environment, and to describe them taking into account their dynamic and static aspects; that is, creating the Communicative Event Diagram and the Business Objects Glossary, respectively. In the following, a series of definitions clarify the concepts upon which the modelling techniques are built.

We refer as communicative interaction to an interaction between actors with the aim of exchanging information. FRISCO report [16] presents a generic model of ISs that considers an IS as a support for communicative interactions. In a previous publication, the authors extend this model in order to deepen the communicative point of view [24]. Depending on the main direction of communication, the following types of communicative interactions can be distinguished:

- Ingoing communicative interactions primarily feed the IS memory with new meaningful information. These interactions often appear in the shape of business forms.
- Outgoing communicative interactions primarily consult IS memory. These interactions often appear in the shape of business indicators, listings and printouts.

The ingoing communicative interactions entail more analytical complexity.

- A communicative event is a set of actions related to information (acquisition, storage, processing, retrieval and/or distribution), which are carried out in a complete and uninterrupted way, on the occasion of an external stimulus [25].

Communication Analysis offers unity criteria to allow identifying communicative events, also facilitating the determination of their granularity. This way, a communicative event can be seen as an ingoing communicative interaction that fulfils the unity criteria. Each unity criterion is related to a communication function (see [26] for detailed information).

Communication Analysis proposes to specify the flow of communicative events by means of the Communicative Event Diagram (CED). The primitives of this modelling technique are shown at the bottom of Figure 6 (CED example) and explained next.

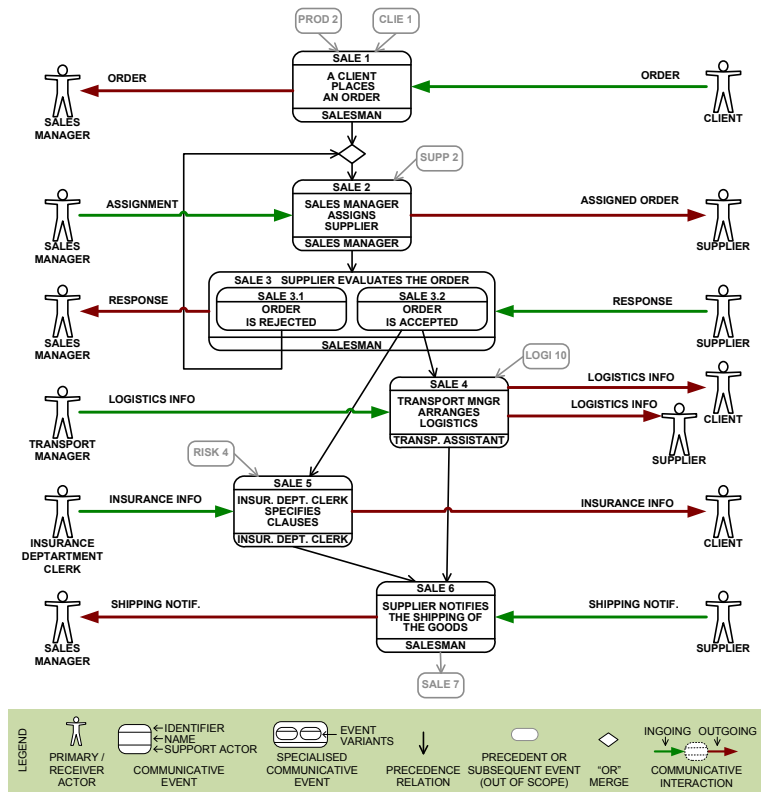


Figure 6. Communicative event diagram of SuperStationery Co. *Sales management* business process (Sale)

Each communicative event is represented as a rounded rectangle and is given an identifier and a descriptive name. The identifier serves for traceability purposes and it is usually a code composed of a mnemonic (related to the system to which the event is ascribed) and a number (e.g. PHO 3). For each event, involved actors are identified. Communication Analysis distinguishes several roles (see theoretical basis in [24]):

- The primary actor triggers the communicative event by establishing contact with the OS and provides the conveyed input information. Therefore, primary actors are modelled as senders of incoming communicative interactions.

- The support actor is in charge of physically interacting with the IS interface in order to encode and edit input messages. Support actors are specified at the bottom of the event rounded rectangle. Sometimes the primary actor and the support actor are different persons. Other times both roles are played by the same person.
- Receiver actors are those who need to be informed of the occurrence on an event. In order to truly understand the meaning of messages in organisations, it is necessary to analyse these actors. They are modelled as receivers of outgoing communicative interactions.
- Reaction processors are those in charge of performing the IS reaction to the message. This role is not depicted in the CED.

The messages associated to communicative events are conveyed via ingoing communicative interactions and outgoing communicative interactions. In the CED, messages are given a name. Communicative interactions are modelled as arrows placed in the horizontal axis. The vertical axis is reserved for precedence relations among communicative events, which are also modelled as arrows.

Communicative events are specialised whenever each specialised variant leads to a different temporal path (i.e. distinct precedence relations). It must be avoided specializing an event as a result of different communication channels, since the message remains the same.

This requirements level also provides a static perspective of business processes, by means of business objects. We refer as business objects to the conceptions of those entities of the Subject System in which the OS is interested. Frequently, stakeholders describe business objects as complex aggregates of properties. Business objects are identified and described in a Business Object Glossary. Also, users are asked to hand out business forms to the analysts, who catalogue them for later form analysis.

On the communicative interaction level, the communicative events that appear in the CED need to be described in detail. Requirements associated to an event are structured by means of an Event Specification Template. The template is composed by a header and three categories of requirements: contact, communicational content and reac-

tion requirements. These categories are related to phatic, referential and cognitive communication functions, respectively.

The header contains general information about the communicative event; that is, the event identifier, its name, a narrative description and, optionally, an explanatory diagram. The event identifier and name come from the CED; event identification needs to be kept consistent throughout the entire analysis and design specification in order to enhance requirements traceability. Since requirements specifications is meant, first of all, to facilitate problem understanding, a narrative description of the event is strongly advised. Also, whenever the event is complex, an explanatory diagram illustrating its associated flow of tasks shall be included. Contact requirements are related to the conditions that are necessary in order to establish communication.

Communicational content requirements specify the message conveyed in an event and related restrictions. With regard to the message, both metalinguistic aspects (e.g. message field structure, optionality of fields) and linguistic aspects (e.g. field domains, example values) need to be specified. Communication Analysis proposes a message modelling technique. Message Structures (MS) is a modelling technique that is based in structured text and allows specifying the message associated to a communicative event[27]. The structure of message fields lies vertically and many other details of the fields can be arranged horizontally; e.g. the information acquisition operation, the field domain, the link with the business object, an example value provided by users.

A communicative event can not be fully understood until its MS is defined in detail. Specifying with precision an event MS forces and helps analysts and users to appropriately mark the event boundary and meaning. Table 1 shows a Messages Structure grammar.

```
message structure
= structure name, '=', initial substructure;
initial substructure
= aggregation substructure | iteration substructure;
aggregation substructure
= '<', substructure list, '>';
iteration substructure
= '{', substructure list, '}';
specialisation substructure
= '[' , substructure list, { '|', substructure list }, ']';
substructure list
```

<pre> = substructure, { '+', substructure }; complex substructure = aggregation substructure iteration substructure specialisation substructure; substructure = substructure name, '=', complex substructure field; </pre>
--

Table 1. EBNF grammar of Message Structures

Table 2 shows an example of a message structure.

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

Table 2. Example of a message structure

Reaction requirements describe how the IS reacts to the communicative event occurrence. Typically, the IS stores new knowledge, extracts all the necessary conclusions that can be inferred from new knowledge, and makes new knowledge and conclusions available to the corresponding actors. Therefore, this category of requirements includes business objects being registered and outgoing communicative interactions being generated by the event, among other requirements.

2.5 OO-Method

OO-Method is an approach for automatic software generation on the basis of OO conceptual modelling. It is supported by the *OLIVANOVA* tool and can decrease development time and increase productivity. Conceptual modelling with OO-Method is independent from the target technological platform (e.g., Java or .Net).

OO-Method consists of the following conceptual models:

Object model: this model specifies the structure and static relationships between the classes of a software system by means of a graphical diagram that can be considered equivalent to UML class diagram; it includes classes, their attributes and methods, and the relationships between the classes.

Dynamic model: this model specifies the dynamic and behavioural side of the classes of the object model by means of graphical diagrams that can be considered equivalent to UML state transitions diagrams; the valid lifecycles of the classes are represented in this model, as well as the possible interactions between the objects (i.e., instances of the classes).

Functional model: this model specifies the semantics of the change of an object state as a result of method execution (e.g., a change in the number of kilometres of a car) by means of a declarative textual specification.

Presentation model: this model specifies the characteristics of the user interface of a software system and how the users will interact with the system; the model is created by means of a pattern-based graphical model through 3 levels of detail, from more general to more specific characteristics.

Applications generated from conceptual modelling with OO-Method have three-layer architecture. The presentation layer contains the software components responsible for presenting users the application interface to interact with a software system. The application layer provides services that implement the functionality of an application. The persistence layer provides services to store and obtain the pieces of data necessary for execution of an application. The software process of OO-Method consists of two stages. First, system analysts (mod-

ellers) create a conceptual schema, which corresponds to a representation of the problem space (i.e., the application domain). A UML-based notation and textual specifications are used. Second, the code of an application is generated on the basis of the Execution Model of *OLIVANOVA*, which corresponds to a representation of the solution space and can be targeted at different technologies. When comparing OO-Method with other approaches for software modelling and development, it deals with the static (data-oriented) and the dynamic (behaviour-oriented) views of an IS. Both views are necessary for complete IS modelling and development. In addition, it relies on an underlying formal model, integrates formal and semi-formal techniques, and (in conjunction with *OLIVANOVA*) allows generation of complete and ready-for-running applications by precisely specifying an IS.

In relation to MDA (Model Driven Architecture; [28]), a detailed description of its correspondence with OO-Method can be found in [29]. The main points are that: 1) an OO-Method conceptual schema corresponds to a Platform-Independent-Model; 2) the execution model corresponds to a Platform-Specific-Model, and; 3) the code generated corresponds to an Implementation Model.

2.6 Analysis and discussion

This chapter intends to present some previous experience about method integration. The works presented in this section had relevance during the development of this thesis. Lessons learned and advices of the works presented in this section were taken into account.

We do not include a state of the art with information about transformation rules definition and ontological alignment, because these topics are out of the scope of this literature review. For more information about these topics please see [11].

About technological decisions, *MOSKitt* have been an example about how to implement the techniques in Eclipse.

The review about Communication Analysis and OO-Method is important to carry out the integration process. This chapter present

an outline of these methods, for more information see the references presented in each one.

3 MDD approach: a metamodel to support the techniques of Communicative Event Diagrams and Message Structures

The development of Information Systems demands requirements methods to establish the needs of the organisation. The paradigm of the Model Driven Development (MDD) can provide to the requirements models some advantages, as to derive conceptual models from it. These conceptual models will be used to generate software in an automatic way. This chapter presents the strategy in order to specify the metamodels of the requirements techniques of the Communication Analysis method. In addition, this chapter presents the metamodel validation as an interesting way to improve and to

correct some mistakes in the metamodel specification. Finally an analysis and discussion is presented to conclude and to distinguish lessons learned.

3.1 Initial state

Communication Analysis proposes a requirements structure that low a stepwise refinements approach to Information System description. The method proposes a static and dynamic perception of reality. The method refers a process level, which refers to business process description from a communicational perspective; the aim is to discover communicative interactions between the IS and its environment and to describe them taking into account their dynamic and static aspects; that is, creating the Communicative Event Diagram (CED) [6]. The flow of communicative events is specified by CED technique. Besides, the method provides a communicative interaction level, which refers to the detailed description of each communicative event.

Communicate Event Diagrams and Message Structures are two novel techniques to analyse the communicative interactions in an information system.

An overview of the Communication Analysis method and the techniques of Communicative Event Diagrams and Message Structures are presented at the primitives of this modelling technique are showed at Chapter 3.

Taking into account the CED primitives and the Message Structures primitives, we can refer two abstraction levels for the syntax:

Abstract syntax: The abstract syntax defines the concepts of language and their relationships. The abstract syntax can be represented by means of a metamodel [30].

Concrete syntax: The concrete syntax defines the graphical appearance of language. The concrete syntax can be represented by means of a textual languages, this mean that it defines how to form sentences. For a graphical language this means that it defines the graphical ap-

pearance of the language concepts and how they may be combined into a model [30].

Following these definitions, we propose the specification of a metamodel to represent the CED language definition and the Message Structures language definition. This abstract syntax definition will be presented at section 3.2.

A proposal for the concrete syntax will be presented at Chapter 5, and we will speak about other ways to represent the concrete syntax.

3.2 The metamodeling strategy

Model Driven Architecture™ or MDA is an approach to using models in software development; and MDA propose the well-known and long established idea of separating the specification of the operation of a system[31].

Although MDA is commonly applied to product software development, we find it interesting to apply this concept for developing methods. Therefore, we have built metamodels with different levels of abstraction.

In order to achieve this proposal, we have designed a MDD framework to develop and to support of Communication Analysis requirements models; this framework contains different stages and activities (for a complete explanation of this framework see the section 1.4).

These activities are taking into account several steps for the metamodel definition according to the MDA guide; thus is considered the following kinds of metamodels:

The requirements for the system are modelled in a computation independent model (**CIM**). The CIM describes the situation in which the system will be used. The CIM is a model of a system that shows the system in the environment in which it will operate, and thus it helps in presenting exactly what the system is expected to do. The CIM specification can be represented by means of a textual explanation, technical reports, etc.

The system is described by means a platform independent model (**PIM**), thus it helps to describe the system but does not show details of its use in a specific technological platform. A PIM might consist of enterprise descriptions and it can follow a particular architectural style (object oriented, aspect oriented, etc.).

The PIM can be transformed in a model that specifies how the system uses the chosen platform. Thus, this is a platform specific model (**PSM**). The PSM provide more details about the implementation. A PIM can be transformed to different PSM; this depends of the end platform.

Due to a metamodel is a model, the MDA approach can be applied to the methods development. This approach pretends to apply the MDA concepts in an orthogonal way over the models in a software development process. These metamodels are aligning to a method textual description or an ontology reference for the method.

We will focus over CIM models (requirements models in a software development process), then, the Communication Analysis requirements models are specified in a metamodel that represents a high level of abstraction of these.

In order to build the Communication Analysis requirements models, the textual description of the method is specified at Chapter 3, the PIM metamodel will be specified at section 3.3, and the PSM metamodel will be specify at section 3.4.

3.3 PIM metamodel specification

According to the framework described at section 1.4 and according to the metamodeling strategy previously explained, the stage 1 should contain a PIM metamodel specification [32].

The phase 1 of the stage 1(See Figure 7) is composed of two activities: The first activity is the ontological analysis of Communication Analysis method. In this activity is studied the conceptual framework about system information then, it is possible to carry out a conceptual alignment of Communication Analysis method (to distinguish the

principal concepts and primitives). Some examples about conceptual framework and ontological analysis are available at[10].

España [33] proposes the ontological alignment for the Communication Analysis Method.

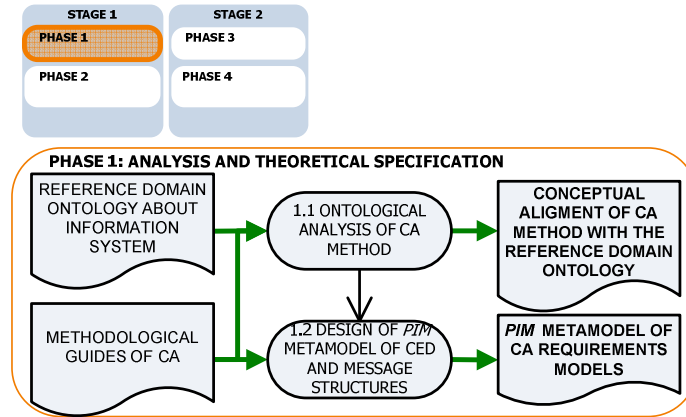


Figure 7. Phase 1 explanation

The second activity is about design of PIM metamodel. The PIM metamodel contains a set of elements (metaclasses) and relationships that represents concepts of the method. Each metaclass and relationship corresponds to a concept of the ontology (for more details see España 2011 [33]). The metamodel lets to build CED and to build message structures. The PIM metamodel is designed following the UML class diagram [34]. We use Microsoft Visio [35] to represent the metamodel in a graphical way. The PIM metamodel is the principal result of the phase 1. This metamodel is showed at Figure 8.

The elements, relationships, cardinalities and roles intend to represent the semantic of the Communication Analysis requirements models. Each modelling decision was taken according to the ontological analysis of the method. For this reason, each element of the metamodel corresponds with one element of the ontology. For more details see [33].

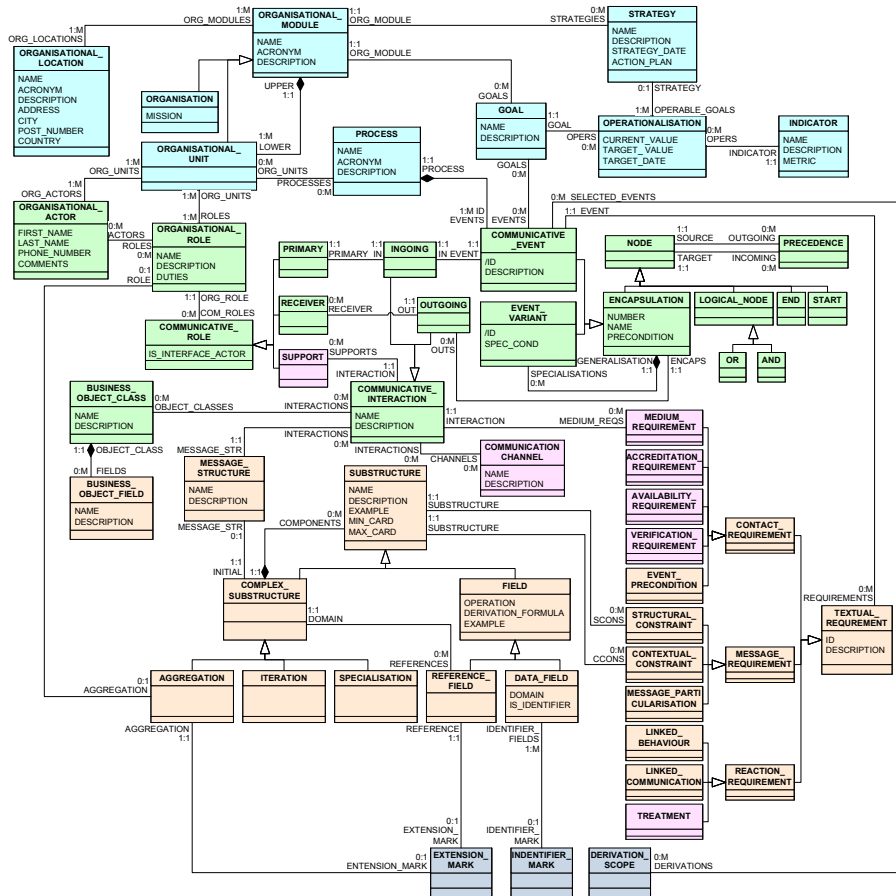


Figure 8. PIM metamodel for communicative event diagrams

3.4 PSM metamodel specification

According to the metamodeling strategy explained at section 3.2, the stage 1 should contain a PIM and a PSM metamodel specification.

The phase 2 of the stage 1 (See Figure 9) is composed of three activities: The first activity is the design of the PSM metamodel. In this activity is used the PIM metamodel obtained in the phase 1 (See section

3.3). The PSM design uses also the Ecore specification [36] and the UML 2.1 tools of Eclipse Modelling Framework (EMF) [36] [37]. The PSM metamodel is the principal result of the phase 2. The second and third activities will be explained at Chapter 5.

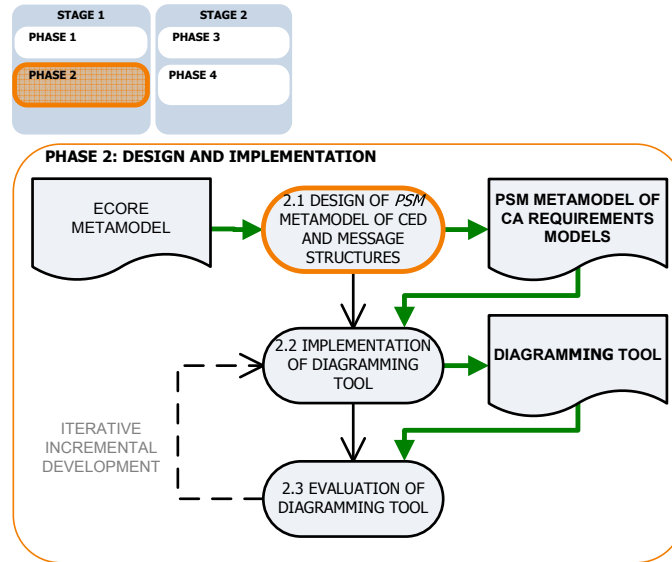


Figure 9. Phase 2 explanation

The PSM metamodel contains a set of elements (metaclasses), which are platform-oriented. This metamodel is an extension of the PIM metamodel. The PSM intends to complement the PIM metamodel with the necessary elements in order to represent the concrete syntax of the requirements models besides, the PSM metamodel includes concepts in order to improve the expressiveness of the concrete syntax. A modelling tool is intended, for this reason some graphical attributes, metaclasses and relationships are added into the PSM metamodel. To achieve this, data types, relationships, roles, metaclasses, attributes etc. were created, and the elements presented at the PIM metamodel were taken into account. This metamodel is showed at Figure 10.

3.4.1 PSM metamodel elements

This section describes (in alphabetical order) each metaclass of the PSM metamodel.

For each metaclass, a definition and a short description explains the context and meaning of each one. A diagram is included for a better understanding of the metaclass context in the metamodel. The attributes of the metaclass are listed, including for each one name and semantics. The constraints of the metaclass are listed, including a short description. A graphical primitive is explained to show the graphical notation that corresponds to each metaclass. Finally, the relationships that the metaclass is involved in are listed from the metaclass perspective, including for each one the name of the relationship if there is one, the role that the class being described plays in said relationship if there is one, the target class to which the metaclass is associated, and its semantics.

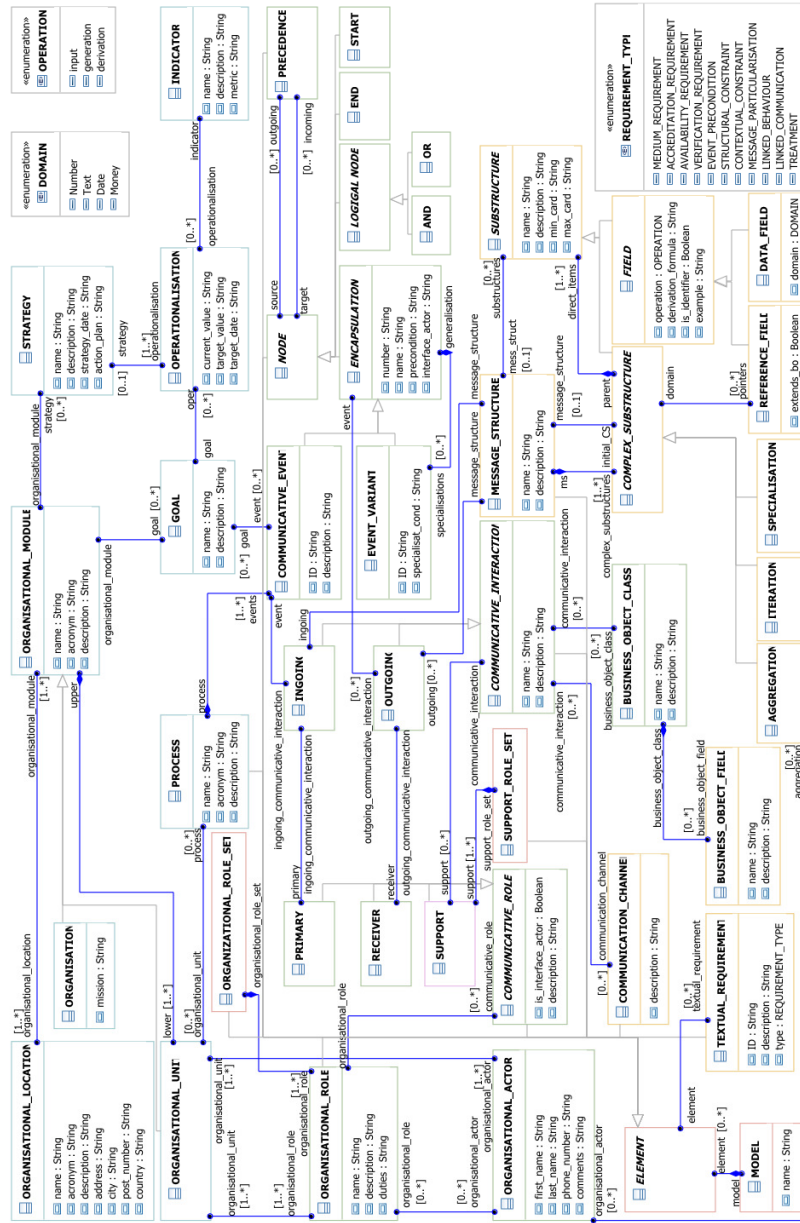


Figure 10. PSM metamodel for communicative event diagrams

List of metaclasses:

3.4.1.1	AGGREGATION metaclass	3.4.1.26	ORGANISATION metaclass
3.4.1.2	AND metaclass	3.4.1.27	ORGANIZATIONAL_ACTOR metaclass
3.4.1.3	BUSINESS_OBJECT_CLASS metaclass	3.4.1.28	ORGANISATIONAL_LOCATION metaclass
3.4.1.4	BUSINESS_OBJECT_FIELD metaclass	3.4.1.29	ORGANISATIONAL_MODULE metaclass
3.4.1.5	COMMUNICATION_CHANNEL metaclass	3.4.1.30	ORGANISATIONAL_ROLE metaclass
3.4.1.6	COMMUNICATIVE_EVENT metaclass	3.4.1.31	ORGANISATIONAL_ROLE_SET metaclass
3.4.1.7	COMMUNICATIVE INTERACTION metaclass	3.4.1.32	ORGANISATIONAL_UNIT metaclass
3.4.1.8	COMMUNICATIVE_ROLE Metaclass	3.4.1.33	OUTGOING metaclass
3.4.1.9	COMPLEX_SUBSTRUCTURE metaclass	3.4.1.34	PRECEDENCE Metaclass
3.4.1.10	DATA_FIELD metaclass	3.4.1.35	PRIMARY metaclass
3.4.1.11	ELEMENT metaclass	3.4.1.36	PROCESS Metaclass
3.4.1.12	ENCAPSULATION metaclass	3.4.1.37	RECEIVER Metaclass
3.4.1.13	END metaclass	3.4.1.38	REFERENCE_FIELD Metaclass
3.4.1.14	EVENT_VARIANT metaclass	3.4.1.39	SPECIALISATION metaclass
3.4.1.15	FIELD metaclass	3.4.1.40	START Metaclass
3.4.1.16	GOAL metaclass	3.4.1.41	STRATEGY metaclass
3.4.1.17	INDICATOR metaclass	3.4.1.42	SUBSTRUCTURE Metaclass
3.4.1.18	INGOING metaclass	3.4.1.43	SUPPORT Metaclass
3.4.1.19	ITERATION metaclass	3.4.1.44	SUPPORT_ROLE_SET metaclass
3.4.1.20	LOGICAL_NODE metaclass	3.4.1.45	TEXTUAL_REQUIREMENT metaclass
3.4.1.21	MESSAGE_STRUCTURE metaclass	3.4.1.46	DOMAIN enumeration
3.4.1.22	MODEL metaclass	3.4.1.47	OPERATION enumeration
3.4.1.23	NODE metaclass	3.4.1.48	REQUIREMENT_TYPE enumeration
3.4.1.24	OPERATIONALISATION metaclass		
3.4.1.25	OR metaclass		

3.4.1.1 AGGREGATION metaclass

It specifies a composition of several substructures in a way that they remain grouped as a whole. It is represented by angle brackets < >.

This metaclass specialises from Complex substructure

Example

For instance, **LINE**=<Product+Price+Quantity> specifies that an order line consists of information about a product, its price and the quantity that the client requests.

Metamodel view of AGGREGATION metaclass

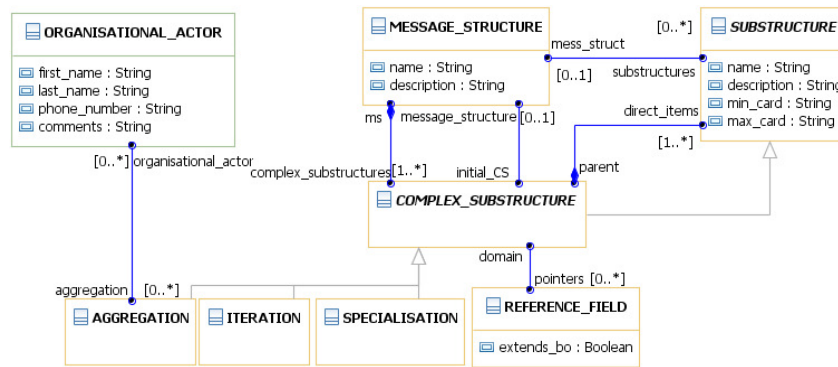


Figure 11. Portion of the metamodel including the metaclass
AGGREGATION

Attributes

NAME	TYPE	DESCRIPTION
name	String	[Inherited from substructure] The name of the aggregation.
description	String	[Inherited from substructure]The description or the aggregation in natural language.
min_card	String	[Inherited from substructure] The minimum cardinality of the aggregation.
max_card	String	[Inherited from substructure] The maximum cardinality of the aggregation.

Table 3. Attributes of the AGGREGATION metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
organisa- tional_actor	ORGANISATIONAL _ACTOR	An aggregation is related to several or- ganisational actors.
direct_items	SUBSTRUCTURE	[Inherited from complex substructure] An aggregation is composed by several sub- structures.
message_structure	MESSAGE_STRUC TURE	[Inherited from complex substructure] An aggregation can be part of one message structure.
ms	MESSAGE_STRUC TURE	[Inherited from complex substructure] An aggregation is contained into one mes- sage structure.
parent	COMPLEX_SUBST UCTURE	[Inherited from substructure] An aggrega- tion can to be a parent complex substuc- ture.
mess_struct	MESSAGE_STRUC TURE	[Inherited from substructure] An aggrega- tion can be related to one message struc- ture.
pointers	REFERENCE FIELD	[Inherited from complex substru] An ag- gregation can to have several pointers from existing reference field.

Table 4 Relationships of the AGGREGATION metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.2 AND metaclass

The *and-fork* and the *and-join* are implicitly represented by two or more precedence relations departing from or arriving to a communicative event, respectively; however, they can be explicitly drawn if needed to express complex logical expressions.

This metaclass specialises from Logical node

Example

C occurs after A and B occur (See Figure 12)

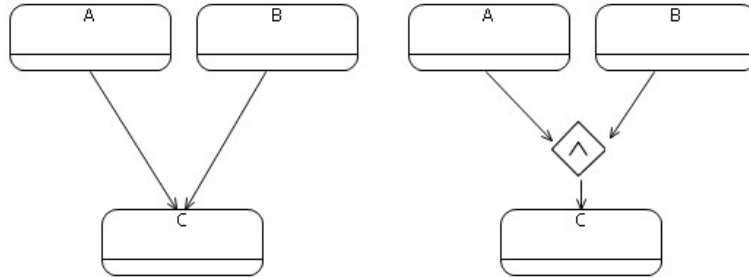


Figure 12. Example of the different AND cases

The Figure 12 shows two ways that represents the AND case. These ways can be used to represent the AND in a CED.

Metamodel view of AND metaclass

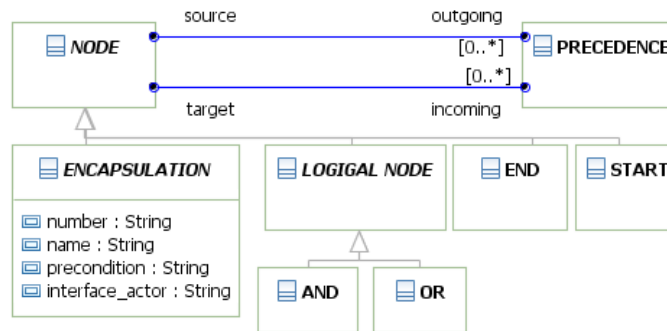


Figure 13. Portion of the metamodel including the metaclass AND

Attributes

This metaclass does not have attributes of its own.

Associations

ROLE	PARTICIPANT	DESCRIPTION
outgoing	PRECEDENCE	[Inherited from Logical node] An And can to have many outgoing precedence from it (See C1).
incoming	PRECEDENCE	[Inherited from Logical node] An And can to have many incoming precedences to it (See C1).

Constraints

C1. For each instance of AND metaclass, is necessary that two or more PRECEDENCE arrive to it, and only one PRECEDENCE goes out from it.

Graphical primitive

Figure 14 shows the graphical representation to use the AND metaclass in a CED. Remember that the semantics of the AND metaclass can be represented in two ways (see Figure 12).



Figure 14 Graphical primitive of AND metaclass

3.4.1.3 BUSINESS_OBJECT_CLASS metaclass

A business object is a composite thing of the system, which an organisational system is interested in acquiring knowledge about it. This metaclass represents a type of business objects.

Example

A class of business objects is a set of business objects of the same type, e.g. Truck.

Metamodel view of BUSINESS_OBJECT_CLASS metaclass

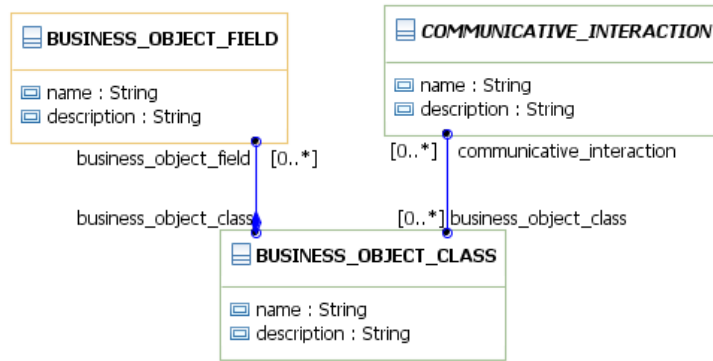


Figure 15. Portion of the metamodel including the metaclass BUSINESS_OBJECT_CLASS

Attributes

NAME	TYPE	DESCRIPTION
name	String	The name of business object class. For example, "TRUCK".
description	String	The description of the business object in natural language.

Table 5. Attributes of the BUSINESS_OBJECT_CLASS metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
business_object_field	BUSINESS_OBJECT_FIELD	A business object class can to be related several business object fields.
communicative_interaction	COMMUNICATIVE_INTERACTION	A business object can to be related communicative interactions that are part of the organisational process.

Table 6. Associations of the BUSINESS_OBJECT_CLASS metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.4 BUSINESS_OBJECT_FIELD metaclass

A business object field is a business object that is part of a business object class.

Metamodel view of BUSINESS_OBJECT_FIELD metaclass

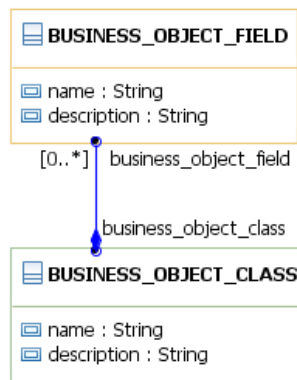


Figure 16. Portion of the metamodel including the metaclass BUSINESS_OBJECT_FIELD

Attributes

NAME	TYPE	DESCRIPTION
name	String	The name of the business object field.
description	String	The description of the business object field in natural language.

Table 7. Attributes of the BUSINESS_OBJECT_FIELD metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
business_object_class	BUSINESS_OBJECT_CLASS	A business object field can to be related with one business object class.

Table 8. Relationships of the BUSINESS_OBJECT_FIELD metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.5 COMMUNICATION_CHANNEL metaclass

We propose the COMMUNICATION_CHANNEL metaclass in order to represent the specific communication medium of a communicative interaction. This medium is used by an ORGANISATIONAL_ACTOR, which communicates an event occurrence.

Examples

The organisational actors can to use a set of devices or mediums in order to communicate information. For example: fax, telephone, email, in person, etc.

Metamodel view for COMMUNICATION_CHANNEL metaclass

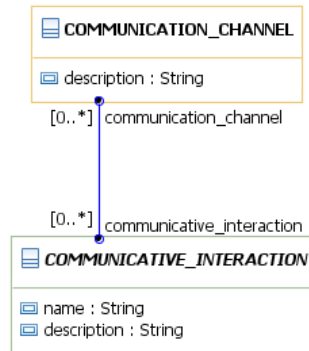


Figure 17. Portion of the metamodel including the metaclass COMMUNICATION_CHANNEL

Attributes

NAME	TYPE	DESCRIPTION
description	String	The description of the communication channel in natural language.

Table 9. Attributes of the COMMUNICATION_CHANNEL metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
communicative_interaction	COMMUNICATIVE_INTERACTION	A communication channel can to be related with many communicative interactions that are part of the organisational process.

Table 10. Relationships of the COMMUNICATION_CHANNEL metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.6 COMMUNICATIVE_EVENT metaclass

A communicative event describes the occurrence of an ingoing communicative interaction and the corresponding reaction of the organisational system. A communicative event describes the information about this event occurrence.

This metaclass specialises from Encapsulation

Examples

Actor B communicates information to Information System; this information is related to the event A. Actor C consults information from Information System.

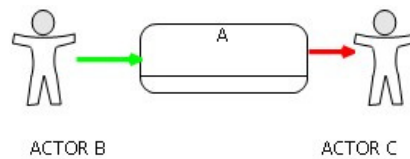


Figure 18. Example of use of COMMUNICATIVE_EVENT a in a CED

Metamodel view for COMMUNICATIVE_EVENT metaclass

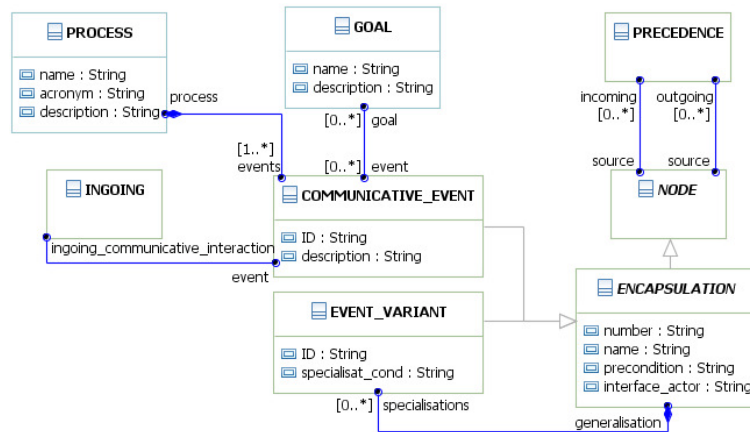


Figure 19. Portion of the metamodel including the metaclass COMMUNICATIVE_EVENT

Attributes

NAME	TYPE	DESCRIPTION
ID	String	The unique identification for each communicative event. It is advisable that the ID corresponds to the union of the value of attribute number and the process acronym.
description	String	The description of the communicative event in natural language.
number	String	[Inherited from Encapsulation] The number of the communicative event.
name	String	[Inherited from Encapsulation] Name that describes the communicative event
precondition	String	[Inherited from Encapsulation] Description of the precondition of the communicative event in natural language
interface_actor	String	[Inherited from Encapsulation] List with the organisational roles that play the role of interface actors.

Table 11. Attributes of the COMMUNICATIVE_EVENT metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
goal	GOAL	The communicative events related to a process intend to fulfil one or several organisational goals.
process	PROCESS	The communicative events are part of one organisational process.
ingo- ing_communicativ e_interaction	INGOING	The occurrence of one communicative event is related to one ingoing communicative interaction.
specialisations	EVENT_VARIANT	[Inherited from Encapsulation] A communicative event can to be compound by several specialisations or event variants.
outgo- ing_communicativ e_interaction	OUTGOING	[Inherited from Encapsulation] A communicative event can to be related by several outgoing communicative interactions.
outgoing	PRECEDENCE	[Inherited from Node] A communicative event can to have many outgoing precedences from it.
incoming	PRECEDENCE	[Inherited from Node] A communicative event can to have many incoming precedences to it.

Table 12. Relationships of the COMMUNICATIVE_EVENT metaclass

Constraints

This metaclass does not have constraints

Graphical primitive

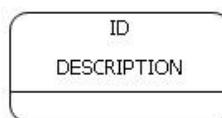


Figure 20. Graphical primitive of the COMMUNICATIVE_EVENT metaclass

3.4.1.7 COMMUNICATIVE INTERACTION metaclass

A communicative interaction is an exchange of messages, these messages contain some data about the subject system. This interchange of message is carried out by at least two organizational actors.

This is an abstract class, which is specialized into Ingoing and Outgoing (see pages 89 and 113 for more details)

Metamodel view

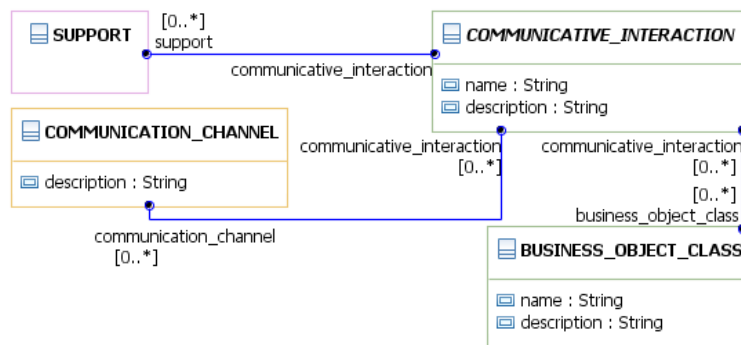


Figure 21. Portion of the metamodel including the metaclass COMMUNICATIVE_INTERACTION

Attributes

NAME	TYPE	DESCRIPTION
name	String	The name of the communicative interaction.
description	String	The description of the communicative interaction in natural language.

Table 13. Attributes of the COMMUNICATIVE_INTERACTION metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
support	SUPPORT	A communicative interaction can

		to be supported by several support roles.
business_object_class	BUSINESS_OBJECT_CLASS	A communicative interaction can be related with several business object class.
communication_channel	COMMUNICATION_CHANNEL	A communicative interaction can be carried out by means of several mediums or communication channel.

Table 14. Relationships of the COMMUNICATIVE_INTERACTION metaclass

Constraints

This metaclass does not have constraints

Graphical primitive

This metaclass does not have graphical primitive

3.4.1.8 COMMUNICATIVE_ROLE Metaclass

Different business actors can to interact with many communicative events, and they may to performance a specific communicative role. This communicative role can be of three types: primary, receiver or support. When a business actor is responsible to communicate the new meaningful information to the information system, this actor performs a primary role. When a business actor requests the retrieval of data from de information system memory, this actor performs a receiver role. When a business actor supports message transfers in some way (ingoing to the communicative event or outgoing from the communicative event), this actor performs a support role.

This is an abstract metaclass, which is specialized into Primary, Receiver and Support (see pages 117, 120 and 131 for more details).

Metamodel view for COMMUNICATIVE_ROLE metaclass

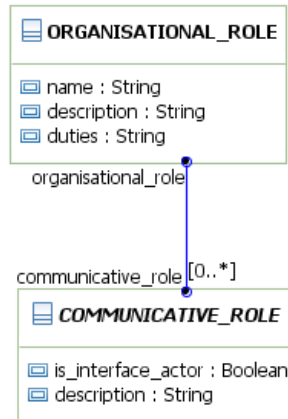


Figure 22. Portion of the metamodel including the metaclass COMMUNICATIVE_ROLE

Attributes

NAME	TYPE	DESCRIPTION
description	String	The description of the communicative role in natural language.
is_interface_actor	Boolean	Indicates whether the communicative role is an interface actor, i.e. whether it plays the role of the interface actor for a communicative event.

Table 15. Attributes of the COMMUNICATIVE_ROLE metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
organisational_role	ORGANISATIONAL_ROLE	A communicative role can be played by one organisational role.

Table 16. Relationships of the COMMUNICATIVE_ROLE metaclass

Constraints

This metaclass has no constraints.

Graphical primitive

This metaclass has no graphical primitive.

3.4.1.9 COMPLEX_SUBSTRUCTURE metaclass

This metaclass is a part of the Messages Structures method [38]. An actor gives information to the IS using the Message Structure guidelines. A Message Structure is composed of substructures.

A complex substructure is a substructure (See section 3.4.1.42 for more details) that contains an internal composition. This internal composition is of different types. The types can take values as aggregation, iteration or specialisation.

This metaclass specialises from Substructure.

Examples

- **AGGREGATION**, it specifies a composition of several substructures in a way that they are grouped as a whole. It is represented by angle brackets `< >`. For instance, `A<B+C+D>` specifies that an order A consists of info B, C and D.
- **ITERATION**, it specifies a set or repetition of the substructures it contains. It is represented by curly brackets `{ }`. For instance, an M is an iteration substructure whit aggregation substructure inside. `M={A<B+C+D>}`.
- **SPECIALISATION**, it specifies one or more variants (i.e. structural alternatives). For instance `MSG=<a+b+OPTION=[c|d]+e>`, both `<a+b+c+e>` and `<a+b+d+e>` are valid messages; see [38] for a more illustrative example.

Metamodel view for COMPLEX_SUBSTRUCTURE metaclass

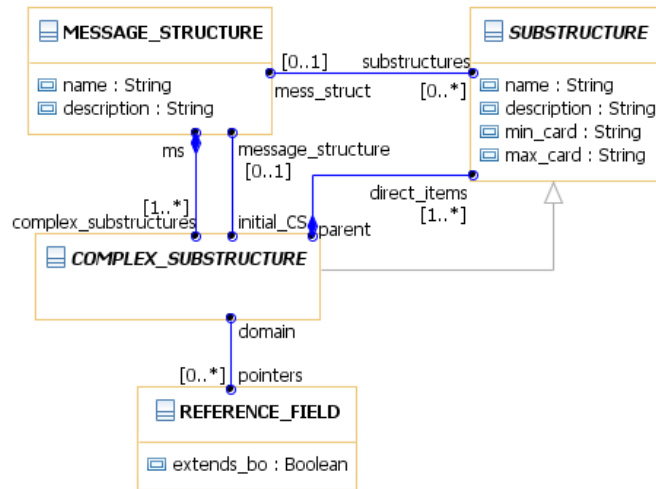


Figure 23. Portion of the metamodel including the metaclass COMPLEX_SUBSTRUCTURE

Attributes

NAME	TYPE	DESCRIPTION
name	String	[Inherited from Substructure] The name of the complex substructure
description	String	[Inherited from Substructure] The description of the complex substructure in natural language
min_card	String	[Inherited from Substructure] Indicates a constraint about minimum cardinality of the complex substructure.
max_card	String	[Inherited from Substructure] Indicates a constraint about maximum cardinality of the complex substructure.

Figure 24. Attributes of the COMPLEX_SUBSTRUCTURE metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
pointers	REFERENCE_FIELD	A complex substructure can to be related to many reference fields.
direct_items	SUBSTRUCTURE	A complex substructure has several direct items.
mess_struct	MESSAGE_STRUCTURE	A complex substructure can to be related a message structure.
message_structure	MESSAGE_STRUCTURE	A complex substructure can to be related one message structure.
ms	MESSAGE_STRUCTURE	A complex substructure is part of one message structure.
parent	COMPLEX_SUBSTRUCTURE	A complex substructure has a parent that is a complex substructure.

Table 17. Relationships of the COMPLEX_SUBSTRUCTURE metaclass

Constraints

This metaclass has no constraints.

Graphical primitive

This metaclass has no graphical primitive.

3.4.1.10 DATA_FIELD metaclass

The Data field metaclass is a part of the explanation of Message Structures [38]. A message structure is composed of substructures. A data field is a substructure. A data field represents a piece of data with basic domain (numbers, text, etc.).

This metaclass specialises from Field.

Examples

The substructure $A \leftarrow B+C+D$ have the data fields B, C, D. These data fields can be Text, numeric or other basic data type.

Metamodel view for DATA_FIELD metaclass

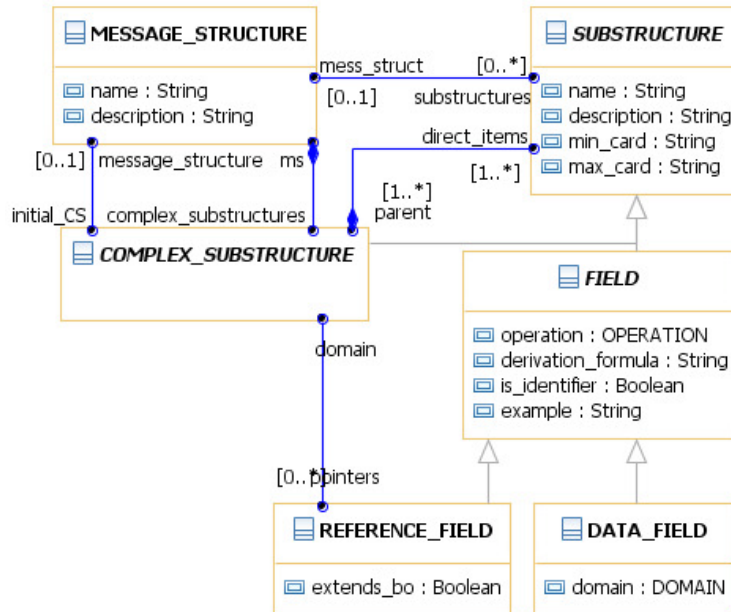


Figure 25. Portion of the metamodel including the metaclass DATA_FIELD

Attributes

NAME	TYPE	DESCRIPTION
domain	DOMAIN	The different domains of the data field are indicated in the metaclass of type enumeration (see metaclass DOMAIN). This domains can be text, numeric, etc.
is_identifier	Boolean	[Inherited from Field] It is a Boolean value. It indicates if a data field is an identifier field of a substructure. This attribute is a mark to support the model transformation.
operation	OPERATION	[Inherited from Field] The operation indicates the origin or the information that the

		field refers. Its values can be “input”, “output” or “derivation”.
derivation_formula	String	[Inherited from Field] if the attribute operation is of type “derivation”, the derivation formula indicates the formula in natural language or OCL.
name	String	[Inherited from Substructure] The name of the data field
description	String	[Inherited from Substructure] The description of the data field in natural language
example	String	[Inherited from Substructure] An example about the data field in natural language.
min_card	String	[Inherited from Substructure] Indicates a constraint about minimum cardinality of the data field.
max_card	String	[Inherited from Substructure] Indicates a constraint about maximum cardinality of the data field.

Table 18. Attributes of the DATA_FIELD metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
mess_struct	MESSAGE_STRUCTURE	A data field can to be related to several message structures.
parent	COMPLEX_SUBSTRUCTURE	A data field is associated to one complex substructure (named parent).

Table 19. Relationships of the DATA_FIELD metaclass

Constraints

This metaclass has no constraints.

Graphical primitive

This metaclass has no graphical primitive.

3.4.1.11 ELEMENT metaclass

The Element metaclass is an abstraction of the whole elements that are part of the communicative event diagram.

This is an abstract class, which is specialised into Communication channel, communicative role, communicative interaction, message structure, node, organisational role, organisational role set, precedence, process, support role set and textual requirements.

Example

A communicative event is an element of the communicative event diagram.

Metamodel view

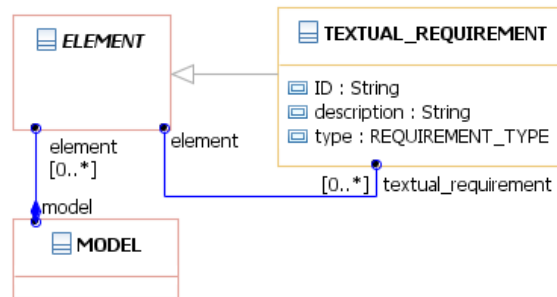


Figure 26. Portion of the metamodel including the metaclass ELEMENT

Attributes

This metaclass has not attributes of its own.

Associations

ROLE	PARTICIPANT	DESCRIPTION
model	MODEL	An element can to be contained in one model.
textual_requirement	TEXTUAL_REQUIREMENT	An element can to have assigned several textual requirements.

nt		
----	--	--

Table 20. Relationships of the ELEMENT metaclass

Constraints

This metaclass has not constraints.

Graphical primitive

This metaclass has not graphical primitive.

3.4.1.12 ENCAPSULATION metaclass

An encapsulation is an abstraction that generalises the communicative event and event variant concepts.

This is an abstract metaclass, which is specialised into Communicative event and Event variant.

Metamodel view for ENCAPSULATION metaclass

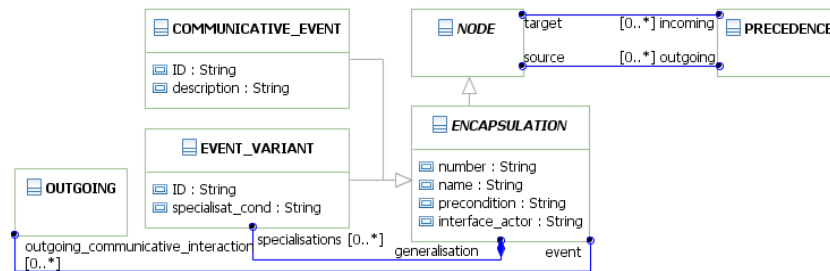


Figure 27. Portion of the metamodel including the metaclass
ENCAPSULATION

Attributes

NAME	TYPE	DESCRIPTION
------	------	-------------

number	String	The number of the encapsulation.
name	String	Name that describes the encapsulation
precondition	String	Description of the precondition of the encapsulation in natural language
interface_actor	String	List with the organisational roles that play the role of interface actors.

Table 21. Attributes of the ENCAPSULATION metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
specialisations	EVENT_VARIANT	An encapsulation can to be compound by several specialisations or event variants.
outgoing_communicative_interaction	OUTGOING	An encapsulation can to be related by several outgoing communicative interactions.
outgoing	PRECEDENCE	[Inherited from Node] An encapsulation can to have many outgoing precedences from it.
incoming	PRECEDENCE	[Inherited from Node] An encapsulation can to have many incoming precedences to it.

Table 22. Relationships of the ENCAPSULATION metaclass

Constraints

C2. The encapsulation number should be different for each encapsulation in a communicative event diagram.

Graphical primitive

This metaclass has no graphical primitive.

3.4.1.13 END metaclass

An end node represents that the communicative event to which it is connected do not have any successor events.

This metaclass specialises from Node.

Metamodel view for END metaclass

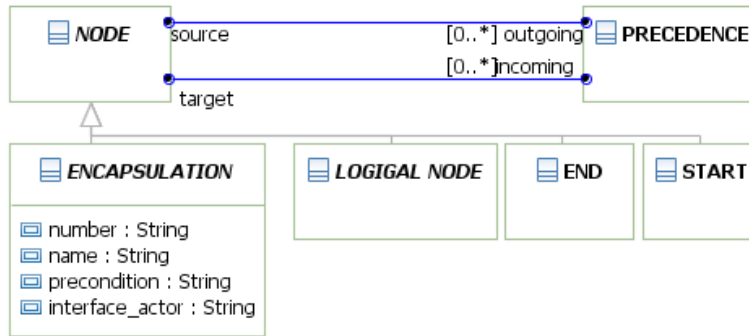


Figure 28. Portion of the metamodel including the metaclass END

Attributes

This metaclass has no attributes of its own.

Associations

ROLE	PARTICIPANT	DESCRIPTION
outgoing	PRECEDENCE	[Inherited from Node] An end can to have many outgoing precedences from it (see C3).
incoming	PRECEDENCE	[Inherited from Node] An end can to have many incoming precedences to it.

Table 23. Relationships of the END metaclass

Constraints

C3. An end cannot have any outgoing precedence.

Graphical primitive



Figure 29. Graphical primitive of the END metaclass

3.4.1.14 EVENT_VARIANT metaclass

An event variant is each alternative behaviour within a specialised communicative event. Each event variant has a corresponding specialisation condition, which is a well-formed formula that can refer to one or several fields of the message structure.

A communicative event can be specialised in different events. This specialisation represents complex communicative event. Through this specialisation is possible to show the all ways of an event occurrence in a particular process.

This metaclass specialises from Encapsulation.

Examples

The Figure 30 shows an example of an event specialization. The communicative event *SALE3* is specialized in the events *SALE3.1* and *SALE3.2*.

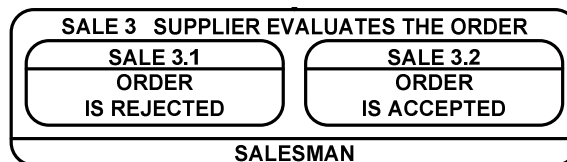


Figure 30. Example of EVENT_VARIANTS in a communicative event diagram

Metamodel view for EVENT_VARIANTS metaclass

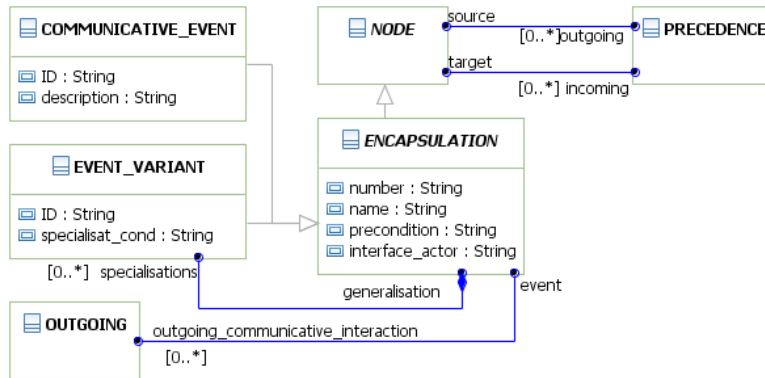


Figure 31. Portion of the metamodel including the metaclass EVENT_VARIANT

Attributes

NAME	TYPE	DESCRIPTION
ID	String	The unique identification for each communicative event. It is advisable that the ID corresponds to the union of the value of attribute number and the process acronym.
specialisat_cond	String	The description of the specialisation condition in natural language.
number	String	[Inherited from Encapsulation] The number of the event variant.
name	String	[Inherited from Encapsulation] Name that describes the event variant.
precondition	String	[Inherited from Encapsulation] Description of the precondition of the event variant in natural language
interface_actor	String	[Inherited from Encapsulation] List with the organisational roles that play the role of interface actors.

Table 24. Attributes of the EVENT_VARIANT metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
generalisation	ENCAPSULATION	An event variant can to be contained into one encapsulation, i.e., communicative event or event variant.
specialisations	EVENT_VARIANT	[Inherited from Encapsulation] An event variant can to be compound by several specialisations or event variants.
outgoing_communicative_interaction	OUTGOING	[Inherited from Encapsulation] An event variant can to be related by several outgoing communicative interactions.
outgoing	PRECEDENCE	[Inherited from Node] An event variant can to have many outgoing precedences from it.
incoming	PRECEDENCE	[Inherited from Node] An event variant can to have many incoming precedences to it.

Table 25. Relationship of the EVENT_VARIANTS metaclass

Constraints

C4. The number of the event variant should be a consecutive number of its corresponding communicative event. For example, if the communicative event that will be specialises is SALE 1, it corresponds to event variants should to have the value 1.1 and 1.2 in its corresponding numbers.

Graphical primitive

The graphical primitive is the like COMMUNICATIVE_EVENT graphical primitive (see Figure 20).

3.4.1.15 FIELD metaclass

A field represents a basic information element of the message structure; it is not composed of other elements. A message structure is

composed of substructures. A field is a substructure (for more details see [38]).

This is an abstract metaclass, which is specialised into Data field and Reference field.

Metamodel view for FIELD metaclass

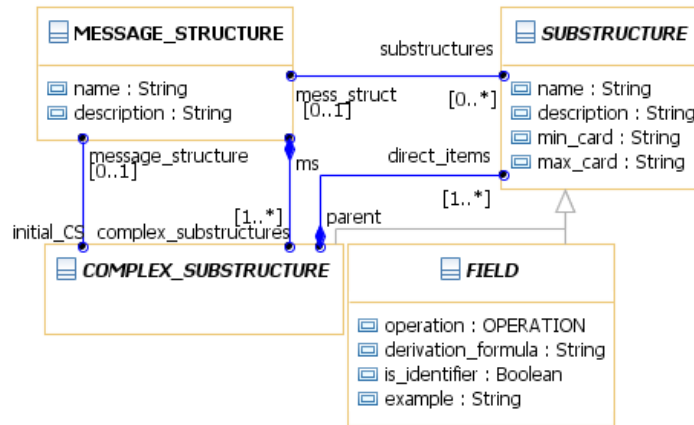


Figure 32. Portion of the metamodel including the metaclass FIELD

Attributes

NAME	TYPE	DESCRIPTION
operation	String	The operation indicates the origin or the information that the field refers. Its values can be “input”, “output” or “derivation”.
derivation_formula	String	If the attribute operation is of type “derivation”, this attribute indicates the derivation formula.
name	String	[Inherited from Substructure] The name of the field
description	String	[Inherited from Substructure] The description of the field in natural language
example	String	[Inherited from Substructure] An example about the field in natural language.
min_card	String	[Inherited from Substructure] Indicates a constraint about minimum cardinality of the

		field.
max_card	String	[Inherited from Substructure] Indicates a constraint about maximum cardinality of the field.
is_identifier	Boolean	Indicates whether the field is an identifier field or not.

Table 26. Attributes of the FIELD metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
mess_struct	MESSAGE_STRUCTURE	A reference field can to be associated to several message structures.
parent	COMPLEX_SUBSTRUCTURE	A reference field is associated to one complex substructure (named parent).

Table 27. Relationships of the FIELD metaclass

Constraints

This metaclass has no constraint.

Graphical primitive

This metaclass has no graphical primitive.

3.4.1.16 GOAL metaclass

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

Metamodel view

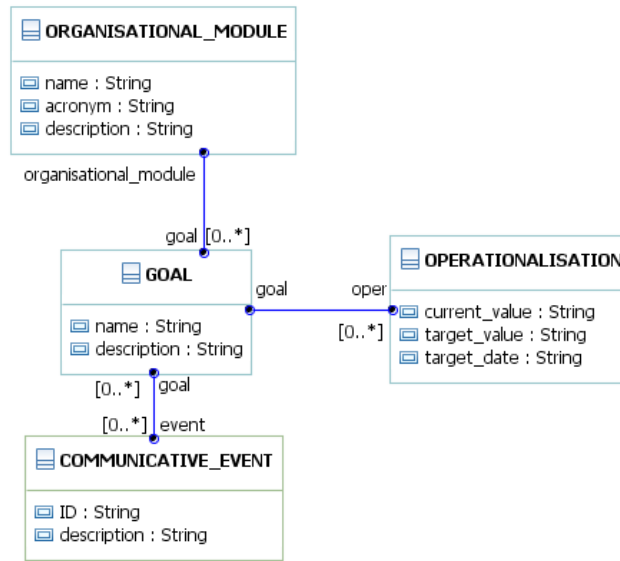


Figure 33. Portion of the metamodel including the metaclass GOAL

Attributes

NAME	TYPE	DESCRIPTION
name	String	The name of the goal in natural language.
description	String	The description of the goal in natural language.

Table 28. Attributes of the GOAL metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
organisational_module	ORGANISATIONAL_MODULE	A goal is related to one organisational module, which is specialised into organisation and organisational unit.
oper	OPERATIONALISATION	A goal is related with sev-

		eral operationalisations.
event	COMMUNICATIVE_EVENT	A goal is related with several communicative events.

Table 29. Relationships of the GOAL metaclass

Constraints

This metaclass does not have constraints

Graphical primitive

This metaclass does not have graphical primitive

3.4.1.17 INDICATOR metaclass

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

Example

For example, a stock Price, consumer confidence, time to market, etc.

Metamodel view

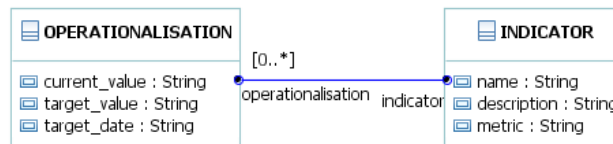


Figure 34. Portion of the metamodel including the metaclass INDICATOR

Attributes

NAME	TYPE	DESCRIPTION
name	String	The name of the indicator.

description	String	The description of the indicator in natural language.
metric	String	The metric related to the indicator in natural language.

Table 30. Attributes of the INDICATOR metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
operationalisation	OPERATIONALISATION	An indicator has several operationalisations.

Table 31. Relationships of the INDICATOR metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.18 INGOING metaclass

The ingoing metaclass represents the ingoing communicative interaction. This ingoing communicative interaction feeds the information system memory with new meaningful information. The information is provided by a communicative role that is played by a primary actor.

This class specialises from Communicative Interaction

Examples

For instance, the placement of an order by a client corresponds to an ingoing communicative interaction, because is new information for the information system.

Metamodel view for PRIMARY metaclass

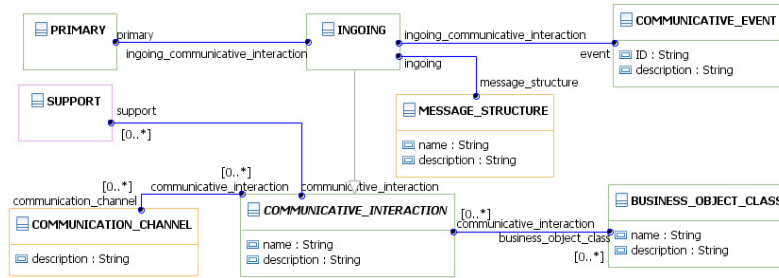


Figure 35. Portion of the metamodel including the metaclass INGOING

Attributes

NAME	TYPE	DESCRIPTION
name	String	[Inherited from Communicative interaction] The name of the communicative interaction.
description	String	[Inherited from Communicative interaction] The description of the communicative interaction in natural language.

Table 32. Attributes of the INGOING metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
event	COMMUNICATIVE_EVENT	An ingoing is related to one and only one communicative event.
primary	PRIMARY	An ingoing is a communicative interaction which communicative role is played by one primary actor.
message_structure	MESSAGE_STRUCTURE	An ingoing do have related one message structure.

support	SUPPORT	[Inherited from Communicative interaction] An ingoing can to be supported by several support roles.
business-object_class	BUSINESS_OBJECT_CLASS	[Inherited from Communicative interaction] An ingoing can to be related with several business object class.
communication_channel	COMMUNICATION_CHANNEL	[Inherited from Communicative interaction] An ingoing can to be carried out by means of several mediums or communication channel.

Table 33. Relationships of the INGOING metaclass

Constraints

This metaclass does not have constraints

Graphical primitive



Figure 36. Graphical primitive of the PRIMARY metaclass

3.4.1.19 ITERATION metaclass

An iteration substructure specifies a set or repetition of the substructures it contains. It is represented by curly brackets { }.

Example

For instance, an order can have several destinations and, for each destination, a set of order lines is defined. Both **DESTINATIONS** and **LINES** are iteration substructures. **LINES**={**LINE**=<Product+Price+Quantity>}

Metamodel view of ITERATION metaclass

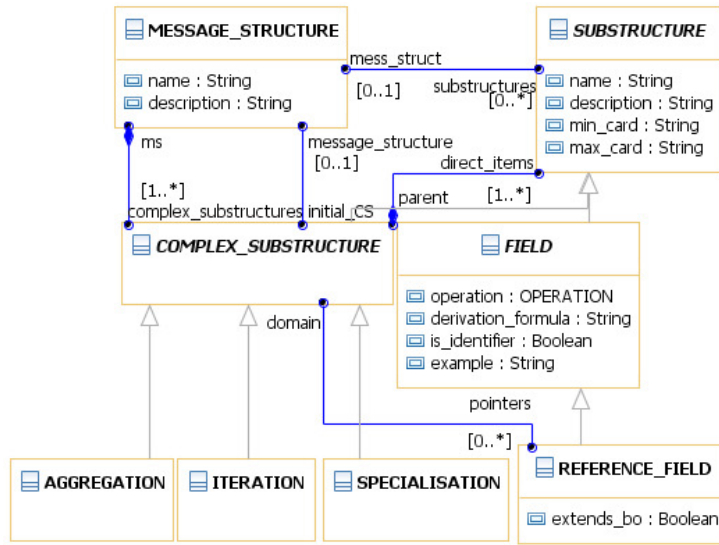


Figure 37. Portion of the metamodel including the metaclass ITERATION

Attributes

NAME	TYPE	DESCRIPTION
name	String	[Inherited from substructure] The name of the iteration.
description	String	[Inherited from substructure]The description or the iteration in natural language.
min_card	String	[Inherited from substructure] The minimum cardinality of the iteration.
max_card	String	[Inherited from substructure] The maximum cardinality of the iteration.

Table 34. Attributes of the ITERATION metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
------	-------------	-------------

direct_items	SUBSTRUCTURE	[Inherited from complex substructure] An iteration is composed by several substructures.
message_structure	MESSAGE_STRUCTURE	[Inherited from complex substructure] An iteration can be part of one message structure.
ms	MESSAGE_STRUCTURE	[Inherited from complex substructure] An iteration is contained into one message structure.
parent	COMPLEX_SUBSTRUCTURE	[Inherited from substructure] An iteration can to be a parent complex substructure.
mess_struct	MESSAGE_STRUCTURE	[Inherited from substructure] An iteration can be related to one message structure.
pointers	REFERENCE FIELD	[Inherited from complex substructure] An iteration can to have several pointers from existing reference field.

Table 35 Relationships of the ITERATION metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.20 LOGICAL_NODE metaclass

A logical node is an abstraction that generalizes the Or and And concepts.

This is an abstract metaclass, which is specialised into Or and And.

Metamodel view for LOGICAL_NODE metaclass

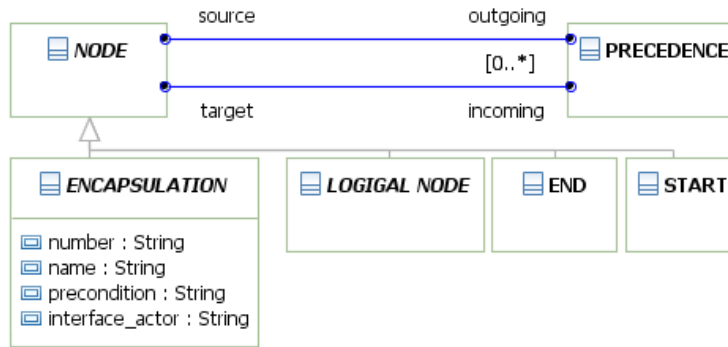


Figure 38. Portion of the metamodel including the metaclass LOGICAL_NODE

Attributes

This metaclass does not have attributes of its own.

Associations²

ROLE	PARTICIPANT	DESCRIPTION
outgoing	PRECEDENCE	[Inherited from Node] A logical node can to have many outgoing precedences from it.
incoming	PRECEDENCE	[Inherited from Node] A logical node can to have many incoming precedences to it.

Table 36. Relationships of the LOGICAL_NODE metaclass

Constraints

This metaclass does not have constraint.

² It is important taking into account the constraints associated to And and Or metaclasses.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.21 MESSAGE_STRUCTURE metaclass

A message structures is a model of the message that corresponds to a communicative interaction.

Example

To see examples about message structures and more details, please to see [39], and the technical report [27].

Metamodel view

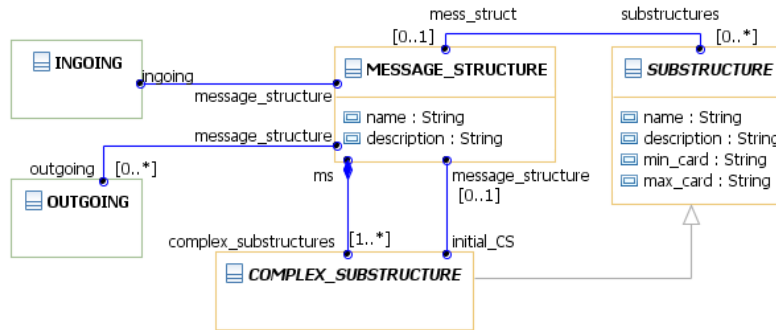


Figure 39. Portion of the metamodel including the metaclass MESSAGE_STRUCTURE

Attributes

NAME	TYPE	DESCRIPTION
name	String	The name of the message structure.
description	String	The description of the message structure in natural language.

Table 37. Attributes of the MESSAGE_STRUCTURE metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
outgoing	OUTGOING	A message structure can to be associated to several outgoing communicative interactions.
ingoing	INGOING	A message structure is related to one ingoing communicative interaction.
substructures	SUBSTRUCTURE	A message structure contains several substructures.
initial_CS	COMPLEX_SUBSTRUCTURE	A message structure has associated one initial complex substructure.
complex_substructure	COMPLEX_SUBSTRUCTURE	A message structure is related with several complex substructures.

Table 38. Relationships of the MESSAGE_STRUCTURE metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.22 MODEL metaclass

The model metaclass is the container of whole elements of the metamodel.

Metamodel view

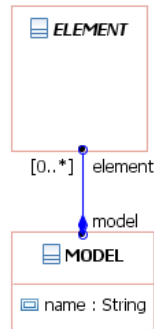


Figure 40. Portion of the metamodel including the metaclass MODEL

Attributes

NAME	SEMANTICS
NAME	Model name

Table 39. Attributes of the MODEL metaclass

Relationships

ROLE	PARTICIPANT	DESCRIPTION
element	ELEMENT	A model can to contain several elements.

Table 40. Relationships of the MODEL metaclass

Constraints

This metaclass has not constraints of its own.

Graphical primitive

This metaclass has not graphical primitive of its own.

3.4.1.23 NODE metaclass

A node is an abstraction in order to generalise concepts that will be associated through precedences relationships.

This is an abstract metaclass, which is specialised into Encapsulation, Logical node, End and Start.

Metamodel view for NODE metaclass

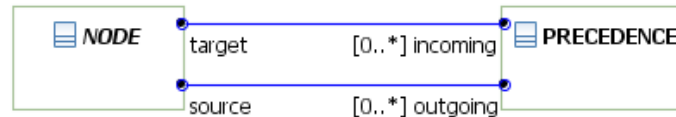


Figure 41. Portion of the metamodel including the metaclass NODE

Attributes

This metaclass does not have attributes of its own.

Relationships

ROLE	PARTICIPANT	DESCRIPTION
outgoing	PRECEDENCE	A node can to have several outgoing precedences.
incoming	PRECEDENCE	A node can to have several incoming precedences.

Table 41. Relationships of the NODE metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.24 OPERATIONALISATION metaclass

The metaclass operationalisation intends to represent the activities for achieving goals that corresponds with some indicators and follows some organizational strategies.

Metamodel view

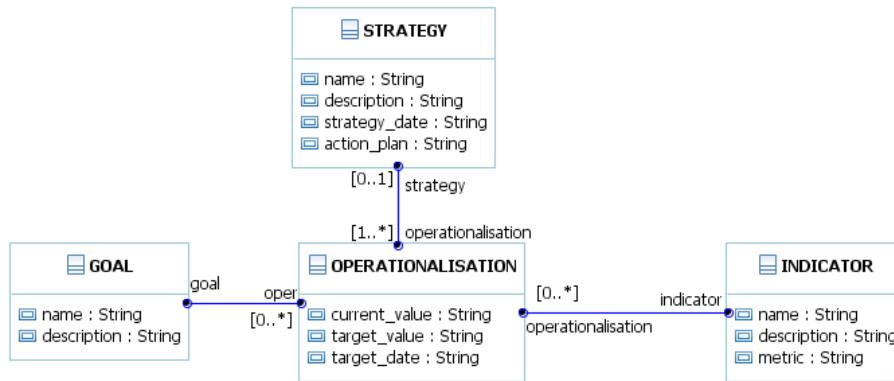


Figure 42. Portion of the metamodel including the metaclass OPERATIONALISATION

Attributes

NAME	TYPE	DESCRIPTION
current_value	String	The current value of the operation
target_value	String	The target value of the operation
target_date	String	The target date of the operation

Table 42. Attributes of the OPERATIONALISATION metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
strategy	STRATEGY	An operation is related to several organisational strategies.
indicator	STRATEGY	An operation is associated to one organisational indicator.

goal	GOAL	An operation is related to one organisational goal.
------	------	---

Table 43. Relationships of the OPERATIONALISATION metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.25 OR metaclass

The *or-merge* indicates that only one of the incoming precedence relations needs to hold. Note that the *or-branch* is implicit and corresponds to event specializations.

This metaclass specialises from Logical node.

Examples

Figure 43 shows an example of an OR case in a communicative event diagram. This example shows that is necessary that occur the communicative event **D** or **F** or **E** to occur the communicative event **A**.

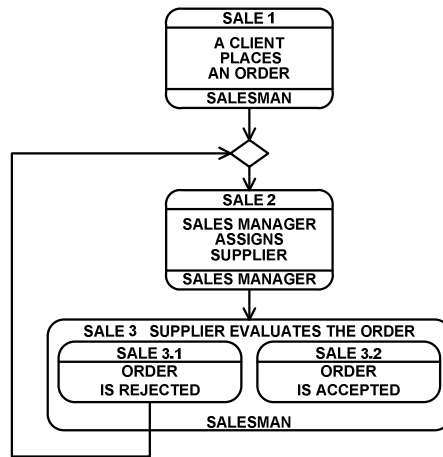


Figure 43. Example of OR in a CED

Metamodel view for OR metaclass

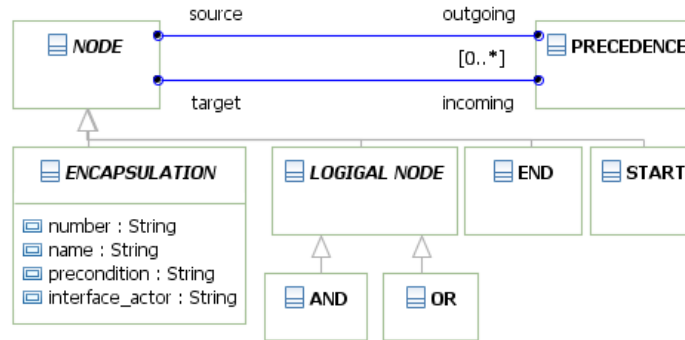


Figure 44. Portion of the metamodel including the metaclass OR

Attributes

This metaclass does not have attributes of its own.

Relationships

ROLE	TO METACLASS	SEMANTICS
outgoing	PRECEDENCE	[Inherited from Logical node] An Or can to have many outgoing precedences from it (See C5).

incoming	PRECEDENCE	[Inherited from Logical node] An Or can to have many incoming precedences to it (See C5).
----------	------------	---

Table 44. Relationships of the OR metaclass

Constraints

C5. For each or, it is necessary that two or more incoming precedences arrive to it; and only one precedence can to come out of an or.

Graphical primitive



Figure 45. Graphical primitive for the OR metaclass

3.4.1.26 ORGANISATION metaclass

An organisation represents the information of a subsystem of the enterprise.

This metaclass specialises from Organisational module.

Metamodel view

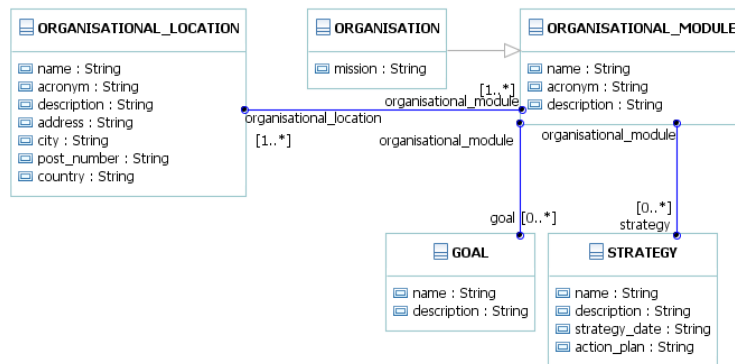


Figure 46. Portion of the metamodel including the metaclass ORGANISATION

Attributes

NAME	TYPE	DESCRIPTION
mission	String	The mission of the organisation.
name	String	[Inherited from Organisational module] The name of the organisation.
acronym	String	[Inherited from Organisational module] The acronym of the organisation.
description	String	[Inherited from Organisational module] The description of the organisation in natural language.

Table 45. Attributes of the ORGANISATION metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
strategy	STRATEGY	[Inherited from Organisational module] An organisation has associated several organisational strategies.
goal	GOAL	[Inherited from Organisational module] An organisation has associated several organisational goals.
lower	ORGANISATIONAL_UNIT	[Inherited from Organisational module] An organisation has several organisational units.
organisa- tional_location	ORGANISATIONAL_LOCATION	[Inherited from Organisational module] An organisation is associated with several organisational location.

Table 46. Relationships of the ORGANISATION metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.27 ORGANIZATIONAL_ACTOR metaclass

Is an actor that interacts with the organisational system in some way. The organisational actors can be no human actors (e.g. a sensor) the most of times organisational actors are human.

Examples

For instance, John, Tony and Peter are members of the enterprise.

Metamodel view for ORGANIZATIONAL_ACTOR metaclass

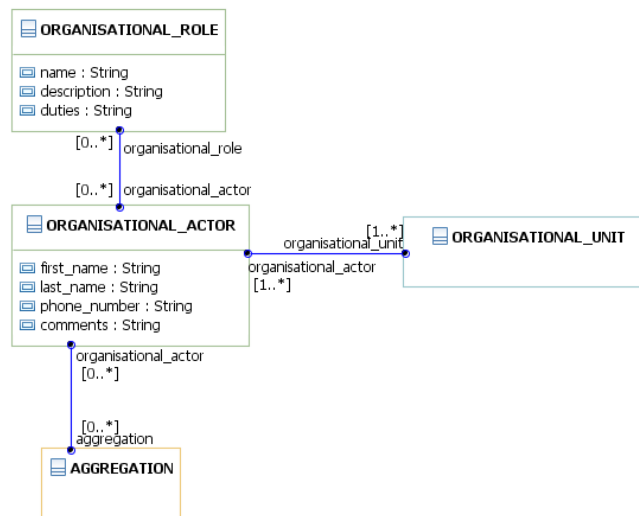


Figure 47. Portion of the metamodel including the metaclass ORGANIZATIONAL_ACTOR

Attributes

NAME	TYPE	DESCRIPTION
first_name	String	The first name of the organisational actor.
last_name	String	The last name of the organisational actor.
phone_number	String	The phone number of the organisational actor.
comments	String	Some comments about the organisational actor.

Table 47. Attributes of the ORGANIZATIONAL_ACTOR metaclass

Relationships

ROLE	PARTICIPANT	DESCRIPTION
organisa- tional_role	ORGANISATION AL_ROLE	An organisational actor can to be related to several organisational roles.
organisa- tional_unit	ORGANISATION AL_UNIT	An organisational actor can to be related to several organisational units.
aggregation	AGGREGATION	An organisational actor can to be associated to several complex substructure of type aggrega- tion.

Table 48. Relationships of the ORGANIZATIONAL_ACTOR metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.28 ORGANISATIONAL_LOCATION metaclass

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

Example

An organisational location is usually defined by the managerial staff and many factors usually influence their situation (e.g. closeness to raw materials and market, availability of communication and power supply infrastructures, government incentives).

Metamodel view

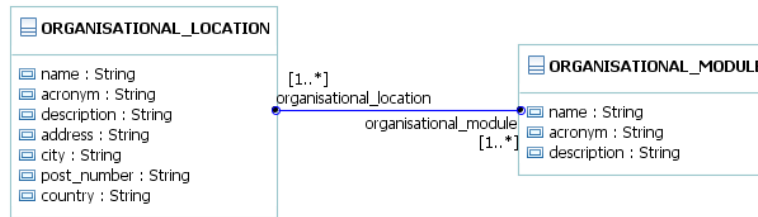


Figure 48. Portion of the metamodel including the metaclass ORGANISATIONAL_LOCATION

Attributes

NAME	TYPE	DESCRIPTION
name	String	The name of the organisational location.
acronym	String	The acronym of the organisational location.
description	String	The description of the organisational location in natural language.
address	String	The address of the organisational location.
city	String	The city where the organisational location is placed.
post_number	String	The post number of the organisational location.
country	String	The country of the organisational location.

Table 49. Attributes of the ORGANISATIONAL_LOCATION metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
organisational module	ORGANISATIONAL_MODULE	One organisational location can be associated to several organisational modules.

Table 50. Relationships of the ORGANISATIONAL_LOCATION metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.29 ORGANISATIONAL_MODULE metaclass

This is an abstract metaclass, which is specialised into Organisation and Organisational unit.

Metamodel view

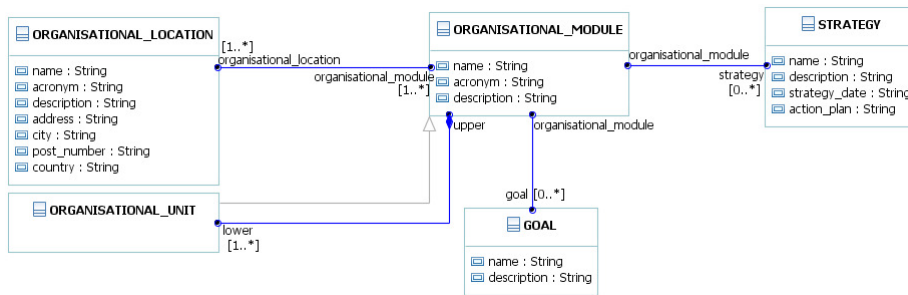


Figure 49. Portion of the metamodel including the metaclass ORGANISATIONAL_MODULE

Attributes

NAME	TYPE	DESCRIPTION
name	String	The name of the organisational module.
acronym	String	The acronym of the organisational module.
description	String	The description about the organisational module in natural language.

Table 51. Attributes of the ORGANISATIONAL_MODULE metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
strategy	STRATEGY	An organisation module has associated several organisational strategies.
goal	GOAL	An organisation module has associated several organisational goals.
lower	ORGANISATIONAL_UNIT	An organisation module has several organisational units.
organisa- tional_loca- tion	ORGANISATIONAL_LOCATION	An organisation module is associated with several organisational locations.

Table 52. Relationships of the ORGANISATIONAL_MODULE metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.30 ORGANISATIONAL_ROLE metaclass

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

Example

For instance, “John and Vicky are department clerks”. Department clerk is an organisational role that can be played by John and Vicky.

Metamodel view

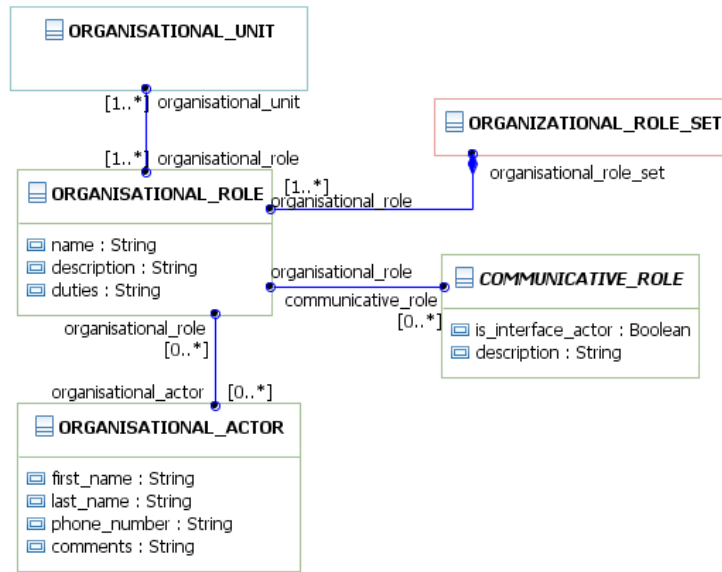


Figure 50. Portion of the metamodel including the metaclass ORGANISATIONAL_ROLE

Attributes

NAME	TYPE	DESCRIPTION
name	String	The name of the organisational role.
description	String	The description of the organisational role in natural language.
duties	String	The description of the duties of each organisational role.

Table 53. Attributes of the ORGANISATIONAL_ROLE metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
organisational_unit	ORGANISATIONAL_UNIT	An organisational role can to be associated to one or

		several organisational units.
organisational_role_set	ORGANISATIONAL_ROLE_SET	An organisational role is related to one organisational role set.
communicative_role	COMMUNICATIVE_ROLE	An organisational role can be played by several communicative roles.
organisational_actor	ORGANISATIONAL_ACTOR	An organisational role can be played by several organisational actors.

Table 54. Relationships of the ORGANISATIONAL_ROLE metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.31 ORGANISATIONAL_ROLE_SET metaclass

This is a container metaclass, which contains objects of type Organisational role.

The objective of this metaclass is to provide a place where is possible to instance graphically objects of type organisational role.

Metamodel view

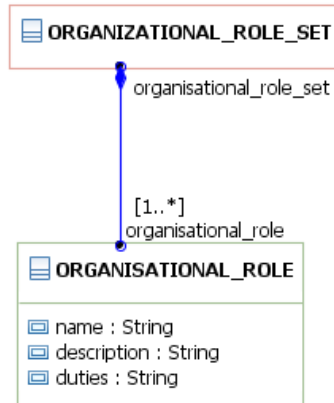


Figure 51. Portion of the metamodel including the metaclass ORGANISATIONAL_ROLE_SET

Attributes

This metaclass does not have attributes of its own.

Associations

ROLE	PARTICIPANT	DESCRIPTION
organisational_role	ORGANISATIONAL_ROLE	An organisational role set can to contain several objects of type organisational role.

Table 55. Relationships of the ORGANISATIONAL_ROLE_SET metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.32 ORGANISATIONAL_UNIT metaclass

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

This class specialises from Organisational module.

Example

The organisational units can be represented by organisational areas, departments, centres, divisions, directorates, teams, etc.

Usually, the organisational units make use of available installations, for example, buildings, warehouses, shopping centres, etc.

Metamodel view

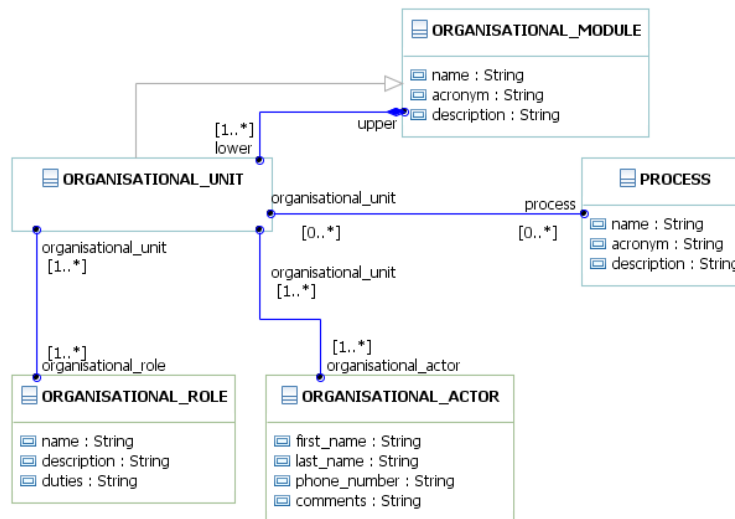


Figure 52. Portion of the metamodel including the metaclass ORGANISATIONAL_UNIT

Attributes

This metaclass does not have attributes of its own.

Associations

ROLE	PARTICIPANT	DESCRIPTION
upper	ORGANISATIONAL_MODULE	An organisational unit is contained in to one organisational module.
process	PROCESS	An organisational unit is related to several organisational processes.
organisational_actor	ORGANISATIONAL_ACTOR	An organisational unit has related several organisational actors.
organisational_role	ORGANISATIONAL_ROLE	An organisational unit has related several organisational roles.

Table 56. Relationships of the ORGANISATIONAL_UNIT metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.33 OUTGOING metaclass

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

Examples

Figure 53 shows a Client that receives information from the communicative event *Sale5*.

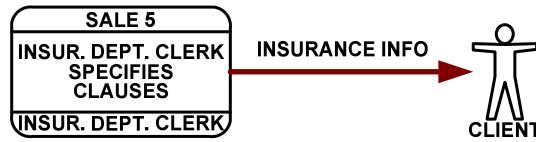


Figure 53. Example of OUTGOING in a communicative event diagram

Metamodel view for OUTGOING metaclass

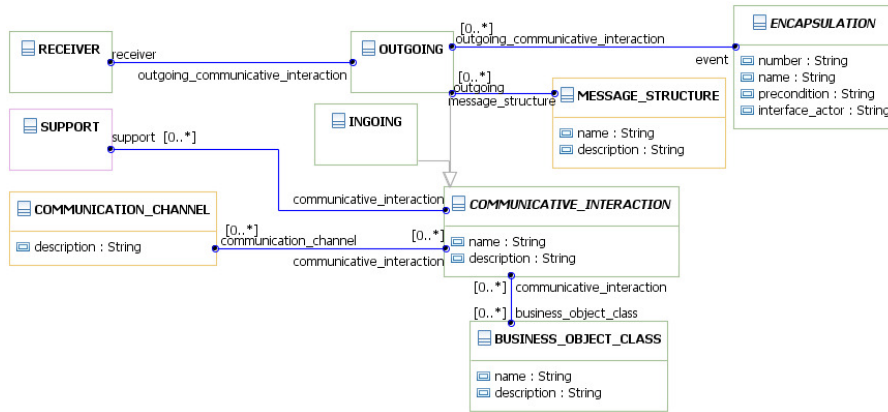


Figure 54. Portion of the metamodel including the metaclass OUTGOING

Attributes

NAME	TYPE	DESCRIPTION
name	String	[Inherited from Communicative interaction] The name of the communicative interaction.
description	String	[Inherited from Communicative interaction] The description of the communicative interaction in natural language.

Table 57. Attributes of the OUTGOING metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
event	ENCAPSULATION	An outgoing is related to one encapsula-

		tion.
receiver	RECEIVER	An outgoing is related to one receiver communicative role.
message_structure	MESSAGE_STRUCTURE	An outgoing has related one message structure.
support	SUPPORT	[Inherited from Communicative interaction] An outgoing can to be supported by several support roles.
business_object_class	BUSINESS_OBJECT_CLASS	[Inherited from Communicative interaction] An outgoing can to be related with several business object class.
communication_channel	COMMUNICATION_CHANNEL	[Inherited from Communicative interaction] An outgoing can to be carried out by means of several mediums or communication channel.

Table 58. Relationships of the OUTGOING metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive



Figure 55. Graphical primitive of the OUTGOING metaclass

3.4.1.34 PRECEDENCE Metaclass

A precedence relation between two communicative events is a rule consisting of a relationship that defines the relative time between them.

Examples

Figure 56 shows an example where for communicative event *Sale6* to occur, *Sale4* has necessarily occurred before.

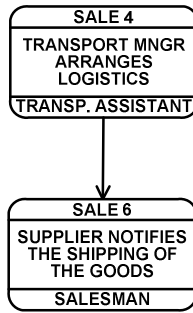


Figure 56. Example of PRECEDENCE in a communicative event diagram

Metamodel view for PRECEDENCE metaclass

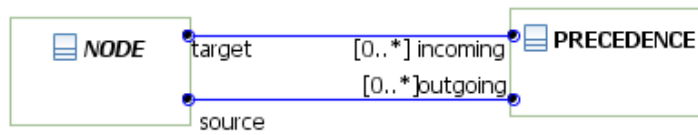


Figure 57. Portion of the metamodel including the metaclass PRECEDENCE

Attributes

This metaclass has no attributes of its own.

Relationships

ROLE	TO METAClass	SEMANTICS
source	NODe	A precedence has only one source node
target	NODe	A precedence has only one target node

Table 59. Relationships of the PRECEDENCE metaclass

Constraints

This metaclass has no constraints of its own.

Graphical primitive



Figure 58. Graphical primitive of the PRECEDENCE metaclass

3.4.1.35 PRIMARY metaclass

A primary role indicates, in the action context of a communicative event, the set of organisational actors that are designated for reporting to the information system an occurrence of a certain type of communicative event.

This metaclass specialises from Communicative role.

Metamodel view

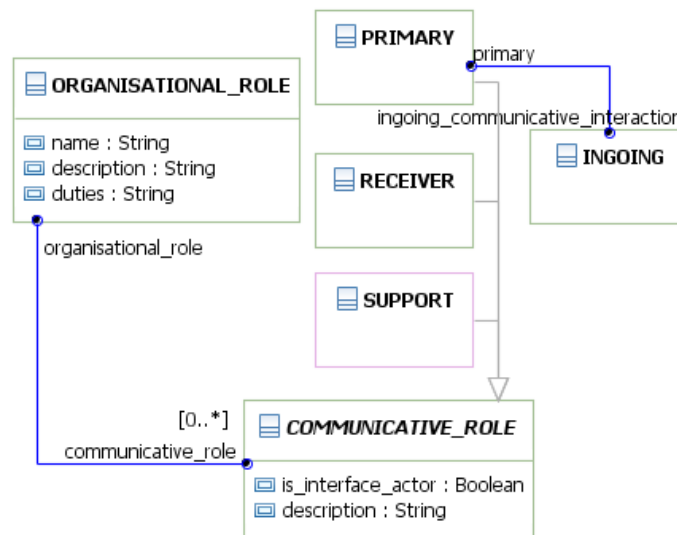


Figure 59. Portion of the metamodel including the metaclass PRIMARY

Attributes

NAME	TYPE	DESCRIPTION
description	String	[Inherited from communicative role] The description of the communicative role in natural language.
is_interface_actor	Boolean	[Inherited from communicative role] Indicates whether the communicative role is an interface actor, i.e. whether it plays the role of the interface actor for a communicative event.

Table 60. Attributes of the PRIMARY metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
organisa- tional_role	ORGANISATIONAL_ROLE	[Inherited from communicative role] A communicative role can be played by one organisational role.
ingo- ing_communicati ve_interaction	INGOING	A primary communicative role is related to one ingoing communicative interaction.

Table 61. Relationships of the PRIMARY metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive



Figure 60. Graphical primitive for the PRIMARY metaclass

3.4.1.36 PROCESS Metaclass

The organisational units are related to organisational process, where the communicative events occur to achieve the organisational goals of the enterprise.

Metamodel view for PROCESS metaclass

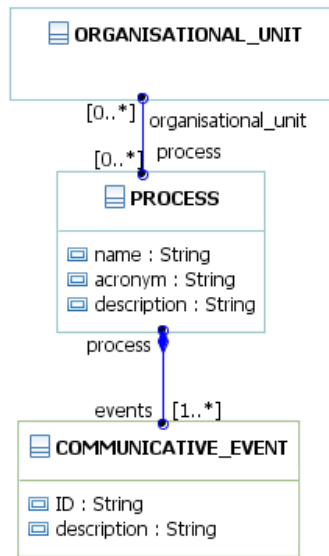


Figure 61. Portion of the metamodel including the metaclass PROCESS

Attributes

NAME	TYPE	DESCRIPTION
name	String	The name of the process.
acronym	String	The acronym of the process.
description	String	The description of the process in natural language.

Table 62. Attributes of the PROCESS metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
organisational_unit	ORGANISATIONAL_UNIT	A process is associated to several organisational units.
events	COMMUNICATIVE_EVENT	A process is related to several communicative events.

Table 63 Relationships of the PROCESS metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.37 RECEIVER Metaclass

The receiver role is played by a set of organisational actors that request the retrieval of data from the information system memory.

This metaclass specialises from Communicative role.

Metamodel view

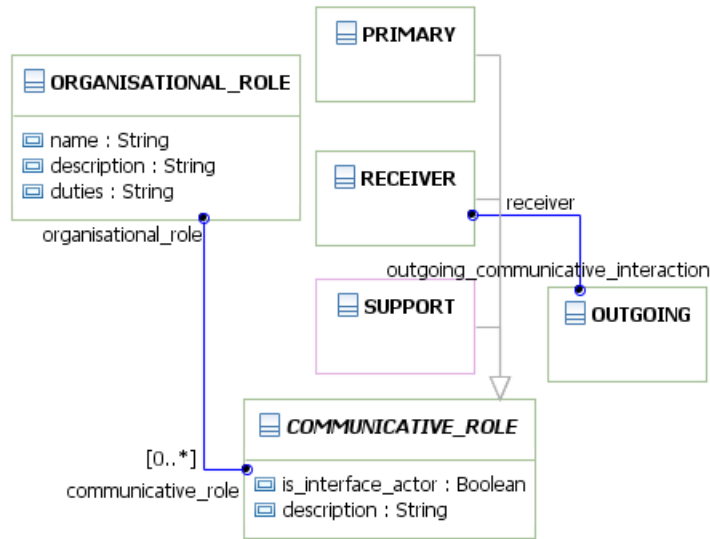


Figure 62. Portion of the metamodel including the metaclass RECEIVER

Attributes

NAME	TYPE	SEMANTICS
description	String	[Inherited from communicative role] The description of the communicative role in natural language.
is_interface_actor	Boolean	[Inherited from communicative role] Indicates whether the communicative role is an interface actor, i.e. whether it plays the role of the interface actor for a communicative event.

Table 64. Attributes of the RECEIVER metaclass

Associations

ROLE	TO METACLASS	SEMANTICS
organisa-tional_role	ORGANISATIONAL_ROLE	[Inherited from communicative role] A communicative role can be played by one

		organisational role.
outgoing_communicative_interaction	OUTGOING	A receiver communicative role is related to one outgoing communicative interaction.

Table 65. Relationships of the RECEIVER metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

Figure 63. Graphical primitive for the RECEIVER metaclass

3.4.1.38 REFERENCE_FIELD Metaclass

The reference field metaclass specifies a field whose domain is a type of business objects.

Examples

The SUBSTRUCTURE $A \langle B+C+D \rangle$ has the DATA_FIELD **B**, and **D**; **C** is a reference field that is a reference to **C** object. This object is already known by de IS.

Metamodel view for REFERENCE_FIELD metaclass

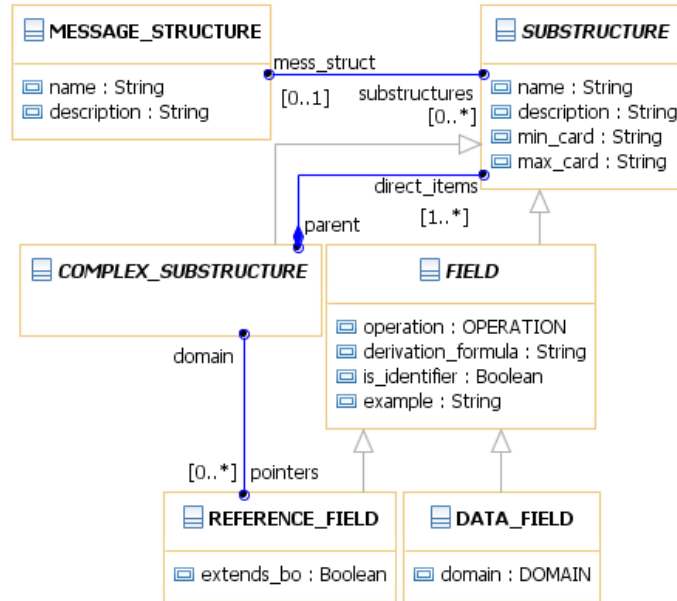


Figure 64. Portion of the metamodel including the metaclass REFERENCE_FIELD

Attributes

NAME	TYPE	SEMANTICS
extends_bo	Boolean	It is a Boolean value that indicates if a reference field extends an existing business object. This attribute is a mark to support the model transformation.
is_identifier	Boolean	[Inherited from Field] It is a Boolean value. It indicates if a data field is an identifier field of a substructure. This attribute is a mark to support the model transformation.
operation	OPERATION	[Inherited from Field] The operation indicates the origin or the information that the reference field refers. Its values can be “input”, “output” or “derivation”.
derivation_formula	String	[Inherited from Field] if the attribute operation is of type “derivation”, the derivation formula indi-

		ates the formula in natural language or OCL.
name	String	[Inherited from Substructure] The name of the reference field
description	String	[Inherited from Substructure] The description of the reference field in natural language
example	String	[Inherited from Substructure] An example about the reference field in natural language.
min_card	String	[Inherited from Substructure] Indicates a constraint about minimum cardinality of the reference field.
max_card	String	[Inherited from Substructure] Indicates a constraint about maximum cardinality of the reference field.

Table 66. Relationships of the REFERENCE_FIELD metaclass

Associations

ROLE	PARTICIPANT	SEMANTICS
mess_struct	MESSAGE_STRUCTURE	A reference field can to be associated to several message structures.
parent	COMPLEX_SUBSTRUCTURE	A reference field is associated to one complex substructure (named parent).
domain	COMPLEX_SUBSTRUCTURE	A reference field refers only one complex substructure (named domain).

Table 67. Relationships of the REFERENCE_FIELD metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.39 SPECIALISATION metaclass

The specialisation specifies one or more variants; that is, structural alternatives.

Metamodel view of SPECIALISATION metaclass

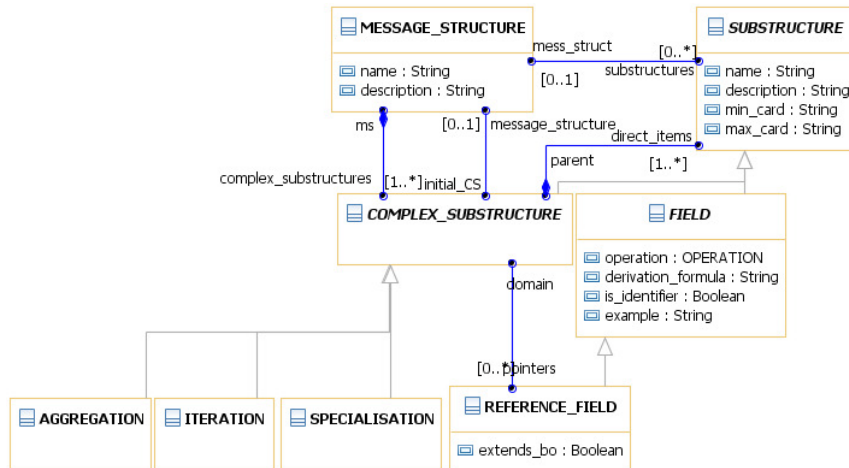


Figure 65. Portion of the metamodel including the metaclass SPECIALISATION

Attributes

NAME	TYPE	DESCRIPTION
name	String	[Inherited from substructure] The name of the specialisation.
description	String	[Inherited from substructure]The description or the specialisation in natural language.
min_card	String	[Inherited from substructure] The minimum cardinality of the specialisation.
max_card	String	[Inherited from substructure] The maximum cardinality of the specialisation.

Table 68. Attributes of the SPECIALISATION metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
direct_items	SUBSTRUCTURE	[Inherited from complex substructure] A specialisation is composed by several substructures.
message_structure	MESSAGE_STRUCTURE	[Inherited from complex substructure] A specialisation can be part of one message structure.
ms	MESSAGE_STRUCTURE	[Inherited from complex substructure] A specialisation is contained into one message structure.
parent	COMPLEX_SUBSTRUCTURE	[Inherited from substructure] A specialisation can to be a parent complex substructure.
mess_struct	MESSAGE_STRUCTURE	[Inherited from substructure] A specialisation can be related to one message structure.
pointers	REFERENCE_FIELD	[Inherited from complex substructure] A specialisation can to have several pointers from existing reference field.

Table 69 Relationships of the SPECIALISATION metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.40 START Metaclass

A start node indicates that the communicative events to which it is connected do not have any precedence events.

The metaclass specialises from Node.

Metamodel view for START metaclass

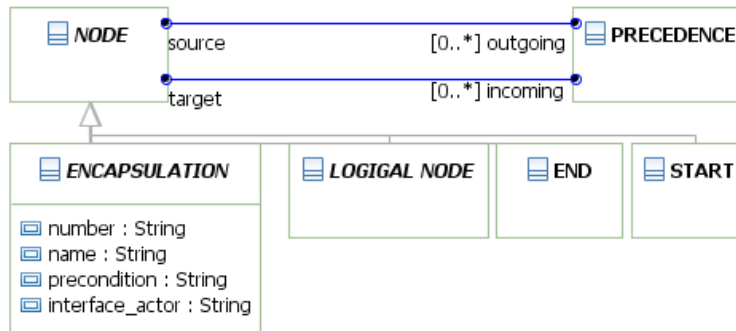


Figure 66. Portion of the metamodel including the metaclass START

Attributes

This metaclass does not have attributes of its own.

Relationships

ROLE	PARTICIPANT	SEMANTICS
outgoing	PRECEDENCE	[Inherited from Node] A start node can to have several outgoing precedences.
incoming	PRECEDENCE	[Inherited from Node] A start node can to have several ingoing precedences according to C6 .

Table 70. Relationships of the START metaclass

Constraints

C6. An START node cannot have ingoing relationships.

Graphical primitive



Figure 67. Graphical primitive of the START metaclass

3.4.1.41 STRATEGY metaclass

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

Metamodel view

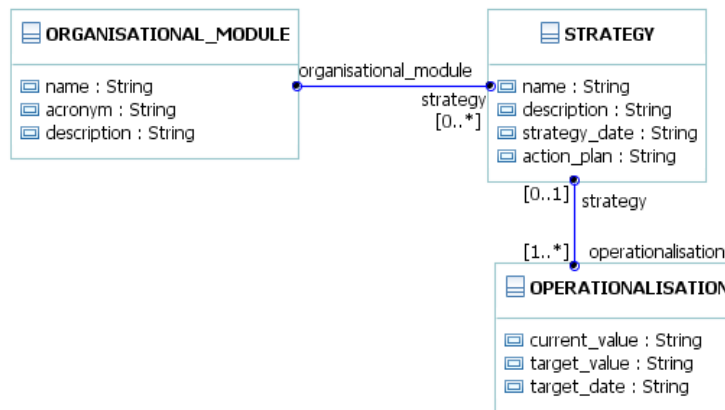


Figure 68. Portion of the metamodel including the metaclass STRATEGY

Attributes

NAME	TYPE	DESCRIPTION
name	String	The name of the strategy.
description	String	The description of the strategy in natural language.
strategy_date	String	The date of register.
action_plan	String	The action plan of the strategy.

Table 71. Attributes of the STRATEGY metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
organisa- tional_module	ORGANISATIONAL_MODULE	A strategy is related to one or- ganisational module.
operationalisa- tion	OPERATIONALISATION	A strategy is associated to sev- eral operationalisations.

Table 72. Relationships of the STRATEGY metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.42 SUBSTRUCTURE Metaclass

The substructure metaclass specifies an element that is part of a mes-
sage structure [38].

This is an abstract metaclass, which is specialised into Complex sub-
structure and Field.

Metamodel view for SUBSTRUCTURE metaclass

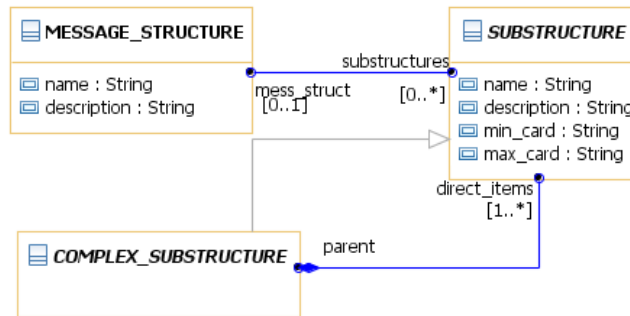


Figure 69. Portion of the metamodel including the metaclass SUBSTRUCTURE

Attributes

NAME	TYPE	DESCRIPTION
name	String	The name of the reference substructure.
description	String	The description of the substructure in natural language.
example	String	An example about the substructure in natural language.
min_card	String	Indicates a constraint about minimum cardinality of the substructure.
max_card	String	Indicates a constraint about maximum cardinality of the substructure.

Table 73. Attributes of the SUBSTRUCTURE metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
mess_struct	MESSAGE_STRUCTURE	A substructure can to be associated to several message structures.
parent	COMPLEX_SUBSTRUCTURE	A substructure is associated to one complex substructure (named parent).

Table 74. Relationships of the SUBSTRUCTURE metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.43 SUPPORT Metaclass

A support role participates as interacting partners, not being a primary actor or a receiver actor.

This metaclass specialises from Communicative role.

Metamodel view for SUPPORT metaclass

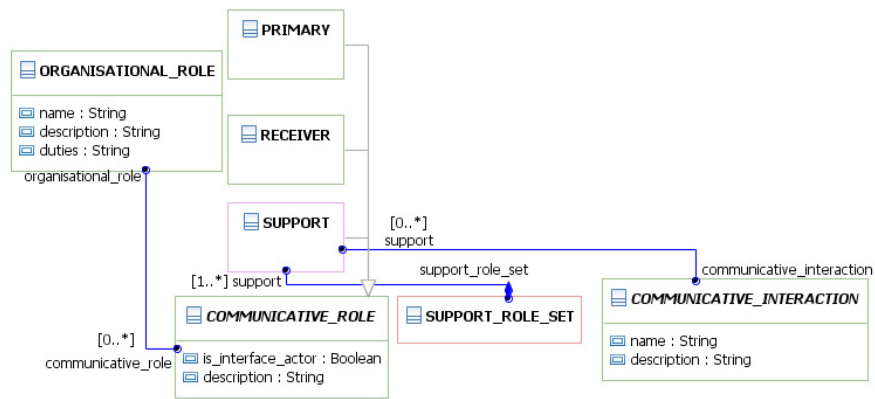


Figure 70. Portion of the metamodel including the metaclass SUPPORT

Attributes

NAME	TYPE	SEMANTICS
description	String	[Inherited from communicative role] The description of the communicative role in natural language.
is_interface_actor	Boolean	[Inherited from communicative role] Indicates whether the communicative role is an interface actor, i.e. whether it plays the role of the inter-

		face actor for a communicative event.
--	--	---------------------------------------

Table 75. Attributes of the SUPPORT metaclass

Associations

ROLE	TO METACLASS	SEMANTICS
organisa- tional_role	ORGANISATIONAL_ ROLE	[Inherited from communicative role] A communicative role can be played by one organisational role.
communica- tive_interaction	COMMUNICATIVE_I NTERACTION	A support communicative role is related to one communicative interaction.
support_role_set	SUPPORT_ROLE_SE T	A support is contained into one support role set.

Table 76. Relationships of the SUPPORT metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.44 SUPPORT_ROLE_SET metaclass

This is a container metaclass, which contains objects of type Support role.

The objective of this metaclass is to provide a place where is possible to instance graphically objects of type support role.

Metamodel view

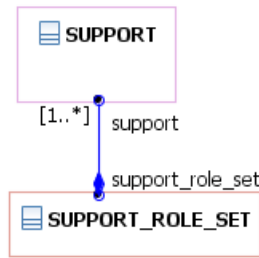


Figure 71. Portion of the metamodel including the metaclass SUPPORT_ROLE_SET

Attributes

This metaclass does not have attributes of its own.

Associations

ROLE	PARTICIPANT	DESCRIPTION
support	SUPPORT	A support role set can to contain several instances of support role.

Table 77. Relationships of the SUPPORT_ROLE_SET metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.45 TEXTUAL_REQUIREMENT metaclass

The textual requirements represent some types of requirements that could be to affect some elements of the metamodel.

Metamodel view

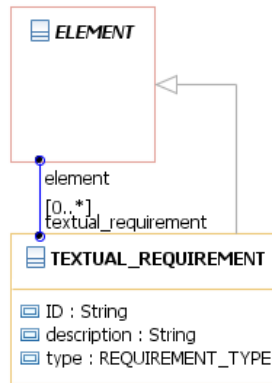


Figure 72. Portion of the metamodel including the metaclass
TEXTUAL_REQUIREMENT

Attributes

NAME	TYPE	DESCRIPTION
ID	String	The identification of the requirement.
description	String	The description of the requirement in natural language.
type	REQUIREMENT_TYPE	The type of the textual requirement.

Table 78. Attributes of the TEXTUAL_REQUIREMENT metaclass

Associations

ROLE	PARTICIPANT	DESCRIPTION
element	ELEMENT	A textual requirement is related to one element of the metamodel.

Table 79. Relationships of the TEXTUAL_REQUIREMENT metaclass

Constraints

This metaclass does not have constraints.

Graphical primitive

This metaclass does not have graphical primitive.

3.4.1.46 DOMAIN enumeration

This enumeration corresponds to the pre-defined domains that exist for all data fields in a message structure.

Metamodel view

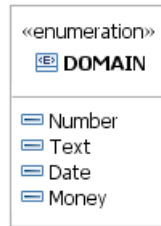


Figure 73. Portion of the metamodel including the metaclass DOMAIN

Elements

ELEMENT	DESCRIPTION
Number	Values are natural, integer, real, autonumeric.
Text	Values are character strings.
Date	Values are time points of variable precision, days, months, years, hours, minutes and seconds.
Money	Values are real.

Table 80. Elements of the DOMAIN metaclass

3.4.1.47 OPERATION enumeration

This enumeration corresponds to the pre-defined operations that exist for all fields in a message structure.

Metamodel view

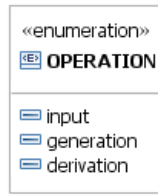


Figure 74. Portion of the metamodel including the metaclass OPERATION

Elements

ELEMENT	DESCRIPTION
input	Value that indicates the information of the field is provided by the primary actor.
generation	Vale that indicates the Information System can automatically generate the information of the field.
derivation	Value that indicates the information of the field is already known by the Information System and, therefore, it can be derived from its memory. This operation can have an associated derivation formula.

Table 81. Elements of the OPERATION metaclass

3.4.1.48 REQUIREMENT_TYPE enumeration

This enumeration corresponds to the pre-defined requirement types that exist for classifying the textual requirements.

Metamodel view

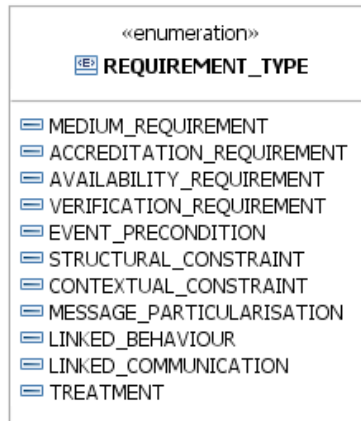


Figure 75. Portion of the metamodel including the metaclass REQUIREMENT_TYPE

Elements

ELEMENT	DESCRIPTION
medium_requirement	This element indicates the medium requirements.
accreditation_requirement	This element indicates the accreditation requirements.
availability_requirement	This element indicates the availability requirements.
verification_requirement	This element indicates the verification requirements.
event_precondition	This element indicates the event precondition requirements.
structural_constraint	This element indicates the structural requirements.
message_particularisation	This element indicates the message particularisation requirements.
linked_behaviour	This element indicates the linked behaviour requirements.
linked_communication	This element indicates the linked communication requirements.
treatment	This element indicates the treatment requirements.

Table 82. Elements of the REQUIREMENT_TYPE metaclass

3.5 Metamodel validation

The PIM metamodel for CED (See subsection 3.3) was built following the concepts explained at Communication Analysis method (See Chapter 3). We have analysed the principal primitives of the communicative event diagrams and we have represented it through meta-classes and relationships in the PIM.

For the metamodel validation, we have established an iterative and incremental procedure. This procedure intends to validate whether the metamodel was implemented correctly, whether the metamodel brings to the Communication Analysis experts a metamodel closer to their ideas and whether the metamodel is closer to the method.

A general view of the metamodel validation is presented at Figure 76.

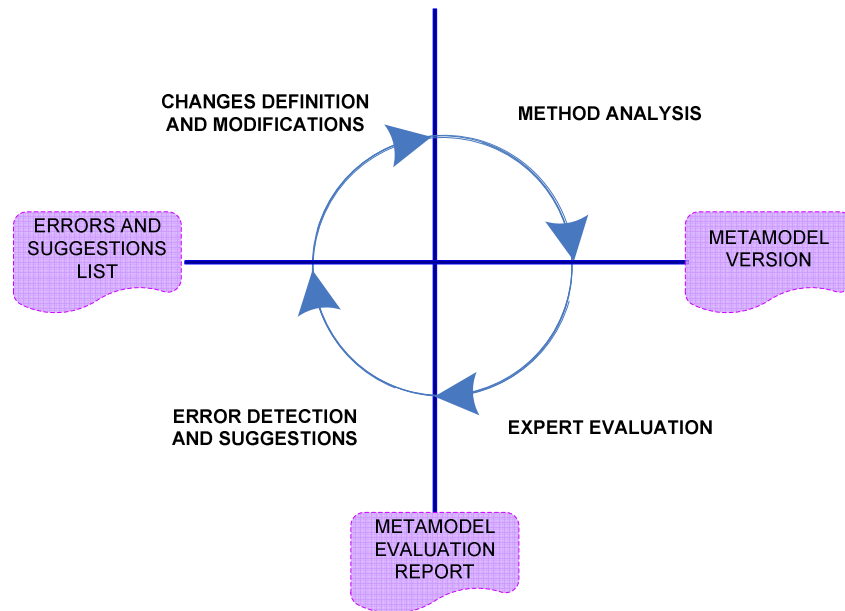


Figure 76. General view of metamodel validation

First, a method analysis activity was necessary in order to design a first *metamodel version*, later; it was revised by an expert in Communication Analysis method. This expert analyse the metamodel and

writes a *metamodel evaluation report*. This report is filtered so as to obtain a *list of errors and suggestions*; thus in this way it is possible to carry out definition changes and modifications for later to generate a new metamodel version aligned with the method and the expert evaluation.

Following this iterative procedure, and after of several iterations, it was possible to obtain a stable metamodel version (PIM metamodel presented at subsection 3.3).

The experts have used the metamodel to model different cases. For this purpose the expert have used Microsoft Visio [35] to model different instances of the CED metamodel.

3.6 Analysis and discussion

After of metamodel specification, we have learnt some lessons about the metamodelling strategy.

It is possible to offer a MDD environment to cover from requirements specification to code generation. It helps to requirements models to be part into the software development.

The MDA guidelines allow us to distinct two levels of metamodelling. The first level is about the reasoning of the method and concepts without to having into account a technological platform (named PIM metamodel). The second level allows us to represent some concepts that are platform-oriented into the metamodel of the concepts of the method.

A metamodel validation was carried out with experts in Communication Analysis method. This validation follows an iterative process, which allowed to improve the metamodel specification and to correct some mistakes.

The PSM metamodel specification is the primary artefact for building a modelling tool in addition, the metamodel is used by several lan-

guage transformation engine. The following chapters present the building of a modelling tool for supporting the modelling of CED and message structures, and the use of a language transformation engine for supporting the requirements models transformation.

4 A modelling tool for Communication Analysis requirements models

To facilitate the use of Communication Analysis method, we propose a technological support to close the gap between the implementation of software systems and the requirements models.

To achieve this aim, we propose to develop a modelling tool for supporting the modelling of communicative event diagrams and message structures (requirements techniques of Communication Analysis method.). This chapter presents the technological support and decisions about the technology to support the method. In addition, we present the metamodel implementation and how to use the graphical editor to build communicative event diagrams and message structures. Furthermore, we present a modelling tool evaluation that allows us to improve the proposal and involve the user experience into the development of the modelling tool.

The Append 2 presents the development of the modelling tool step by step, which includes the design of the metamodels for specifying

the concrete syntax for the models. This chapter concludes with some analysis and discussions about this phase of the project.

4.1 Technological support

Eclipse is an open source software development project, which purpose is to provide a highly integrated tool platform. The work in Eclipse consists of a core project, which includes a generic framework for tool integration, and a Java development environment built using it. Other projects extend the core framework to support specific kinds of tools and development environments. The projects in Eclipse are implemented in Java and run on many operating systems including Windows and Linux [36].

Eclipse.org [37] is a consortium of a number of companies that have made a commitment to provide substantial support to the Eclipse project in terms of time, expertise, technology, or knowledge.

The Eclipse platform is a framework for building integrate development environments (IDEs). The Eclipse platform itself and the tools that extend it are both composed of plug-ins. A simple tool may consist of a single plug-in, but more complex tools are typically divide into several [36].

Eclipse Modelling Framework (EMF) is a modelling framework for Eclipse. EMF relates modelling concepts directly to their implementations, thereby bringing to Eclipse and Java Developers in general, the benefits of modelling with a low cost of entry. Thus, EMF is a framework and code generation facility that lets you define a model. EMF unifies the three important technologies: Java, XML and UML. Regardless of which one is used to define it, an EMF model is the common high level representation that “glues” them all together [36] (see Figure 77).

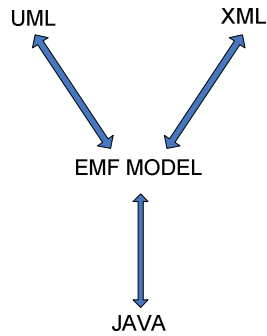


Figure 77. EMF defines Java, XML and UML

The model used to represent models in EMF is called Ecore (See PSM implementation at subsection 4.2.1). Ecore is itself an EMF model, and thus is its own metamodel.

A simplified subset of the Ecore model is shown in Figure 78. This subset contains a set of metaclasses that represent another model. Ecore is a subset of UML [36].

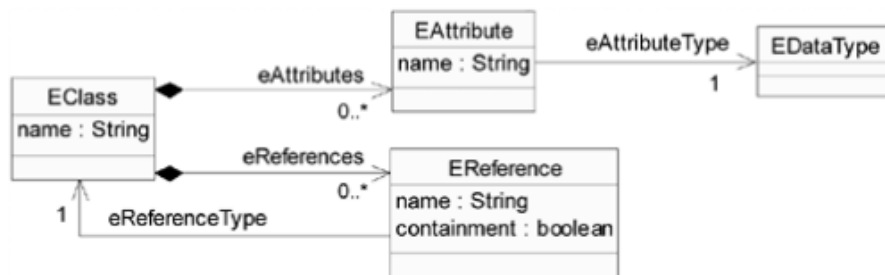


Figure 78. Subset of the Ecore model

A modelling tool is intended, for this reason a PSM metamodel was defined (see subsection 3.4). This PSM metamodel corresponds to an EMF specification.

The following subsections explain how was the PSM metamodel implementation, this development is supported by Eclipse technology; specifically EMF and Graphical Modelling Framework (GMF) [40].

4.2 PSM metamodel implementation

According to the framework described at section 1.4 (The framework of integration), the phase 2 of the proposal includes a task about the implementation of a modelling tool (see Figure 79). This task includes the implementation of the PSM metamodel into the chosen technology.

We have used Eclipse UML 2.1 tools [40] to model the PSM metamodel. We have preferred to implement the PSM metamodel in the UML 2.1 Eclipse tools because the graphical representation let us to have a complete vision of the metamodel. The Ecore metamodel is available in a graphical representation too, but the use of the graphical tools for Ecore metamodels are difficult to use; especially at the moment to specify relationships and cardinalities. Moreover, the use of UML 2.1 Eclipse tools let us obtain the Ecore model automatically.

Having the Ecore model, we can to use the GMF so as to build a graphical interface to model communicative event diagrams and message structures. The implementation of the PSM metamodel will be shown below.

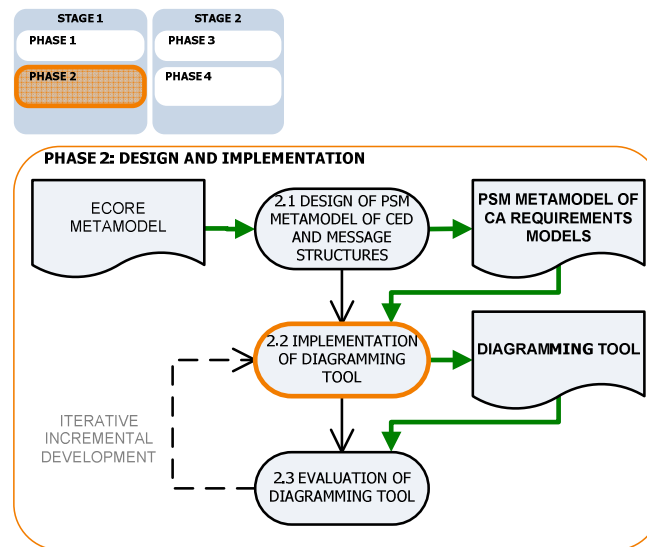


Figure 79. Stage 1 phase 2 explanation

4.2.1 Use of Eclipse Modelling Framework tools

First, create an UML 2.1 project. Go to **file - new - other**. In the **New** window, select the **Class Diagram** wizard for UML 2.1 Diagrams project (see Figure 80).

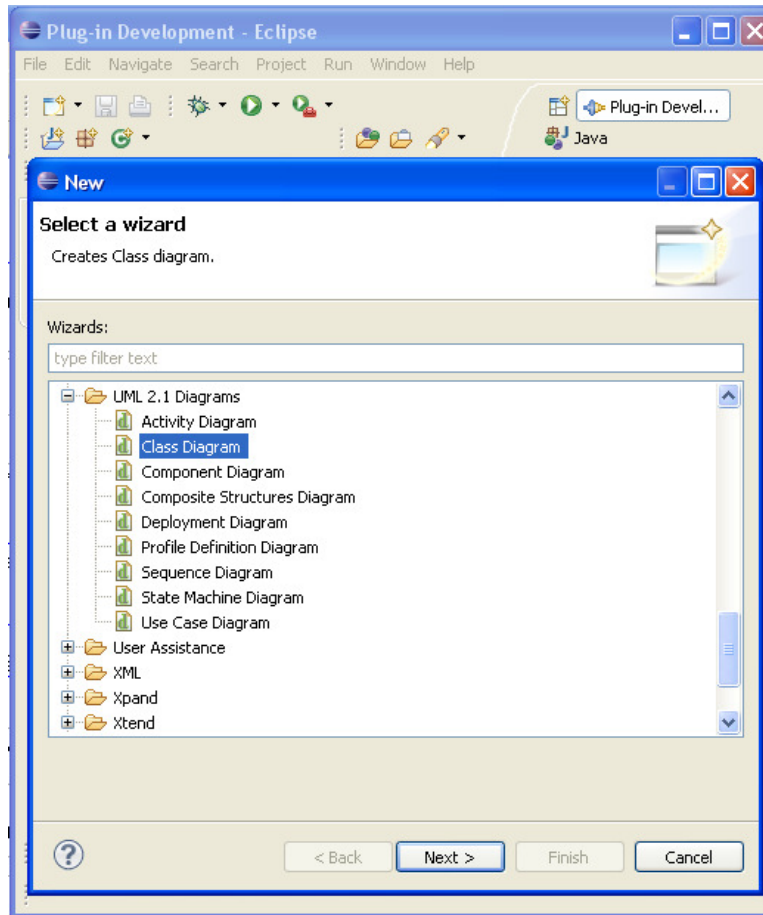


Figure 80. Creation of a UML class diagram Project

Two files are created (*umlclass* type and *uml* type). The *umlclass* file is used to model de PSM metamodel previously created (See section 3.4). The PSM metamodel is specified graphically using the UML 2.1 tools.

To take the elements from the *palette* and put it in to the workspace to represent the metaclasses, attributes, relationships, cardinalities, data types, roles, constraints, etc. While the metamodel is being created graphically, the *uml* file contains the metamodel information in a tree way (See Figure 81). To illustrate, the Figure 81 presents the graphical view and the tree view of the metamodel. The orange box emphasizes the metaclass `COMMUNICATIVE_EVENT` in both views.

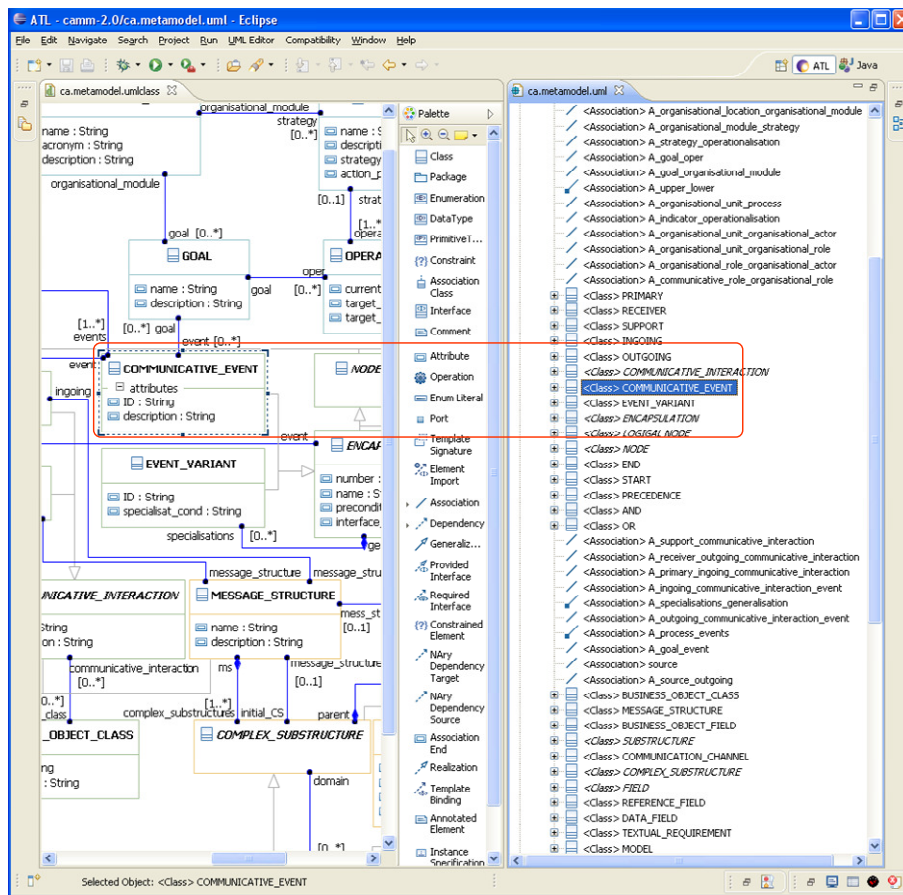


Figure 81. Creation of the PSM metamodel with UML 2.1 tools

The complete specification of the PSM metamodel is presented at Figure 82.



Figure 82. Complete description of the PSM metamodel in a tree way

To obtain the Ecore specification, go to menu; option **UML Editor – Convert To – Ecore Model** (See Figure 83).

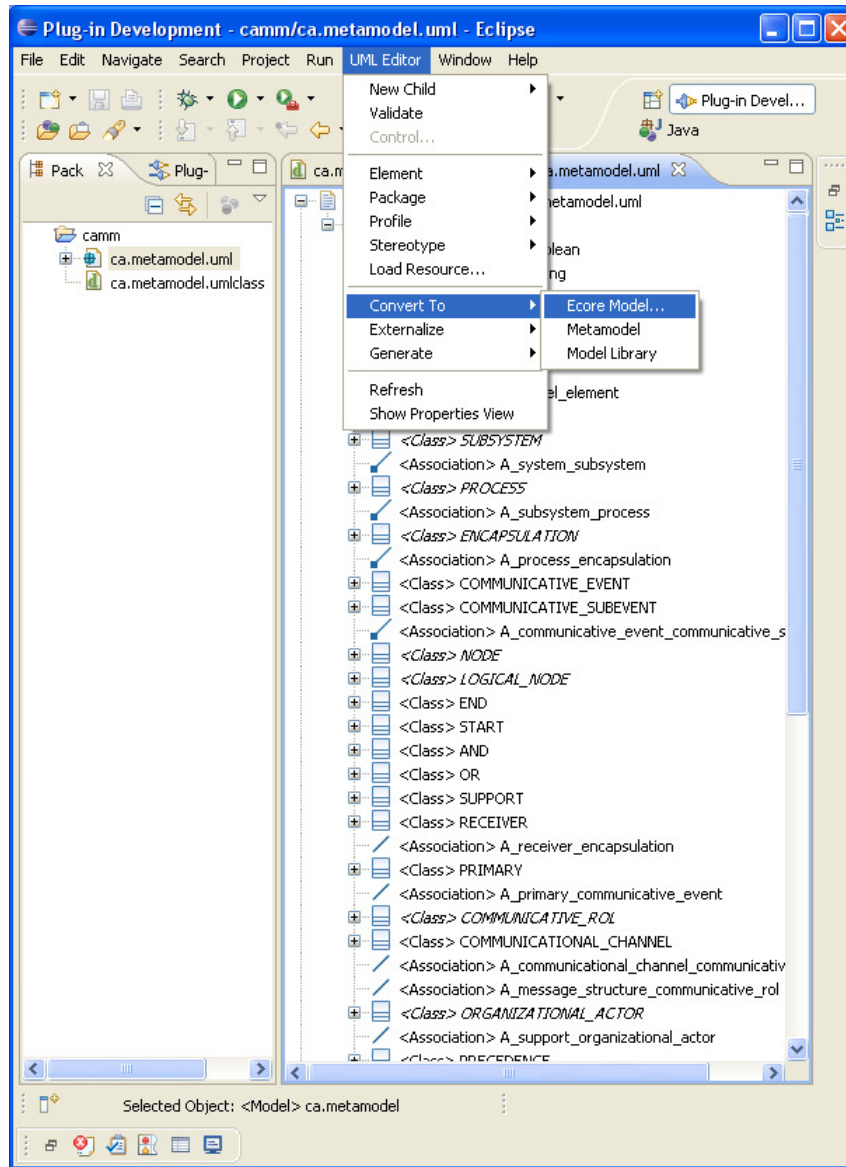


Figure 83. Ecore model creation

Thus the Ecore specification for the PSM metamodel is obtained (See Figure 84).

The Ecore metamodel is intended because this metamodel will be the principal artefact in order to implement the modelling tool.

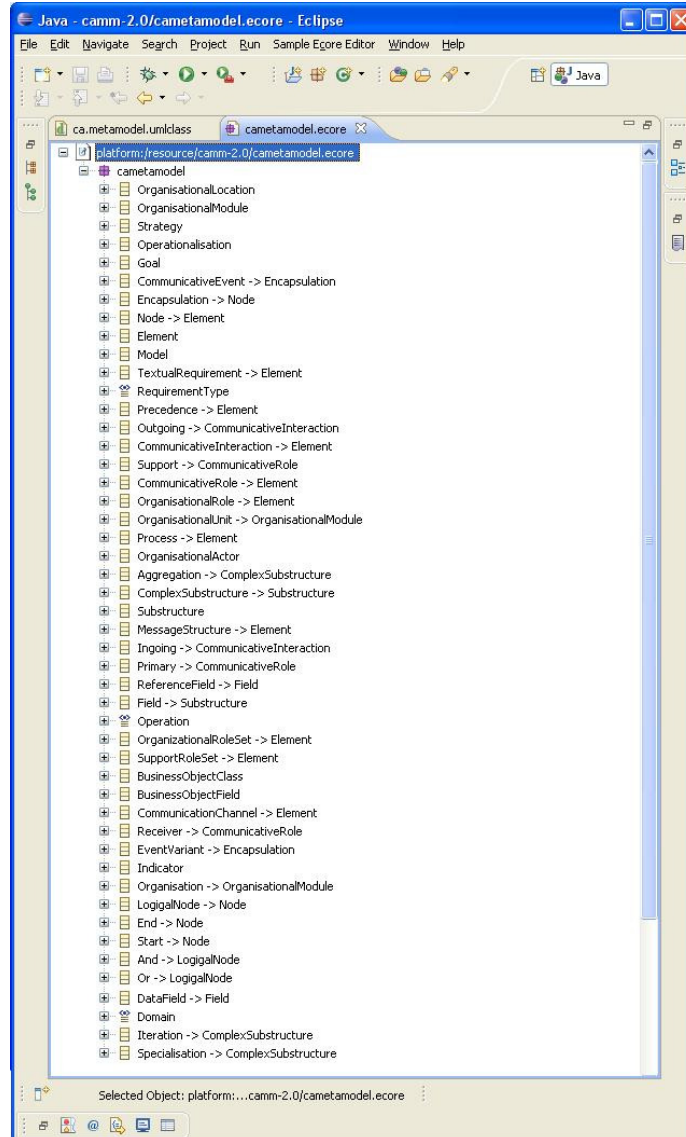


Figure 84. PSM metamodel in the Ecore specification

4.3 Design of the graphical editor for communicative event diagrams and message structures

The Graphical Editing Framework (GEF) provides technology to create rich graphical editors and views for the Eclipse Workbench User Interface [40]. The Graphical Modelling Framework (GMF) is an Eclipse project that aims to provide a generative bridge between the EMF and GEF [40].

GMF is an Eclipse project with the potential to become a keystone framework for the rapid development of standardized Eclipse graphical modelling editors. Architects and developers involved in the development of graphical editors or of plug-ins integrating both EMF and GEF technologies should consider building their editors against the GMF Runtime component. This framework let us to build modelling tools based on Eclipse editors like UML editor, Ecore editor, BPM Editor, etc. The Framework can be divided en two main components: the tooling and the runtime. The tooling consists of editors to create/edit models describing the notational, semantic and tooling aspects of a graphical editor. The generated plug-ins depend on the GMF Runtime component to produce a world class extensible graphical editor [41].

We have followed a workflow in order to create a graphical modelling environment for communicative event diagrams and message structures. The Figure 129 shows the workflow followed. This workflow was built according to the Eclipse Tutorials [37].

We can to distinguish three important phases. The first phase is the *definition of domain models*. These set of models intent specify the non-graphical information managed by the editor. The second phase is the *definition of diagram models*. These models define graphical elements to be displayed in the editor. The Third phase is the *generation of graphical editor*. This phase basically takes the models previously created in order to generate the java code that will be representing the graphical editor.

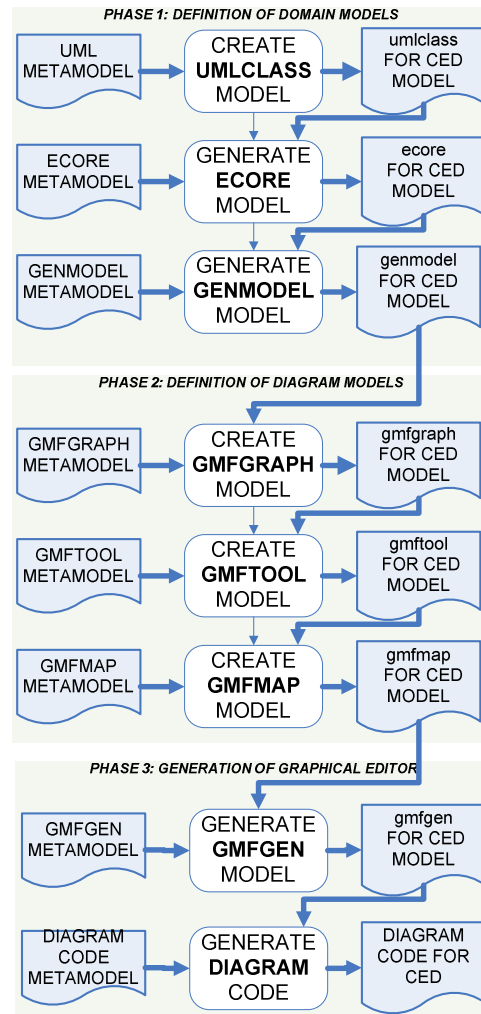


Figure 85. Workflow to create the modelling environment for CED

At the Appendix 1, we present an analysis about each phase and we will explain how was made each activity.

4.4 How to use the modelling tool

This subsection intends to show the use of the graphical editor for communicative event diagrams and message structures. Then, we have used a fictional example of a process named SuperStationery Co. A textual explanation of the process is presented at the following subsection. We have selected a part of the business process to explain the use in the modelling tool. The steps to build the corresponding communicative event diagram and to specify the message structures in the modelling tool are explained below.

Now, we will introduce a textual description that corresponds to part of the business processes of a fictional small and medium enterprise named SuperStationery Co. This company provides stationery and office material to its clients. The company acts as intermediary.

SuperStationery Co. Textual description

SuperStationery Co. is a company that provides stationery and office material to its clients. The company acts as an intermediary: the company has a catalogue of products that are bought from suppliers and sold to clients. Most clients call the Sales Department, where they are attended by a salesman. Then the client requests one or several products that are to be sent to one or many destinations. The salesman takes note of the order (see the Order form at Table 84). Other clients place orders by email or by fax. Then the Sales Manager reviews the order and assigns it to one of the many suppliers that work with the company, using his own judgement (the Sales Manager notes down the Supplier section of the Order form; additionally, the company wants to record the assignment date). An order form is sent by fax to the supplier. The supplier receives the order form and checks whether they have enough stock or not. In case they have enough stock of all the products requested in the client order, they accept the order (the supplier indicates the planned delivery date –that is, the date at which the supplier commits to deliver the order- and the salesman also notes down the response date); otherwise, they reject it. In case the order is rejected, the Sales Manager assigns it to a different supplier (this can happen many times until the order is accepted).

Once the order is accepted, the salesman sends a copy of the order to the Transport Department and the Insurance Department. In the Transport Department, the Transport Manager arranges how the goods will be carried to the destinations; this implies selecting one of the truck drivers hired by the company and deciding the order in which the truck will visit each of the client destinations. The Transport Manager prefers to work in paper and pencil; then he gives the logistics information to his assistant, the assistant fills the logistics form and sends it to both the client and the supplier. In the Insurance Department, the clerk specifies the insurance clauses, stapling them to the order form. SuperStationery has contracted an insurance policy with an insurance company. The policy has a set of generic insurance clauses. For each order, the Insurance Department clerk can specify additional clauses that extend or restrict the coverage. The clerk sends the order form and the insurance information back to the Sales Department, where the salesman faxes the insurance information to the client. When the transportation vehicle (usually a truck, but sometimes a van) picks up the goods from the supplier's warehouse, the supplier phones the company to report that the shipments are on their way to their destinations (a timestamp is recorded).

As the company prospers, the amount of orders increases and, thus, the company needs more truck drivers to deliver the goods in time. Therefore, from time to time the transport manager hires a new truck driver. Truck drivers have their own truck.

The company has a director and is divided into four departments (see Figure 86).

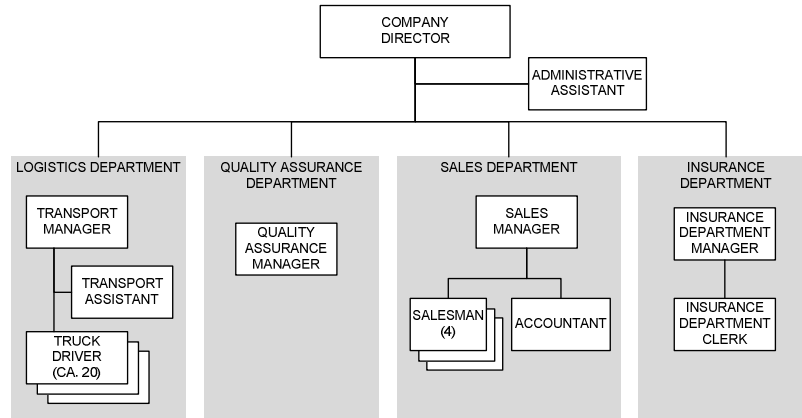


Figure 86. SuperStationery Co. organization chart

The company work practice has been decomposed into business processes (see Table 83)

BUSINESS PROCESS		DEPARTMENTS				
Acronym	Name	Director	Logistics	Quality A.	Sales	Insurance
Clie	Client management	X			X	
Prod	Product management	X		X		
Logi	Logistics		X			
Sale	Sales management		X		X	X
Risk	Risk management					X
Acco	Accounting				X	
Supp	Supplier management	X			X	

Table 83. SuperStationery Co. business processes

The analysis of the process description about SuperStationery Co. could be represented in a communicative event diagram. The Figure 87 shows the communicative event diagram for Sale business process modelled in Visio tool[35]. A complete explanation about building

communicative event diagram for SuperStationery Co. is available at [42].

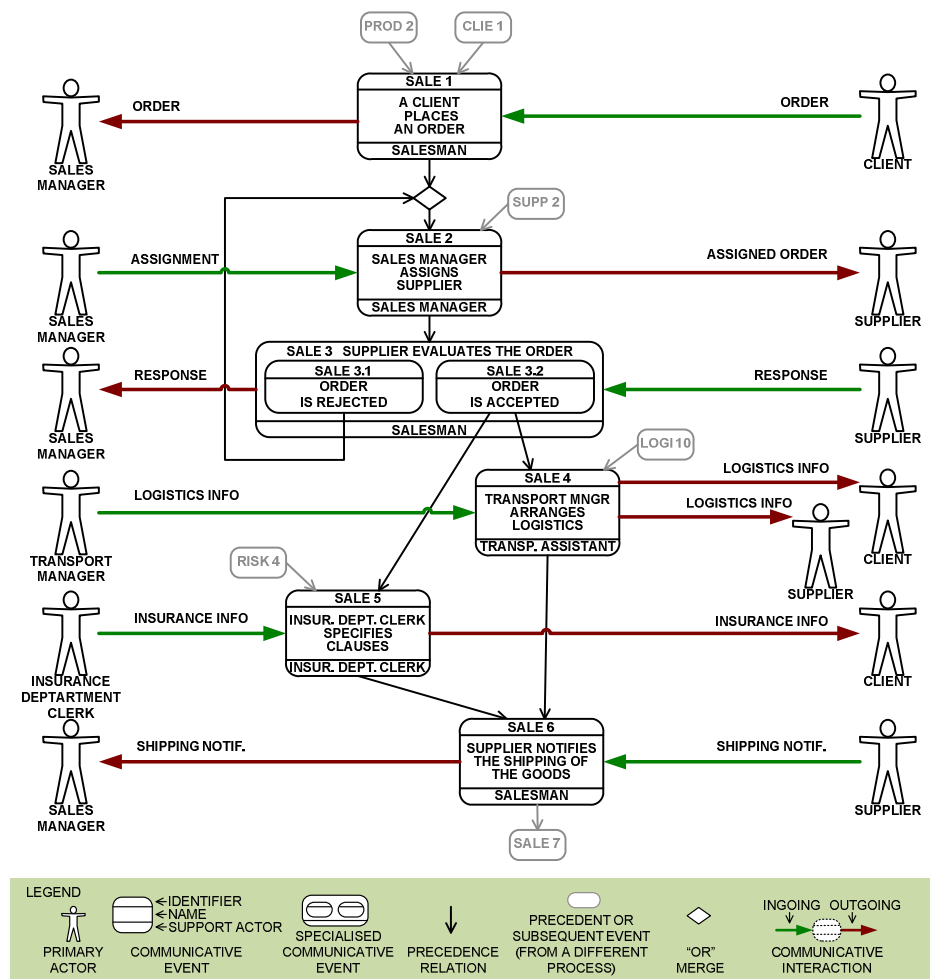


Figure 87. Communicative event diagram of SuperStationery Co. Sales manager business process (Sale)

The aim is to show how to use the modelling tool, thus, for practical purposes; we have decided to present the analysis of one communicative event of the Sale process. The purpose is to illustrate its representation in the modelling tool.

We have chosen the communicative event Sale 1 (*A CLIENT PLACES AN ORDER*) to guide the use into the modelling environment.

First, We will analyse the event description template of the Sale 1 [6]. This includes the analysis of forms or another documents related to the event, and the analysis of it related message structure [38].

Event description template

Sale 1. A client places an order		
General Information		
Goals		
<p>The objective of the organisation is to attend the clients when they request goods.</p> <p>From the point of view of the information system, the objective of this event is to record the order that the client places, and to let the Sales Manager know that a new order has arrived.</p>		
Description		
<p>Most clients call the Sales Department, where they are attended by a salesman. Then the client requests one or several products that are to be sent to one or many destinations. The salesman takes note of the order. Other clients place orders by email or by fax.</p>		
Contact requirements		
Actor responsibilities		
<ul style="list-style-type: none"> • Primary actor: Client • Communication channel: In person, by phone, by fax • Support actor: Salesman 		
Temporal requirements		
<ul style="list-style-type: none"> • Occurrence temporal restrictions: Only working days during reception hours (09:00-18:00) • Frequency of occurrence: 500 orders per week 		
Business forms		
ORDER		
Order number: 10352	Request date:	31-08-2009
Payment type: <input checked="" type="checkbox"/> Cash	<input type="checkbox"/> Credit	<input type="checkbox"/> Cheque
		Planned delivery date:05-09-2009

Client					
VAT number: 56746163-R					
Name: John Papiro Jr.					
Telephone: 030 81 48 31					
Supplier					
Code: OFFIRAP					
Name: Office Rapid Ltd.					
Address: Brandenburgen street, 46, 2983 Millhaven					
Destination: Blvd. Blue mountain, 35-14A, 2363 Toontown					
Person in charge: Brayden Hitchcock					
#	Code	Product name	Price	Q	Amount
1	ST39455	Rounded scissors (cebra) box-100	250 €	35	889,00 €
2	ST6399	Stáplescoop 26-22 blister 500	5,60 €	60	336,00 €
3	CA479-9	Stereofoam cups box-50 (pack 120)	18,75 €	10	187,50 €
					1412,50 €
Destination: Greenhouses street, 20, 2989 Millhaven					
Person in charge: Luke Padbury					
#	Code	product name	price	Q	Amount
1	ST6399	etaples cooper 26-22 bliste9000	5,60 €	3	444,50 €
2	CA746-3	Sugar lumps 1kg	2,30 €	3	6,90 €
					451,40 €
Total					1863,90 €

Table 84. Order form

Communication requirements**Message structure**

FIELD	OP	DOMAIN	EXAMPLE VALUE
ORDER =			
< Order number +	g	number	10352
Request date +	i	date	31-08-2009
Payment type +	i	text	Cash

<pre> Client + DESTINATIONS = { DESTINATION = < Address + Person in charge + LINES = { LINE = < Product + Price + Quantity > } } > </pre>	<pre> i i i i i i i i </pre>	<pre> Client Client address text Product money number </pre>	<pre> 56746163-R, John Papiro Jr. Blvd. Blue mountain, 35-14A, 2363 Brayden Hitchcock ST39455, Rounded scissors (cebra) box-100 25,40 € 35 </pre>
---	------------------------------	--	---

Table 85. Message structure specification of communicative event Sale1

Structural restrictions

- One order can have many destinations.
- One destination can have many lines.

Contextual restrictions

- Orders are identified by Order number.
- The product price in the line takes its value from the current price of the product in the catalogue.

Reaction Requirements

Treatments

- The order is updated.

Linked communications

- The order form is sent to the supplier.

After of the analysis of the event description template, we can represent the communicative event Sale 1 using the modelling tool.

Creation of workspace

To use the Eclipse modelling tool, it is necessary launch it, in this way, we go to the Eclipse environment, to the workspace with our project, then right-click the diagram file and select **Run As -> Eclipse Application** (see Figure 144). This action opens a new instance of Eclipse workspace. This workspace does not have projects (This happen the first time when the workspace was not launched before).

Into the new instance, Click the menu option **File** and select **New -> Project -> General -> Project**. Assign the **Project name** (for instance "SuperStationery-Example") and click the **Finish** button.

Then, right-click the SuperStationery project file and selects **New -> Other -> Example EMF Model Creation Wizards -> Cametamodel Model**. Assign the **File name** (for instance "SuperStationery") and click the **Next** button. Then, select the **Model Object** to create, here chose in the combo box the object named **Model**. The XML Encoding might be **UTF-8**. Click the **Finish** button. At this point, the workspace looks like Figure 145.

Now, becoming to the textual description about SuperStationery Co. we want to focus the communicative event Sale 1 (A client places an order). Thus, the information presented at the event description template is used to complement the communicative event diagram.

Several information of the event description template could be stored in an Eclipse project through it representation in the communicative event diagram. This is because the metamodel specifies primitives for communicative event diagram, and the event specification template (i.e., message structure specification, see the subsection 3.4 for more details about the metamodel specification).

We intend to specify the event Sale 1 in the Eclipse workspace previously created.

The Figure 88 presents the composition of the modelling tool. In the project explorer we can manage the projects (assign name project, create new project, change to other project, etc). In the Modelling

space we can to model the communicative event diagrams. The tool palette provides the modelling elements.

Now, we can to start the modelling of the communicative event Sale 1. So, there are two ways to create a new communicative event. Go to the palette and click the element named “Communicative event”, then, click in the modelling space. A new communicative event appears in the modelling space. Another way is to put the mouse on the modelling space, immediately, a contextual menu will appear, when this contextual menu appears, click on the communicative event symbol. A new communicative event appears in the modelling space.

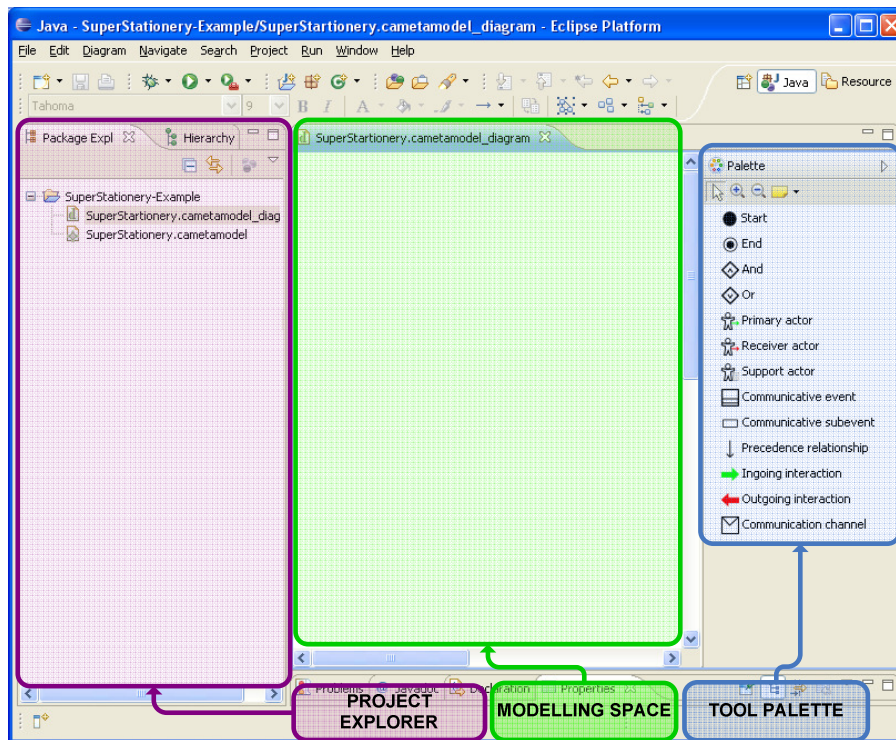


Figure 88. Composition of the CED graphical editor

To define the properties of the communicative event, right-click the communicative event and to select the option **Show Properties view**.

This action opens new tab properties. There is possible to specify the name, the Id, the number, and the whole properties of communicative events (see Figure 89).

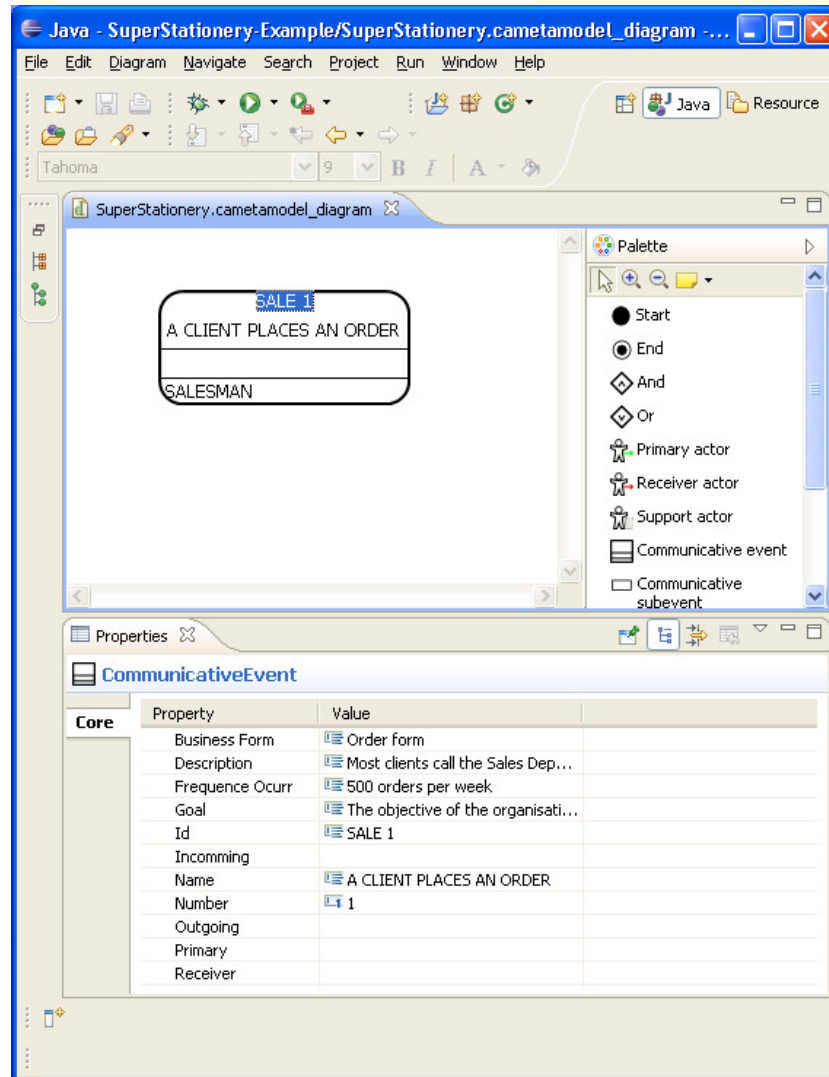


Figure 89. Specifying the communicative event SALE 1

The event Sale 1 has the support actor named Salesman. The receiver actor can be added to the communicative event in two ways. From

the contextual menu, put the mouse over the communicative event and a contextual menu will appear to add a new support actor. Another way is to add the support actor from the palette. To click the element support actor and click on the communicative event to add the new support actor. In the tab properties is possible to specify the description and the other properties of support element.

The fields incoming, outgoing, primary and receiver are related to the relationships of the communicative event.

The primary actor of the communicative event Sale 1 is Client. To assign the primary actor is possible to follow the same steps that were explained for communicative event. Go to the palette and click the element named "Primary actor", later, click in the modelling space. A primary actor element appears on the modelling space (see Figure 90).

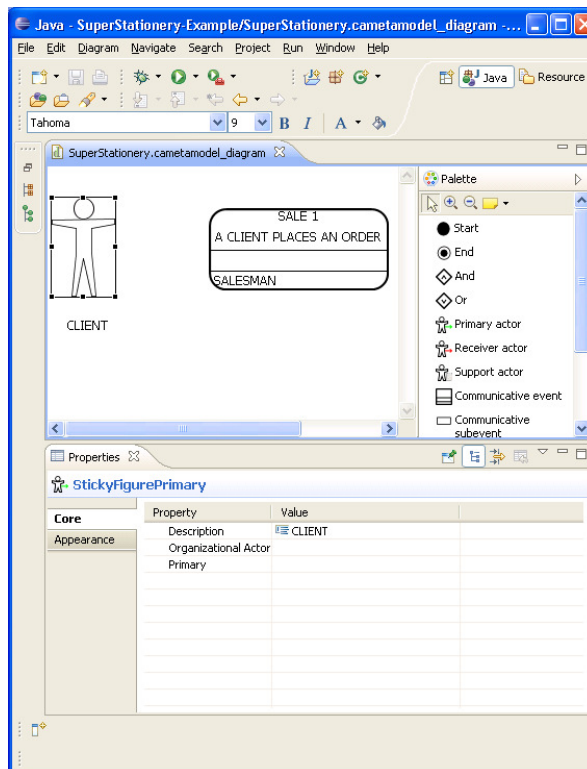


Figure 90. Specifying a primary actor for SALE 1

It is possible to add a primary actor from contextual menu that appears when the mouse is over the modelling space.

In the tab properties is possible to specify the description and the organizational actor. The primary field specifies the relationships with a communicative event.

The receiver actor of the communicative event Sale1 is Sales Manager. To add the receiver actor, follow the steps like primary actor (see Figure 91).

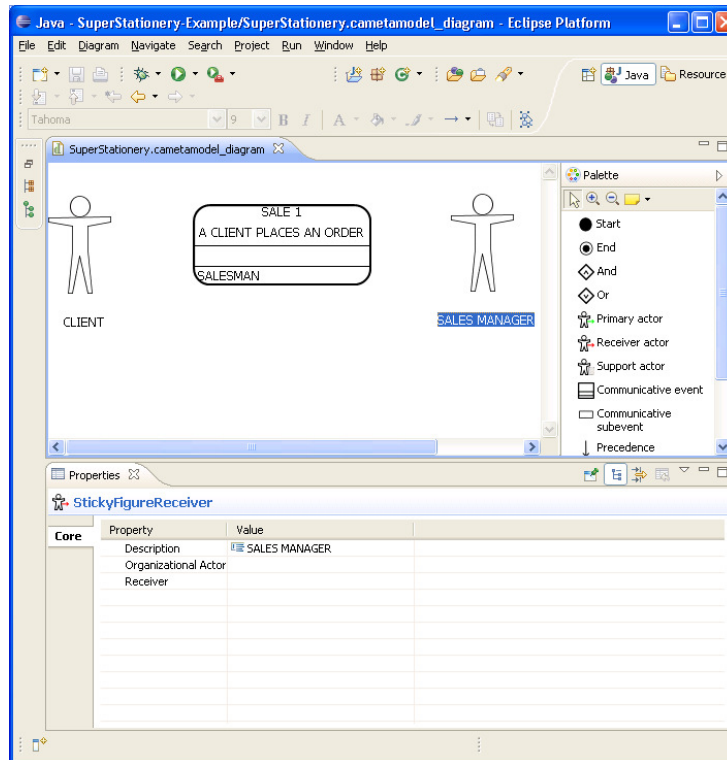


Figure 91. Specifying a receiver actor for SALE 1

Now, the relationships among the actors and the communicative events can be specified. So as to click the element named "Ingoing interaction" to add the relationship between the primary actor and the communicative event. Click on the primary actor and to bring the mouse to the communicative event. Another way is to put the mouse

over the primary actor and to bring the mouse to the communicative event. These steps can be followed to add the outgoing interaction between the communicative event and the receiver actor. The state of the CED for Sale 1 is showed at Figure 92.

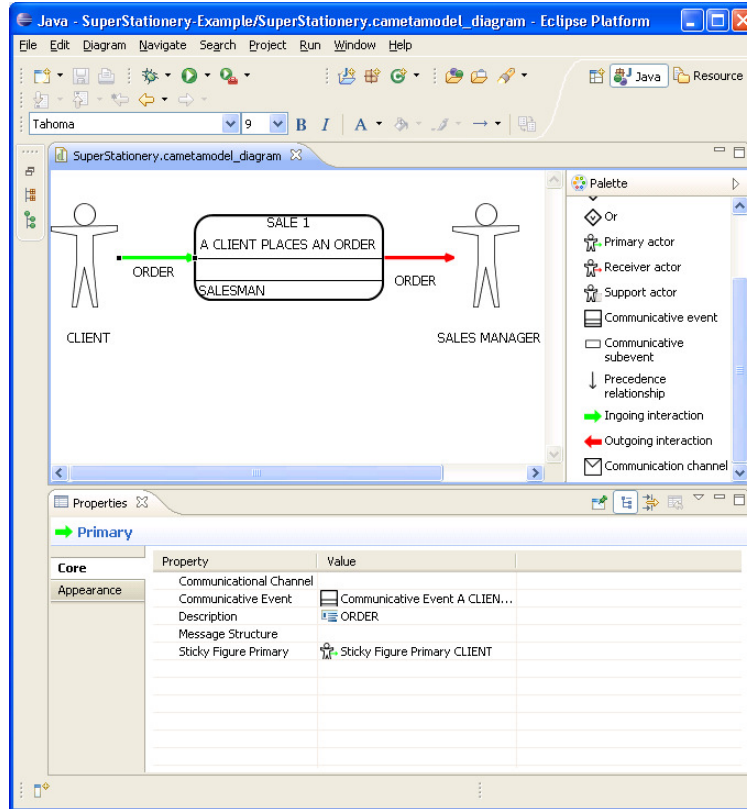


Figure 92. Specifying ingoing and outgoing interactions for SALE 1

Thus, after of to analyse the other communicative events that are part of the Sale business process, we can to have the CED showed at Figure 93.

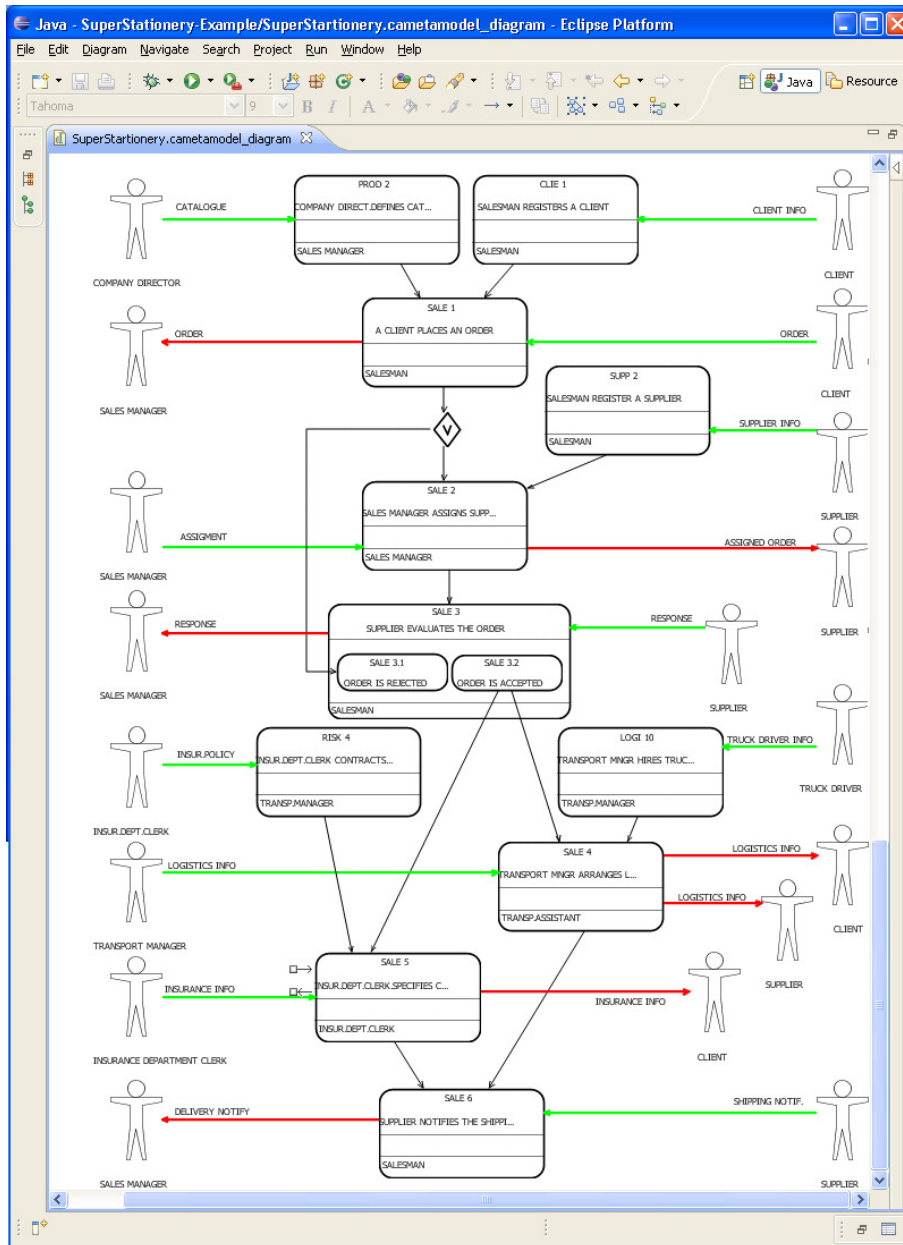


Figure 93. Communicative Event diagram for Sale business process modelled in the graphical editor.

Support to Message Structure

In the event description template for **Sale 1**, is specified the message structure. We have developed two supporting tools to include the message structure specification in the CED: one uses the Xtext technology, and the other uses the Eclipse Modelling Framework [38].

Support to Message Structures with Xtext

Message Structures is a modelling language based on structured text, that can be specified using the Extended Backus-Naur Form notation [38]. This characteristic facilitates the development of a domain-specific language (DSL) tool. Figure 94.a shows the Message Structure grammar as defined in the Xtext environment, an Eclipse-based environment for the development of textual DSLs [43].

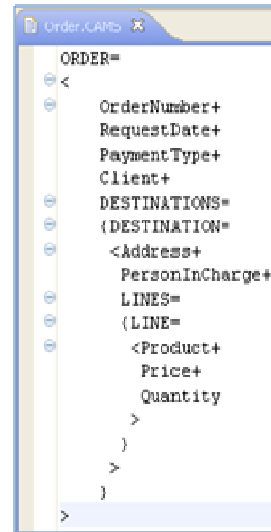
This environment allows the automatic generation of textual editors for the defined DSLs. Figure 94.b shows the specification of the message structure **ORDER**, using the Xtext tool. An advantage of this environment is that it can be integrated with other Eclipse-based modelling tools.

```

grammar org.xtext.example.mydsl.CAMS with
org.eclipse.xtext.common.Terminals
generate cAMS "http://www.xtext.org/example/mydsl/CAMS"
MessageStruc:
strucName +=StrucName
(initialSubstruc +=InitialSubstruc);
StrucName:
strucName=ID '=';
InitialSubstr:
(aggregationSubstruc +=AggregationSubstruc) |
(iterationSubstruc +=IterationSubstruc);
AggregationSubstruc:
'<'(substrucList +=SubstrucList)'>';
IterationSubstruc:
'{'(substrucList +=SubstrucList)'}';
SpecialisationSubstruc:
'['(substrucList +=SubstrucList)
'|' (substrucList +=SubstrucList)']';
SubstrucList:
(substruc+=Substruc) ('+'(substruc+=Substruc))*;
Substruc:
(field +=Field) |
substrucName=ID'='
(complexSubstruc+=ComplexSubstruc);
Field:
fieldName=ID;
ComplexSubstruc:
(aggregationSubstruc+=AggregationSubstruc) |
(iterationSubstruc+=IterationSubstruc) |
(specialisationSubstruc+=SpecialisationSubstruc);

```

a) DSL definition in Xtext for Message Structures



b) Example of a message structure

Figure 94. Support to Message Structures with the Xtext environment

Support to Message Structures with Eclipse Modelling Framework

We have presented a metamodel that specifies the communicative event diagrams and message structures as part of the Communication Analysis method. This metamodel was designed to allow modelling message structures. The Figure 95 shows a part of this metamodel. The metaclasses that have black border represent the primitives for message structures.

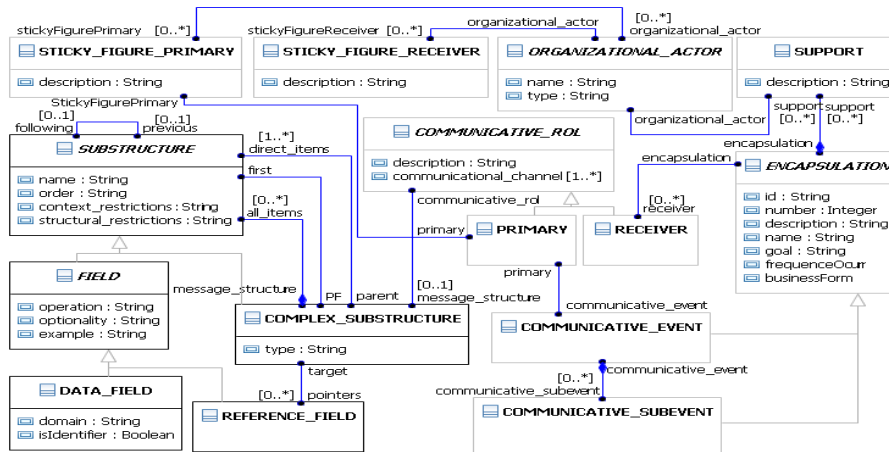


Figure 95. Support to Message Structures with EMF

Figure 96 shows the message structure as an instantiation of the metamodel, in the form of an Ecore tree. The tree graphically represents the composition of complex substructures, leaving the operators = and + implicit. The type of each substructure is stored in the property type of the metaclass COMPLEX_SUBSTRUCTURE (e.g. the tab named Properties shows that the complex substructure DESTINATIONS is of type iteration).

On the one hand, the implementation in Xtext ensures the compliance with the EBNF grammar for Message Structures and it offers an editorial environment that is more efficient and usable. On the other hand, the implementation in EMF extends the CASE tool for Communication Analysis; moreover, its Ecore metamodel offers the possibility of defining model to model transformations using languages such as ATL Transformation Language (ATL [44]) or Query/View/Transformation (QVT [45]). In any case, both implementation approaches are complementary.

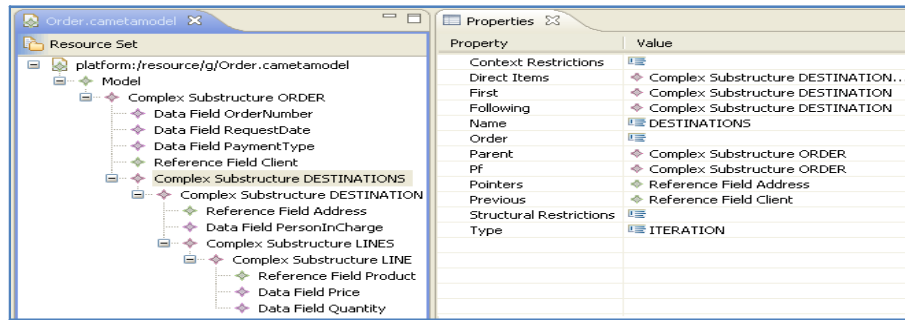


Figure 96. Example of message structure supported by the EMF environment

4.5 Modelling tool validation

Due to the increasing interest in the MDA paradigm, the conceptual models have become the backbone of the software development process. So far some methods exist to develop a user interface according to a MDA-compliant method, none of them explicitly connects usability to their process activities[46].

Many approaches to evaluate the usability of software systems have been proposed in the last years [47], [48], [49], [50], [51], [52]. Most of them focus on defining a set of attributes that explains usability and on developing guidelines and heuristics for testing it.

According to the integration framework, the Figure 97 presents the activity of evaluation of the diagramming tool as part of this proposal.

This subsection intends to show two ways were carried out to validate the diagramming tool. The diagramming tool should to support the activities of the analyst. This means that, the diagramming tool should to provide a usable interface. The goal is to increment the productivity and efficiency of the analyst[32]. With this goal in mind, we have proposed two types of validations: a heuristic evaluation of usability and expressiveness and a usability evaluation with users (user testing). These validations are explained below:

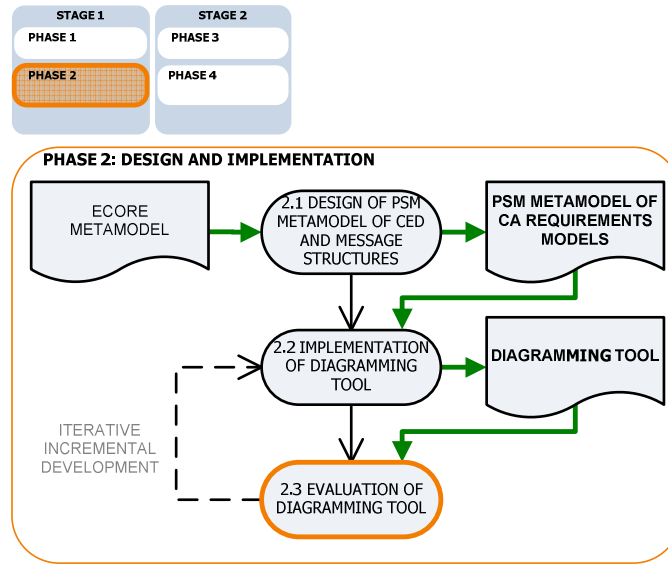


Figure 97. Activity of evaluation of diagramming tool

4.5.1 A heuristic evaluation of usability and expressiveness

For the heuristic evaluation of usability and expressiveness, an expert in the Communication Analysis method has tested the modelling tool. With this test, we intend to have a first report about the usability and expressiveness of the modelling tool from an expert point of view.

The ten principles proposed by Nielsen [53] have been used as heuristic rules (see Table 86).

<p>1. VISIBILITY OF SYSTEM STATUS</p>	<p>The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.</p>
<p>2. MATCH BETWEEN SYSTEM AND THE REAL WORLD</p>	<p>The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information</p>

	appear in a natural and logical order.
3. USER CONTROL AND FREEDOM	Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
4. CONSISTENCY AND STANDARDS	Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5. ERROR PREVENTION	Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
6. RECOGNITION RATHER THAN RECALL	Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
7. FLEXIBILITY AND EFFICIENCY OF USE	Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8. AESTHETIC AND MINIMALIST DESIGN	Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
9. HELP USERS RECOGNIZE,	Error messages should be expressed in

DIAGNOSE, AND RECOVER FROM ERRORS	plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
10. HELP AND DOCUMENTATION	Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Table 86. Heuristics rules

The classification of usability problems (CUP) scheme is used for the purpose of specifying the usability problems found during the heuristic evaluation [54]. This classification provides a clear framework for the analysis of the usability of the modelling tool for Communication Analysis method. It facilitates the classification of problems found during the heuristic usability evaluation. As recommended in [55], the CUP scheme has been adapted to the needs of the current evaluation; the fields of the resulting template are explained below (see Table 87).

ID	Unique identifier for referencing each usability problem. In this field, puts a numeric value that is increasing as the number of problems that have been found.
DESCRIPTION	Simplified textual description of the problem of usability found.
TRIGGER	Description of the heuristic rule applied to the problem of usability found. In this field, put the number of heuristic rule is related to the usability problem found (see Table 86)
IMPACT	Analysis of problems encountered in the implementation and impact values. The allowed values to describe the impact are: 1. Minor: Problems involved in the execution of the tasks of

	<p>the application, but these problems do not interfere with which they are carried out.</p> <p>2. Moderate: Problems that significantly prevent the completion of the application's tasks, where users can find an alternate way to complete the task.</p> <p>3. Severe: Problems that do not allow the user to successfully complete the tasks being carried out in the application.</p> <p>In this field, puts the number of values to describe of impact of usability problem found.</p>
PREVENTION OF USABILITY PROBLEMS	<p>Ideas on how you can be carried out various activities in the implementation.</p> <p>In this field, indicate a simplified textual description about a better way for activity carry out.</p>

Table 87. Description of the adapted classification of usability problems (CUP) scheme

After of the evaluation carried out by the expert, it was found four problems of expressiveness and 25 problems of usability. The expert took five hours to evaluate the modelling tool and to write the report into the CUP scheme (see Table 87).

4.5.2 An usability evaluation with users (user testing)

User testing provides valuable design feedback, but generally this feedback arrives in the later stages of development when it is difficult to modify the design. For this reason we propose involve the user experience with the modelling tool into the first stages of the development.

After of the heuristic evaluation, we carried out a usability evaluation with users. This usability evaluation was designed with the advice of the professor Jean Vanderdonckt. He is professor at the Université Catholique de Louvain (UCL). He is head of Belgian Laboratory of Computer-Human Interaction (BCHI) and Coordinator of the

UsiXML Consortium [56]. He is author of several research papers about human-computer interaction and one book [57].

The usability evaluation with users was carried out after to solve some problems found during the heuristic evaluation (see subsection 4.5.1). An improve version of the modelling tool was used to carry out the evaluation with users and this version was used to design the exercises for the usability evaluation of the modelling tool.

We have identified clearly two profiles, these profiles will be defined below following the technique of PERSONA from Alan Cooper [58]. We have followed a research paper that describe the advantages of use this technique to define profiles, moreover, this analysis allow us to focus on the end users of our evaluation, their task, activities and knowledge[59].


The use of persona allow us to identify clearly the different profiles for our user testing activity, in addition, the task, questionnaires, activities and the other aspects of the user testing evaluation was been adapted to these profiles. We think that the profiles identification it was an important step for the activity of the usability evaluation, especially for the team of the design of the exercise, because this helped us to communicate the ideas according the defined profiles and also to provide an exercise according to the user capabilities.

Users description

The usability evaluation was carried out with master students of software engineering. These students have a profile close to the profile of the end-user of the modelling tool.

We can to distinguish two types of users. There are described below (Figure 98 and Figure 99):

Ana




Ana is 28 years old. She works as analyst of the information system at a pension insurance company.

Ana carries out several activities to maintain the software application that support the pension business. These activities include the elicitation of the requirements specification. She uses the Communication Analysis method to elicitate and to analyse the requirements. This means that she has several meetings with the end-users of the system, and after of this, she draws a plan to develop the requirements captured.

Ana uses different tools to support the manage of the requirements. She uses Microsoft Visio to specify the communicative event diagram models, and also she uses Microsoft Word and Excel to specifies the message structures and event description template.

Figure 98. Description of the profile 1: Ana

Jhon



Jhon is 28 years old. He works as analyst of the information system at a pension insurance organization.

John carries out several activities to maintain the software application that support the pension business. These activities include the elicitation of the requirements specification. He does not uses requirements elicitation method to elicitate and to analyse the requirements. This means that the He have several meetings with the end-users of the system, and after of this, He draws a plan to develop the requirements specified.

For this activities, Jhon does not uses modeling tools that help him with the manage of the requirements. He uses Microsoft Word and Excel to specifies the information that he wants to analyse.

Figure 99. Description of the profile 2: Jhon

At last, with the profiles clearly identify, we have proceeded to design the scenario for the usability evaluation with the users.

To follow, we describe the activities, a short description of each one and the allowed time respectively.

Design of the scenario for the usability evaluation

The activities that were carried out are described in the planning showed on Table 88.

ACTIVITY	DESCRIPTION	ALLOTTED TIME
Demographic questionnaire	General information about the subjects is collected.	10 minutes
Demonstration	A researcher demonstrates the usage of the modelling tool by carrying out a task similar to the one that the user will perform.	5-10 minutes
Exploring the application	Users are given some time to discover by themselves the various features of the modelling tool.	10 minutes
Task performance	Subjects are given a task that they have to carry out by using different features of the modelling tool.	25-30 minutes
Break	Users take a break.	5 minutes
Usability questionnaire	Subjects complete the CSUQ questionnaire and some additional free-response questions.	15-20 minutes
Focus group	A focus group (a type of moderated team inter-	25-30 minutes

	view) is conducted. The aim is to gather qualitative information about the impressions of the subjects about the modelling tool; for this purpose, the user responses to free-responses questions are discussed.	
--	--	--

Table 88. Planning of the usability evaluation

First, the students filled the demographic questionnaire (see Figure 100).

Demographic Questionnaire

Q1. Mark your gender:
 Male: ___ Female: ___

Q2. Indicate your age: ___

Please answer the following questions about your previous experience with modelling tools.

Explanation of the scale:
 1 = No experience, I have never used these type of tools
 2 = I have used these type of tools to create simple models for training purposes (e.g. in undergraduate or master courses)
 3 = I have used these type of tools to create moderately complex examples for training purposes
 4 = I use these type of tools occasionally in my professional projects to create mid-sized projects
 5 = I use these type of tools almost daily in my professional projects

		1	2	3	4	5
Q3	What is your experience with general-purpose diagramming tools (e.g. Microsoft Visio, Dia)					
Q4	What is your experience with CASE modelling tools (e.g. Rational Rose, ArgoUML)					

Please answer the following question about your previous experience with modelling methods.

Explanation of the scale:
 1 = I have never user this method
 2 = I have seen examples in class
 3 = I have solved small exercises
 4 = I have solved moderately complex cases
 5 = I have solved real cases professionally

		1	2	3	4	5
Q5	Business process modelling, in general (using any kind of notation)					
Q6	Communication Analysis models (a specific requirements method)					

Figure 100. Demographic questionnaire

The results were analysed and are presented below. The Gender, Age and some questions about previous knowledge were specified into the demographic questionnaire.

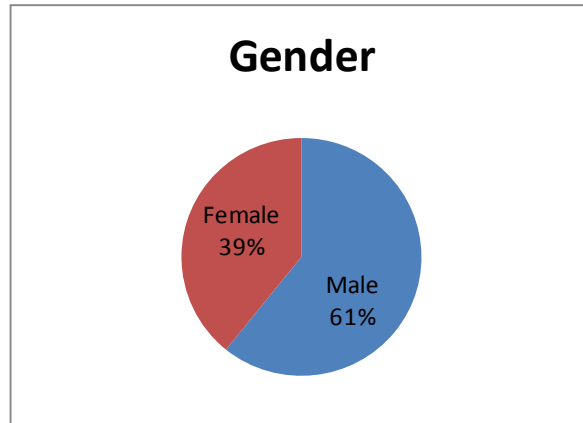


Figure 101. Percentage of male and female.

The median of the age of the participants is 27 years old.

The level of knowledge of the participant in modelling tools, CASE tools and requirements models are presented below:

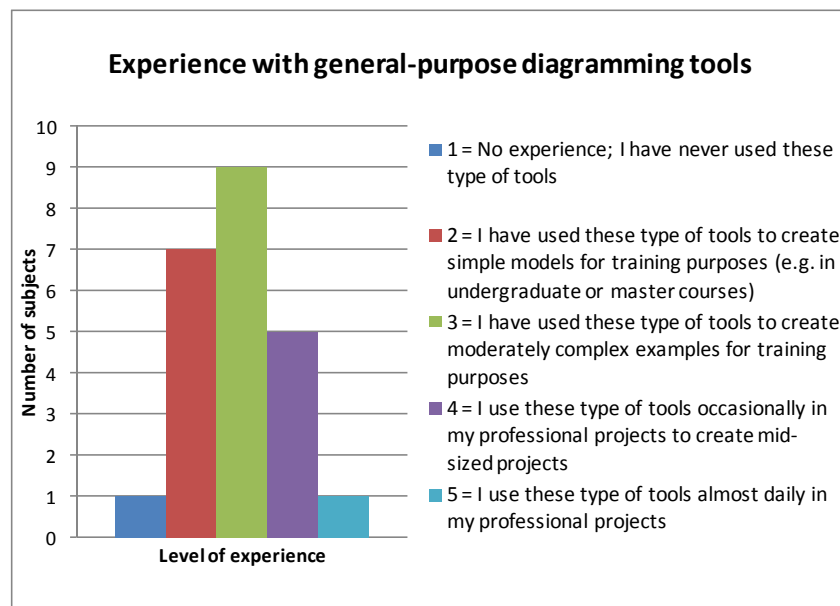


Figure 102. Experience with general purpose diagramming tools

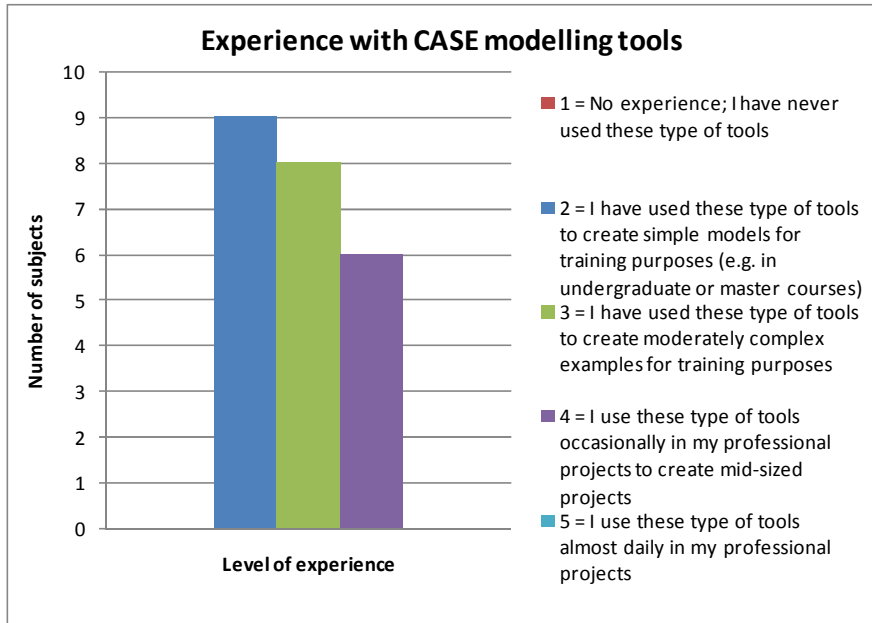


Figure 103. Experience with CASE modelling tools

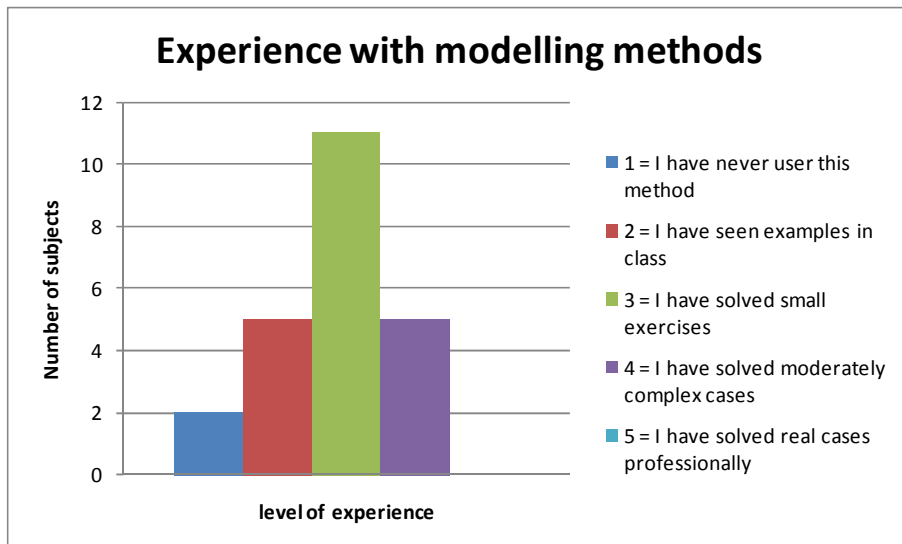


Figure 104. Experience with modelling methods

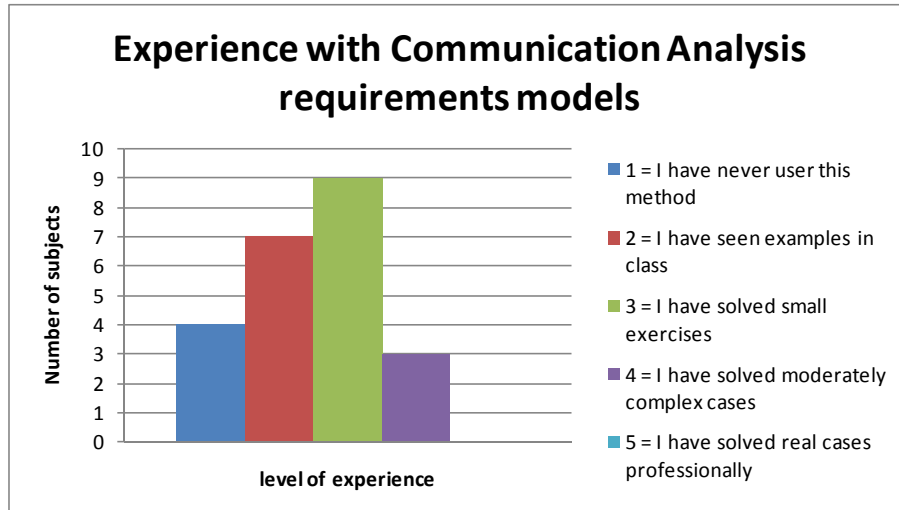


Figure 105. Experience with Communication Analysis requirements models

These results allow us to know about the level of expertise of the students in general. According to these graphics, the users have been familiarised with diagramming tools, modelling methods, CASE tools and they have experience with the requirements models of Communication Analysis method. Thus, the participant of the experiment is persons that can to carry out the exercise of the experiment without problems and also they can give us feedback according their previous experience with requirement methods, modelling tools, etc.

Then, the researcher starts the demonstration explaining how to represent a communicative event diagram into the modelling tool.

Later, the users can to explore the modelling tool and to ask some questions to the researcher.

Thus, the communicative event diagram showed at Figure 106 was given to the users in a print format. The users should to performance the task into the modelling tool, they has 30 minutes for drawing the model into the modelling tool.

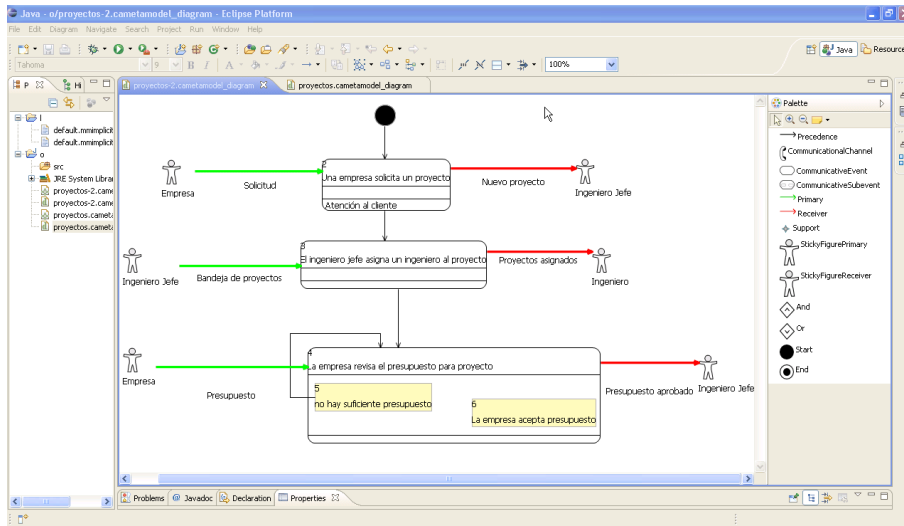


Figure 106. Task done in the modelling tool.

After the break, the users response, the computer system usability questionnaire CSUQ [60]. These questions ask to the users about their points of view about the modelling tool. Through the CSUQ questionnaire is possible to identify three factors that corresponds with some questions of the CSUQ; these factors corresponds with some human factors related to the Information System (see Figure 107). These factors are System Usefulness (SYSUSE), Information Quality (INFOQUAL), and Interface Quality (ITERQUAL). The answers of the CSUQ questionnaire are in a scale from 1 to 7, then, we can to analyse the mean of the answers for each factor and the mean of the overall.

		DISAGREE				AGREE		
		1	2	3	4	5	6	7
1.	Overall, I am satisfied with how easy it is to use this system.							
2.	It was simple to use this system.							
3.	I can effectively complete my work using this system.							
4.	I am able to complete my work quickly using this system.							
5.	I am able to efficiently complete my work using this system.							
6.	I feel comfortable using this system.							
7.	It was easy to learn to use this system.							
8.	I believe I became productive quickly using this system.							
9.	The system gives error messages that clearly tell me how to fix problems.							
10.	Whenever I make a mistake using the system, I recover easily and quickly.							
11.	The information (such as online help, on-screen messages, and other documentation) provided with this system is clear							
12.	It is easy to find the information I needed.							
13.	The information provided for the system is easy to understand.							
14.	The information is effective in helping me complete the tasks and scenarios.							
15.	The organization of information on the system screens is clear.							
16.	The interface of this system is pleasant.							
17.	I like using the interface of this system.							
18.	This system has all the functions and capabilities I expect it to have.							
19.	Overall, I am satisfied with this system.							

Figure 107. Computer System Usability Questionnaire (CSUQ)

The mean of the answers of the CSUQ questionnaire is presented in a box plot presented at Figure 108.

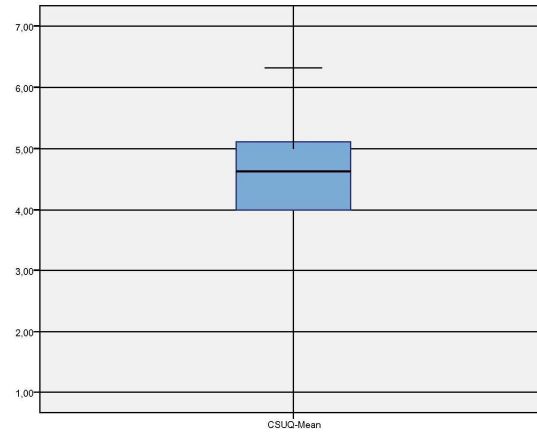


Figure 108. Mean of the answers of the CSUQ questionnaire

The answers of the CSUQ questionnaire have a mean of 4.57. Having into account the maximum value of the CSUQ scale is 7; with this result we can infer that the most of the answers had a positive result.

The SYSUSE factor indicates the perception of the users about the system usefulness. This factor corresponds with the questions 1 to 8.

The mean of the answers is of 5,53, the Figure 109 presents a box plot with the values of the answers.

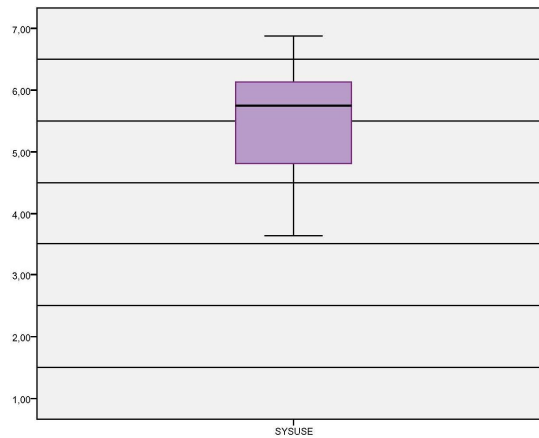


Figure 109. Mean of the answers of the SYSUSE factor

The INFOQUAL factor indicates the perception of the users about the information quality of the system. This factor corresponds with the questions 9 to 15. The mean of the answers is of 3, 59, the Figure 110 presents a box plot with the values of the answers.

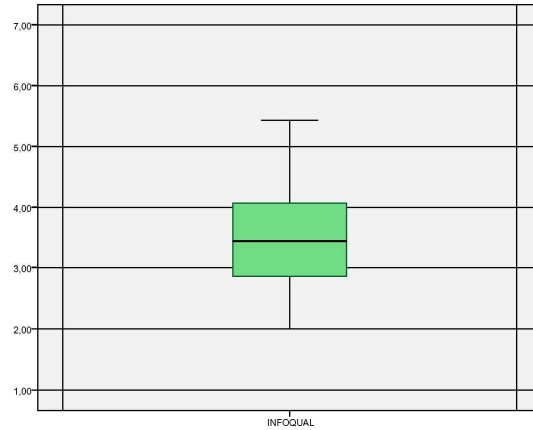


Figure 110. Mean of the answers of the INFOQUAL factor

The INTERQUAL factor indicates the perception of the users about the interface quality of the system. This factor corresponds with the questions 16 to 18. The mean of the answers is of 4, 43, the Figure 111 presents a box plot with the values of the answers.

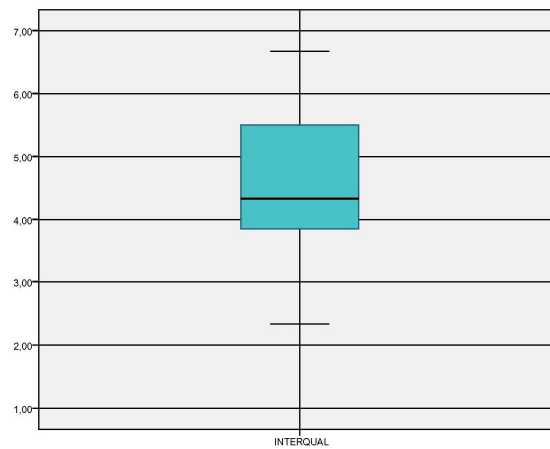


Figure 111. Mean of the answers of the INTERQUAL factor

Later, the students filled the questionnaire with some additional questions (Figure 112).

Additional questions	
<p>A. Please list three positive features of the modelling tool (i.e. three characteristics of the tool that you appreciate)</p> <div style="border: 1px solid black; padding: 5px;"> <p>1</p> <p>2</p> <p>3</p> </div>	<p>E. When you need to make changes to previous work, how easy is it to make the change? Which particular changes and why?</p> <div style="border: 1px solid black; height: 60px;"></div>
<p>B. Please list three negative features or drawbacks of the modelling tool (i.e. three characteristics of the tool that you dislike)</p> <div style="border: 1px solid black; padding: 5px;"> <p>1</p> <p>2</p> <p>3</p> </div>	<p>F. Do some kinds of mistake seem particularly common or easy to make? Which ones?</p> <div style="border: 1px solid black; height: 60px;"></div>
<p>C. Please estimate what percentage of the task you have successfully performed?</p> <hr style="width: 100%;"/>	<p>G. After completing this questionnaire, can you think of obvious ways that the design of the system could be improved? What are they?</p> <div style="border: 1px solid black; height: 60px;"></div>
<p>D. What kind of features of the tool or modelling elements are more difficult to see or find?</p> <div style="border: 1px solid black; height: 60px;"></div>	

Figure 112. Additional free-response questions

The additional questions are classified in this manner:

The questions A, B, and D were classified according to the usability model [61]. The attributes of this usability model constitutes a taxonomy of the aspects that are part of the usability of software application. We have follow the usability model to build a complete and detailed taxonomy, then, the results of the questionnaires was added in order to obtain the results showed at Table 89 and Table 90.

The answers were classified according to the attributes of the usability model. Negative and positive features were been into account.

FEATURE	Q
POSITIVE FEATURES	
1. Learnability	2
1.1. Help Facilities	0
1.2. Predictability	0
1.3. Informative Feedback	0
1.4. Memorability	0
TOTAL	2
2. Understandability	1
2.1. Legibility	4
2.2. Readability	2
2.3. Familiarity	0
2.4. Workload Reduction	7
2.5. User Guidance	0
TOTAL	14
3. Operability	1
3.1. Installability	0
3.2. Data Validity	0
3.3. Controlability	1
3.4. Capability of Adaptation	1
3.5. Consistency	0
3.6. Error Management	0
3.7. State System Monitoring	0
TOTAL	3
4. Attractiveness	0
4.1. Background Color Uniformity	0
4.2. Font Color Uniformity	0
4.3. Font Style Uniformity	0
4.4. Font Size Uniformity	0
4.5. UI Position Uniformity	0
4.6. Subjective Appealing	0
TOTAL	0
5. Compliance	0
5.1. Degree of Fulfillment with the ISO/IEC 9126	0
5.2. Degree of Fulfillment with the ISO 9241-10	0
5.3. Degree of Fulfillment with the Microsoft style guide	0
5.4. Degree of Fulfillment with the Java style guide	0
TOTAL	0

Table 89. Positive features

FEATURES	Q
----------	---

NEGATIVE FEATURES	Q
-------------------	---

1. Learnability	0	4.1. Background Color Uniformity	0
1.1. Help Facilities	1	4.2. Font Color Uniformity	0
1.2. Predictability	3	4.3. Font Style Uniformity	0
1.3. Informative Feedback	1	4.4. Font Size Uniformity	0
1.4. Memorability	0	4.5. UI Position Uniformity	0
TOTAL	5	4.6. Subjective Appealing	0
2. Understandability	0	TOTAL	0
2.1. Legibility	2	5. Compliance	0
2.2. Readability	0	5.1. Degree of Fulfillment with the ISO/IEC 9126	0
2.3. Familiarity	0	5.2. Degree of Fulfillment with the ISO 9241-10	0
2.4. Workload Reduction	3	5.3. Degree of Fulfillment with the Microsoft style guide	0
2.5. User Guidance	0	5.4. Degree of Fulfillment with the Java style guide	0
TOTAL	5	TOTAL	0
3. Operability	0		
3.1. Installability	0		
3.2. Data Validity	0		
3.3. Controlability	5		
3.4. Capability of Adaptation	2		
3.5. Consistency	0		
	3		
3.6. Error Management			
3.7. State System Monitoring	0		
TOTAL	10		
4. Attractiveness	0		

Table 90. Negative features

The answers of the question C indicate us the perception of the user about the percentage of the task successfully performed (see Figure 113).

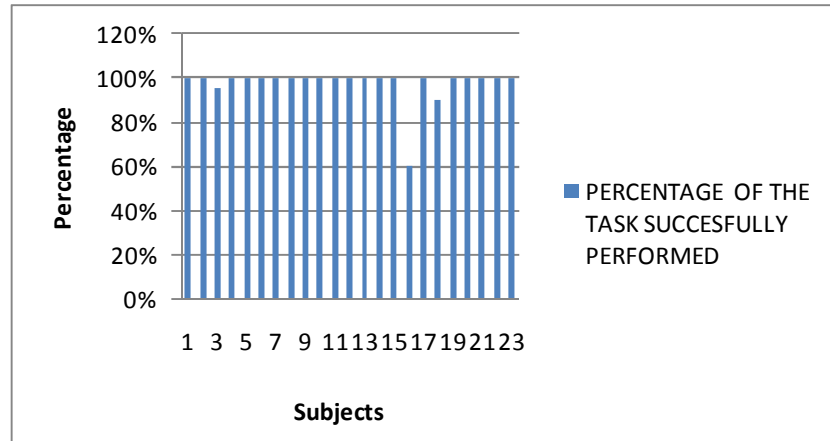


Figure 113. Percentage of the task successfully performed

In general, the perception of the users about the percentage of the task successfully performed is of 100%. This indicates that the users have perceived that they can do the task completely.

The questions E and F were classified according to the cognitive dimensions of information artefacts purposed by [62].

Cognitive dimensions are a tool to aid non-HCI specialists in evaluating usability of information-based artefacts (summative evaluation). Since they are addressed to non-specialists, they consciously aim for broad-brush treatment rather than lengthy, detailed analysis. Their checklist approach helps to ensure that serious problems are not overlooked.

The dimensions treated in this evaluation were the viscosity, visibility and juxtaposability and the error-proneness.

The viscosity is the resistance to change, the cost of making small changes.

We have classified the questions about the viscosity taking into account positive and negative sides; in this manner we have two groups of answers:

- There is no problem when making changes to existing diagrams.
- There are some problems when making changes to existing diagrams.

The first statement had 10 answers related to it.
The second statement had 6 answers related to it.

The visibility and juxtaposability is the ability to view components easily and the ability to place any two components side by side.

We have classified the questions about the visibility and juxtaposability taking into account positive and negative sides; in this manner we have two groups of answers:

- There are not features or modelling elements that were difficult to see or to find.
- There are features or modelling elements that were difficult to see or to find.

The first statement had 5 answers related to it.
The second statement had 16 answers related to it.

The error-proneness is the possibility that the notation invites mistakes.

We have classified the questions about the error-proneness taking into account positive and negative sides; in this manner we have two groups of answers:

- There are possible seemed mistakes common or easy to make.
- There are not mistakes common or easy to make.

The first statement had 15 answers related to it.
The second statement had 0 answers related to it.

After the classification of the answers, we have identified problems related to the usability and expressiveness. Some problems are related to the PSM metamodel, other problems are related to the Eclipse technology. Finally, the users declared that the modelling tool was easy to use, light and agile. And according to the answer C, the users can carry out the totality of the modelling exercise.

Finally, a focus group was conducted. Each user spoke aloud about his/her perceptions about the modelling tool. The users spoke about positive and negative characteristics of the modelling tool. Some sodas were given to the focus group participants to motivate the dis-

cussion. A microphone was located at the center of the room to record the opinions and comments during the focus group.



Figure 114. Focus group participants

A lot of comments and suggestions were discussed by the users. This suggestions and comments were saved and classified in an Excel file after the focus group for its future analysis.

4.6 Analysis and discussion

The development of this project phase has carried out several advances to aim the principal objective: to integrate a requirements method into a MDD environment. The specifications of the meta-models was the first step, now with the modelling tool is possible to use and to distribute a tool for showing how to use the method in a technological platform. During the development of this phase of the project we have found several challenges, for instance we have decide what is the better technological platform for supporting the requirements models, this having into account the others steps of the project as the transformation models. Another challenge was the preparation of the heuristic and usability evaluation, this exercise was so difficult because into the literature, we can not to find how to test modelling tools, how to carry out exercise as user-testing for modelling tools and what kind of heuristics were more appropriate for this type of evaluations. The usability evaluation with master students left us several lessons learned, some of these lesson are corresponding to the modelling tool and the role of the user, other lessons are about the

design of the exercise. Thanks to the advice of the professor Jean Vanderdonck who helps us a lot to design the exercise, his support and experience left us a lot of lessons learned and mark a point to start for offering a modelling tool usable and friendly with the users. We have planed to repeat the exercise of user-testing with the modelling tool improved.

5 Supporting the model transformation: From Communication Analysis requirements models to OO-Method object model

Models are now part of an increasing number of engineering processes. However, in most cases they are confined to a simple documentation role instead of being actively integrated into the engineering process. The MDD approach considers the models as first class entities and also considers the tools, repositories, etc. can be represented as models [37]. In order to these ideas, model transformation appears as a central operation. Model transformation aims to provide a mean to specify the way to produce target models from a number of source models. In this way, we have proposed this chapter with several activities that aim to transform the requirements models of the Communication Analysis method to conceptual models of the OO-Method. The strategy will be presented, the technological sup-

port, examples of the use of the transformation module and a proposal of the validation of the transformation models. In addition is presented a module of traceability support. Finally we conclude with an analysis and discussions about lessons learned.

5.1.1 Initial state

Having into account the proposal for integrating requirements models into MDD environments, We propose a stage with several activities that aim to transform requirements models specified in communicative event diagrams and message structures (two novel techniques of the Communication Analysis method) to conceptual models (conceptual models of OO-Method). This thesis proposes to follow the activities presented in stage 2 (design and implementation), please see Figure 115.

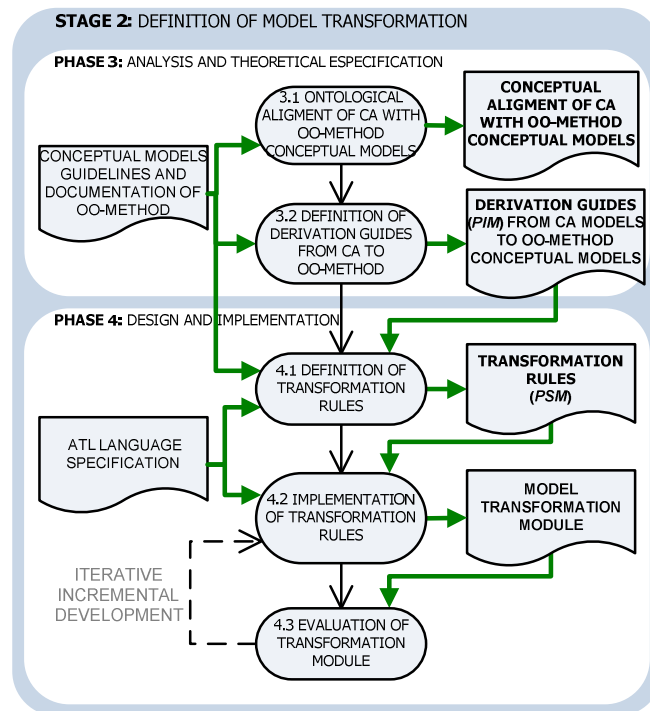


Figure 115. Stage 2 explanation

The activities corresponding to the phase 3 are part of previous work of España et al [6], [63] [64]. The ontological alignment of PIM meta-model is based on the conceptual alignment of communication analysis (see phase 1). In [6], [26] and [12] are explained the Communication Analysis perspective. In [29] is explained the OO-Method perspective.

The definition of derivation guides are presented in [63] [65]. The derivation guides are presented in natural language. This let to analyse the derivation guidelines without concepts about implementation and technological target platform (see Figure 116).

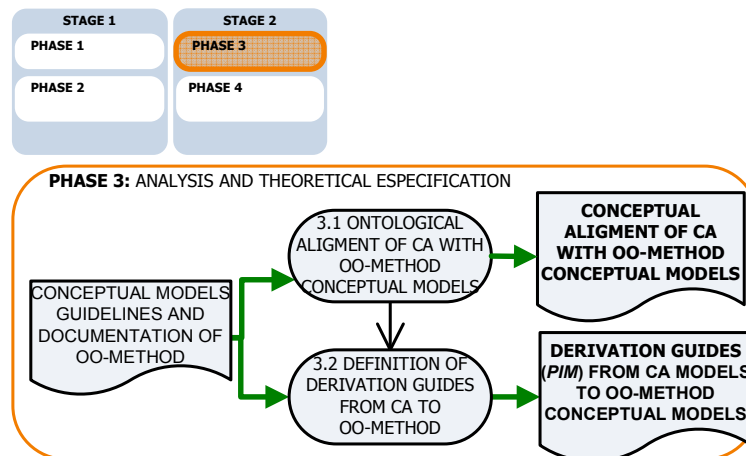


Figure 116. Phase 3 explanation

The phase 4 contains the activities about the implementation of derivation guides in a technological platform. The first activity intends to establish a set of transformation rules. The derivation guides are product of the phase 3, and these are represented in natural language. This derivation guides are used to define a set of transformation rules that can be implemented. First is interesting to have this transformation rules in a pseudocode, because it is possible to implement the transformation rules into different languages depending the target platform.

So as to implement the transformation rules, an analysis of different technological support for models transformation was necessary.

OMG purpose the Query/Views/Transformations (QVT) RFP standard [45]. Others proposal evolved the OMG process, ATLAS Transformation Language is one of them [44]. This proposal are supported by Eclipse [66] [67] and both follow a similar operational context and share some common features (see Figure 117).

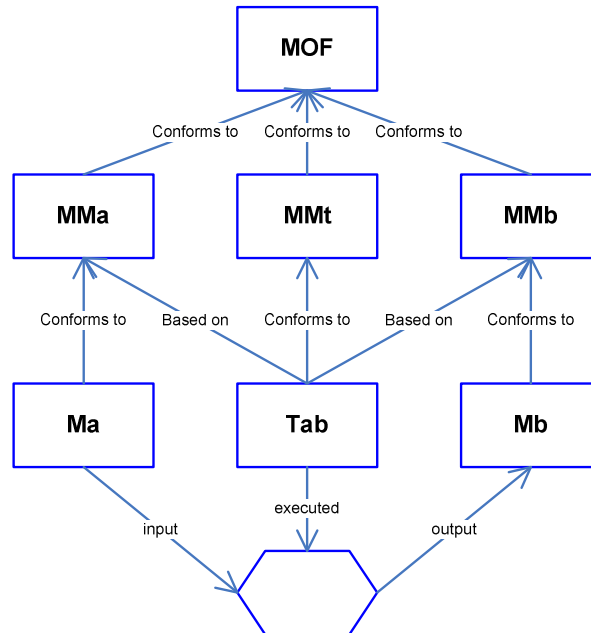


Figure 117. Operational context of ATL and QVT³

Tab is a transformation program which execution results in automatic creation of **Mb** from **Ma**. These three entities are all models conforming to **MMt**, **MMb**, and **MMa** MOF metamodels respectively. **MMt** corresponds to the abstract syntax of the transformation language. QVT and ATL provide their own metamodels defining their abstract syntaxes [68].

No a single language can be adapted to all application domains. This means that it is not different for model engineering and especially for model transformation. Simple problems can often be solved

³ Figure taken from Jouault and Kurtev [68].

using one language; more complex problems sometimes need several. Therefore, it is important to acquire knowledge about the strong and weak points of existing languages and their applicability [68].

QVT operates in the layered MOF-based metamodelling architecture prescribed by OMG. The abstract syntax of QVT is defined as a MOF 2.0 metamodel. This metamodel defines three sublanguages for transforming models. OCL 2.0 [69] is used for querying models.

The QVT language is formed of three languages with declarative and imperative constructs. The languages are named *Relations*, *Core* and *Operational Mappings*.

The languages *Relations* and *Core* are declarative languages at two different levels of abstraction. *Operational Mappings* is an imperative language that extends *Relations* and *Core* languages.

These languages can be implemented in a set or in a separate way; this depends of the transformation rules that will be specified.

ATL architecture is composed of three layers: the ATLAS Model Weaving (AMW) [70] [71], ATL and ATL Virtual Machine (ATL VM).

ATL provides both declarative and imperative constructs and is therefore a hybrid model transformation language.

The coming sections present the implementation of the rules transformation. We decided to support the rules transformation by ATL language. The rules use declarative language and imperative language to express the heuristics. Then, we consider ATL as the best option according to its architecture and hybrid language (support of declarative and imperative sentences). The technological support will be presented at section 5.2. Other advantage of the ATL is the implementation on the Eclipse platform. This let to use the metamodels that have been built. The rules implementation will be presented at section 5.3. The case of SuperStationery Co specified at the technical report available at [42] present the rules transformation carried out of a manual derivation way. The section 5.4 presents the results of to apply the ATL rules to obtain the conceptual model for SuperStationary Co lab demo. Section 5.5 presents a proposal to validate the correctness and completeness of the models resulting of to apply the

ATL transformation rules versus the models resulting of to apply the rules of a manual way. Section 5.6 presents a proposal for managing the traceability between the models. Finally, section 5.7 presents analysis and conclusions of this chapter.

5.2 Technological support

ATL is a hybrid model transformation language developed as a part of the ATLAS Model Management Architecture. ATL is supported by a set of development tools built on top of the Eclipse environment: a compiler, a virtual machine, an editor, and a debugger.[44]. According to the Figure 117, the abstract syntax of ATL takes place in **MMt**, this metamodel corresponds to ATL abstract syntax. Transformation programs are written using ATL concrete syntax.

The declarative part of ATL is based on the notion of *matched rule*. A rule consists of a source pattern matched over source models and of a target pattern that gets created in target models for every match.

ATL offers two imperative constructs: *called rule* and *action block*. A called rule is explicitly called, like a procedure, but its body may be composed of a declarative target pattern. Matched rules and called rules may be used together in a single transformation program. Action blocks are sequences of imperative instructions that can be used in either matched or called rules.

Transformation programs written in ATL are inherently unidirectional. Source models are only navigable and target models are not navigable.

An execution engine and development tools (code editor, compiler, debugger, etc.) are available on Generative Modelling Technologies (GMT) project [72]. GMT is the official research incubator project of the top-level Eclipse Modelling Project.

The execution support for ATL is on a virtual machine. The virtual machine is implemented on Eclipse Modelling Framework (EMF) [40] and Netbeans MetaData Repository (MDR) [73]. Virtual machine operations are adapted to OCL helpers implementation [69].

ATL is developed on top of the Eclipse platform, the ATL Integrated Environment (IDE) provides a number of standard development tools (syntax highlighting, debugger, etc.) that aims to ease development of ATL transformations [66].

To below, we present the implementation of the rules transformation in ATL, in order to obtain the conceptual models from specifications that follow the Communication Analysis method.

5.3 Rules implementation

The rules transformation intend to derive the OO-Method conceptual models from Communication Analysis requirement models. The derivation guidelines have been published in conference papers [63] and in a technical report [65].

The activity to implement the transformation rules is part of the phase 4 of the proposal (please see Figure 118). In this figure is possible to see how interact the different resulting products of the other activities (PSM metamodels, specifications, definitions, etc.).

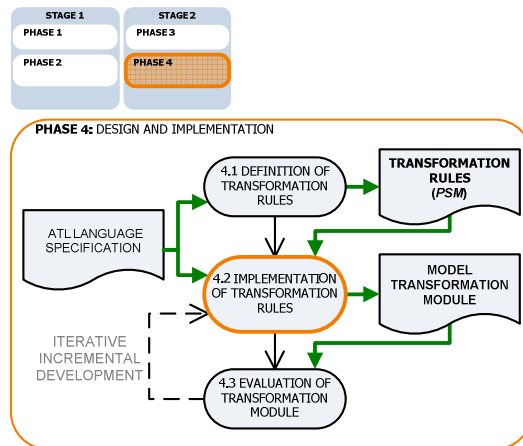


Figure 118. Phase 4, step of implementation of transformation rules

After to analyse the derivation guidelines, it was possible to represent each rule on ATL. In this way, each rule mentioned at [65] corresponds with a rule in the ATL code.

The PSM metamodel for Communication Analysis requirements models (communicative event diagrams and message structures) presented at section 4 was used as the source metamodel for transforming process (ecore metamodel).

The target metamodel for transforming process is the UML metamodel of MDT-UML2Tools of GMF [37]. The conceptual model metamodel of OO-Method not is represented in an Eclipse technology, for this reason, we have decide to use the UML metamodel of Eclipse. At section about future works (please see page 217), we purpose some alternatives to close the requirements models to conceptual models. Interoperability concepts to derive the conceptual models of OO-Method from requirements models of Communication Analysis could be a solution [74]. Although, the target model is the UML metamodel, it is still possible to see a resulting model so close to OO-Method specification. This happens because several primitives of OO-Method are specified in the UML metamodel.

The ATL code of the rules transformation is showed in the Append 2.

5.4 Transformation example

Taking into account the example showed at section 4.3, we wants to transform this communicative event diagram and message structures to its corresponding conceptual model.

The manual derivation of the conceptual model and the complete specification of the SuperStationery Co. lab demo is available in [42]⁴.

⁴ The technical report “Integrating of Communication Analysis and the OO-Method: Manual derivation of the conceptual model. The SuperStationery Co. lab demo” contains the com-

Thus, taking The SuperStationery Co. lab demo, we will see how is possible to obtain the conceptual models in an automatic way through to execute the ATL transformation code (please see the code at append 2).

To below, we present how to execute the ATL transformation code to see the conceptual model in two different specifications provides by Eclipse: tree representation and graphical representation. After to specify the communicative event diagram and to specify its corresponding message structures into the modelling tool, the transformation module can be executed.

Then, returning to the example seen in the section 4.3, to remember that the communicative event diagram and message structures for The SuperStationery Co. were specified into the modelling tool.

The following steps indicate how to execute the ATL code.

- The communicative event diagram with The SuperStationery Co. specification should be exported in a XMI format.
- Open the ATL module and Import the XMI of The SuperStationery Co. The workspace looks like the Figure 119.

plete specification for deriving the conceptual model (communicative event diagrams, message structures and event specification templates). We do not consider necessary to present this information in this thesis for simplified the explanation.

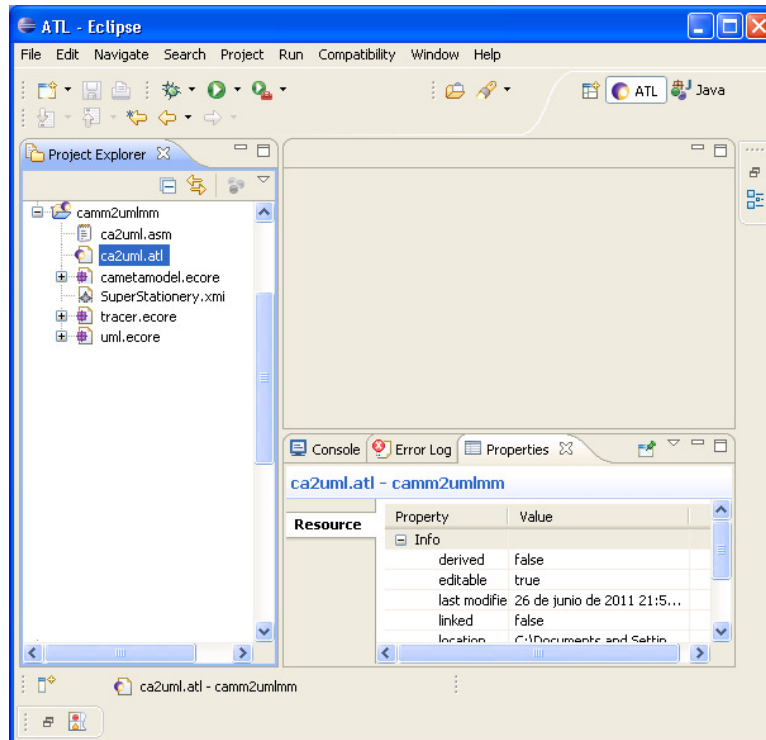


Figure 119. Workspace ATL

The workspace contains:

- **ca2uml.atl** -> File with the ATL code.
- **cametamodel.ecore** -> File with the Communication Analysis requirements metamodels.
- **SuperStationery.xml** -> Communication Analysis requirements model for The SuperStationery Co.
- **tracer.ecore** -> File with the traceability metamodel (the traceability metamodel will be explained at section 5.6).
- **uml.ecore** -> File with the UML metamodel.
- Right-click the **ca2uml.atl** file and to select **Run As -> 1 ATL transformation**.
- The run configurations should be established according to the Figure 120.

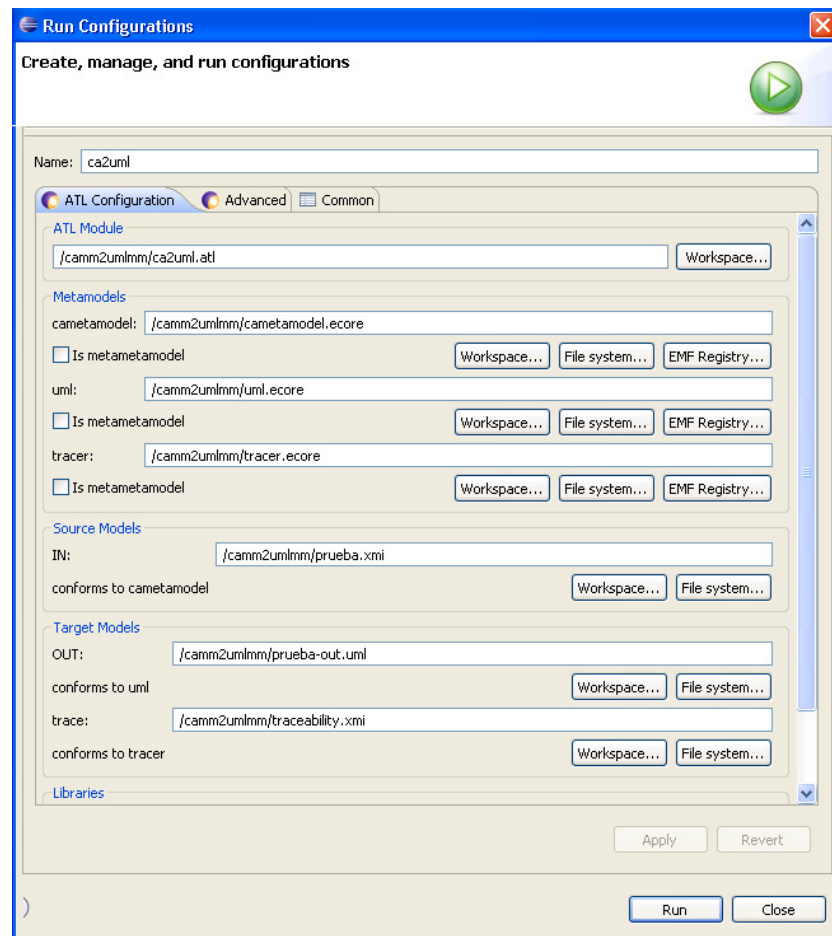


Figure 120. Run configuration for ATL transformation

- Two new files should to appear: ***SuperStationery.uml*** and ***traceability.xmi***.

The file *SuperStationery.uml* contains the derived conceptual model for SuperStationery Co. This conceptual model is represented in a tree view (please see Figure 121).

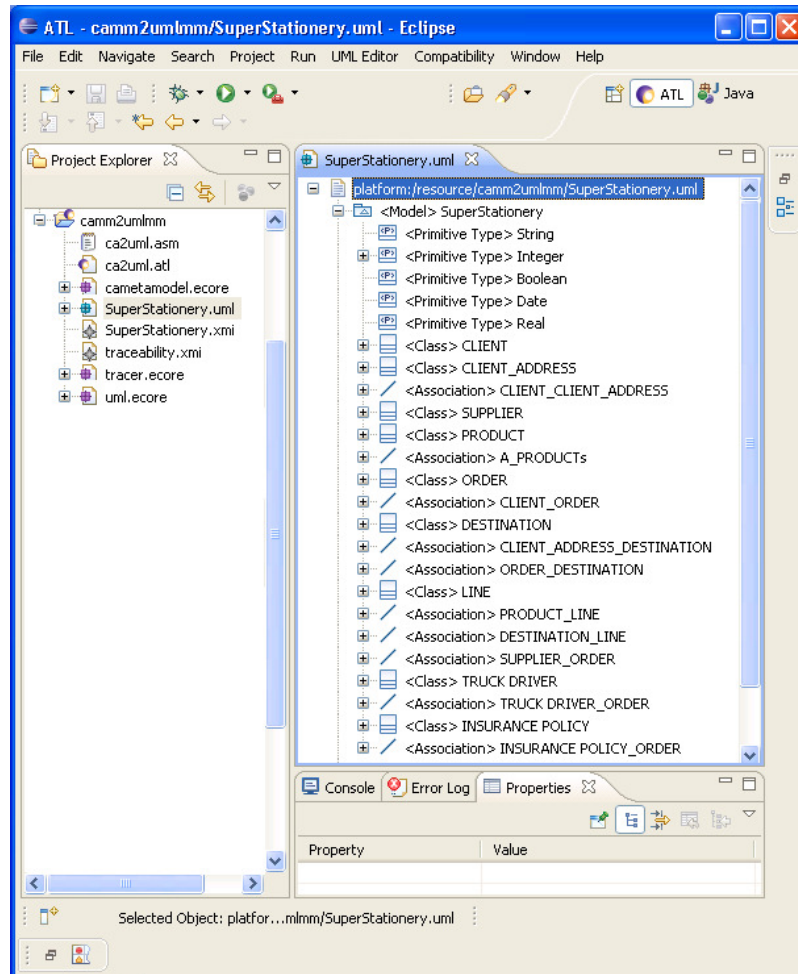


Figure 121. SuperStationery Co. Class diagram tree view

The file traceability.xmi contains information about the traceability among the elements of the metamodels. The traceability metamodel and traceability model will be explained at section 5.6.

- Right-click the SuperStationery.uml file and to select **Initialize Class Diagram**. This action generates the conceptual model in a graphical representation (please see Figure 122).

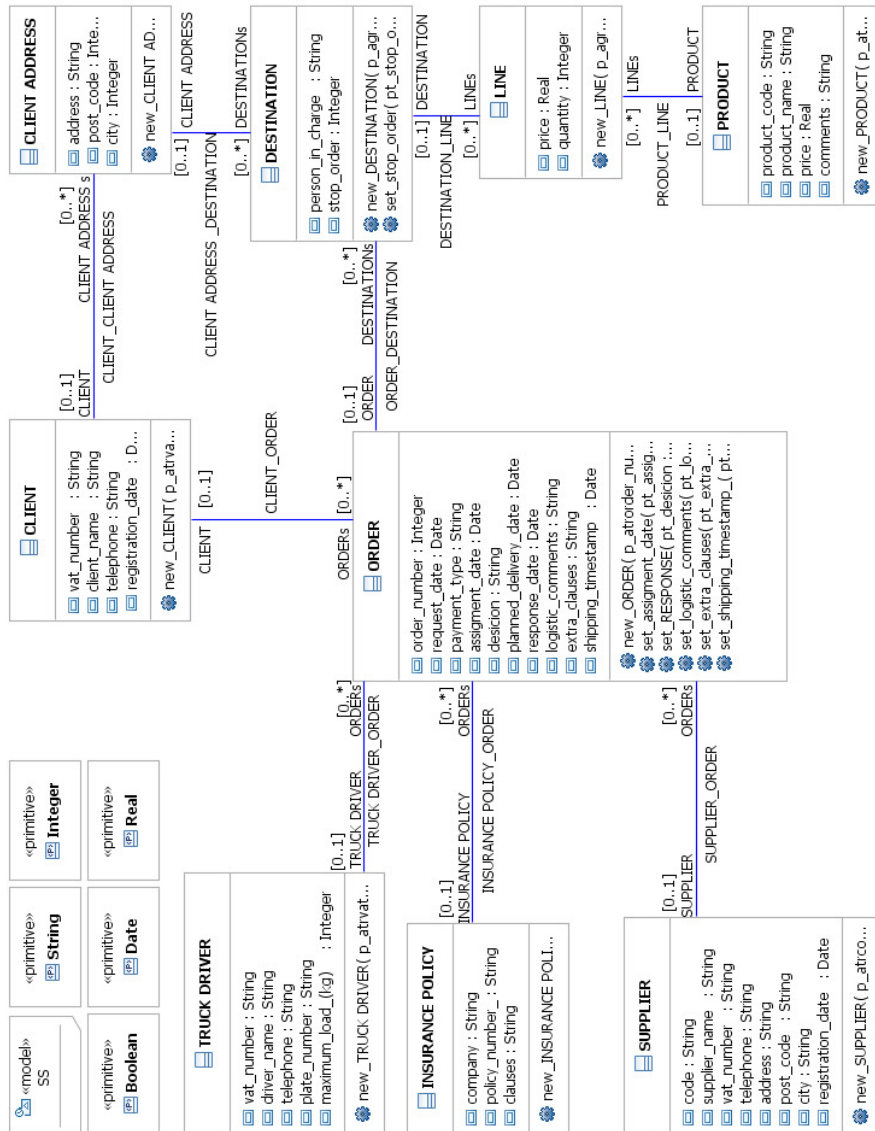


Figure 122. SuperStationery Co. Class diagram in UML 2.0

5.5 Validation of the transformation proposal

The technological implementation of the rules transformation provides the advantage to integrate the requirements models into the engineering processes.

Thus, with the conceptual models in an electronic format, it is possible to carry out interoperability activities to represent the generated conceptual models in the conceptual models of OLIVANOVA.

We consider the implementation of the rules transformation as an important step to achieve to close the gap between the requirements models and the software application. For this reason the last activity of the phase 4 is the evaluation of the module transformation. It is necessary to provide a transformation module improved in an incremental way.

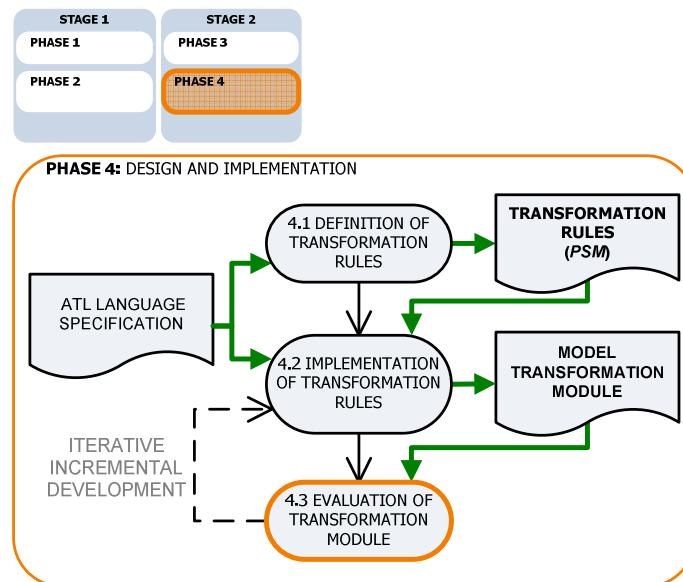


Figure 123. Activity of evaluation of the transformation module

Previous experiments have been carried out with master students to evaluate the completeness and correctness of the generated conceptual models using the rules transformation in a manual way.

We intend to check the agility and facility to use the transformation module versus to use the transformation rules in a manual way.

We plane to carry out an exercise with master students. Previous experiments have tested the manual use of the transformation rules. We consider interesting to analyse the correctness and completeness of the generated conceptual models using the implemented transformation rules.

And also, we plane to carry out a usability evaluation and expressiveness of the modelling tool with the transformation module. We plane to use the same framework that was described at section 4.5.

5.6 Traceability support

Model transformation is expressed in terms of heuristics. These heuristics define how a set of source models are transformed in a set of target models. These heuristics are defined in term of metamodels. The model weaving is an important operation in MDE. Model weaving intends to handle fine-grained relationships between elements of distinct models, establishing links between them. These links are captured by a weaving model. It conforms to a metamodel that specifies the link semantics[71].

There are tools for supporting model weaving. Didonet Del Fabro et al [71] purpose AMW weaving tool. AMW is available in the Eclipse platform (GMT project [72]) and it reuses part of the infrastructure of the ATL IDE [66]. It provides an API to access models and metamodels that are based on the Ecore [36] metamodel. This implementation offers a user interface for weaving task.

Due to the complexity of the transformation rules for transforming requirements models into conceptual models, we consider necessary to specify our own metamodel traceability. The inherent complexity of the rules not was adaptable to the AMW weaving tool. The weaving tool lets to specify the matching between elements. The ATL

transformation code contains imperative code, this difficult to express it in a declarative way, or a matching way.

We intends to follow the MDE principle “everything is a model” [31]. For this reason, it is possible to consider a metamodel traceability for our metamodels. The Figure 124 presents our proposal of traceability metamodel. We intend not to introduce metaclasses to the source and target metamodels. Following the idea presented by Jouault [75], the metamodels specified with Ecore metamodel [36] are linked with this metamodel. For this reason, the elements of each metamodel can to inherit of the class EOBJECT.

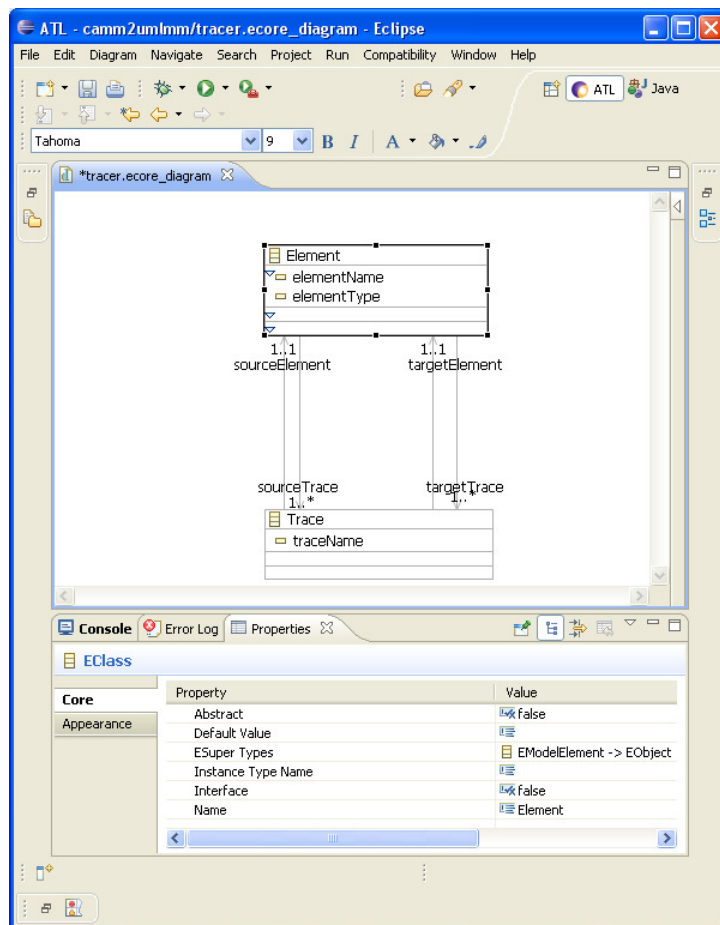


Figure 124. Traceability metamodel

Thus, the metamodel of Communication Analysis requirements models (please see section 4) also have an inheritance relationship with the Ecore class EOBJECT. The same for the UML metamodel (see Figure 125 and Figure 126).

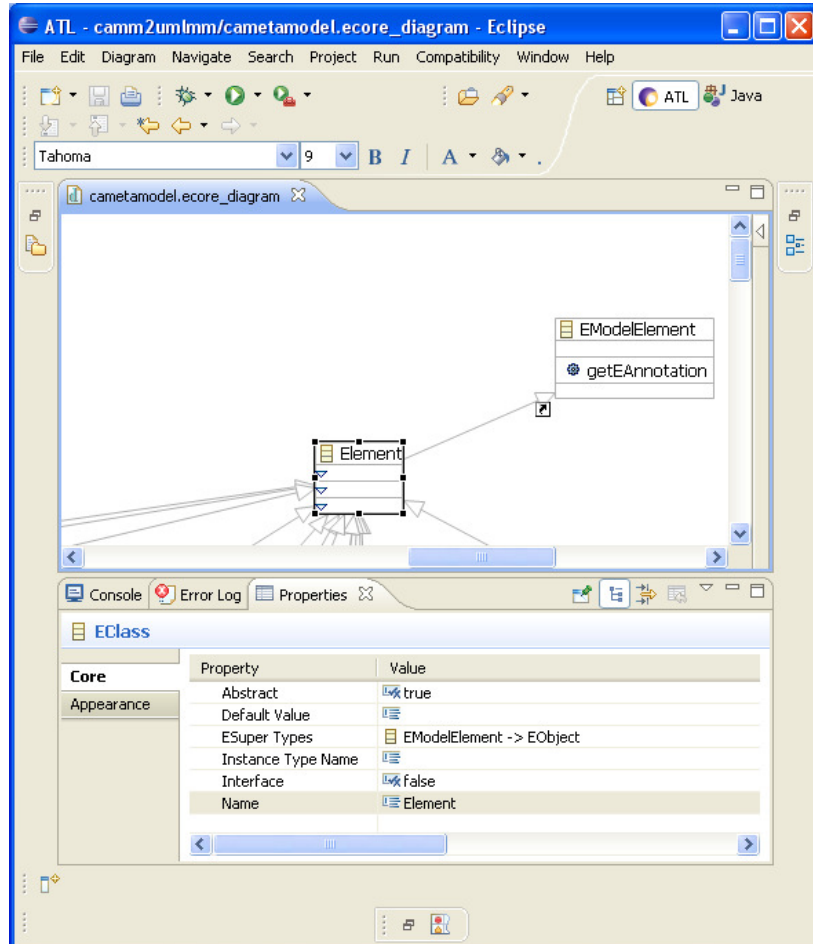


Figure 125. Inheritance relationship between the ELEMENT metaclass and EMODELELEMENT metaclass

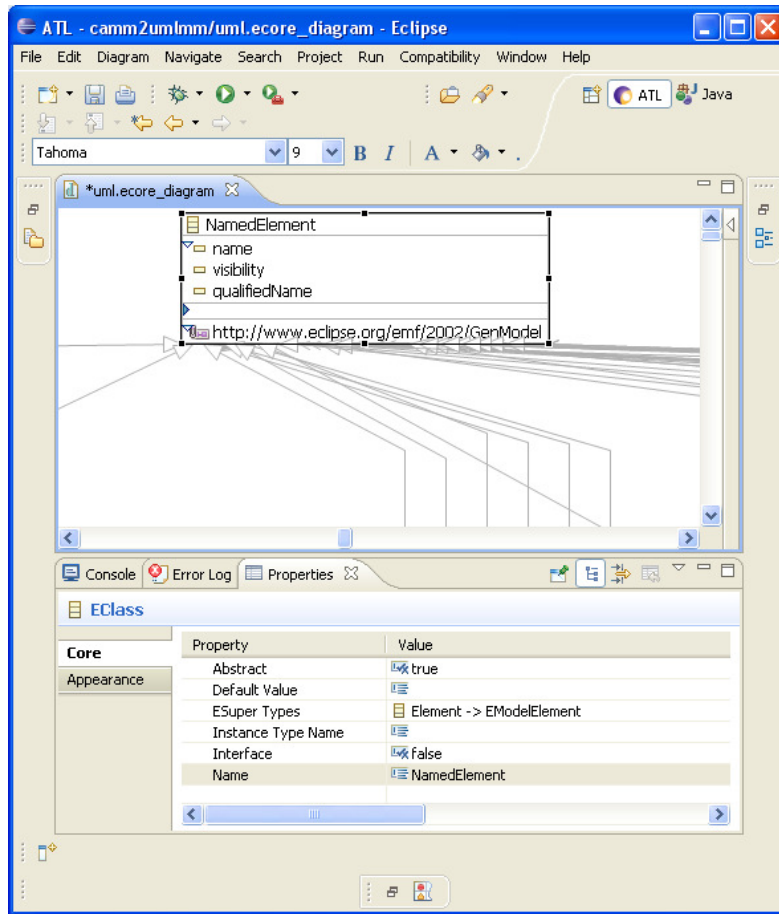


Figure 126. Inheritance relationship between the NAMEDELEMENT metaclass and EMODELELEMENT metaclass

According to the traceability metamodel, the ATL code contains sentences to save information about each transformation. For instance, Figure 127 shows the header of the rule to create a class from aggregation substructure⁵. This header contains information about the elements (source and target), and save information about the name of the element and its type.

⁵ The complete ATL code is presented at Append 1.

Then, the ATL code generates the conceptual model and the information about the traceability between models. The information about the traceability is saved in a XMI format

```
rule process_create_class(active_substructure: cametamodel!Substructure){
  to
    class: uml!Class(name<-active_substructure.name),
    elementSrc: tracer!Element(elementName<- active_substructure.name,
      elementType <- 'Aggregation substructure'),
    elementTrg: tracer!Element(elementName<- class.name,
      elementType <- 'Class',targetTrace<-Sequence(elementTrg)),
    trace: tracer!Trace{
      traceName<- 'Create Class',
      targetElement <- elementTrg}
```

Figure 127. ATL rule to create class from aggregation substructure

Executing the code showed at Figure 127, The SuperStationery Co. example has the traceability information presented at Figure 128.



```
1<?xml version="1.0" encoding="ISO-8859-1"?>
2<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:tracer="tracer">
3 <tracer:Element sourceTrace="/0" elementName="CLIENT" elementType="Aggregation substructure"/>
4 <tracer:Element targetTrace="/1" elementName="CLIENT" elementType="Class"/>
5 <tracer:Trace traceName="Create Class" targetElement="/1" sourceElement="/0"/>
```

Figure 128. Example of traceability information for The SuperStationery Co.

It is possible to add more traceability information. Thus, the traceability metamodel lets to add the information that the analyst wants to save. The ATL rules can to contain a lot of traceability rules according to the metamodel. Then, the traceability metamodel is adaptable to different kind of traceability information.

5.7 Analysis and discussion

Model transformation aims to provide a mean to specify the way to produce target models from a number of source models. In this way, we have proposed a stage with several activities that aim to transform requirements models specified in communicative event dia-

grams and message structures (two novel techniques of the Communication Analysis method) to conceptual models (conceptual models of OO-Method). This thesis proposes to follow the activities presented in stage 2 (design and implementation) of the general framework (please see Figure 115).

To achieve this objectives, we have implemented the transformation rules purposed by España et al [63]. Then, a study about transformations engine was carried out. QVT [45] is a standard purposed by the OMG for model transformation, and it is implemented in Eclipse environment [76]. ATL is other transformation engine developed in the GMT project of Eclipse [72]. The result of this study let us to choose ATL as the better option to implement the. This is because the rules are expressed in declarative and imperative sentences. The hybrid nature of ATL provides patterns and a language to specify this kind of heuristics [66].

A transformation example is presented for illustrating the result of to execute the transformation rules.

The imperative code of the transformation rules facilitates the implementation of the heuristics. The transformation rules are implemented using declarative and imperative code, for this reason we decide to use ATL.

6 Conclusions

After to develop the activities related to this master thesis, We analyse the result of our work. This chapter presents the main conclusions that can be highlight after development of this thesis.

This chapter presents contributions related to the objectives and research questions presented at chapter 1. In addition, we present a list of publications related to this thesis, which confirm the acceptance level and relevance of this project. In addition, we present a list of technical reports that was prepared during development of this thesis. This technical reports are a more detailed documents, where is possible to find more information abut different topics related to this thesis.

Furthermore, a list of before publications of this thesis is presented. These publications are the research background of the author of this thesis.

Finally, we conclude with future work.

6.1 Contributions

The main research goal of this thesis is to propose an approach to integrate Communication Analysis method and OO Method. This proposal presents a general framework that guides the integration activity and provides software artefacts that support the integration between the requirement method and the conceptual method. This thesis proposes to use the MDD paradigm to develop technological environments for requirements methods, which provide several advantages as: agile development of software, availability of open source platforms for MDD developments (e. g, Eclipse) and possibility of software automatic generation, etc. The principal contributions of this master thesis are presented below.

Generic model-driven framework to integrate requirements engineering methods. A model-driven framework to guide the integration activities of both, requirements engineering methods and conceptual models. This framework presents the phases, stages and activities that are necessary to carry out an integration process. This generic framework follows a model-driven architecture perspective; modelling artefacts (e.g. metamodels, modelling tools, etc.) are suggested to achieve the integration.

Integration of Communication Analysis and OO-Method. The generic model-driven framework is instanced to achieve the main objective of this thesis: integrate Communication Analysis and OO-Method. Development of this thesis presents the integration process and presents the different software artefacts obtained.

Technological artefacts to support the Communication Analysis techniques. A set of metamodels that describe techniques of Communication Analysis were built. In addition, a modelling tool to support the method techniques was proposed. These artefacts intend to offer technological support for Communication Analysis techniques. These artefacts are used later at the moment to build a support for integrating methods.

Technological artefacts to support the integration between Communication Analysis and OO-Method. A set of transformation rules were designed. These transformation rules were specified in ATL. ATL is a transformation language engine supported by Eclipse. Thus, a module to support the models transformation between Communication Analysis and OO-Method is presented.

A usability evaluation proposal to evaluate modelling tools. A usability evaluation design for modelling tools is presented. The evaluations include the point of view of the Experts of the methods and the point of view of the potential users of the modelling tool. The usability evaluation was an important experience to improve the modelling tool.

6.2 Publications

Publications related to this thesis

- **Ruiz, M.**, España, S., González, A, Pastor, O. Análisis de Comunicaciones como un enfoque de requisitos para el desarrollo dirigido por modelos.VII Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM'10) Valencia, España September 7, 2010
- España, S., **Ruiz, M.**, González, A, Pastor, O. Integrando Técnicas Avanzadas de Modelado de Requisitos en MOSKitt. MOSKittDay - eclipseDay 2010 Valencia, EspañaNovember 30-31, 2010
- **Ruiz, M.**, González, A, España, S., Pastor, O. Message Structures: a modelling technique for information systems analysis and design. 14th Workshop On Requirements Engineering (WER 2011). Rio de Janeiro, Brazil April 27-29, 2011
- España, S., **Ruiz, M.**, Pastor, O., González, A. Systematic derivation of state machines from communication-oriented business process models. IEEE Fifth International Conference on

Research Challenges in Information Science (RCIS 2011) Guadeloupe, France May 19-21, 2011

- González, A, España, S., **Ruiz, M.**, Pastor, O. Systematic derivation of class diagrams from communication-oriented business process models. International Workshop on Business Process Modelling, Development and Support (BPMDS 2011). London, United Kingdom June 20-21, 2011
- **Ruiz, M.**, Ameller, D., España, S., Botella, P., Franch, X., Pastor, O. Ingeniería de requisitos orientada a servicios: características, retos y un marco metodológico. VII Jornadas de Ciencia e Ingeniería de Servicios. A Coruña, España Septiembre 5-9, 2011

Technical reports related to this thesis

- España, S.; González, A.; Pastor, Ó.; **Ruiz, M.** Integration of Communication Analysis and the OO Method: Rules for the manual derivation of the Conceptual Model. Informe técnico ProS-TR-2001-04. arXiv.org: <http://arxiv.org/abs/1103.3686>
- España, S.; González, A.; Pastor, Ó.; **Ruiz, M.** A practical guide to Message Structures: a modelling technique for information systems analysis and design. Informe técnico ProS-TR-2011-02. arXiv.org <http://arxiv.org/abs/1101.5341>
- España, S.; González, A.; Pastor, Ó.; **Ruiz, M.** Integration of Communication Analysis and the OO Method: Manual derivation of the Conceptual Model. The SuperStationery Co. lab demo. Informe técnico ProS-TR-2011-01. arXiv.org <http://arxiv.org/abs/1101.0105>

Publications before this thesis

- Zapata, C., **Ruiz, M.**, Pastor, O. Desde Esquemas Preconceptuales hacia OO-Method. Revista Facultad de Ingeniería, Universidad de Antioquia; Medellín Clave: A Volumen: 56 Páginas, inicial: 203 final: 212 Fecha: 2010 Editorial (si libro): ISSN 0120 – 6230 Lugar de publicación: Medellín, Colombia
- Zapata, C., **Ruiz, M.**, Villa, F. UNC-Diagramador una herramienta upper CASE para la obtención de diagramas UML

desde Esquemas Preconceptuales. Revista Universidad Eafit
Clave: A Volumen: 43 Páginas, inicial: 68 final: 80 Fecha:
2007 Editorial (si libro): ISSN 0120-341X Lugar de publicación:
Medellín, Colombia

6.3 Future work

In order to reduce the gap between the industries and the academy, we will intend to carry out a comparative study between manual model transformation and automatic model transformation. So as to do this study, the previous evaluation of the modelling tool, and the experiments carried out following manual model transformation could be analysed.

In addition, a study about the BPMN acceptance in the industrial environment is intended. This study intends to present the acceptance level of BPMN vs. modelling languages produced in the academy, for instance the communicative event diagrams (CED) of the Communication Analysis method.

Moreover, we intend to propose a PhD project that includes exploiting the ideas proposed in this master thesis.

7 References

7.1 References of this thesis

1. I. The Standish Group International, "The Standish Group, CHAOS Summary 2010," ed, 2010.
2. W. Bussen and M. D. Myers. Executive information system failure: a New Zealand case study. *Journal of Information Technology*, vol. 12(2), 1997, pp. 145-153.
3. S. T. Foster Jr. and C. R. Franz. User involvement during information systems development: a comparison of analyst and user perceptions of system acceptance *Journal of Engineering and Technology Management*, vol. 16(3-4), 1999, pp. 329-348.
4. N. Juristo, A. M. Moreno, and A. Silva. Is the European industry moving toward solving requirements engineering problems? . *Software IEEE*, vol. 19(6), 2002, pp. 70-77.
5. B. Nuseibeh and S. Easterbrook, "Requirements engineering: a roadmap." In: Conference on The Future of Software Engineering (ICSE'00), New York, USA, 2000.
6. S. España, A. González, and Ó. Pastor, "Communication Analysis: a requirements engineering method for information systems." In: 21st International Conference on Advanced Information Systems (CAiSE'09), Amsterdam, The Netherlands, 2009.

7. D. Zowghi and C. Coulin. Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. *Engineering and Managing Software Requirements*, 2005, pp. 19-46.
8. S. Kabanda and M. Adigun, "Extending model driven architecture benefits to requirements engineering." In: South African institute of computer scientists and information technologists on IT (SAICSIT '06), Republic of South Africa, 2006.
9. M. Alferez, U. Kulesza, A. Sousa, J. Santos, A. Moreira, J. Araújo, and V. Amaral, "A Model-driven Approach for Software Product Lines Requirements Engineering," in *Software Engineering and Knowledge Engineering SEKE (2008)* 2008, pp. 779-784.
10. O. Pastor, S. España, and A. Gonzalez, "An Ontological-Based Approach to Analyze Software Production Methods." In: United Information Systems Conference (UNISCON 2008), Klagenfurt, Austria, 2008.
11. S. España, "Methodological integration of Communication Analysis into a Model-Driven software development framework," PhD. Computer Science, Departamento de Sistemas Informáticos y Computación (DSIC), Universitat Politècnica de València, Valencia, 2011.
12. A. Gonzalez, S. España, and O. Pastor, "Towards a Communicational Perspective for Enterprise Information Systems Modelling." In: The Practice of Enterprise Modeling: from Business Strategies to Enterprise Architectures (PoEM), Stockholm, Sweden, 2008.
13. M. Bunge, *Philosophy of science: from explanation to justification* vol. 2: Transaction Publishers., 1998.
14. R. Wieringa, "Design science as nested problem solving." In: 4th International Conference on Design Science Research in Information Systems and Technology (DESRIST '09), New York, USA, 2009.
15. K. Kronlof, *Method integration: concepts and case studies*. Chichester, UK.: John Wiley and Sons Ltd., 1993.
16. K. Kumar and R. J. Welke, *Methodology EngineeringR: a proposal for situation-specific methodology construction*. NY, USA: John Wiley & Sons, 1992.

17. M. Saeki. A meta-model for method integration. *Information and Software Technology*, vol. 39:1997, pp. 925-932.
18. G. Giachetti, "Supporting Automatic Interoperability in Model Driven Development Processes," Departamento de Sistemas Informáticos y Computación (DSIC), Universitat Politècnica de València, Valencia, Spain, 2011.
19. B. Nuseibeh, "A Multi-Perspective Framework for Method Integration," Department of Computing, Imperial College of Science, Technology and Medicine, London, UK, 1994.
20. J. L. de la Vara, "Business Process-Based Requirements Specification and Object-Oriented Conceptual Modelling of Information Systems," Departamento de Sistemas Informáticos y Computación (DSIC), Universitat Politécnica de València, Valencia, Spain, 2011.
21. AuraPortal. *AuraPortal BPMS*. Available: <http://www.auraportal.com>
22. MOSKitt. *Modelling Software Kitt*. Available: <http://www.moskitt.org/>
23. I. T. Corporation. *Visio Stencil and Template for UML 2.2*. Available: <http://softwarestencils.com/uml/index.html>
24. O. Pastor, A. Gonzalez, and S. España. Conceptual alignment of software production methods. *Conceptual modelling in information Systems engineering*, 2007, pp. 209-228.
25. A. Gonzalez, "Algunas consideraciones sobre el uso de la abstracción en el análisis de los sistemas de información de gestión," Departamento de Sistemas Informáticos y Computación (DSIC), Universitat Politècnica de València, Valencia, Spain, 2004.
26. A. Gonzalez, S. España, and O. Pastor, "Unity criteria for business process modelling: A theoretical argumentation for a software engineering recurrent problem." In: Third International Conference on Research Challenges in Information Science (RCIS 2009), Fes, Morocco, 2009.
27. S. España, A. Gonzalez, O. Pastor, and M. Ruiz, "Message Structures: a modelling technique for information systems analysis and design." Pros Research Center, Valencia, España, 2011.
28. OMG, "MDA Guide," in *How is MDA used?*, ed: OMG, 2003, pp. 1-62.

29. O. Pastor and J. C. Molina, *Model-Driven Architecture in practice: a software production environment based on conceptual modeling*. New York: Springer, 2007.
30. J. den Haan, "The Enterprise Architect, Building an agile enterprise," in *MDE - Model Driven Engineering - reference guide*, J. d. Haan, Ed., ed, 2009.
31. OMG, "MDA Guide," in *How is MDA used?*, ed: OMG, 2003, pp. 1-62.
32. M. Ruiz, S. España, A. Gonzalez, and O. Pastor, "Análisis de Comunicaciones como un enfoque de requisitos para el desarrollo dirigido por modelos." In: VII Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM 2010), Jornadas de Ingeniería de Software y Bases de Datos (JISBD) Valencia, España, 2010.
33. S. España, PhD. Computer Science, Departamento de Sistemas Informáticos y Computación (DSIC), Universitat Politècnica de València, Valencia, 2011.
34. OMG, "OMG Unified Modeling Language™ (OMG UML), Infrastructure," ed, 2010, pp. 1-226.
35. Microsoft. (2010, *Microsoft Visio 2010*. Available: <http://visiotoolbox.com/2010/>
36. F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. J. Grose, *Eclipse Modeling Framework: A Developer's Guide*: Addison-Wesley Professional, 2003.
37. Eclipse.org. *Model Development Tools (MDT) / UML2*. Available: <http://wiki.eclipse.org/MDT-UML2>
38. S. España, A. González, Ó. Pastor, and M. Ruiz, "Integration of Communication Analysis and the OO-Method: Manual derivation of the conceptual model. The SuperStationery Co. lab demo." Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Spain, Available at <http://hci.dsic.upv.es/ca/SuperStationery-TR-v2.0.pdf> 2011.
39. A. González, M. Ruiz, S. España, and Ó. Pastor, "Message Structures a modelling technique for information systems analysis and design." In: XIV Workshop on Requirements Engineering (WER'11), Rio de Janeiro - Brasil, 2011.
40. Eclipse.org. *Eclipse Modeling Framework (EMF)*. Available: <http://www.eclipse.org/modeling/emf/>

41. F. Plante, "Introducing the GMF Runtime." IBM, 2006.
42. S. España, A. González, Ó. Pastor, and M. Ruiz, "Integration of Communication Analysis and the OO-Method: Manual derivation of the conceptual model. The SuperStationery Co. lab demo." 2011.
43. H. Behrens, M. Clay, S. Efftinge, M. Eysholdt, P. Friese, J. Köhnlein, K. Wannheden, S. Zarnekow, and and contributors. (2010, 2010-11). *Xtext user guide (v 1.0.1)*. Available: http://www.eclipse.org/Xtext/documentation/1_0_1/xtext.pdf
44. F. Jouault and I. Kurtev, "Transforming models with ATL." In: Satellite Events MODELS 2005, Montego Bay, Jamaica, 2005.
45. OMG, "Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification," ed, 2011.
46. S. Abrahao and E. Insfran, "Early Usability Evaluation in Model Driven Architecture Environments." In: Sixth International Conference on Quality Software (QSIC'06), 2006.
47. L. Constantine and L. A. D. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. New York: Addison-Wesley Pub Co, 1999.
48. D. Hix and H. R. Hartson, *Developing User Interfaces: Ensuring Usability Through Product and Process*. New York: John Wiley and Sons, 1993.
49. ISO, "ISO-IEC 9126-1: Software Engineering - Product Quality - Part 1: Quality Model," ed, 2001.
50. J. Nielsen, *Usability engineering*. London: Academic Press, 1993.
51. B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 3rd ed. Boston, MA, USA: Addison-Wesley, 1997.
52. D. Wixon and C. Wilson, "The usability engineering framework for product design and evaluation," in *Handbook of Human-Computer Interaction*, M. Helander, *et al.*, Eds., ed: Elsevier Science B.V, 1997.

53. J. Nielsen, *Heuristic evaluation, in Usability inspection methods*. New York, USA: John Wiley and Sons, 1994.
54. E. T. Hvannberg and E. L.-C. Law, "Classification of usability problems (CUP) scheme." In: Human Computer Interaction INTERACT'03, 2003.
55. S. G. Vilbergsdóttir, E. T. Hvannberg, and E. L.-C. Law, "Classification of Usability Problems (CUP) Scheme: Augmentation and Exploitation." In: NordiCHI 2006: Changing Roles, Oslo, Norway, 2006.
56. UsiXML. (2011), *USiXML USer Interface eXtensible Markup Language*. Available: <http://www.usixml.org>
57. A. Seffah, J. Vanderdonckt, and M. Desmarais, *Human-Centered Software Engineering, Software Engineering Models, Patterns and Architectures for HCI*: Springer, 2009.
58. A. Cooper, "Putting personas under the microscope," in *Journal, A blog about design, business and the world we live in.*, ed, 2009.
59. F. Long, "Real or Imaginary; The effectiveness of using personas in product design," in *Proceedings of the Irish Ergonomics Society Annual Conference (IES Conference 2009)*, Dublin, Ireland, 2009, pp. 1-10.
60. J. R. Lewis. IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, vol. 7(1), Jan-March 1995 1995, pp. 57-78.
61. S. España, I. Pederiva, I. Panach, S. Abrahao, and O. Pastor, "Evaluación de la Usabilidad en un Entorno de Arquitecturas Orientadas a Modelos." In: IDEAS 2006, Argentina, 2006.
62. T. Green and A. Blackwell, "Cognitive Dimensions of Information Artefacts: a tutorial " In: BCS HCI conference, 1998.
63. A. Gonzalez, S. España, M. Ruiz, and O. Pastor, "Systematic derivation of class diagrams from communication-oriented business process models." In: 12th edition of the Business Process Modeling, Development, and Support (BPMDS) series, in conjunction with CAiSE'11, London, UK, 2011.

64. S. España, M. Ruiz, O. Pastor, and A. Gonzalez, "Systematic derivation of state machines from communication-oriented business process models." In: Fifth IEEE International Conference on Research Challenges in Information Science, Guadeloupe, France, 2011.
65. S. España, A. Gonzalez, O. Pastor, and M. Ruiz, "Rules for the manual derivation of the Conceptual Model." ProS Research Centre, Universitat Poliècnica de València, València, Spain, 2011.
66. Eclipse.org. *ATL, the Atlas Transformation Language*. Available: <http://www.eclipse.org/atl/>
67. Eclipse.org. *M2M/QVT Declarative (QVTd)*. Available: http://wiki.eclipse.org/M2M/Relational_QVT_Language_%28QVTR%29
68. F. Jouault and I. Kurtev, "On the architectural alignment of ATL and QVT." In: The 21st Annual ACM Symposium on Applied Computing SAC'06, Dijon, France, 2006.
69. OMG, "Object Constraint Language 2.0 (OCL)," ed, 2006.
70. M. Didonet Del Fabro, J. Bézivin, F. Jouault, and P. Valduriez. Applying Generic Model Management to Data Mapping. *Journées Bases de Données Avancées (BDA05)*, 2006.
71. M. Didonet Del Fabro, J. Bézivin, F. Jouault, E. Breton, and G. Gueltas, "AMW: A Generic Model Weaver." In: 1ères Journées sur l'Ingénierie Dirigée par les Modèles (IDM 05), Paris, 2005.
72. Eclipse.org. *GMT Project*. Available: <http://www.eclipse.org/gmt/>
73. NetBeans.org. NetBeans.org open source project.
74. B. Marín, G. Giachetti, and O. Pastor, "Intercambio de modelos UML y OO-Method." In: Congreso Iberoamericano en "Software Engineering" (CibSE'07), Isla de Margarita, Venezuela, 2007.
75. F. Jouault, "Loosely Coupled Traceability for ATL." In: European Conference on Model Driven Architecture (ECMDA) workshop on traceability (2005) Nuremberg, Germany, 2005.

76. Eclipse.org. *Eclipse.org*. Available: <http://www.eclipse.org/>

8 Append 1

8.1 Development of the modelling tool: step by step

The Graphical Editing Framework (GEF) provides technology to create rich graphical editors and views for the Eclipse Workbench User Interface [40]. The Graphical Modelling Framework (GMF) is an Eclipse project that aims to provide a generative bridge between the EMF and GEF [40].

GMF is an Eclipse project with the potential to become a keystone framework for the rapid development of standardized Eclipse graphical modelling editors. Architects and developers involved in the development of graphical editors or of plug-ins integrating both EMF and GEF technologies should consider building their editors against the GMF Runtime component. This framework let us to build modelling tools based on Eclipse editors like UML editor, Ecore editor, BPM Editor, etc. The Framework can be divided en two main components: the tooling and the runtime. The tooling consists of editors to create/edit models describing the notational, semantic and tooling aspects of a graphical editor. The generated plug-ins depend on the GMF Runtime component to produce a world class extensible graphical editor [41].

We have followed a workflow in order to create a graphical modelling environment for communicative event diagrams and message structures. The Figure 129 shows the workflow followed. This workflow was built according to the Eclipse Tutorials [37].

We can distinguish three important phases. The first phase is the *definition of domain models*. These set of models intent specify the non-graphical information managed by the editor. The second phase is the *definition of diagram models*. These models define graphical elements to be displayed in the editor. The Third phase is the *generation of graphical editor*. This phase basically takes the models previously created in order to generate the java code that will be representing the graphical editor.

Now, we will analyze each phase and we will explain how was made each activity.

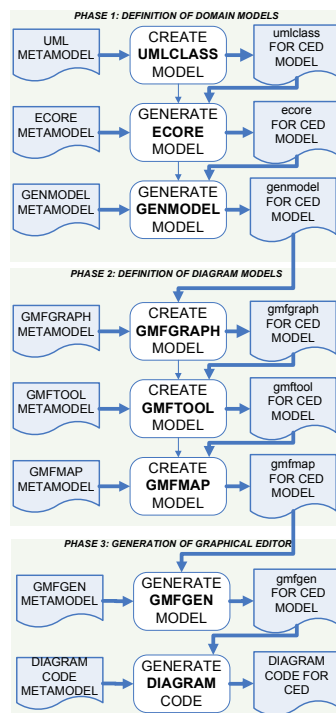


Figure 129. Workflow to create the modelling environment for CED

Phase 1: Definition of domain models

The phase 1 consists about the creation of the Domain models. The CED metamodel was specified through of the UML class model (see section 3.4). Later, the Ecore model was generated. Now, the first step consists in to create a new EMF project. This means to select **File -> New -> Project -> Eclipse Modelling Framework -> EMF Project**. Choose the Ecore model resource (The Ecore model previously created), and click the finish button to close the wizard and to establish the generation. The genmodel for CED looks like Figure 130.

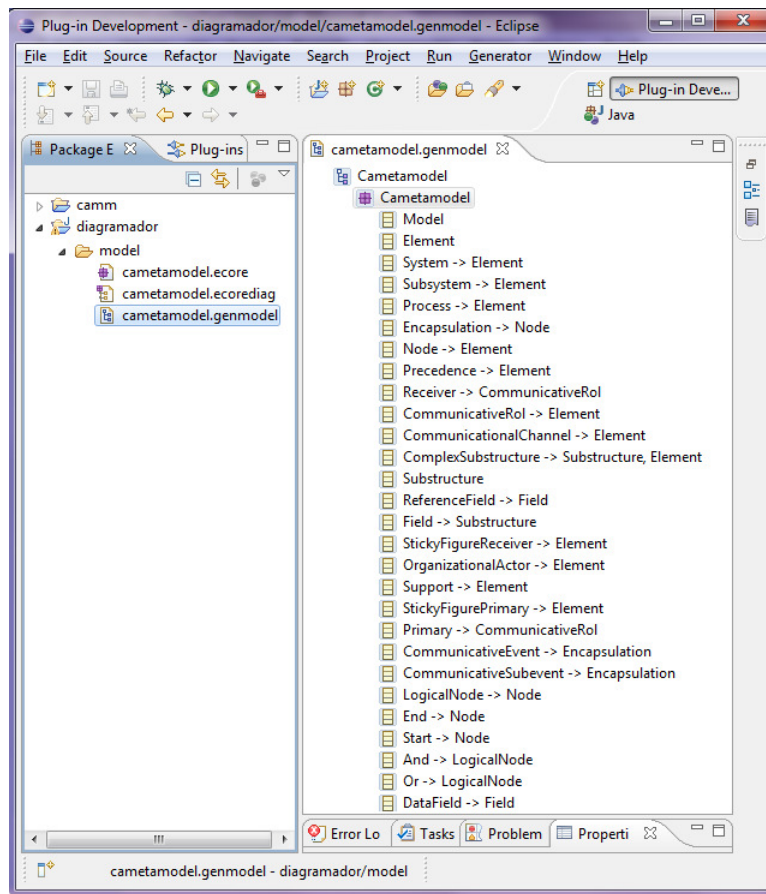


Figure 130. Genmodel model for CED

At this point, it is possible to generate the editor code. EMF provides a textual editor to implement models according to the Ecore model.

This textual editor is a concrete syntax for the CED (See Chapter 0 for more explanation about the concrete and abstract syntax for CED).

In order to obtain the textual editor, right-click the genmodel file and select **Generate Model Code**, later, select **Generate Edit Code** and **Generate Editor Code**. With these actions, we can to obtain a **source file**, an **edit file** and an **editor file**. These files contain the classes with the model implementation and the editor implementation.

We can to use the textual editor to specify CED in a tree view. To specify CED in a tree view, follow these steps:

- Right button over the editor project select **Run As -> Eclipse Application** (see Figure 131). This action to run other instance of Eclipse application with the plug-in that we have created. This plug-in contains the metamodels to create CED.

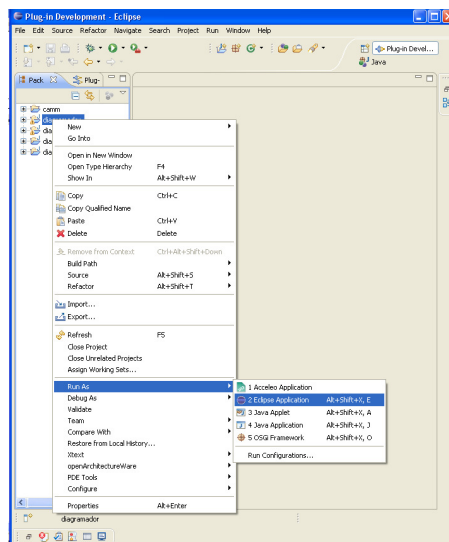


Figure 131. How to run the textual editor for CED

- In the new Eclipse environment, create a new EMF project. Select **File -> New -> Project -> Eclipse Modelling Framework -> Empty EMF Project** (See Figure 132). Assign a name to the new EMF project.

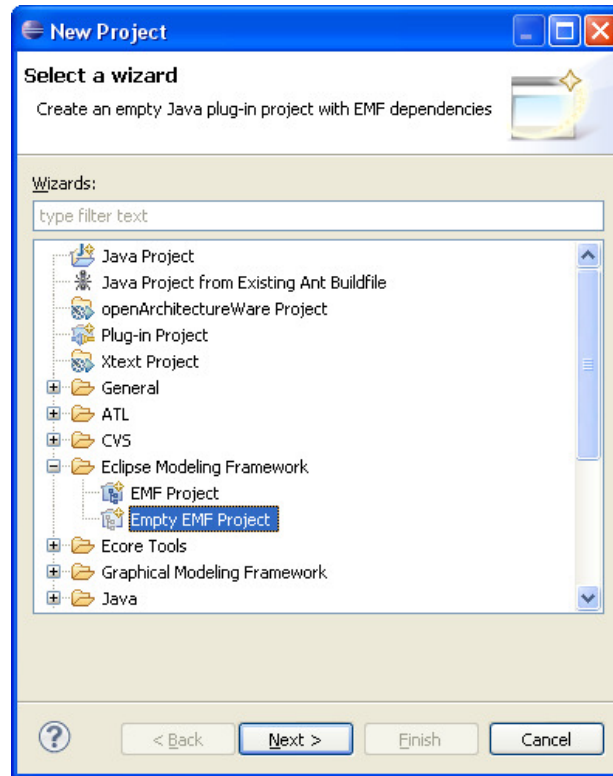


Figure 132. Creation of a new empty EMF project

- Select **File -> New -> Other -> Example EMF Model Creation Wizard -> Cametamodel Model**. This action opens the wizard associated to CED models (See Figure 133). Click the **Next** button.

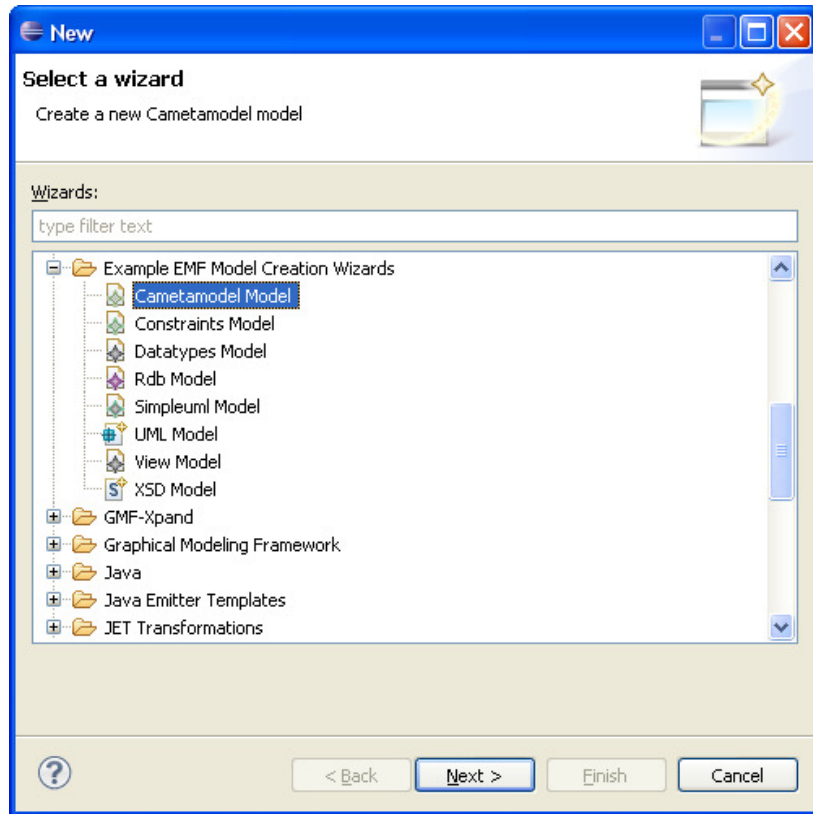


Figure 133. Selection of the CED creation wizard

- In the wizard, to create the new CED, select the EMF project previously created and to assign a name to the model. Notice that the model extension is “.cametamodel”, this name corresponds with the metamodel name that defines the model (see Figure 134). Click the **Next** button.

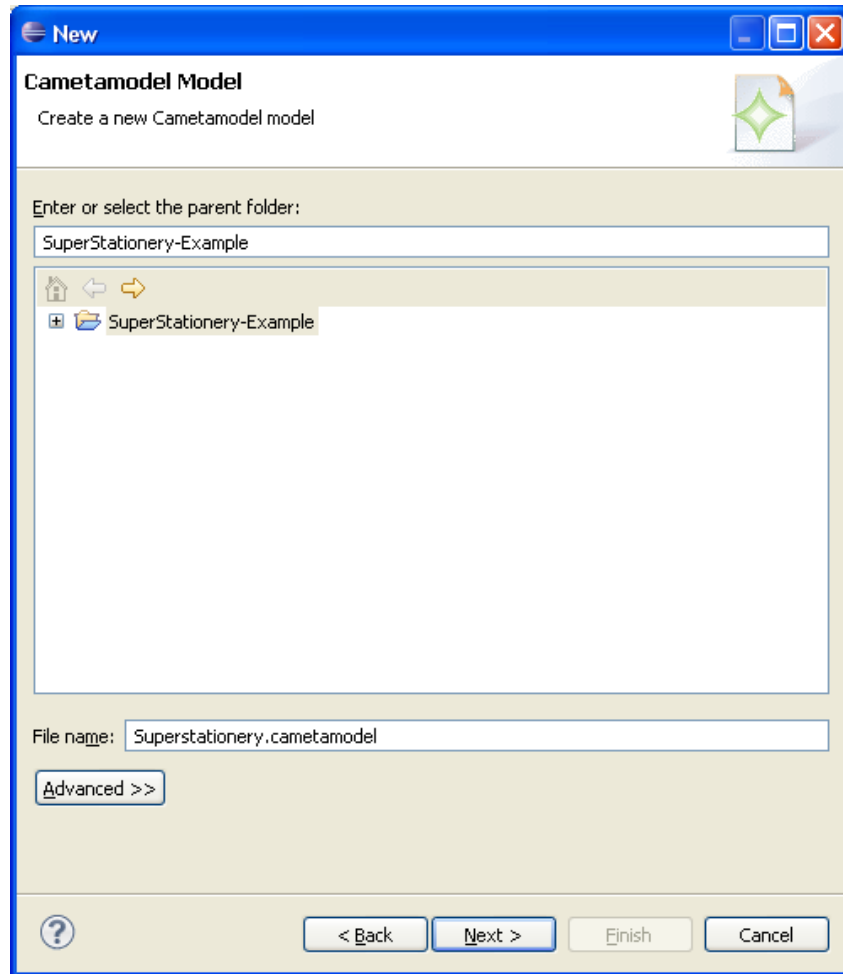


Figure 134. Creation of the new CED model

- In this step, you should select a model object to create. Choose in the **Model Object** section **Model** element. This object indicates the root element of the model that will be created. The value for the **XML Encoding** section should be **UTF_8** (see Figure 135). Click the **Finish** button.

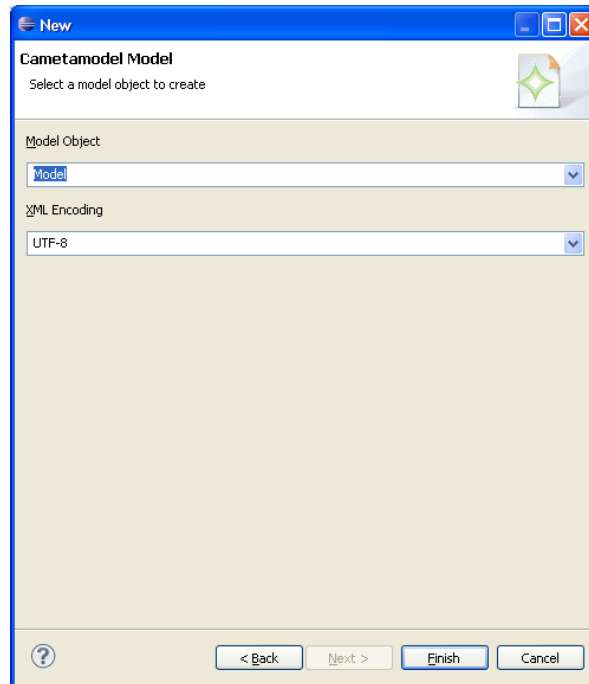


Figure 135. Selection of the model object to create

Now, we have a new file named "SuperStationery.cametamodel". This file has hierarchical structure. This file contains a root element named *Model*. This element can contain the elements that we have defined in the metamodel specification.

The initial state of the SuperStarionary project is showed at Figure 136.

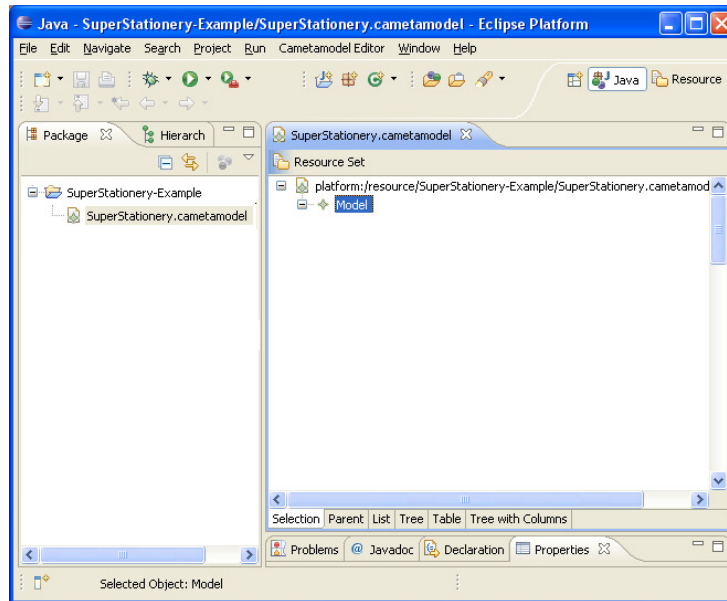


Figure 136. Initial state of the SuperStationery model

The element *Model* is container for the other elements that are part of the SuperStationery model.

In order to add elements, right button over the *Model* element and select **New Child**. This action shows a menu whit a set of elements that could be added to the Model (see Figure 137).

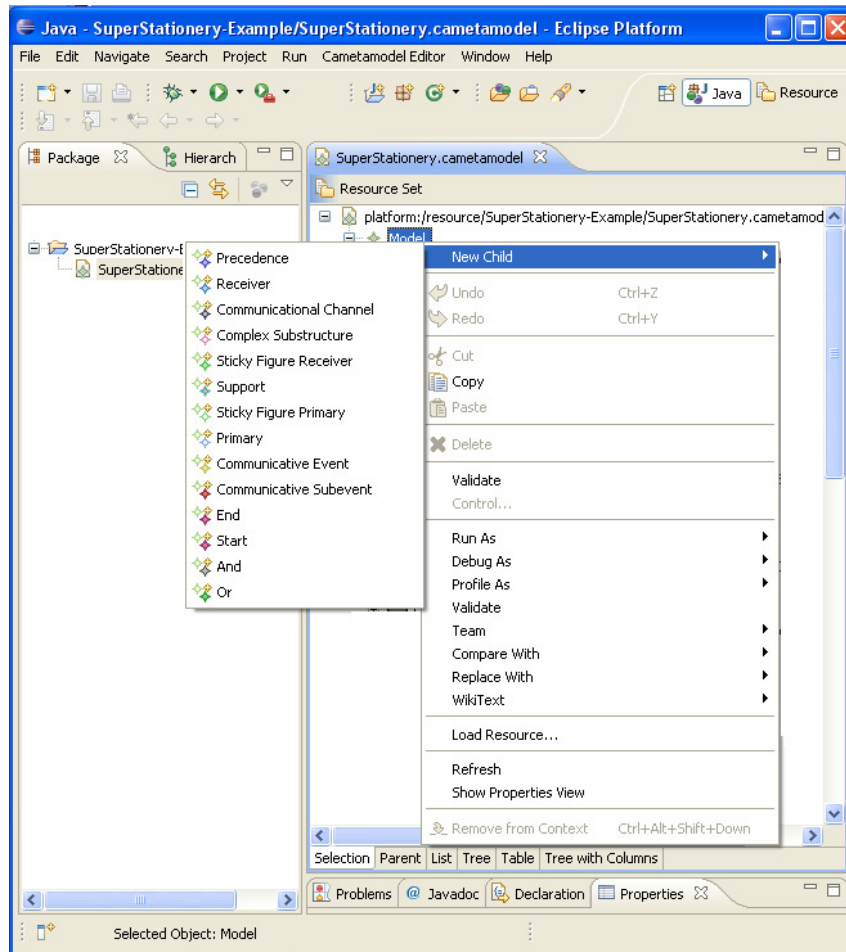


Figure 137. Add elements to the Model

For instance, if we want to add a new communicative event, we need to go to the option *Communicative Event*. Automatically, in the model appears a communicative event and a **Properties** tab is available to edit the information about this element (see Figure 138 for a communicative event example).

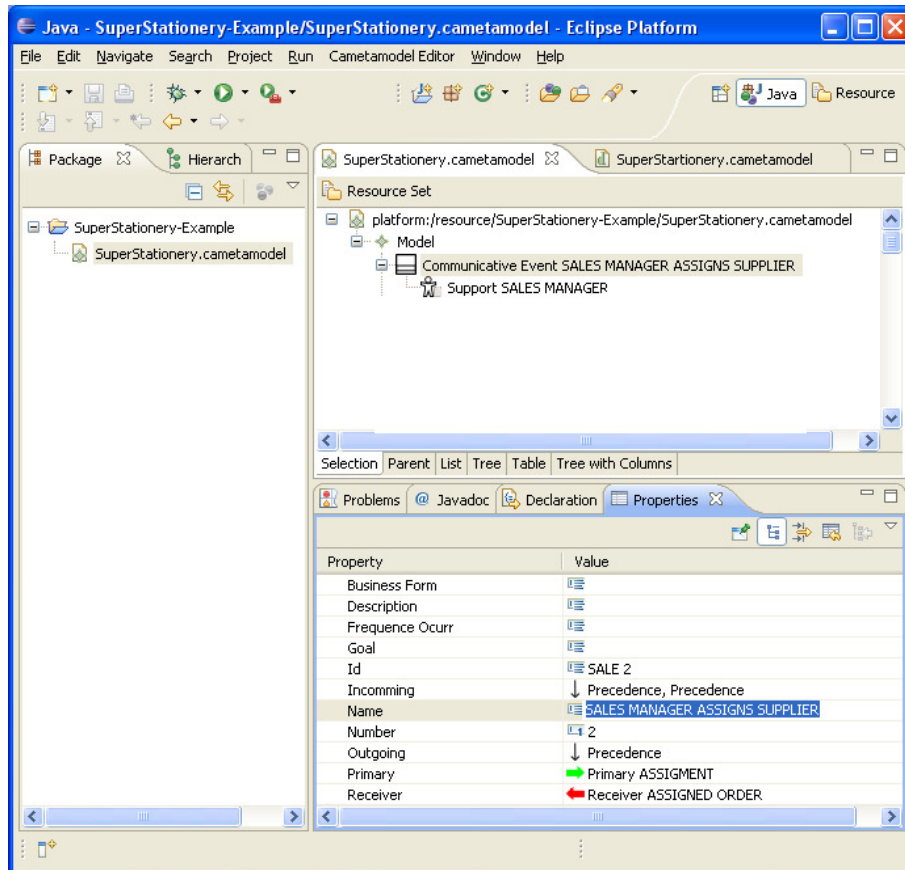


Figure 138. Example of creation of a Communicative Event

This activity should be carried for each element that will be part of the model.

The Figure 139 shows the complete example about SuperStationery Co Model implemented in a textual editor.

We do not recommend the use of the textual editor for CED because this way could be so slow. The textual editor does not provide a flexible interface and it does not provide a preview of the model. For this reason some mistakes and fails could be made in the model.



Figure 139. SuperStationery example in a text editor

Phase 2: Definition of diagram models

The phase 2 consists of the definition of diagram models. This means that the CED metamodel built previously will be represented in a graphical editor.

For the definition of a graphical editor, it is necessary to follow three steps: **Create a gmfgraph model, a gmftool model and a gmfmap model.**

In the gmfgraph model we have specified the figures, nodes, links and all kind of elements that we want to display in the CED editor.

Then, to specify a gmfgraph model, select the Ecore model (see Figure 140), right-click the ecore file and select **New -> Other -> Graphical Modelling Framework -> Simple Graphical Definition Model**. This action opens a wizard where is specified the kind of figure for each element of the metamodel.

A graphical figure is designed for each element. The gmfgraph model for CED looks like the Figure 140.

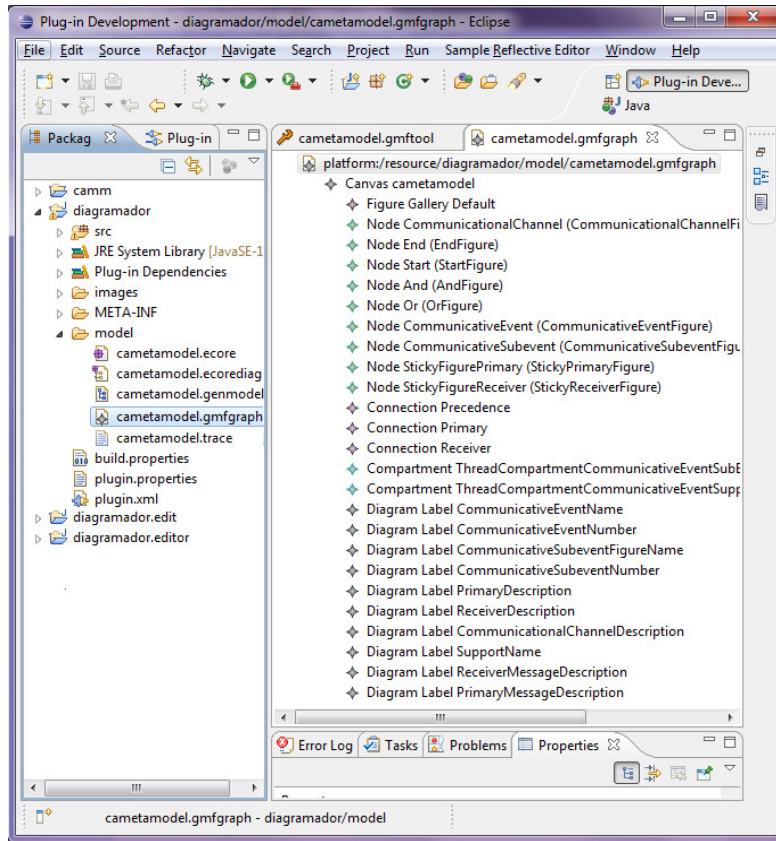


Figure 140. gmfgraph for CED

In the gmftool model we have specified the palette, creation tools, actions, etc for the graphical elements.

Then, to specify a gmftool model, select the Ecore model (see Figure 141), right-click the ecore file and select **New -> Other -> Graphical Modelling Framework -> Simple Tooling Definition Model**. In fact, the steps are identical as the graphical definition model. This action opens a wizard where is specified the kind of figure for each element. The name label for the palette is selected. The gmftool model for CED looks like Figure 141.

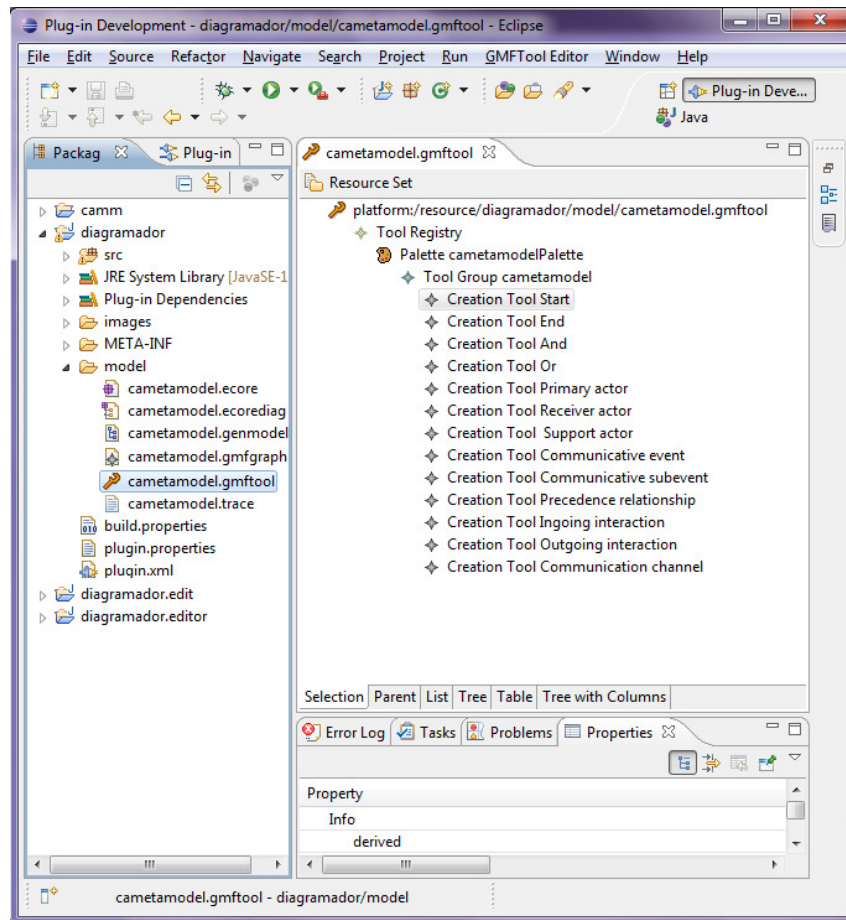


Figure 141. gmftool for CED

In the gmfmap model we have specified the union between the three models, the domain, the graphical and the tooling model (Ecore model, gmfgraph model and gmftool model). This is the principal activity. The gmfmap model will be used as input to a transformation step which will produce the generation model (Phase 3).

Then, to specify a gmfmap model, select the Ecore model (see Figure 142), right-click the ecore file and select **New -> Other -> Graphical Modelling Framework -> Guide Mapping Model Creation Wizard**. This action opens a wizard. The Ecore model, gmfgraph and gmftool models

are selected. Later, is necessary to relate each element of each model and to assign the properties. The gmfmap model for CED looks like Figure 142.

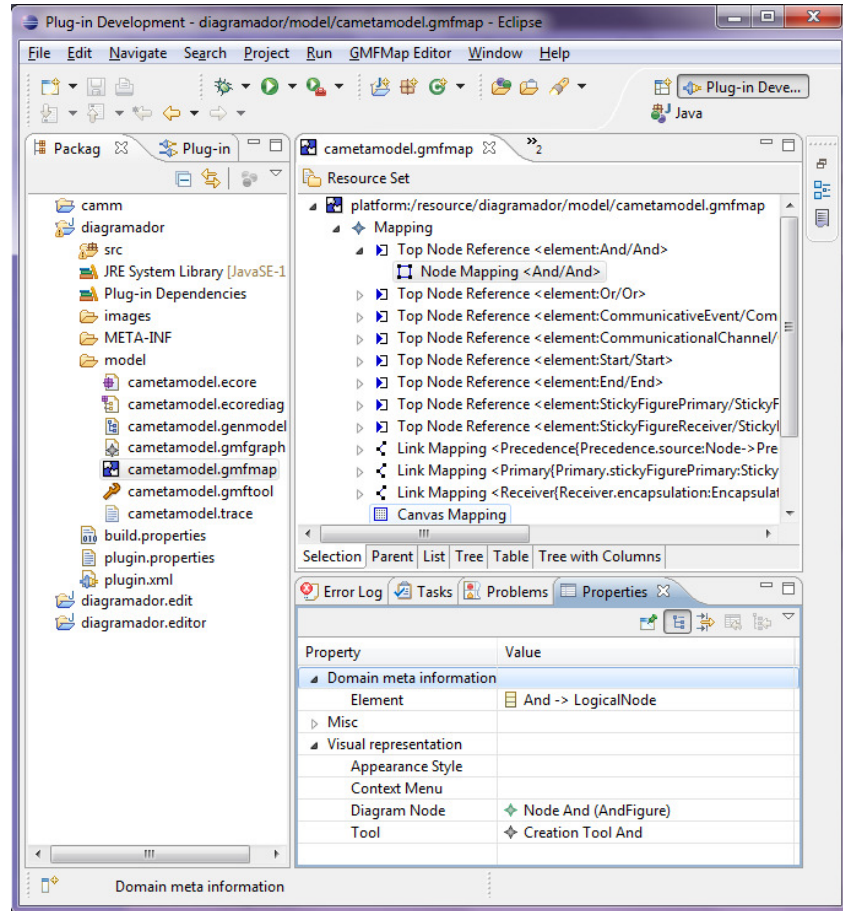


Figure 142. gmfmap for CED

Phase 3: Generation of graphical editor

The phase 3 consists in generate the last model where is possible to specify editors characteristics and generate the graphical editor code.

The generator model (gmfgen) sets the properties for code generation. For generate this model, right-click the mapping file and select **Create generator model**. Then a file named `cametamodel.gmfgen` is created.

The generator model includes the default values for all of its properties (see Figure 143).

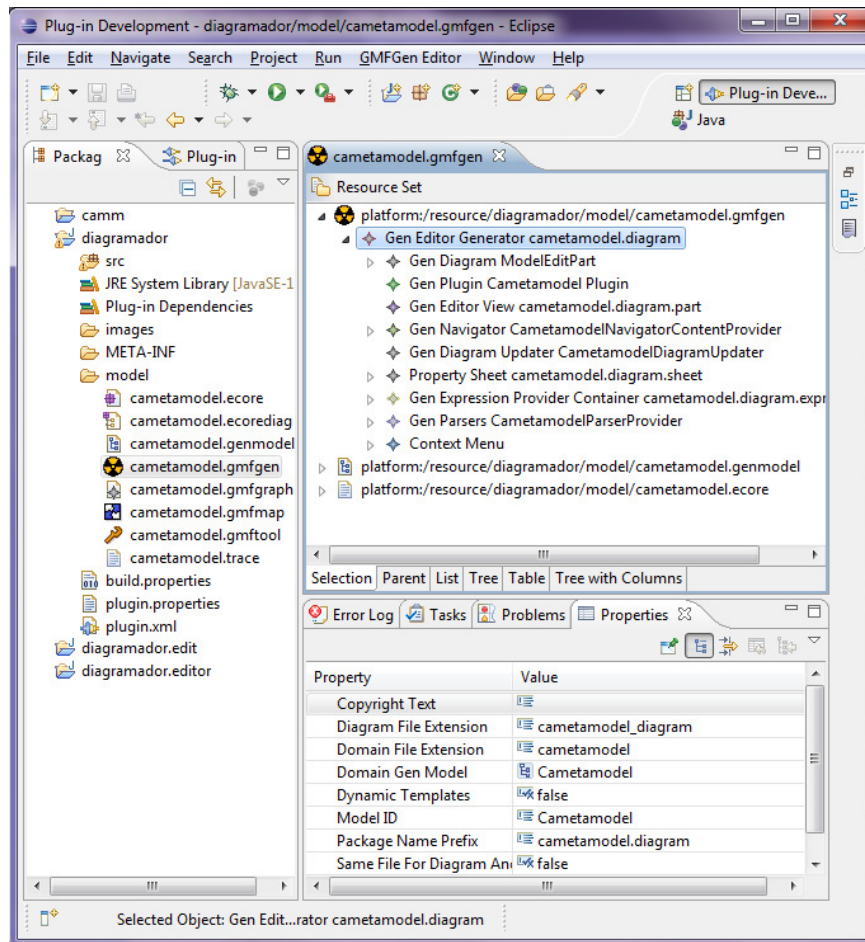


Figure 143. gmfgen for CED

Now, right-click the `cametamodel.gmfgen` file and select **Generate diagram code**. Later, a message dialog appears with the message

“Code generation completed successfully”. New files were been created with the diagram code. This code could be customized for particularly proposes. We have customized this code to offer more graphical facilities and services.

Now, we can to launch the graphical editor for CED. So, right-click the diagram file and select **Run As -> Eclipse Application** (see Figure 144). This action opens a new instance of Eclipse. This Eclipse instance has incorporated the plug-in to model CED.

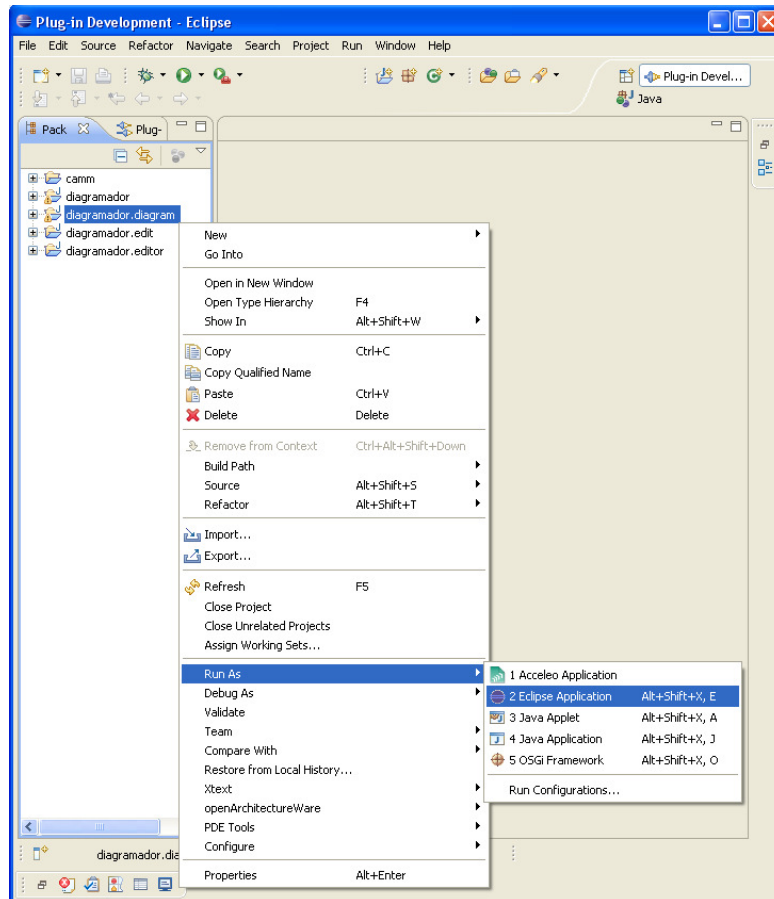


Figure 144. Launch of the Eclipse application with the plug-in to model CED

After the Eclipse application is launched, we can see the graphical interface of the modelling tool for CED. It looks like Figure 145.

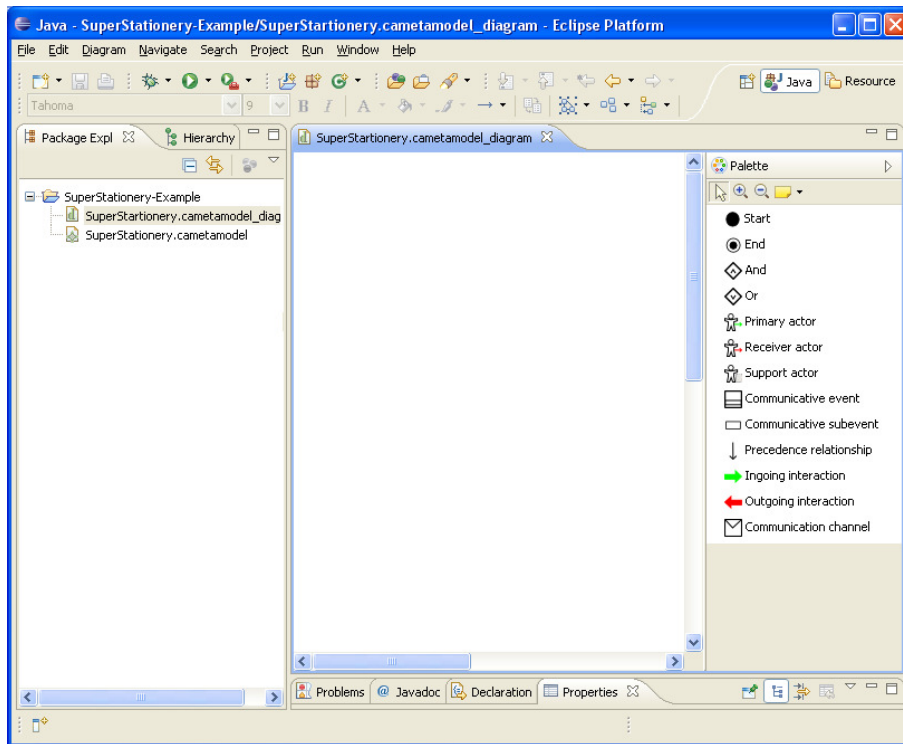


Figure 145. Interface of CED Modelling tool

9 Append 2

9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
-- Module: transformation rules
-- Communication Analysis requirements models to OO-
Method conceptual model
-- Authors: Sergio España Cubillo
--           Arturo González
--           Óscar Pastor López
--           Marcela Ruiz
-- Corresponding authors:
Sergio España -> sergio.espana@pros.upv.es
Marcela Ruiz  -> lruiz@pros.upv.es
-- Version: 2.0
-- Final Version: No
-- Release date: Aug 2011

module ca2uml;

create OUT: uml, trace: tracer from IN: cametamodel;

--HELPERs
-- This helper Returns true if a substructure is a Field
type
```

248 9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
helper def: is_of_type_field(sub: cameta-
model!Substructure): Boolean=
    if (sub.oclIsKindOf (cametamodel!Field)) then
        true
    else
        false
    endif;
-- This helper Returns true if a substructure is a Data-
Field type
helper def: is_of_type_data_field(sub: cameta-
model!Substructure): Boolean=
    if (sub.oclIsKindOf (cametamodel!DataField)) then
        true
    else
        false
    endif;
-- This helper Returns true if a substructure is a Ref-
erenceField type
helper def: is_of_type_reference_field(sub: cameta-
model!Substructure): Boolean=
    if (sub.oclIsKindOf (cametamodel!ReferenceField)) then

        true
    else
        false
    endif;
-- This helper Returns true if a substructure is a Com-
plexSubstructure type
helper def: is_of_type_complex_substructure(sub: cameta-
model!Substructure): Boolean=
    if (sub.oclIsKindOf (cametamodel!ComplexSubstructure)
) then
        true
    else
        false
    endif;
-- This helper returns true if a substructure is a ini-
tial substrucutre
helper def: is_initial_substructure(sub: cameta-
model!ComplexSubstructure): Boolean=
    if (sub.parent = OclUndefined) then
        true
    else
        false
    endif;
```

```

--This helper returns true if the substructure is of
AGGREGATION type
helper def: is_of_type_aggregation(sub: cameta-
model!ComplexSubstructure): Boolean=
    if (sub.oclIsKindOf (cametamodel!Aggregation)) then

        true
    else
        false
    endif;
--This helper returns true if the substructure is of
ITERATION type
helper def: is_of_type_iteration(sub: cameta-
model!ComplexSubstructure): Boolean=
    if (sub.oclIsKindOf (cametamodel!Iteration)) then
        true
    else
        false
    endif;
--This helper returns true if the substructure is of
type SPECIALIZATION
helper def: is_of_type_specialization(sub: cameta-
model!ComplexSubstructure): Boolean=
    if (sub.oclIsKindOf (cametamodel!Specialisation)) then

        true
    else
        false
    endif;
--
--This helper returns the precedent substructure of an
enter substructure
helper def: get_precedent_substructure(sub: cameta-
model!ComplexSubstructure): cameta-
model!ComplexSubstructure =
    if (self.is_of_type_aggregation (sub.parent)) then
        sub.parent
    else if (self.is_of_type_iteration (sub.parent)) then
        sub.parent.parent
    else OclUndefined
    endif
endif;

-- This helper returns the plural of any word
helper def: plural(word: String): String = word + 's';

```

250 9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
-- This helper returns the lower case and replace blank
space by underscore
helper def: format_to_lower_underscore (word: String) :
String = (word.toLowerCase()).replaceAll(' ', '_');

--Tuple of Substructure and class
helper def: set_tuple_sub2class(sub: cameta-
model!Substructure, class : uml!Class):TupleType
(sub: cametamodel!Substructure, class : uml!Class)
= Tuple{sub: cametamodel!Substructure = sub, class :
uml!Class = class};
--Tuple of DataField and Attribute
helper def: set_tuple_datf2attr(datField: cameta-
model!DataField, attribute : uml!Property):TupleType
(datField: cametamodel!DataField, attribute :
uml!Property) = Tuple{datField: cametamodel!DataField =
datField, attribute : uml!Property = attribute};
--Tuple of DataField and Association
helper def: set_tuple_datf2rel(datField: cameta-
model!DataField, relationship :
uml!Association):TupleType
(datField: cametamodel!DataField, relationship :
uml!Association) = Tuple{datField: cametamodel!DataField
= datField, relationship : uml!Association = relation-
ship};

-- Sets of tuples
helper def: subs_classes_set : Sequence (TupleType(sub:
cametamodel!Substructure, class : uml!Class)) =
Sequence {Tu-
ple {sub: cametamodel!Substructure = cameta-
model!Substructure, class : uml!Class = uml!Class}};
helper def: datF_attr_set : Sequence (Tuple-
Type(datField: cametamodel!DataField, attribute :
uml!Property)) =
Sequence {Tu-
ple {datField: cametamodel!DataField = cameta-
model!DataField, attribute : uml!Property =
uml!Property}};
helper def: datF_rel_set : Sequence (TupleType(datField:
cametamodel!DataField, relationship : uml!Association))
=
Sequence {Tu-
ple {datField: cametamodel!DataField = cameta-
model!DataField, relationship : uml!Association =
uml!Association}};
```

```

-- These helpers are like global variables
helper def: conceptual_model: uml!Model = OclUndefined;
helper def: class_case : String = '';
helper def: active_class : uml!Class = OclUndefined;
helper def: active_attribute : uml!Property = OclUn-
defined;
helper def: attribute_count : Integer = 0;
helper def: active_aggregation: uml!Association = OclUn-
defined;
helper def: argument_count : Integer = 0;
helper def: active_event: uml!Operation = OclUndefined;
--Data Types
helper def: data_type_string : uml!PrimitiveType =
OclUndefined;
helper def: data_type_integer : uml!PrimitiveType =
OclUndefined;
helper def: data_type_boolean : uml!PrimitiveType =
OclUndefined;
helper def: data_type_date : uml!PrimitiveType = OclUn-
defined;
helper def: data_type_real : uml!PrimitiveType = OclUn-
defined;

entrypoint rule mainRule(){

    using{
        --Parameters
        active_substructures : cameta-
model!ComplexSubstructure = cameta-
model!ComplexSubstructure.allInstances();
        requirements_model: cametamodel!Model= cametamo-
del!Model.allInstances();
    }

    to
    model: uml!Model

    do{

        self.conceptual_model <- model;

        for(p in requirements_model){
            self.conceptual_model.name<- p.name;
        }
    }
}

```

252 9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
--Create Data Types
self.create_data_types();

--Process active_substructure
for(active_substructure in ac-
tive_substructures){
    self.process_structure(active_substructure);
}
}

--Creation of Data Types Attributes
rule create_data_types(){
    to
        dataTypeString: uml!PrimitiveType,
        dataTypeInteger: uml!PrimitiveType,
        dataTypeBoolean: uml!PrimitiveType,
        dataTypeDate: uml!PrimitiveType,
        dataTypeReal: uml!PrimitiveType
    do{
        self.data_type_string<-dataTypeString;
        self.data_type_string.name<-'String';

        self.conceptual_model.packagedElement.add(self.data
_type_string);
        self.data_type_integer<-dataTypeInteger;
        self.data_type_integer.name<-'Integer';

        self.conceptual_model.packagedElement.add(self.data
_type_integer);
        self.data_type_boolean<-dataTypeBoolean;
        self.data_type_boolean.name<-'Boolean';

        self.conceptual_model.packagedElement.add(self.data
_type_boolean);
        self.data_type_date<-dataTypeDate;
        self.data_type_date.name<-'Date';

        self.conceptual_model.packagedElement.add(self.data
_type_date);
        self.data_type_real<-dataTypeReal;
        self.data_type_real.name<-'Real';

        self.conceptual_model.packagedElement.add(self.data
_type_real);
    }
```



```

    }
  }
  --Process transformation: Process Structure
  rule process_structure (active_substructure: cameta-
model!ComplexSubstructure){

    do{
      --Class creation
      self.class_case <- '';
      --if(self.is_of_type_aggregation (ac-
tive_substructure)){
        for(rf in ac-
tive_substructure.directItems){
          --Checking the existence of a refer-
ence field that extends business object

          if(self.is_of_type_reference_field(rf)){
            --if the attribute extends_bo is
equal to True, then a class previously created needs to
be selected.

            if(rf.extendsBo = true){
              for(substructure in
self.subs_classes_set){
                if(rf.domain = sub-
structure.sub){-- if the corresponding substructure was
processed

                  if(substructure.class <> OclUndefined){-- if the
class was defined

                    self.class_case <- 'pre_existing';

                    self.active_class <- substructure.class;
                    --Add an event
to the active_class

                    self.process_add_editing_event(active_substructure,
rf);--Add editing events (prefix set_)
                }
              }
            }
          }
        }
      }
    }
  }
}

```

254 9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
-- If the active substructure do not contain reference field that extends business object, then a new class is created.
    if(self.class_case <> 'pre_existing'){
        if(self.is_of_type_aggregation (active_substructure)){
            self.class_case <-
'newly_created';

            self.process_create_class(active_substructure);--
Creation of a new class
            --Add an event to the active_class
            self.process_add_creation_event(active_substructure);--Creation event
        }
        --End of class creation
        --Iterate and process all substructure items (Creation of attributes and relationships)
        for(active_item in active_substructure.directItems){
            --If the substructure item is a data field then create a new attribute

            if(self.is_of_type_data_field(active_item)){

                self.process_create_atrrIBUTE(active_item);
            }
            --If the substructure item is a reference field and its attribute extendsBo = false, then create a relationship between its corresponding class and the complex substructure that is contained
            else
            if(self.is_of_type_reference_field(active_item)){
                if(active_item.extendsBo = false){

                    self.process_create_relationship_case1(active_item)
                }
            }
        }
    }
}
```

```

        --If the substructure is not an initial
        substructure then create a relationship between the ac-
        tive class and the "precedent" class
            if (not
self.is_initial_substructure(active_substructure)){
                if(self.is_of_type_aggregation (ac-
active_substructure)){
                    if(self.is_of_type_iteration (ac-
active_substructure.parent)){--If the active substructure
is contained into an iteration substructure
                        if(not
self.is_initial_substructure(active_substructure.parent)
){
                            self.process_create_relationship_case2(active_subst
ructure);
                                }
                            } else{--The active substructure
does not contained into an iteration substructure
                                self.process_create_relationship_case2(active_subst
ructure);
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
        --Create Agents
    }
}

rule process_create_class(active_substructure: cameta-
model!Substructure){
    to
        class: uml!Class(name<-
active_substructure.name),
        elementSrc: tracer!Element(elementName<- ac-
active_substructure.name, elementType <- 'Aggregation sub-
structure'),
        elementTrg: tracer!Element(elementName<-
class.name,elementType <- 'Class',targetTrace<-
Sequence{elementTrg}),
        trace: tracer!Trace(
            traceName<- 'Create
Class',
            targetElement <- ele-
mentTrg)
        do {

```

256 9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
--Stablish active_class
self.active_class <- class;

--Traceability
elementSrc.refSetValue('sourceTrace', Sequence{elementSrc});
trace.refSetValue('sourceElement', elementSrc);

--Set of tuples of substructure and class
--
self.subs_classes_set.add(self.set_tuple_sub2class(active_substructure, self.active_class));

self.subs_classes_set.add(self.set_tuple_sub2class(active_substructure, self.active_class));

--Add the new class to the Model

self.conceptual_model.packagedElement.add(self.active_class);
}
}
rule process_create_attribute(active_item: cameta-model!DataField){
  to
    property: uml!Property(name <- self.format_to_lower_underscore(active_item.name),
      elementSrc: tracer!Element(elementName<- active_item.name, elementType <- 'DataField'),
      elementTrg: tracer!Element(elementName<- active_item.name, elementType <- 'Attribute', targetTrace<- Sequence{elementTrg}),
      trace: tracer!Trace(
        traceName<- 'Create Attribute',
        targetElement<- elementTrg)
  do{
    --Stablish active_attribute
    self.active_attribute <- property;

    --Body

    self.active_class.ownedAttribute.add(self.active_attribute);-- ownedAttribute is a set of properties
```

```

        self.attribute_count <- self.attribute_count +
1;
        --Attributes data type
        if(active_item.domain = #text){
            self.active_attribute.type<-
self.data_type_string;
        }else{
            if(active_item.domain = #date){
                self.active_attribute.type<-
self.data_type_date;
            }else{
                if(active_item.domain = #money){
                    self.active_attribute.type<-
self.data_type_real;
                }else{
                    self.active_attribute.type<-
self.data_type_integer;
                }
            }
        }

        --Traceability
        trace.refSetValue('sourceElement', elementSrc);
        elementSrc.refSetValue('sourceTrace', Se-
quence{elementSrc});

        --Tuple of DataField and Attribute

        self.datF_attr_set.add(self.set_tuple_datf2attr(act
ive_item, self.active_attribute));

    }
}
--Create Association
--Case1: ReferenceField to Association
rule process_create_relationship_case1(active_item:
cametamodel!ReferenceField){
    to
        association: uml!Association,
        attributeComponent: uml!Property,
        attributeCompound: uml!Property,
        minCardinalityComponent: uml!LiteralInteger,
        maxCardinalityComponent:
uml!LiteralUnlimitedNatural,
        minCardinalityCompound: uml!LiteralInteger,

```

258 9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
maxCardinalityCompound:
uml!LiteralUnlimitedNatural,
    elementSrc: tracer!Element(elementName<- ac-
tive_item.name, elementType <- 'Reference field'),
    elementTrg: tracer!Element(elementName<- ac-
tive_item.name,elementType <- 'Associa-
tion',targetTrace<-Sequence{elementTrg}),
    trace: tracer!Trace(
        traceName<- 'Create As-
sociation',
        targetElement<- ele-
mentTrg)
    do{
        --Stablish active_aggregation
        self.active_aggregation<-association;

        --Body

        --Attribute of Component Class
        attributeComponent.association <-
self.active_aggregation;
        attributeComponent.owningAssociation<-
self.active_aggregation;
        attributeComponent.type <- self.active_class;
        --The compound class is created before the com-
ponent class, thus, the minimum cardinality in the com-
ponent class is necesarily 0
        minCardinalityComponent.value<-0;
        attributeComponent.lowerValue<- minCardinality-
Component;
        --The maximum cardinality in the component
class depends on the user requirements or on the Analyst
criterion
        maxCardinalityComponent.value<- -1;
        attributeComponent.upperValue<- maxCardinality-
Component;-- (-1 = M in OO-Method metamodel)
        --Set the role name
        if(maxCardinalityComponent.value = -1){
            attributeComponent.name<-
self.plural(self.active_class.name);
        }else{
            attributeComponent.name<-
self.active_class.name;
        }

        --Attribute of Compound Class
```

```

        for (p in self.subs_classes_set){ --
Traceability
        if(p.sub = active_item.domain){
            --Set the name of the association
            self.active_aggregation.name <-
p.class.name + '_' + self.active_class.name;
            --Set the role name
            attributeCompound.name<-
p.class.name;
            attributeCompound.association <-
self.active_aggregation;
            attributeCompound.owningAssociation<-
self.active_aggregation;
            attributeCompound.type <- p.class;
            --The minimum cardinality in the
compound class depends on the user requirements or the
Analyst criterion
            if(self.class_case =
'newly_created'){
                minCardinalityCompound.value<-0;
                attributeCompound.lowerValue<-
minCardinalityCompound; --Less restrictive
            }else if(self.class_case =
'pre_existing'){
                minCardinalityCompound.value<-0;
                attributeCompound.lowerValue<-
minCardinalityCompound;--The minimum cardinality in the
compound class is necesarily 0
            }
            --The maximum cardinality is necesar-
ily 1
            maxCardinalityCompound.value<- 1;
            attributeCompound.upperValue<- max-
CardinalityCompound;
        }
    }

    self.active_aggregation.memberEnd.add(attributeComp
onent);

    self.active_aggregation.ownedEnd.add(attributeCompo
nent);

    --Traceability
    trace.refSetValue('sourceElement', elementSrc);

```

260 9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
        elementSrc.refSetValue('sourceTrace', Sequence{elementSrc});

        --Tuple of DataField and Relationship

        self.datF_rel_set.add(self.set_tuple_datf2rel(activ
e_item, self.active_aggregation));

        --Add the new Relationship to the Model

        self.conceptual_model.packagedElement.add(self.acti
ve_aggregation);
    }
}
--Create Association
--Case2: Nested ComplexSubstructure to Association
rule proc-
ess_create_relationship_case2(active_substructure:
cametamodel!ComplexSubstructure){
    to
        association: uml!Association,
        attributeComponent: uml!Property,
        attributeCompound: uml!Property,
        minCardinalityComponent: uml!LiteralInteger,
        maxCardinalityComponent:
uml!LiteralUnlimitedNatural,
        minCardinalityCompound: uml!LiteralInteger,
        maxCardinalityCompound:
uml!LiteralUnlimitedNatural,
        elementSrc: tracer!Element(elementName<- ac-
tive_substructure.name, elementType<- 'Nesting substruc-
ture'),
        elementTrg: tracer!Element(elementName<- ac-
tive_substructure.name,elementType <- 'Associa-
tion',targetTrace<-Sequence{elementTrg}),
        trace: tracer!Trace(
            traceName<- 'Create As-
sociation',
            targetElement<- ele-
mentTrg)
    do{
        --Stablish active_aggregation
        self.active_aggregation<-association;
        --Body
        --Attribute of Compound Class
        for(p in self.subs_classes_set){
```



```

        if(p.sub =
self.get_precedent_substructure(active_substructure)){
            self.active_aggregation.name <-
p.class.name+ '_' + self.active_class.name;
            attributeCompound.association <-
self.active_aggregation;
            attributeCompound.owningAssociation<-
self.active_aggregation;
            attributeCompound.type <- p.class;
            --The minimum cardinality in the
compound class depends on the user requirements or the
Analyst criterion
            minCardinalityCompound.value<-0;
            attributeCompound.lowerValue<- min-
CardinalityCompound; --Less restrictive
            --The minimum cardinality in the
compound class depends on the user requirements or the
Analyst criterion
            maxCardinalityCompound.value<- 1;--
For the moment
            attributeCompound.upperValue<- max-
CardinalityCompound;
            if(maxCardinalityCompound.value = -
1){
                attributeCompound.name<-
self.plural(p.class.name);
            }else{
                attributeCompound.name<-
p.class.name;
            }
        }
    }

    --Attribute of Component Class created before
to the component class
    attributeComponent.association <-
self.active_aggregation;
    attributeComponent.owningAssociation<-
self.active_aggregation;
    attributeComponent.type <-
self.active_class;
    --The compound class is created before the com-
ponent class, thus, the minimum cardinality in the com-
ponent class is necessarily 0
    minCardinalityComponent.value<-0;

```

262 9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
        attributeComponent.lowerValue<- minCardi-
nalityComponent;
        --It could happent that more components
are added in the future (in later event)

        if(self.is_of_type_aggregation(active_substructure.
parent)){
            maxCardinalityComponent.value<- 1;--
For the moment
            attributeComponent.upperValue<- max-
CardinalityComponent;
        }else{

            if(self.is_of_type_iteration(active_substructure.pa
rent)){
                maxCardinalityComponent.value<- -
1;
                attributeComponent.upperValue<-
maxCardinalityComponent;
            }
            if(maxCardinalityComponent.value = -1){
                attributeComponent.name<-
self.plural(self.active_class.name);
            }else{
                attributeComponent.name<-
self.active_class.name;
            }

--Traceability
            trace.refSetValue('sourceElement', elementSrc);
            elementSrc.refSetValue('sourceTrace', Se-
quence{elementSrc});

            --Tuple of DataField and Relationship

            self.datF_rel_set.add(self.set_tuple_datf2rel(activ
e_substructure, self.active_aggregation));

            --Add the new Relationship to the Model

            self.conceptual_model.packagedElement.add(self.acti
ve_aggregation);

        }
    }
```

```

rule process_add_creation_event (active_substructure:
cametamodel!ComplexSubstructure){
  to
    operation: uml!Operation,
    elementSrc: tracer!Element(elementName<- ac-
tive_substructure.name, elementType<- 'Aggregation sub-
structure'),
    elementTrg: tracer!Element(elementName<- ac-
tive_substructure.name, elementType <- 'Event',targetTrace<-
Sequence{elementTrg}),
    trace: tracer!Trace(
      operation,
      traceName<- 'Create Op-
eration',
      targetElement<- ele-
mentTrg)
  do{
    --Stablish active_event
    self.active_event<-operation;

    --Body
    self.active_event.class<-self.active_class;
    if(self.class_case = 'newly_created'){
      self.active_event.name<- 'new_' +
self.active_class.name;
      --Adding data value inbound arguments and
object value inbound arguments
      for(p in active_substructure.directItems){
        if(self.is_of_type_data_field (p)){

          self.process_add_data_value_inbound(p,
self.active_event);
        }else{

          if(self.is_of_type_reference_field(p)){

            self.process_add_object_value_inbound(p,
self.active_event);
          }
        }
      }

      --Traceability
      trace.refSetValue('sourceElement', elementSrc);
    }
  }
}

```

264 9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
        elementSrc.refSetValue('sourceTrace', Sequence{elementSrc});
    }
}
rule process_add_editing_event(active_substructure:
cametamodel!ComplexSubstructure, reference_field:cametamodel!ReferenceField){
    to
        operation: uml!Operation,
        elementSrc: tracer!Element(elementName<- active_substructure.name, elementType<- 'Aggregation sub-
structure'),
        elementTrg: tracer!Element(elementName<- active_substructure,elementType <- 'Event',targetTrace<-
Sequence{elementTrg}),
        trace: tracer!Trace(
            traceName<- 'Create Op-
eration',
            targetElement<- ele-
mentTrg)
    do{
        --Stablish active_event
        self.active_event<-operation;
        --body
        self.active_event.class<-self.active_class;
        for(p in active_substructure.directItems){
            if(self.is_of_type_data_field (p)){
                self.argument_count <-
self.argument_count + 1;
            }
            if(self.argument_count = 1){
                for(p in active_substructure.directItems){
                    if(self.is_of_type_data_field (p)){
                        self.active_event.name<- 'set_'
+self.format_to_lower_underscore(p.name);

                        self.process_add_data_value_inbound(p,
self.active_event);
                    }
                }
                self.argument_count <- 0;
            }else{
                if(self.argument_count > 1){
```

```

        for(p in active_substructure.directItems){
            if(self.is_of_type_data_field(p)){
                self.active_event.name<-
                'set_' + active_substructure.name;

                self.process_add_data_value_inbound(p,
                self.active_event);
            }
        }
        self.argument_count <- 0;
    }

    self.process_add_object_value_inbound(reference_field, self.active_event);
    --Traceability
    trace.refSetValue('sourceElement', elementSrc);
    elementSrc.refSetValue('sourceTrace', Sequence{elementSrc});
}
}
rule process_add_shared_event_change (active_substructure: cametamodel!ComplexSubstructure){
    to
        operation: uml!Operation,
        elementSrc: tracer!Element(elementName<- active_substructure.name, element_type<- 'Aggregation substructure'),
        elementTrg: tracer!Element(elementName<- active_substructure.name, element_type <- 'Event', targetTrace<- Sequence{elementTrg}),
        trace: tracer!Trace(
            traceName<- 'Create shared Operation',
            targetElement<- elementTrg)
    do{
        --Stablish active_event
        self.active_event<-operation;

        --Body
        self.active_event.class<-self.active_class;
        if(self.class_case = 'newly_created'){

```

266 9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
        self.active_event.name<-'new_' +
self.active_class.name;
        --Adding data value inbound arguments and
object value inbound arguments
        for(p in active_substructure.directItems){
            if(self.is_of_type_data_field (p)){

                self.process_add_data_value_inbound(p,
self.active_event);
            }else{

                if(self.is_of_type_reference_field(p)){

                    self.process_add_object_value_inbound(p,
self.active_event);
                }
            }
        }

        --Traceability
        trace.refSetValue('sourceElement', elementSrc);
        elementSrc.refSetValue('sourceTrace', Se-
quence{elementSrc});
    }
}

rule process_add_shared_event_ins_del (ac-
tive_substructure: cametamodel!ComplexSubstructure){
    to
        operation: uml!Operation,
        elementSrc: tracer!Element(elementName<- ac-
tive_substructure.name, elementType<- 'Aggregation sub-
structure'),
        elementTrg: tracer!Element(elementName<- ac-
tive_substructure.name, elementType <- 'Event', targetTrace<-
Sequence{elementTrg}),
        trace: tracer!Trace(
            traceName<- 'Create
shared Operation',
            targetElement<- ele-
mentTrg)
    do{
        --Stablsh active_event
        self.active_event<-operation;
    }
}
```

```

        --Body
        self.active_event.class<-self.active_class;
        if(self.class_case = 'newly_created'){
            self.active_event.name<-'new_' +
self.active_class.name;
            --Adding data value inbound arguments and
object value inbound arguments
            for(p in active_substructure.directItems){
                if(self.is_of_type_data_field (p)){

                    self.process_add_data_value_inbound(p,
self.active_event);
                }else{

                    if(self.is_of_type_reference_field(p)){

                        self.process_add_object_value_inbound(p,
self.active_event);
                    }
                }
            }
        }

        --Traceability
        trace.refSetValue('sourceElement', elementSrc);
        elementSrc.refSetValue('sourceTrace', Se-
quence{elementSrc});
    }
}

rule process_add_data_value_inbound(dataField: cameta-
model!DataField, operation: uml!Operation){
    to
        parameter: uml!Parameter,
        elementSrc: tracer!Element(elementName<- data-
Field.name, elementType<- 'Data Field'),
        elementTrg: tracer!Element(elementName<- data-
Field,elementType <- 'Data value inbound',targetTrace<-
Sequence{elementTrg}),
        trace: tracer!Trace(
            traceName<- 'Create
Data value inbound',
            targetElement<- ele-
mentTrg)

```

268 9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
do{
  --Body
  if(self.class_case = 'newly_created'){
    parameter.name <-
'p_atr'+self.format_to_lower_underscore(dataField.name);
    parameter.operation<-operation;
    --Paramter inbound data type
    if(dataField.domain = #text){
      parameter.type<-
self.data_type_string;
    }else{
      if(dataField.domain = #date){
        parameter.type<-
self.data_type_date;
      }else{
        if(dataField.domain = #money){
          parameter.type<-
self.data_type_real;
        }else{
          parameter.type<-
self.data_type_integer;
        }
      }
    }
  }else{
    if(self.class_case = 'pre_existing'){
      parameter.name <-
'pt_'+self.format_to_lower_underscore(dataField.name);
      parameter.operation<-operation;
      --Paramter inbound data type
      if(dataField.domain = #text){
        parameter.type<-
self.data_type_string;
      }else{
        if(dataField.domain = #date){
          parameter.type<-
self.data_type_date;
        }else{
          if(dataField.domain =
#money){
            parameter.type<-
self.data_type_real;
          }else{
            parameter.type<-
self.data_type_integer;
          }
        }
      }
    }
  }
}
```



```

    }
  }
}

--Traceability
trace.refSetValue('sourceElement', elementSrc);
elementSrc.refSetValue('sourceTrace', Sequence{elementSrc});
}
}
rule process_add_object_value_inbound(referenceField:
cametamodel!ReferenceField, operation: uml!Operation){
  to
    parameter: uml!Parameter,
    elementSrc: tracer!Element(elementName<- referenceField.name,
elementType<- 'Reference Field'),
    elementTrg: tracer!Element(elementName<- referenceField.name,
elementType <- 'Object value inbound', targetTrace<-Sequence{elementTrg}),
    trace: tracer!Trace(
      traceName<- 'Create Object value inbound',
      targetElement<- elementTrg)
  do{
    --Body
    if(self.class_case = 'newly_created'){
      for(p in self.subs_classes_set){
        if(p.sub = referenceField.domain){--
Check the existence of the class
          parameter.name <-
'pAgr'+self.format_to_lower_underscore(referenceField.domain.name);
          parameter.operation<-operation;
          --Parameter inbound object type
          parameter.type<-p.class;
        }
      }
    }else{
      if(self.class_case = 'pre_existing'){
        for(p in self.subs_classes_set){
          if(p.sub = referenceField.domain){--
Check the existence of the class
            parameter.name <-
'p_this'+p.class.name;

```

270 9.1 ATL rules to transform Communication Analysis requirements models to OO-Method class diagram

```
        parameter.operation<-operation;
        --Paramter inbound object type
        parameter.type<-p.class;
    }
}
}
--Traceability
trace.refSetValue('sourceElement', elementSrc);
elementSrc.refSetValue('sourceTrace', Se-
quence{elementSrc});
}
}
```

