

Combining Design and Engineering Methodology for
Organizations with the Rational Unified Process

Ilias Begetis



Combining Design and Engineering Methodology for Organizations with the Rational Unified Process

Master Thesis in Computer Science MSc Program

By

Ilias Begetis

 **TU Delft** Delft University of Technology
Information System Design Group
Faculty of EEMCS
Delft University of Technology
Delft, The Netherlands
www.ewi.tudelft.nl

 **Mprise**
Mprise B.V.
Newtonstraat 2, 3902 HP
Veenendaal, The Netherlands
www.mprise.nl

Author

Candidate: Ilias Begetis, BSc
Student number: 1395075
Email: ibegetis@gmail.com

Title

Combining DEMO with the Rational Unified Process

MSc presentation

28th April 2010

Graduation Committee

Prof. dr. ir. J.L.G. Dietz (chair)	Information Systems Design Group, EEMCS Delft University of Technology
Ir. J. de Jong (company)	Mprise B.V.
Dr. J. Barjis (member)	Systems Engineering, TPM Delft University of Technology
Ir. B.R. Sodoyer (member)	Information Systems Design Group, EEMCS Delft University of Technology

Abstract

The software development has changed dramatically the last two decades. Software was primarily built in house 30 years ago, aiming to fit the needs of a specific user. During the 80's the tendency changed with the foundation of software houses that were specializing in the development of "off the self" software, fitting the needs of a wider group of users, thus achieving scale economy, cheaper software with better quality. The last years, the explosion of the internet usage has transferred all applications to the "cloud" exploiting the faster and cheaper than ever hardware and network. Furthermore, software systems such as Content Management Systems and Enterprise Resource Planning have transformed information system development into a process that connects the right components of functionality together.

However, no matter what the advances are, tailored software is still required. Organizations, like businesses and institutions, with a variety in characteristics like delivered services, size, people, business processes and operating rules will always have a need for a customized system that fits their needs. Thus, building software has become more complex not in terms of available technological solutions but in terms of determining user needs. There are still excellent flawless software systems that solve the wrong problem. Therefore, enriching software engineering processes with business modeling techniques has been one way to cope with this problem.

One of the most famous software engineering processes is the Rational Unified Process (RUP) which includes its own business modeling technique. In this thesis we try to combine DEMO and RUP in order to exploit the advantages of both methodologies which will ultimately assist practitioners in the development of quality software that solves the right problem. Our effort starts with the identification of a common scientific background, continues with devising a framework of assisting the combination and study of the methodologies. Then, the combined methodology is used in a case study in order to test in practice the new methodology.

Keywords: DEMO, Rational Unified Process, multi-methodology, business modeling, software engineering, software engineering process

Preface

Writing this preface while spending my last moments as a student and as a Netherlander forces me to go through all this time I've spent educating myself. Einstein once said that education is what remains when you have forgotten everything that you've learned in school. We will stick with him in these lines and nothing will be mentioned of what I have learned. There are plenty of pages following describing this.

What come first are the people. Literally since the first moment the first person that I met and worked together for two years is Thuan; among others a friend, colleague, project manager, house lord, side man, co-swimmer, co-biker. He made my studying experience easier although I should apologize for some of his white hair! Next is Puspa. A major contributor to this thesis which is something I am grateful for. Also a colleague at TU Delft and Mprise and of course a housemate. I consider myself lucky for spending all this time with her. Also, our dear colleague Dianna who I only remember her smiling, and pretty Yan! I hope I will meet them again before I leave! I will never forget any of my international friends and I promise that I will try to visit them in their home countries!

Finally, all 16 roommates that I had during this period and of course all my Greek friends especially Soto and Aristide who both have a very comfortable couch!

Next comes the support! After having two years of housing problems I would like to thank Joop de Jong for providing a great house for as long as it was required in order to focus on my thesis. Many thanks go to Ioannis as well for helping me in my first housing difficulties. Also, all these friends I left back in Greece for their remote support; Doc, Mitso, Bakakos, Yoshi, Georgia and of course my little cousin Aristeia! However, most of the support comes from my village and my two beloved parents. This thesis is a result of their patient as well and is dedicated to them! I can't wait to meet them again!

Finally, is all this fun I had during these years! All the people I met from around the world, all the places I have been to, all the parties and activities I had, all the bike paths I followed! I shall not thank anyone in specific but I ultimately this is what I will remember the most after few years!

While finalizing this preface I would like to thank my supervisors Joop de Jong and Professor Jan Dietz for accepting my proposal and providing me a wise supervision.

I hope you will enjoy reading this report!

Delft, 28th of April, 2010.

Ilias Begetis

Contents

Abstract	v
Preface	vii
Contents	ix
List of Figures	xii
List of Tables	xiv
1. Introduction	15
1.1. <i>Problem Statement</i>	16
1.2. <i>Research Questions</i>	16
1.3. <i>Research Approach</i>	17
1.4. <i>Thesis Outline</i>	18
2. Theoretical Background	19
2.1. <i>Systems and Systems theory</i>	19
2.2. <i>System Definition(s)</i>	19
2.3. <i>System Classifications</i>	21
2.4. <i>Methodology</i>	25
2.5. <i>Contribution to the Research</i>	28
3. Business Modeling in RUP	29
3.1. <i>Introduction</i>	29
3.2. <i>Scope of Business Modeling</i>	30
3.3. <i>The Business Modeling Discipline</i>	30
3.3.1. <i>Workflow</i>	30
3.3.2. <i>Roles & Artifacts</i>	32
3.3.2.1. <i>Roles</i>	32
3.3.2.2. <i>Artifacts</i>	32
<i>Documental Artifacts</i>	33
<i>Modeling Artifacts</i>	33
3.4. <i>The Business Use Case model</i>	33
3.5. <i>The Business Analysis model</i>	36
3.6. <i>Building the Business Model</i>	38
3.6.1. <i>Building the Business Use Case Model</i>	40
3.6.2. <i>Building the Business Analysis Model</i>	43
3.7. <i>The Role of Business Model</i>	48
3.7.1. <i>The Requirements Discipline</i>	48
3.7.2. <i>The Analysis & Design Discipline</i>	50
3.7.3. <i>Deriving System Models from the Business Model</i>	52
3.7.3.1. <i>Supporting Business Workers</i>	52
3.7.3.2. <i>Automated Business Workers</i>	53
3.7.3.3. <i>Analysis Model</i>	53
3.8. <i>Contribution to the Research</i>	54

4.	A Combination Framework	55
4.1.	<i>Theoretical Background</i>	55
4.2.	<i>Adapting the Framework</i>	58
4.2.1.	The Separation of the World	59
4.2.2.	The Intervention Phases	60
4.2.3.	The World Division Incompatibility	62
4.2.3.1.	The Role of the ICT in the Operation of the Organization	67
4.2.4.	The Adapted Combination Framework	70
4.2.5.	The Methodological Framework	71
4.3.	<i>Linking the Methodologies Together</i>	72
5.	Applying the Combination Framework	73
5.1.	<i>Introduction</i>	73
5.2.	<i>DEMO Grid</i>	73
5.2.1.	Way of thinking	73
5.2.2.	Way of modeling	74
5.2.3.	Way of working	75
5.2.4.	Way of controlling	76
5.2.5.	Way of supporting	77
5.3.	<i>RUP Grid</i>	78
5.3.1.	Way of thinking	80
5.3.2.	Way of modeling	81
5.3.3.	Way of working	84
5.3.4.	Way of controlling	87
5.3.5.	Way of supporting	88
6.	Combining DEMO with RUP	91
6.1.	<i>The combined frameworks</i>	91
6.2.	<i>Way of thinking</i>	91
6.3.	<i>Way of working</i>	94
6.4.	<i>Way of controlling</i>	95
6.5.	<i>Way of supporting</i>	96
6.6.	<i>Way of modeling</i>	96
6.6.1.	“Mimicking B-actors” scenario	97
6.6.1.1.	Building the Use Case Model of the B-application	98
6.6.1.2.	Building the Analysis Model of the B-application	101
6.6.2.	“Supporting B-actors” scenario	104
7.	Applying to the Case Study	107
7.1.	<i>Applying to the case study</i>	107
7.1.1.	The Case Study	107
7.2.	<i>The “way of working”</i>	108
7.3.	<i>Transforming DEMO models to RUP models</i>	109
7.4.	<i>Building the Analysis Model</i>	111
7.5.	<i>The added value of the case study</i>	113
8.	Discussion and Conclusions	115
8.1.	<i>Disadvantages of UML in Business Modeling</i>	115
8.2.	<i>The advantages of the multi-methodology</i>	116

Appendixes	119
<i>Appendix A – Class diagrams in UML</i>	119
<i>Appendix B – Discipline details in RUP</i>	120
<i>Appendix C – UML profiles</i>	121
<i>Appendix D – The narrative description of the case study</i>	122
<i>Appendix E – The DEMO models of Mprise [Sandhyaduhita, 2009]</i>	134
The Interaction Model (ATD +TRT) of Mprise	134
The Process Model of Mprise	136
The State Model of Mprise	139
The Interstricion Model of Mprise	141
Abbreviations	145
References	147

List of Figures

Figure 1 - Research Approach	18
Figure 2 - A system typology of the universe by Checkland (1993).....	25
Figure 3 - The framework of methodologies of Seligmann [Bronts et al., 1995].....	28
Figure 4 - The “Business Modeling” workflow	31
Figure 5 - Roles and Artifacts in Business Modeling	32
Figure 6 - The artifacts developed and their relationship	39
Figure 7 - Actor classification example	40
Figure 8 - Modeling the outsourcing of a process	40
Figure 9 - 'include' stereotype usage	41
Figure 10 - 'extend' stereotype usage.....	41
Figure 11 – Activity Diagram for a BUC specification	42
Figure 12 - A partition of the goal hierarchy	43
Figure 13 - The business class diagram	44
Figure 14 - The lifecycle diagram of “Training Schedule” entity.....	45
Figure 15 - Using statecharts to denote assignment of roles	45
Figure 16 - The realization of a BUC with a sequence diagram	47
Figure 17 - The process of capturing managing requirements	50
Figure 18 - Transforming BUC diagram into system use case diagram	52
Figure 19 - Deriving system use case from the business model.....	53
Figure 20 - Automating business processes	53
Figure 21 - Transforming BE into analysis classes.....	54
Figure 22 - The division of the world by Habermas	56
Figure 23 - The multi-methodology framework	57
Figure 24 – An example of using the framework	57
Figure 25 - The layering of computer systems.....	60
Figure 26 - The system design process	61
Figure 27 - Ontology, technology and implementation	62
Figure 28 - The System Development Process	62
Figure 29 - The Organization Theorem.....	63
Figure 30 - The integration of an organization [De Jong, 2010]	64
Figure 31 - Combining Organization theorem and SDP [Dietz, 2008].....	64
Figure 32 - The set of actor kind combinations between the B-organization and the I-organization [De Jong, 2010].....	65
Figure 33 - The set of actor kind combinations in the D-organization [De Jong, 2010]	65
Figure 34 - Archiving P-facts [De Jong, 2010].....	66
Figure 35 - The support of ICT in every aspect system.....	66
Figure 36 - The role of ICT in the operation of the I-organization	69
Figure 37 - The role of ICT in the operation of the D-organization.....	69
Figure 38 - The ICT systems that are of interest for the RUP practitioner	71
Figure 39 - Graphical representation of the operation axiom.....	74
Figure 40 - The basic pattern of a transaction.....	74
Figure 41 - The ontological aspect models and their diagrams	75
Figure 42 - The cross-model tables	77
Figure 43 - Software architecture model of RUP	82
Figure 44 - UML diagrams categorization [OMG, 2009]	84
Figure 45 - 2dimensional structure of RUP	84
Figure 46 - An iterative process	86
Figure 47 - Roles, activities and artifacts.....	87
Figure 48 - RUP Product framework	89
Figure 49 - Introducing DEMO in the RUP iterative nature	94
Figure 50 - Transforming the IAM into a use case diagram.....	98
Figure 51 - Transforming PSD into sequence diagram	98
Figure 52 - Transforming the complete pattern into an activity diagram.....	99
Figure 53 - A more complex case of transformation	100
Figure 54 - Transforming a SM into an analysis model	102
Figure 55 - Transforming a specialization	103
Figure 56 - An extension of a binary fact type.....	103
Figure 57 - An extension creating an association class	103
Figure 58 - An extension as the property of on class.....	103

Figure 59 – Transformation of an aggregation	104
Figure 60 - Transforming a union into a generalization class	104
Figure 61 - Use case model of the I-organization	105
Figure 62 - Use case of the B-application	110
Figure 63 - Analysis model of the B- and I-application	112
Figure 64 - State-machine diagram for the Enrollment class	112
Figure 65 - State machine diagram for the Course class	113
Figure 66 - Class diagram.....	119
Figure 67 - Object Diagram.....	119
Figure 68 - The structure of RUP	121
Figure 69 - The UML extensions mechanisms	122
Figure 70 - ATD for the standard course	134
Figure 71 - ATD for client specific course	134
Figure 72 - ATD for development and marketing	135
Figure 73 - TRT of ATD of Mprise	135
Figure 74 - PSD for the enrollment process.....	136
Figure 75 - PSD of the Standard Operation Process.....	136
Figure 76 - PSD for the Client-Course-Kind Production Process	137
Figure 77 - PSD for the Client Specific Course Operation.....	138
Figure 78 - PSD for the inovation process	138
Figure 79 - PSD for the Maintenance Process	139
Figure 80 - PSD for the Marketing Process	139
Figure 81 - The SM of Mprise.....	140
Figure 82 - ABD of the Standard Course.....	141
Figure 83 - OCD of the Standard Course.....	141
Figure 84 - ABD of the Client Specific Course	142
Figure 85 - OCD of the Client Specific Course	142
Figure 86 - ABD of the Development and Marketing	143
Figure 87 - OCD for the Development and Marketing.....	143

List of Tables

Table 1 - Thesis Outline-----	18
Table 2 - The system classification of Jordan (1968)-----	23
Table 3 - The system taxonomy of Boulding (1956)-----	24
Table 4 - Distinct classifications of methodology combinations-----	55
Table 5 - Adaptation of the combination framework-----	70
Table 6 - Applying the combination framework to DEMO-----	73
Table 7 - Applying the combination framework to RUP-----	78
Table 8 - UML views and diagrams relationship [Rumbaugh et al., 2005]-----	83
Table 9 - A combination of DEMO with RUP-----	91
Table 10 - The B-application actors and use cases-----	109
Table 11 - Class relationships-----	120

1. Introduction

The object oriented programming paradigm has offered a new way of developing software. The software is composed of objects which offer functionality to the programmer, offering a specific interface while hiding the internal construction of the object. Thus, common complicated parts of application programming do not have to be programmed again, removing errors and increasing maintainability of the system.

Furthermore, the new tendency is the Component-based Software Engineering where system is constructed using individual software components that encapsulate data and a set of semantically related functions. Some software systems are thus easy to be developed simply by combining the appropriate components. One example is the Content Management Systems which can build a web portal only by installing the right components. Also, the developer may not have to interfere with a programming language since he can do the development using a Graphical User Interface.

However there are still software systems that are easy to develop and free of coding errors but still solve the wrong problem. This is mainly because of the poor requirement elicitation. Software practitioners have coped with this issue by building a software prototype early in the development process which helps the development team and the users to understand better the requirements. Another way to cope with this flaw of the software engineering process is to model the business that will use the software program. By modeling and therefore understanding better the construction of the business it is easier to identify the functional requirements of the system that will be used. However, the construction of a software system is radically different than the construction of a business; a methodological swift is required when modeling a business or a software system. Doing this methodological swift may provide the engineers of a process that takes into account the distinct nature of these two system classes.

The Rational Unified Process (RUP) is a *software engineering process* which provides guidelines for a software development effort and is configurable for a variety of project types. It is also a *software development approach*, since it captures principles and commercially proven best practises. RUP activities are focused on the creation of *models* instead of paper documents. It is a guide for the effective use of Unified Modelling Language (UML) which is used in order to create models of the developing system. Also, it is a *product*. It offers tools that automate and facilitate the process' activities, tool mentors, a web based knowledge base with extensive guidelines, templates for the artifacts, and a customizable process framework [Rational, 2001].

RUP is also a *software engineering process* [Kruchten, 2000]. RUP is well-defined and well-structured; it provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget [Kruchten, 2000].

Design and Engineering Methodology for Organizations (DEMO) is a methodology that has proven itself powerful in modelling an organization, whether this is a business, a governmental authority or a non profit institution. DEMO approaches business¹ modelling from a theoretical perspective, based on the outcomes on the

¹ Although modeling in DEMO applies to a broader set of organizations than businesses we will refer to DEMO modeling as business modeling conventionally

philosophical notion of ontology. Also, it copes with the complexity of an organization by introducing distinct aspect models which adopt a specific point of view on the organization. Furthermore, it uses a diagrammatic language which is based on the theoretical background. The visualization procedure of DEMO is supported by the tool Xemod, which is developed by Mprise (former XPrise). Finally, DEMO adopts the Generic System Development Process which describes the process of designing and implementing systems and the role of ontology in this process.

1.1. Problem Statement

RUP appears to be a popular software engineering process among software developers. UML is the de facto standard in software design visualization and can model a software system in a concrete and consistent way.

But, standard UML does not provide a dedicated diagram for business modelling, thus the business modelling workflow faces a major drawback, the absence of a diagrammatic language that can capture the architecture of a business.

A first research has found some efforts to do business modelling using UML and RUP. They are mainly extensions of the existing diagrams, like use case diagrams and sequence diagrams in order to fit the business world [Baker, 2001] [Eriksson et al., 2000]. Also, an extension of RUP has been made that extends to the enterprise domain [Ambler et al., 2005]. This is a new methodology which has extended the 2-dimensional structure of RUP in order to create EUP (Enterprise Unified Process).

The multi-methodology framework that is discussed above can guide as through the development of a methodology that will combine the advantages of DEMO with the advantages of RUP. Thus, the main research goal of the project should be summarized as:

HOW TO EFFECTIVELY COMBINE DEMO WITH RUP

The combination effort will be based on the work of Mingers and Brocklesby (1997) who built a framework that can assist practitioners to combine methodologies. They have recognized the different levels of interfering methodologies which can be seen in Table 4. In this problem statement we choose not to commit ourselves in a multi-methodology level. Stating that we will combine these two methodologies means that we will create either a combination or a multi-methodology with respect to Table 4. More explanation can be found in the respective paragraphs.

1.2. Research Questions

The research questions which are driving and shaping the scope and the research approach of this project are stated in the following lines. These questions have been answered by this project, either explicitly or implicitly.

- i. What are the business modelling methods, techniques, models and diagrams that are used in the RUP process?
- ii. What are the advantages of DEMO over these techniques?
- iii. How can we effectively combine these two methodologies?
- iv. What framework should we construct in order to facilitate the combination?
- v. How can the tools, used by these two methodologies, be combined?
- vi. What are the advantages of the combined methodology?

1.3. Research Approach

Our research methodology was based on the research of Mingers and Brocklesby (1997) who introduced the term of multi-methodology as the most interweaving level of combining methodologies. Also, a case study is used that gives us a testing platform. This case study is Mprise division of training, which until recently was a separate business entity.

The steps we will follow are:

- i. *Identify a common scientific background.* In order to be able to study a methodology an investigation has been done about methodologies themselves. Thus, a terminology about methodology could be developed by identifying its parts. Also, a common scientific background should be found that would help us make objective comparisons between the two different paradigms of the methodologies. This background is system theory, a discipline that studies systems.
- ii. *A study of RUP and its business modelling techniques.* A study of RUP was done in order to reveal the business modelling techniques and diagrams. The technique has been applied to our case study. This, on the one hand, gave us an insight of the role of the Business Modelling workflow of RUP. On the other hand it helped us in building the framework of a multi-methodology and gave us the potential advantages and disadvantages that DEMO can offer to RUP.
- iii. *Creating a mapping framework for combining methodologies.* Based on the research on multi-methodology a mapping framework was created that was the base of a more thorough study of the methodologies and it will guide the project towards a systematic combination of the methodologies.
- iv. *Fit methodologies to the framework.* After building the framework a thorough study of the methodologies was done in order to identify the elements of each methodology and fit them to the framework.
- v. *Combining the methodologies.* After creating the framework and identifying its elements in the methodologies we combine them in a single intervention.
- vi. *Testing the new multi-methodology.* The developed methodology is tested in the case study in order to prove that it is practical. Also, it was proven that the theoretical study only was inadequate to reveal all the aspects of the combination.

The following diagram shows the sequence of the activities and the main artifacts that were used during the each activity. Some activities were done in parallel so to validate the theoretical part with specific cases.

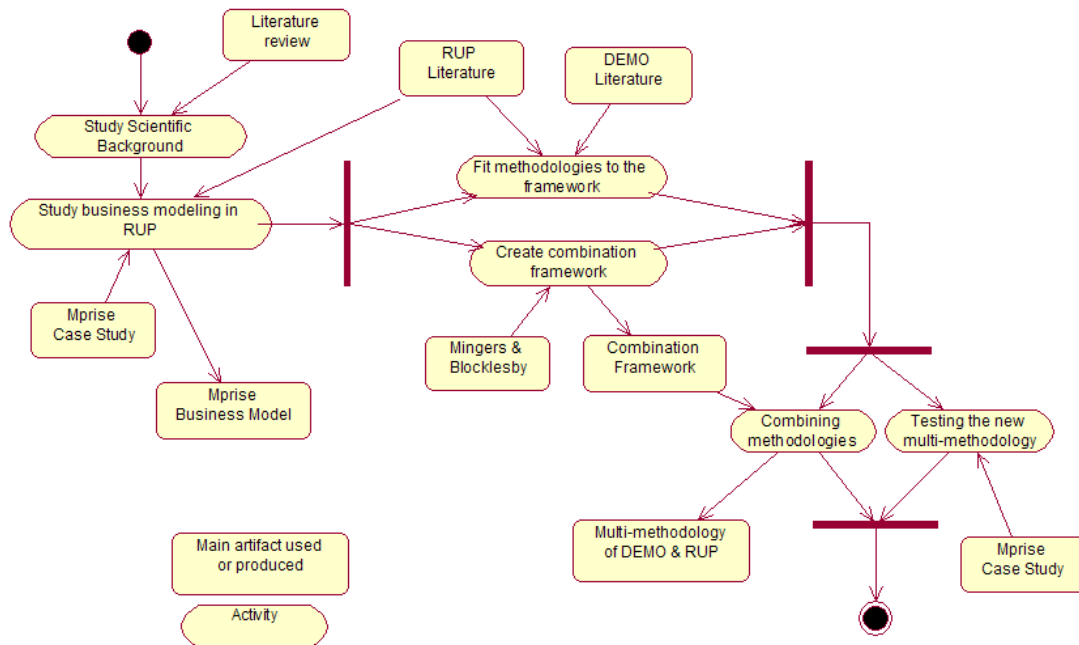


Figure 1 - Research Approach

1.4. Thesis Outline

The above activities have produced some artifacts. Most of them are part of the current thesis report. The following table shows the contents of each chapter and which of the above activities are related with each chapter.

Chapter	Related Activity	Content
Chapter 2	i.	Systems theory & theories about methodology.
Chapter 3	ii.	Business Modelling using RUP's technique & the respective UML profile
Chapter 4	iii.	Study of Mingers & Blocklesby's framework. Adaptation to the current project.
Chapter 5	iv.	Applying the adapted framework to both DEMO & RUP
Chapter 6	v.	Combining DEMO & RUP into a multi-methodology using the applied frameworks.
Chapter 7	vi.	Applying the multi-methodology to the case study and verify its soundness and utility.
Chapter 8	--	Summarizing the whole report and concluding the main findings.

Table 1 - Thesis Outline

2. Theoretical Background

2.1. *Systems and Systems theory*

During a research about systems and systems theory the first conclusion that the researcher will come up with is that the term system is one of the most broadly used terms within science, commerce and society. Almost all authors of the field mention that the use of the term “system”, as used during our daily life, usually has no content, it is not aligned with the definitions and it is often used in the wrong context. The misuse of the term has settled it meaningless and contentless [Flood et al., 1991] [Flood, 1990]. Also, the effort to create a new scientific discipline for the study and theory of systems meets obstacles in the generality of the term. As Boulding (1965) clearly stated “we always pay generality by sacrificing content and all we can say about practically everything is almost nothing”. However, the General System Theory was introduced in that paper and has evolved to a discipline with its own domain of inquiry, a body of knowledge and methodologies [Klir et al., 2003].

The system thinking was the transition from “machine age” to “system age” [Flood, 1990] after the inadequacy of machine thinking, as expressed through reductionism (breaking down elements into separate parts, building up from the parts) to explain notions such as emergence of properties and irreducible wholes that were appearing in the living organizations.

2.2. *System Definition(s)*

The research for a system definition has traced a plethora of them. Everyone is around the notions of element, property, relationship, interaction, irreducible whole, purpose, emerged property, boundary and environment. The main idea derived from all the definitions is that a system is an organized collection of elements, which possesses properties. The elements of the system influence each other either by exchanging information, energy or physical materials. The system has emerged properties, meaning that it has properties and accomplishes functions that the elements alone do not have. Systems interact with their environment exchanging information, energy or materials either in a clear and strict way or in an open and more unsettled way. Their boundary delimits the system from the environment. Finally, systems may accomplish a purpose for functioning whether this is easy observable or not. We will justify this paragraph in the following lines by quoting some system definitions that guided us.

The first definition that we will quote comes from Klir & Elias (2003). They define system as:

“A system, say system S , is an ordered pair $S=(A, R)$, where A denotes a set of relevant things and R denotes a relation among the things in set A .”
--

This definition introduces the terms “elements” (things) and relation among elements as fundamental in any kind of system. However, as the authors immediately identify this definition is too general and therefore of little pragmatic value. Thus, they suggest that “specific classes of ordered pairs (A, R) , relevant to recognized problems must be introduced”. These classes can be separated by two fundamental criteria:

- The certain kinds of things of each class.
- The certain kinds of relations of each class.

The first criterion has as example the classification of traditional science and technology into disciplines and specializations each focusing on certain kinds of elements. The second criterion creates classes of systems which are characterized by a

specific kind of relation with no commitment on the kinds of the elements that are related with each other. However, “the only things that need be common to all systems are identifiable entities and identifiable connections between them. In all other ways systems can vary unlimitedly” [Jordan, 1960]. Thus, “the core meaning (of a system) includes a set of entities and connections between them” [Jordan, 1960]. This definition is quoted first because it is considered fundamental since it recognizes the principle of identifying something as a system. Indeed, the notions of element and relationship are present in all the definitions, substantiating the above statements.

The second definition comes from Maier & Rechtin (2002) and states that a system is:

“A set of different elements so connected or related as to perform a unique function not performable by the elements alone”

This definition introduces the notion of emergence, properties that the whole holds which are not met by the elements individually. In the work of Georgiou (2003) a thorough discussion is done around the idea of emergence. In this paper he identifies twelve aspects of emergent properties in the literature. As he quotes from Checkland¹, “the most fundamental systems idea is that of emergent properties”. The different interpretations of the idea of emergent property as used in systems theory are summarized in the following lines. Thus the emergent property [Georgiou, 2003]:

- i. is attributable to a system as a whole
- ii. is an attribute with which interrelating elements of a system are associated
- iii. is an epistemological moment to interrelating elements of a system
- iv. is irreducible to knowledge of the interrelating elements of a system and is thus on a different epistemological dimension to such knowledge
- v. is irreducible to the potentially infinite points of view available about the system and is thus on a different epistemological dimension to such perspectives
- vi. is instantly conceivable as a whole
- vii. is a function of the relationship between observer and the system observed
- viii. can be reached through a plurality of epistemological routes
- ix. is a singular reference point (or singular set of reference points) without which a system is not posited as of any particular type
- x. is a singular reference point (or singular set of reference points) without which the elements of a system can not be readily understood as interrelating
- xi. is a singular reference point (or singular set of reference points) without which the system can not be readily understood as existing qua system
- xii. is the identity of a system.

The third definition is the one from Bertalanffy (1976) which comes from the book that introduced the General System Theory. He defines system simply as:

“A set of elements standing in interrelation among themselves and with the environment”

This definition focuses more in the interaction of the system with its environment and is influenced by the transition from the mechanistic thinking to system thinking. The mechanistic thinking was based on the closed system view [Flood, 1990] where no relationships are found or made between the elements of a system and its external environment [Flood et al., 1993]. On the other hand, system thinking is based on the open system view where a system exchanges material, information and (or) energy

¹ Checkland P (1981). *Rethinking a systems approach*. Journal of Applied Systems Analysis Vol. 8

with its environment across a boundary [Flood et al., 1993]. There are more views about systems, discussed in more detail in Flood [et al., 1993].

The substance of every theory is determined by the metaphors on which they are based. Metaphors are used to increase the understanding of phenomena by using analogies with which we are familiar. They flesh out “abstract organizing structures (models) through which we can conceive of a world around us” [Flood, 1990].

The last system definition that we will quote comes from Dietz (2006, 2008). This work is based on Bunge² and is based on the distinction of the ontological and teleological system notions which are concerned with the construction/operation and the function/behaviour of a system respectively. First we will discuss the ontological system notion and later the teleological.

Firstly, the definition of a system is based on the difference of aggregate and system. Both are collections of items but in an aggregation “the items are not held together by interaction bonds” where in a system there are relationships between elements which moreover make elements to influence each other. “The existence of just a relationship between the elements is insufficient”; the relationship should influence the related elements.

Further, a system should have the following properties: a composition, an environment, a structure and a production.

- The *composition* of a system is a set of proper elements, which are “those parts of it that can be engaged independently in mutually influencing relations”. Moreover, the category of the system is determined by the type of these relations.
- The *environment* of a system is “the set of proper elements from the same category that are not contained in its composition but that act on or are acted upon by elements of the composition”.
- The *structure* of a system “is the set of mutually influencing relations, as determined by the system category, among the system’s elements as well as between them and the elements in the environment”.
- Last, the *production* of a system is “what is brought about by the elements in the composition and transferred to the elements in the environment through interaction links”.
- The composition, the structure and the environment are collectively called the *construction* of the system,
- The composition and the structural relations between its elements are called the *kernel* of the system.

The teleological system notion is adequate for the purpose of using or controlling a system, it reveals the function that every designed system has and therefore the purpose of the system. Through its function a system can accomplish different purposes thus the purpose of a system is a relationship between a system and a stakeholder or user of the system. Indeed, the construction of a system is objective; it is not dependent on the viewpoint or purpose of the user.

2.3. System Classifications

In the previous section we mentioned the distinction between open and closed systems according to the degree of interaction with their environment. This distinction is an attempt to classify systems based on a certain characteristic. As we also mentioned, Klir & Elias (2003) categorize systems based on the type of elements or the type of

² Bunge M.A. (1979). Treatise on basic philosophy, Vol. 4: A world of systems. D. Reidel Publishing Company, 1977.

relations, with the scientific disciplines as an example of the former. Another example is the classification into concrete and abstract systems where the former are systems composed of tangible elements while the latter contain intangible elements such as concepts and terms. More classifications exist in more particular areas of study such as biology and mechanical engineering.

However there are some attempts to create a general classification of systems in which all the different categories of systems can fit. In the following paragraphs we will mention some of these.

The first classification attempt we will discuss comes from Jordan (1968). In this paper he creates “a classification into bipolar types that are somehow similar to dimensions”. These dimensions describe the information needed to specify an instance of a system: *rate of change*, *purpose* and *connectivity*. Each dimension defines a pair of opposite properties (values) that can be used in the specification of a system instance.

The ‘*rate of change*’ dimension classifies systems to those that change over time and those that do not, using the ‘*functional (dynamic)*’ and ‘*structural (static)*’ properties. These properties are determined by the time span under attention since everything decays after an infinite time span. A system is classified as static or dynamic depending on the connections between the entities comprising the system; if they can be understood from knowledge of the state of the system for any instant within a time period is static. Otherwise it is dynamic.

The ‘*purpose*’ dimension refers to “a distinguishable pattern of action” which drives the system to its goal state. The goal state is usually hindered or impeded by the system’s environment. The “purposive behavior” of a system can take two forms. On the one hand it can be directed towards the environment where the system, in order to reach its goal state, either modifies the environment or overcomes barriers posed by the environment or tries to circumvent and bypass these barriers. On the other hand it is directed towards the system itself, trying not to reach the equilibrium point, a state where system entropy is the maximum, by using homeostasis to reduce the entropy. The properties ‘*purposive*’ and ‘*non purposive*’ are used to specify this dimension.

Finally, the ‘*connectivity*’ dimension refers to the density of relations between the elements of a system. A system that is not densely connected and that no change is done to the remaining elements upon the change or removal of an element or relation is called ‘mechanical’. On the other hand the ‘organismic’ systems are those that are perceived intrinsically as a different system when an element or relation is removed or extirpated.

This dimension of connectivity encompasses the ideas of the mechanical and organic metaphor of systems. The mechanical metaphor (or closed system view) states that a system can be decomposed in standardized parts each with a definite function [Flood et al., 1991]. The system operates in repetitive circles, performing predetermined actions while the environment is hardly considered and entropy never decreases [Flood, 1990]. The organic metaphor (or open system view) incorporates ideas from biology, adopting the concepts of ‘level of resolution’, irreducible whole, complex network of elements and exchange with the environment [Flood, 1990].

Using the 3 dimensions and the two properties of each dimension a framework of 8 cells is being created where every cell maps to a combination of values for every dimension. A system can then fit to one of the cells depending on the value of every dimension. The author then classifies examples of systems that meet the requirements of each cell into the framework. He also recognizes that when talking about systems

we should only use the dimensional description of every system in order to avoid ‘verbal magic’ since it contains all the needed information to describe a system. A summary of the examples that Jordan has used in his framework is described in the following table.

Cell		Example
1.	Structural Purposive Mechanical	A road network.
2.	Structural Purposive Organismic	A suspension bridge.
3.	Structural Non-purposive Mechanical	A mountain range.
4.	Structural Non-purposive Organismic	A bubble or any other physical system in equilibrium.
5.	Functional Purposive Mechanical	A production line.
6.	Functional Purposive Organismic	Living organisms.
7.	Functional Non-purposive Mechanical	The changing flow of water as a result of a change in the river bed.
8.	Functional Non-purposive Organismic	The space-time continuum.

Table 2 - The system classification of Jordan (1968)

Another system classification comes from Boulding (1956). This paper introduces the General System Theory (GST) which will provide a general framework or structure of systems where the different disciplines will fit their corpus of knowledge in ordered and coherent manner. Thus, an interdisciplinary movement is created with form and structure whose concern is “the general relationships of the empirical world” and which will enable one expert within a discipline to communicate with experts from other disciplines.

Boulding mentions two approaches, rather complementary with each other, that can be used in the development of a GST framework. The first is to look over the empirical world and pick up certain general phenomena which are found in many different disciplines and then build up general theoretical models relevant to these phenomena. The other approach is to “arrange the empirical fields in a hierarchy of complexity of organization of their basic ‘individual’ unit of behavior and try to develop a level of abstraction appropriate to each”. Using the second approach mainly, he starts from an intuitive explanation of the levels of complexity and relates the scientific disciplines into each level. The resulting framework is briefly described in the following table, depicted from Chekland (1993).

Level	Characteristics	Examples	Relevant Disciplines	
1.	Structures Frameworks	Static	Crystal structures, bridges	Description, verbal or pictorial in any discipline
2.	Clock-works	Predetermined motion (may exhibit equilibrium)	Clocks, machines, the solar system	Physics, classical natural science
3.	Control mechanisms	Closed-loop control	Thermostats, homeostasis mechanisms in organisms	Control theory, cybernetics
4.	Open systems	Structurally self-maintaining	Flames, biological cells	Theory of metabolism (information theory)
5.	Lower organisms	Organized whole with functional parts, 'blue-printed' growth, reproduction	Plants	Botany
6.	Animals	A brain to guide total behavior, ability to learn	Birds and beasts	Zoology
7.	Man	Self-consciousness, knowledge of knowledge, symbolic language	Human beings	Biology, psychology
8.	Socio-cultural systems	Roles, communication, transmission of values	Families, the boy scouts, drinking clubs, nations	History, sociology, anthropology, behavioral science
9.	Transcendental systems	Inescapable unknowables	The idea of God	?

Table 3 - The system taxonomy of Boulding (1956)

The complexity is increasing from level 1 to level 9. By complexity Boulding means the difficulty of an outside observer to predict behavior of the system. This difficulty arises by the dependency of the behavior from the decisions made by consciousness and un-programmed decisions that emerge in conscious systems. The higher level systems encompasses the lower level systems and thus exhibit all the properties of the lower levels, such as an animal is made of cells and cells are operating based on a chemical system. In each defined level emergent properties arise, which makes the higher levels to be more than the aggregation of combination of the lower levels systems.

Boulding states that adequate theoretical models extend up to the 4th level and above this level system models are inadequate. Though, empirical knowledge is inadequate at practically all levels and that “the description of complex structures is still far from solved” even at the lowest levels and while moving to the upper level systems this description becomes even more noticeable. Finally he suggests that this framework and its implications should “prevent us from accepting as final a level of theoretical analysis which is below the level of empirical world which we are investigating”. Though a high level system incorporates all the lower levels and therefore an insight to these levels will provide valuable information.

Checkland (1993) has also created a so called “system typology” where he divides the world into five system classes which make up “a system map of the universe”. The following figure shows these classes and their relationships [Checkland, 1993].

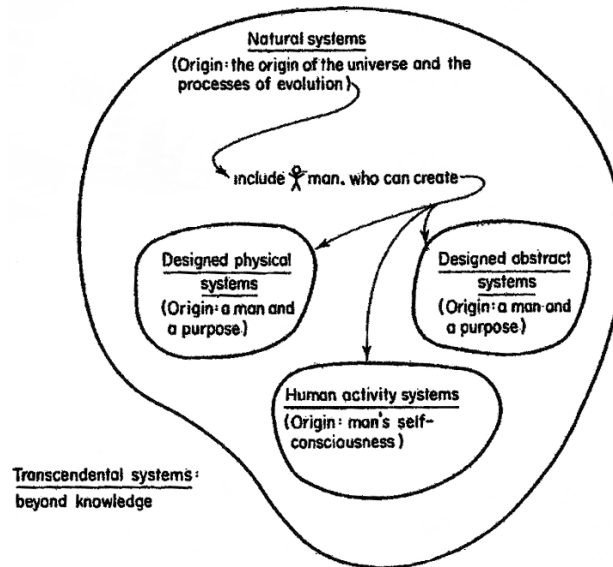


Figure 2 - A system typology of the universe by Checkland (1993)

Checkland’s typology starts from the *natural systems* which are the physical systems which make up the universe. These systems include physical, chemical, biological, solar and planetary systems. They pre-existed of the human and they are the result of the evolutionary processes. Natural systems may be investigated, observed and described by humans.

The *human* is included in the typology since he is responsible for the creation of the other three classes of systems. The keyword describing his actions is ‘purpose’.

The *designed physical systems* are made by human as a result of one of his purposes and exist to serve a purpose although sometimes it may be difficult to recognize it. They exist because a need for them in some human activity system.

The *designed abstract systems* are “the ordered conscious product of the human mind”. They are of intangible nature but can be captured in natural or designed physical systems such as books or CDs. Designed systems may be created and used.

The *human activity systems* are less intangible systems, but clearly observable since they consist of “innumerable sets of human activities more or less consciously ordered in wholes as a result of some underlying purpose or mission”. These activities can be enough to define the human activity system. Also, human activity systems are associated with other systems, usually designed physical systems, which their existence forms the human activity system as an irreducible entity. These systems are the main reason that makes an observer to identify a set of activities as an entity. Another characteristic of a human activity system is that they are dependent by the human intervention and human action in order to keep their identity.

Finally, the transcendental systems exist which are beyond the knowledge of the human and nothing can be said about them.

2.4. Methodology

The Marian-Webster dictionary defines methodology as:

- i. "the analysis of the principles of methods, rules, and postulates employed by a discipline"

- ii. "the systematic study of methods that are, can be, or have been applied within a discipline"
- iii. "a particular procedure or set of procedures"

From the above three definitions it is obvious that the term "methodology" has different meanings according to the context. In philosophy of science methodology is used to denote two concepts.

Within every scientific domain there exist three basic components [Klir et al., 2003]:

- A domain of inquiry
- A body of knowledge
- A methodology

The domain of inquiry refers to the particular class of systems that are examined by this domain [Klir et al., 2003]. This domain possesses a body of knowledge which is the existing knowledge that regards it. Thus, the methodology of this domain is "a coherent collection of methods for the acquisition of new knowledge within the domain as well as utilization of the knowledge for dealing with problems relevant to the domain" [Klir et al., 2003]

Thus, according to [Klir et al., 2003] the role of methodology is to create new knowledge within a domain. Mingers (1997) extends the role of methodology by stating that methodology "is a structured set of guidelines or activities in order to assist people in undertaking research or intervention". The new notion is that of the "intervention"; that a methodology is used not only to create new knowledge within a scientific domain but can also be used by problem solvers who are applying existing knowledge in order to solve a problem or create artifacts that will solve this problem, just like chemical engineers use the knowledge of pure chemistry in order to create a new chemical product which will solve a problem.

Based on the problem, the type of the solution for the problem or the type of agents that undertake the effort to solve the problem the methodology can be a procedure with specific steps and activities or a framework which only provides guidance and the main steps that have to be done. An example of the former are the software engineering processes, such as Rational Unified Process (RUP), which provides a clear procedure with clear steps and activities that have to be done. A software engineering process is "the total set of software engineering activities needed to transform user's requirements into software" [Humphrey, 1989]. To avoid the recursion of the definition the same author defines software engineering as "the disciplined application of engineering, scientific, and mathematical principles and methods to the economical production of quality software". The software engineering processes are another example of a problem solver (software developer) which intervenes in a problematic situation using knowledge from computer science.

Also, methodology is "a theory about how to conduct inquiry rationally" [Laudan, 1986]. It also includes the study of inquiry, it studies the way through which something happens.

Mingers (1997) divides a methodology into the following parts:

Paradigm is a very general set of philosophical assumptions that define the nature of possible research and intervention. Methodologies embody the philosophical assumptions of a paradigm and they can either be developed consciously as a methodology or they may emerge as guidelines of using good practices and specific techniques.

The different paradigms can be distinguished along three dimensions: ontology, epistemology, and praxiology.

- *Ontology* is the types of entities assumed to exist and the nature of that existence;
- *Epistemology* is the possibilities of, and limitations on, our knowledge of the world;
- *Praxiology* is how we should act in an informed and reflective manner.

A *technique* (or method³) is a specific activity that has a clear and well-defined purpose within the context of a methodology. Techniques may be complementary to each other and combine in a whole or they may be substitutes, using one technique instead of another. Techniques are the particular way of performing the activities that methodologies specify to be undertaken.

A *tool* is an artefact, often computer software, which can be used in performing a particular technique.

An *agent* is someone actively engaged in dealing with a particular problem situation. This term is used as an aggregation of the various terms used among different methodologies, like analyst, actor, practitioner, problem-solver, intervener and facilitator and since all of these terms disclose something about the activity that the agent has to accomplish the term agent is used instead.

Another framework for analyzing methodologies comes from Seligmann et al. (1989) and Wijers (1991) and is shown in Figure 3. Contrarily to the previous one, this is considered explicitly with information system (IS) methodologies. According to this framework a methodology consists of six components that each addresses specific aspects of the intervention process. These components are called “ways” and describe different aspects of a methodology.

- The “*way of thinking*” reflects the underlying principles of the methodology by delineating how we observe the application domain of the methodology. It also identifies the philosophical principles of the methodology and it may be referred as paradigm or underlying perspective or philosophy [Wijers, 1991], [Bronts et al., 1995].
- The “*way of working*” describes the process of intervention or developing a solution. It defines the possible tasks, their sub-tasks as well as the sequence of execution. It can also provide guidelines. Instructions and suggestions on how these tasks should be performed [Bronts et al., 1995]. The tasks can be divided into modelling and non-modelling tasks with the former focusing more on the construction or editing of models [Sattari, 2010]. These tasks are described in more detail on the “way of modelling”, something that justifies their relationship in the following figure. These two components constitute the operational level of the methodology.
- The “*way of modelling*” contains the dominant modelling techniques that are used in the representation of certain aspects of the methodology, whether this is the problem, its solution or an abstraction of the solution. Models are the most identifiable aspects of methodologies [Sattari, 2010] and one of the main artifacts of the methodology. The description of this component is abstract and includes the models, their concepts, their structure and relationships and their role in the process [Bronts, 1995].
- The “*way of controlling*” is concerned with the project management activities and techniques used through the project. It constitutes the managerial level of the methodology and it is highly related with the operational level since it

³ The authors substituted the term method with the term technique in order to reduce confusion between the terms methodology and method.

affects the operation of the process and on the same time needs feedback from the operational level in order to be performed.

- The “*way of supporting*” refers to the support of the managerial and operational levels of a methodology, possibly by tools which may include software, machines and communication systems or even pen and paper.
- The “*way of communicating*” describes how the abstract notions from the “*way of modelling*” are visualized and communicated to the different persons that are involved during the execution of a methodology [Bronts, 1995]. Usually it provides a graphical notation or the assumptions used in the modelling language. Thus, combining a “*way of communicating*” with a “*way of modelling*” provides the practitioner of a methodology with a modelling technique [Bronts, 1995].

The above components and their relations are displayed in the following figure:

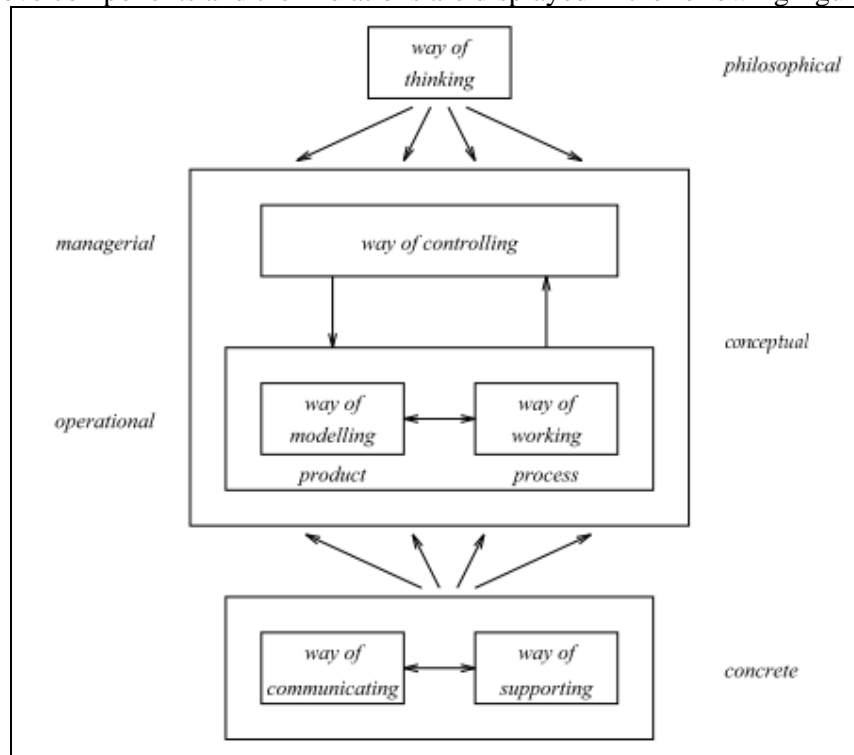


Figure 3 - The framework of methodologies of Seligmann [Bronts et al., 1995]

2.5. Contribution to the Research

This chapter provides as with fundamental knowledge on systems and methodologies. This provides a common scientific background on the domains of the two selected methodologies. The domain of DEMO on the one hand is businesses and organizations. The domain of RUP is software systems. Both domains refer to systems. Thus, systems theory provides us with a common scientific background which will help us to identify commonalities in the methodologies.

The second part of this chapter provides us with knowledge on the methodologies; what is their purpose and what are their elements. The two frameworks that have been identified will help us study and decompose the methodologies. Consequently, the combination will be accommodated and made in a sound theoretical basis.

The contribution of this chapter will be made clear as the reader moves forward to the chapters that describe the combination effort.

3. Business Modeling in RUP

3.1. Introduction

The RUP static description contains the “Business Modeling” discipline as the first one in a process execution and thus contains the initial activities that should be done in a project¹. Business modeling is a set of activities whose goal is to help project members to *visualize* and *understand* business processes and guide them during the construction of the system [Baker, 2001].

However, this discipline is not mandatory and many projects choose not to go through this [Rational, 2001]. Three general reasons for the necessity of a business modeling effort are also given by Baker (2001). Business modeling is required when the effort is towards to business process re-engineering, improvement or automation. Kruchten (2005) suggests that the business models created within the business modeling discipline are more useful “when there are more people directly involved in using the system” and “more information will be handled by the system”. This is the case for information systems that support the daily operations of an organization (business) and involve people who use them as a tool to accomplish their daily work. The role of business modeling in such cases is to analyze the organizational system where the software system will operate, the environment of the system. This is not the case for off-the-self, general purpose software such as office and multimedia applications which are general purpose, single user applications.

Kroll (2003) adopts a goal oriented view on the necessity of business modeling in a project. The business modeling efforts should be done when the software is being built in order to support a business and the business is not well understood. The amount of work depends on the impact of the software system to the business and the available knowledge about the business. If the business is well understood or the impact is small, like improving the efficiency of an algorithm or changing an application’s interface, then business modeling is redundant. However, even if this knowledge exists then it should be somehow transformed to system requirements. Also, in case of e-business transformation where the operation of an organization is totally altered this knowledge may become obsolete during the process of re-engineering the business thus an iterative transformation of the business models is more purposeful.

The purpose of the business modeling is not limited only to the above reasons. Kruchten (2005) and Rational (2003) contain a list of the goals of business modeling within RUP. These are:

- To understand the structure and the dynamics of the organization in which a system is to be deployed and how a to-be-deployed software system will fit into the organization.
- To assess the impact of organizational change.
- To understand current problems in the target organization and identify improvement potentials.
- To ensure that customers, end users, developers and other parties have a common understanding of the organization.
- To derive the software system requirements needed to support the target organization.

¹ The first activities done are the configuration and initialization of the process. Business modeling activities are the first referring activities towards the implementation of the system.

3.2. Scope of Business Modeling

The effort of modeling the business, as also stated before, is depended on the business knowledge available and the impact of the software to the business; the context and the needs of the developing system [Kruchten, 2005]. Thus, the scope of the business modeling is described in the following scenarios from [Kruchten, 2005] and [Rational, 2003] related to the characteristics of the developing system:

Organization chart: This is the case when the application to be developed is not intended to change or extensively affect the organization's operation. It contains only a simple charting of the organization and its processes, suitable to derive requirements for the application and the business modeling activities are part of the software engineering process, performed during the inception phase.

Domain modeling: In this case only a model of the information in the business level is built, without considering the workflows of the business. The domain modeling is also part of the software engineering process, performed in the early phases of inception and elaboration. It is suitable for applications that are mainly managing and presenting information.

One business, many systems: If the project is undertaken in order to create a large system or a family of applications then the business modeling can be undertaken as a separate project which will provide input to the several software projects. The developed models will help in the elicitation of functional requirements for the software and will facilitate the building of the architecture of the software.

Generic business model: When software is developed while considering that more than one business will use it then the business modeling effort should be taken in order to understand and manage differences in how the organizations will use the application and it will be easier to prioritize the application's functionality. Also, a business model can align the organization regarding their way of operating in order to avoid requirements that are too complex for the system, although this alignment won't always be an option.

New business: This is the case when a new line of business is planned to start, which will require the support of information systems. The role of business modeling is to elicitate requirements for the system and also to determine the feasibility of the new line of business. Usually, in this case the business modeling will be considered an individual project.

Revamp: In revamp, the organization is completely revamping the business operations and processes. The business modeling then, as well as software development, is a part of several projects needed, grouped under the term "Business Process Reengineering". Among them are: envision the new business, reverse engineer the existing business, forward engineer the new business, install the new business.

3.3. The Business Modeling Discipline

3.3.1. Workflow

The workflow of the business modelling expressed in an UML activity diagram, is the following² [Kruchten, 2005]:

² See appendix of how RUP details its disciplines.

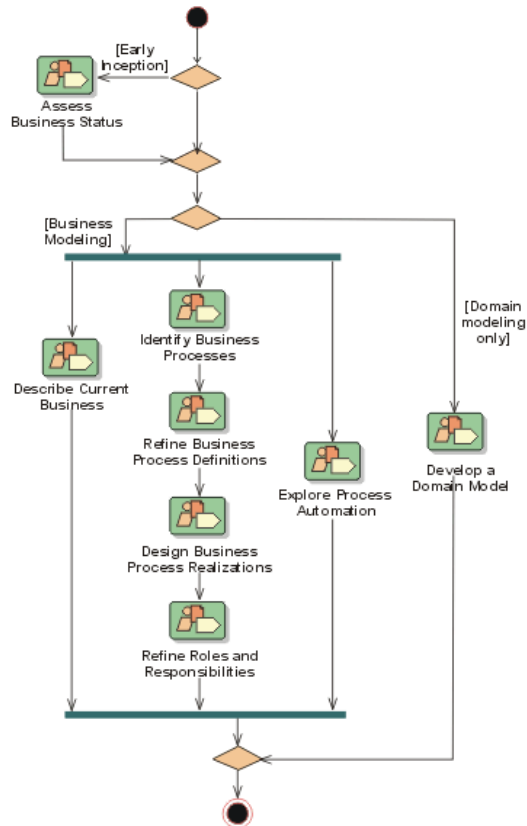


Figure 4 - The “Business Modeling” workflow

Depending on the purpose of the business modelling (scenarios 1-6) and the stage of the development lifecycle, a different path of the workflow can be followed [Rational, 2003], [Kroll, 2003].

In the early inception phase of the process lifecycle, the status of the target organization is assessed and improvement areas are identified. This assessment is the basis for decisions about the way of working in the next iterations, which is based on one of the previous scenarios, and captured in the ‘Business Vision’ and ‘Target Organizations Assessment’ artifacts.

If only a domain model is required (scenario 2) then the ‘Domain modelling only’ flow should be followed. A domain model is composed from the business entities³ of the organization. If no major changes will occur to the business processes then ‘scenario 1’ is followed and only a charting of the processes is needed in order to derive system requirements. In such case, the business modelling flow is followed, skipping the ‘Describe Current Business’ activity.

If a new software system will be deployed then the current business should be modelled in order to understand how the software will fit the organization.

If the purpose is the improvement or re-engineering or making significant changes to the business (scenarios 3, 4, 6) then both the current and the target business should be modelled. The result would be the ‘Business Architecture Document’ artifact which will contain an overview of the architecturally significant aspects of the business.

³ This term in the context of RUP is explained later.

Finally, if a new business is developed from scratch (scenario 5) then the models will describe the business under design and the description of the current business should be skipped.

3.3.2. Roles & Artifacts

The artifacts of the business modeling discipline, related with the roles that execute them are shown in the following figure [Kruchten, 2005].

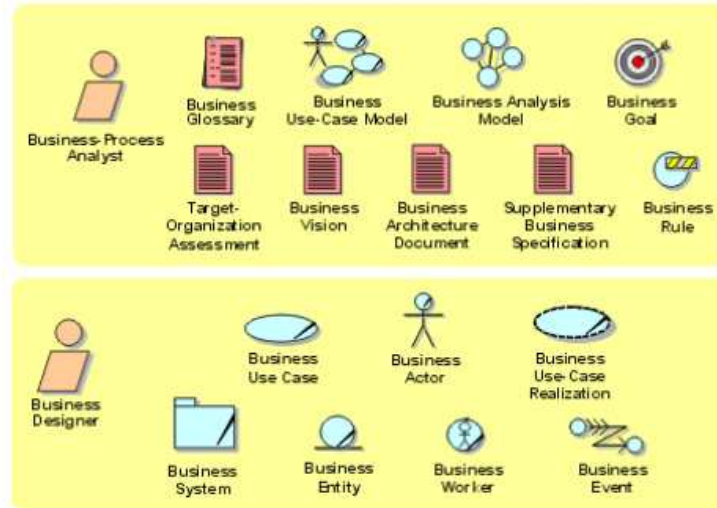


Figure 5 - Roles and Artifacts in Business Modeling

A short explanation of the above roles and artifacts is following, depicted mainly from [Kruchten, 2005] and [Rational, 2003]. For a more detailed description the reader should refer to these references.

3.3.2.1. Roles

Business Process Analyst: The business-process analyst is responsible for defining the business architecture, the business use cases and actors and how they interact by outlining and delimiting the organization by identifying what business actors and business use cases exist.

Business designer: The business designer role details the specification of a part of the organization by specifying the workflow of business use cases in terms of business workers and business entities. The details of business use case can also be specified using a textual flow of events description, an activity or a sequence diagram.

Also, roles that are involved in this discipline are the *various stakeholders* who represent the organization and may participate in the business modelling process by providing knowledge about the organization and the *business reviewer* who reviews the artifacts.

3.3.2.2. Artifacts

The artifacts are divided into two categories, documents and models which are easily recognized by the icons of the previous figure. The key artifacts of the discipline are the ‘Business Vision’, the ‘Business Use Case Model’ and the ‘Business Analysis Model’ [Kruchten, 2005]. A brief description of the minor artifacts follows, sorted alphabetically. A more detailed exists in Kruchten (2005) and Rational (2003). Due to their importance a whole paragraph has been dedicated to business use case and business analysis models.

Documental Artifacts

Business Architecture Document: Provides a comprehensive overview of the architecturally significant aspects of the business from a number of different perspectives.

Business Glossary: Defines important terms used in the business modelling part of the project.

Business Vision: Defines the set of goals and objectives at which the business modelling effort is aimed.

Supplementary Business Specifications: Presents quantifiers of the business not included in the Business Use Case Model or the Business Analysis Model, or constraints or restrictions to which the business must comply.

Target-Organization Assessment: Describes the current status of the organization in which the system is to be deployed in terms of current processes, tools, people's competencies, people's attitudes, customers, competitors, technical trends, problems, and improvement areas.

Modeling Artifacts

Business Event: Represents a significant occurrence in the activities of the business that requires immediate action. It is represented as a stereotyped signal.

Business Goal: A requirement that must be satisfied by the business in the form of the desired value of a particular measure. It is an optional artifact represented as a stereotyped UML class.

Business Rule: A declaration of policy or a condition that must be satisfied. It should be used only when there are many or complex conditions guiding business operations. It can be either modelled as a stereotyped constraint or in a document format.

Business System: Encapsulates a set of roles and resources that together fulfil a specific purpose and defines a set of responsibilities with which that purpose can be achieved. It is represented as a stereotyped package.

3.4. The Business Use Case model

The artifacts of Figure 5 are the deliverables of the business modelling discipline but not all of them are required in every RUP configuration. Though, the essential elements are the modelling elements; the business use-case model and the business analysis model, which together constitute the model of a business [Baker, 2001]. Both show the business processes from a different perspective. A business use-case diagram describes *what* the business does and the business object model describes *how* it does it. These two models are using the business modelling profile⁴ of UML [Johnston, 2004]. In the following paragraphs we will examine the two models and their main elements, starting from the business use case model which starts the modelling process.

The *Business Use Case Model* is “a model of the business goals and intended functions. It is used as an essential input to identify roles and deliverables in the organization” [Rational, 2003]. It shows *what* the business does from an external value added point of view. The main elements of this model are the business actor and the business use case [Rational, 2003] and the business use case diagram which relates the first two elements. After describing these two elements we will return to the BUC model for further discussion.

⁴ See Appendix C – UML profiles

A *Business Actor (BA)* “represents a role played in relation to the business by someone or something in the business environment” [Rational, 2003], where “someone or something” denotes that a BA can be a human, an organization, a machine or a system. The main characteristic of a BA is that he is placed outside the business boundary and interacts with it [Johnston, 2004], putting a demand on it while interested in the business output [Heumann, 2001]. Therefore he only has an external understanding of the business and its behaviour. Furthermore, he represents a role (or a set of roles) rather than an individual or a specific system [Johnston, 2004]. The name should be declarative of their role related to the business and not of a person; the person or system undertaking the role should be easy replaceable [Rational, 2003]. Finally, also in Rational (2003) it is strongly suggested that an actor should be involved with a business use case; otherwise he is redundant and should be removed. Also, if he interacts with several, completely different BUCs then he encompasses several roles and should be split to more actors. On the opposite, if more than one actor is related to one BUC then a shared role can be created using generalization relationships [Rational, 2003].

A *Business Use Case (BUC)* “defines a set of business use-case instances in which each instance is a sequence of actions that a business performs that yields an observable result of value to a particular business actor” [Rational, 2003]. More specifically a BUC defines a business process [Heumann, 2001] from an external, value-added point of view [Johnston, 2004] that performs a sequence of actions and produces an observable result of value to a particular BA [Ng, 2004]. Thus, BUCs are normally involved with at least one BA [Rational, 2003] and fulfill a goal [Behrens, 2004] which is specified from the point of view of the BA and the business [Rational, 2003]. The name of the BUC should briefly signify the result of the process, usually from the related actor’s point of view [Rational, 2003].

Each BUC defines a set of process instances where every instance can follow a different workflow and not every workflow will eventually be successful; the goal of the BUC may not be reached [Rational, 2003]. Thus, a BUC defines a class, as perceived from object oriented paradigm, which includes the instances of the essential process, which reaches the desired goal/value, as well as the alternative flows which may occur in exceptional cases depending on events and facts that may occur, regardless of terminating successfully or not [Johnston, 2004]. So BUCs unify all these possible similar workflows into one class that abstract details, based on the value that these instances create to the BA and in such a way that they are complete from an outside perspective [Rational, 2003]. This means that a BUC workflow starts with an externally clear trigger (event or message) and finishes with an externally clear result or message. This result may not always be the goal of the BUC. Ng (2002) believes that this is the true value of business modeling with BUCs: “understanding how seemingly piecemeal workflows relate to each other”. BUC workflows can run across different organizations, such as suppliers and service providers, which are outside the boundary of the business [Rational, 2003]. These outsourced actions of a workflow do not change the essence of a BUC since they are considered realization details of the workflow. Finally, a BUC workflow can run for a long period of time [Rational, 2003]. This can affect the identification of performance goals which generate requirements for the building system.

The *Business Use Case Specification* [Heumann, 2001], [Ng, 2002] documents the business use case details. Since the modeling artifact of the BUC has space only for the name the specification is maintained in a separate document. It captures all the

details that a BUC abstracts [Heumann, 2001]. Among the most important items that are included are the name and the goal/value of the BUC [Heumann, 2001]. Also, the preconditions and postconditions capture the necessary conditions before and the effects after the execution of a BUC [Rational, 2003] and therefore they implicitly state the sequence of BUC execution. Also items such as performance goals, the risks of executing and/or implementing the BUC, special requirements that may apply to the BUC and extension points are mentioned in the BUC specification [Heumann, 2001], [Ng, 2002], [Rational, 2003].

Finally, the most important part is the description of the main and alternative workflows of the BUC [Heumann, 2001]. This documents the main and alternative workflows using textual description or/and UML diagrams, usually activity or sequence diagrams [Heumann, 2001]. The first step towards the documentation of a BUC is the identification of the initial conditions before the execution of a BUC and the goal of the BUC [Ng, 2002]. Then the main flow documents the steps from the initial condition directly to the goal [Ng, 2002]. When the main flow is identified then all the different situations that might arise at each step are identified and documented as alternative flows [Ng, 2002]. This identification can be done either by interviewing business workers or brainstorming possible exceptions.

The main characteristic is that the workflow should describe what the business does to deliver value to the business actor and not how it does it [Heumann, 2001]. The distinction between the “how” and the “what” often can be unclear and not easy to be done [Ng, 2004]. The same author proposes on that matter that “BUC specification should be used to distinguish process invariants (fixed elements) from what can be re-engineered”, therefore remove the implementation details from the description since the implementation of a system is always a subject of re-engineering. Furthermore, Johnston (2004) provides 4 guidelines that help the business designer on how to state the “what” and not the “how:

- i. avoid natural language expressions that have a technical connotation,
- ii. do not imply technical implementation of the flow,
- iii. follow a simple synchronous request/response communication model
- iv. the sequence of steps does not imply the order of the functionality and is often mandatory only in relation to the next interaction with an actor

Also, the workflow description should be clear and easy to understand for people outside the business modeling team and it shouldn't contain actors, activities, business workers or BUC that are outside the scope of the BUC or not related to it [Rational, 2003].

After describing the basic elements of the BUC model we should elaborate on this a little bit more. The BUC model illustrates “how the business is used by its customers and partners” [Rational, 2003]. It does so by relating BAs with BUC in BUC diagrams. A BA is always related to a BUC, though a BUC may not be related to an actor. Also a BUC can be classified into 3 categories: core, management and supporting. Thus a core BUC is always related to a BA but a managerial or a supporting may not be [Rational, 2003]. Still, all the BUCs are oriented in providing value to the BAs, the only thing that changes between the above categories is the triggering source; a BA for a core, a time interval or an event for a supporting or management. Furthermore, BUC model contains the direction and intent of the business; this is done respectively with a business goal hierarchy and with the added value and means of interaction with the stakeholders of the business [Johnston, 2004]. The role of the BUC model in the system design and construction process is to

understand the way the business interacts with its environment and provide the context for the software development and the software itself [Johnston, 2004].

3.5. The Business Analysis model⁵

The second most important element of a business model is the business analysis model. The BUC model shows what the business does, while this models “how” it does this: it “describes the realization of business use cases by interacting business workers and business entities” [Rational, 2003]. It is an abstraction of the way business workers and business entities are related and how business workers collaborate with each other in order to perform the business use cases [Rational, 2003]. The main concepts of business workers, business entities and BUC realization are described further in the following paragraphs.

A *Business Entity* (BE) “represents a significant and persistent piece of information that is manipulated by business actors and business workers” [Rational, 2003]. It represents a “thing” handled or used by business workers [Baker, 2001] while they execute their activities and tasks within a BUC. The use of “represent” denotes that a business entity is not the actual, tangible thing i.e. a “classroom” but a representation, an abstraction of it. However intangible “things” can also be considered as a business entity as well. Furthermore, any piece of information that can be considered a property of something is probably not a business entity itself but an attribute of it [Johnston, 2004]; an attribute can be considered a piece of information about an instance of an entity [Rational, 2003]. The problem of identifying attributes is not so simple since it depends on the scoping of the system as well as the design objective. A general rule from Rational (2003) is that the business designer “models a phenomenon as an attribute if no more than one object needs to have direct access to it or if the only natural way to access it is through the object”.

Another characteristic of the BEs is that they are passive [Rational, 2003] and thus they do not initiate action and interaction on their own. Also, they can be used in many different business use case realizations [Johnston, 2004] and accessed, inspected, manipulated, produced by more than one business workers and thus they provide a basis for information sharing among the business workers [Ng, 2004]. Also, entities have a lifecycle. The lifecycle of entities span across different BUC [Behrens, 2004] and thus it gives an overview of where and how the entities are manipulated.

Finally, the set of all BEs serves as a business *domain model* [Ng, 2004] since they capture the most important type of objects in the context of the domain (the business) [Behrens, 2004]. In a domain model, besides the entity and a brief description of them two more requirements can be captured: the relationships between the business entities and their multiplicities as well as their constraints and derivation rules (e.g. age is computed by birth date) [Behrens, 2004]. A class diagram of the entities is required to support the description and show the relationships among them. The importance of the business domain model is that it provides a common vocabulary of terms with precise meanings for the use case specifications and thus creates a common and shared understanding [Behrens, 2004], [Johnston, 2004]. The domain model can be considered a partition of the static subset of the business analysis model [Johnston, 2004].

⁵ Several authors name this model as “Object” model [Heumann, 2001], [Baker, 2001], [Ng, 2004]. We will stick to the official RUP documentation and name it “Analysis” instead.

A *Business Worker (BW)* is “an abstraction of a human or software system that represents a role performed within BUC realizations” [Rational, 2003]. It is a role or a set of roles undertaken from humans or software systems or machines within the organization [Baker, 2001]. The role of a business worker within a business model is to collaborate with other BWs, use and manipulate BEs and interact with the BAs in order to realize BUCs [Johnston, 2004]. BWs are initiated when a BUC in which they are involved is started and exist as long as the workflow of the BUC executes [Rational, 2003]. The most important element of a BW is the operations he has to execute, the specific activities and tasks performed within a BUC. These operations are activated by messages from other BWs, BAs or business events that occur within the business [Rational, 2003]. BWs are not a compulsory artifact and should be modelled only when changes to the organization need to be considered [Rational, 2003].

A *Business Use Case Realization (BUR)* “describes how business workers, business entities, and business events collaborate to perform a particular business use case” [Rational, 2003]. The BUR describes *how* the steps of the workflow of a BUC are performed, in contrast with the BUC which describes what steps are performed. BUCs give an outside perspective of a business while the BURs gives an inside perspective [Johnston, 2004]. However, it is not always easy to define what is an outside and what is an inside perspective. Furthermore, similarly to BUC specification, a BUR should contain all the alternative workflows of a BUC. Also, there is always a BW that handles the interaction between a BA and the business and acts as the interface of the business [Rational, 2003].

Ng (2004) recognizes three approaches in realizing BUCs. These are:

- Focus on work processes
The focus is on the work processes, identifying BWs, BAs and their responsibilities. A sequence diagram can be used to describe the realization of the basic flow of a BUC. Also, activity diagrams with swimlanes that partition activities according to the responsible BWs can be used. The benefits are that the process is being understood without involving system or information details and identify which BWs, BAs and their responsibilities are time consuming, resource intensive or error prone in order to prioritize automation needs.
- Focus on process automation
The focus is on exploring the automation of BAs and BWs responsibilities. More specifically when and how they use business systems, what are the responsibilities of business systems in support of BAs and BWs. The realization is modelled with a sequence diagram which shows the messages exchanged between BAs and BWs with the business systems of the organization. This view gives an understanding of how business systems are used by BAs and BWs as part of their responsibilities and facilitates the derivation of system use cases.
- Focus on information processes.
The focus is on information processes, how business entities are manipulated. In this case a sequence diagram showing the interactions between BWs and BEs is used. A collaboration diagram can also be used.

3.6. *Building the Business Model*

Based on the above theory we created the business model of the Mprise. The model is based on the narrative description of Mprise which has been created by Puspa Sandhyaduhita for her graduation project.

The artifacts created and the relationships between them are modeled in the following UML class diagram⁶. The classes of this diagram are representations of the above concepts and thus they also provide a short description of the above theory.

⁶ For an introduction to class diagrams see Appendix A – Class diagrams in UML

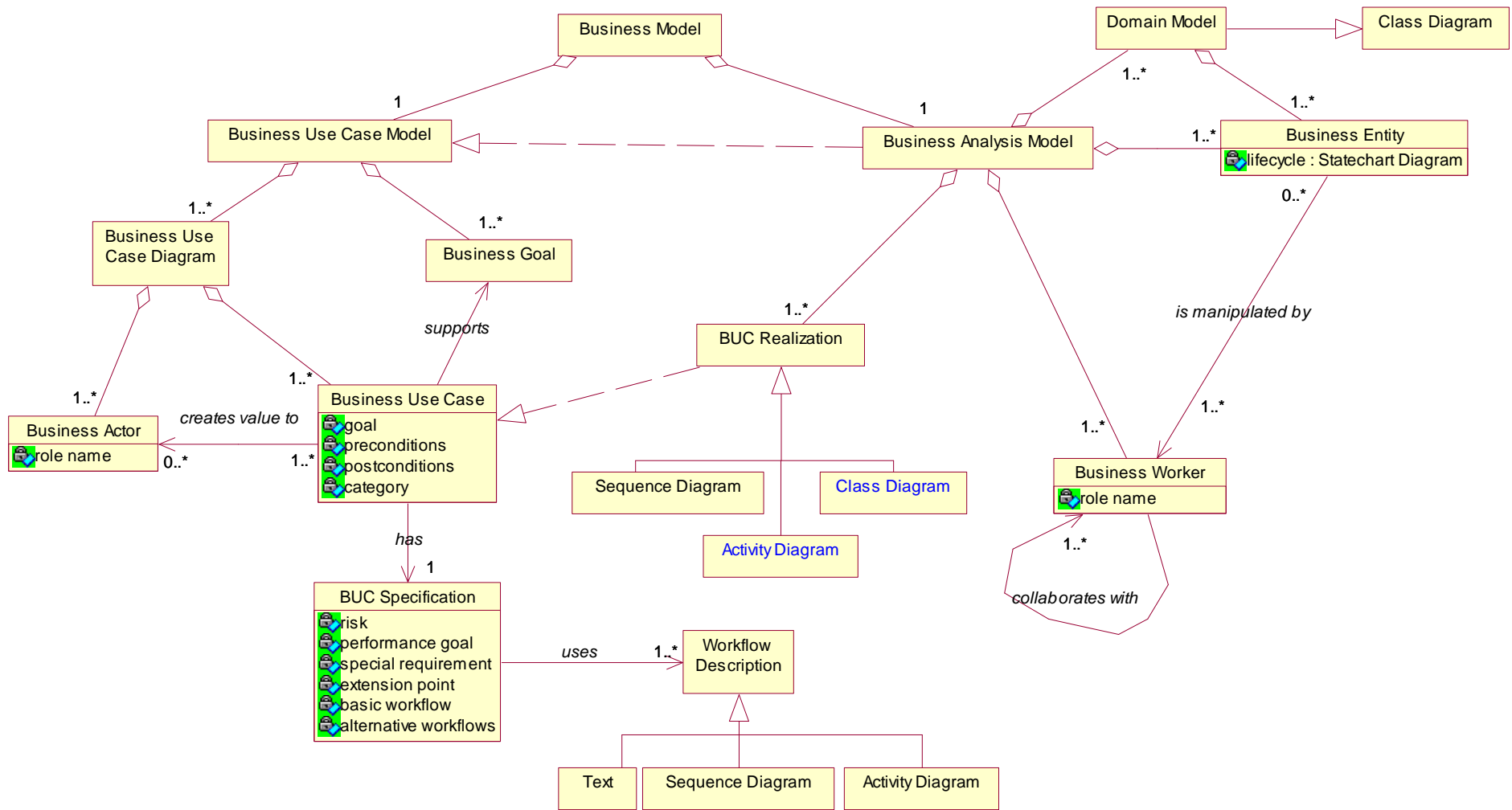


Figure 6 - The artifacts developed and their relationship

3.6.1. Building the Business Use Case Model

In this paragraph we will explain the process of building the business use case model, the guidelines we followed and some artifacts created. The complete report of the business use case model which contains all the diagrams exists in different documents for further reference. The reader should have in mind that RUP is an iterative process and thus the sequence of the steps is not strictly linear. Some artifacts developed in sequence but were not completed until some other artifacts had been started.

The first step is to identify business actors [Ng, 2004]. This step is quite straightforward since the boundary of the business is quite clear and easily identifiable. Also, generalization relationships have been used in order to join common roles undertaken by different individuals. An example is the “client” BA which is a role that can be taken from several types of actors, classified from attributes such as the size of the business or the type of usage of software systems. A part of the diagram is shown.

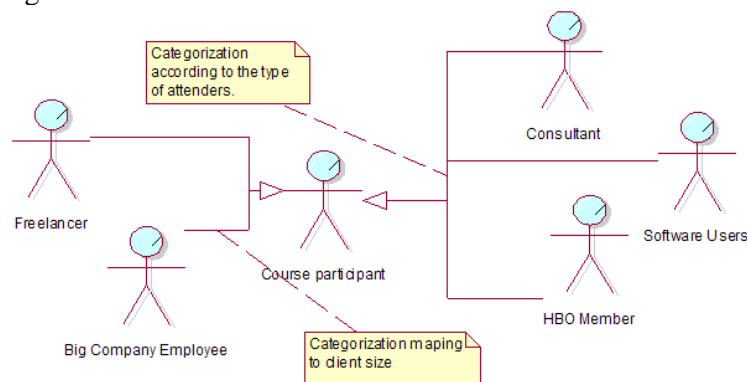


Figure 7 - Actor classification example

Ng (2004) identifies the problem that there is always the case that something can either be a BA or a BW and the separation may not be clear. Indeed this was the case for the suppliers of Mprise. Also, it is a rule of RUP that a BA should be involved in a BUC otherwise he is redundant. However, a BUC can not be identified for a supplier since a BUC does not deliver value to them. There is no obvious goal reached by i.e. outsourcing the attendants’ lunch in a company from the point of view of Mprise. It is a step in a BUC. The supplier is only related to the realization of a BUC. But he is still outside the scope of the business and he can only be modeled as a BA. One solution is to relate suppliers with the BUC who are involved with, in a BUC diagram. This is illustrated in the figure:

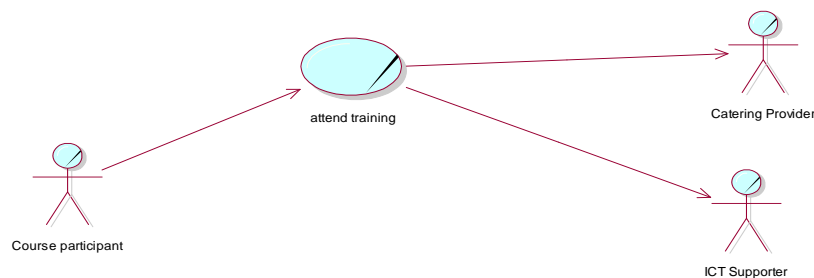


Figure 8 - Modeling the outsourcing of a process

However, this solution illustrates details about the “how” of the business and it’s not so relevant about the “what”. A more sound solution is to involve this kind of BAs in

the realization of the BUC in the Business Object model. This involvement will be shown in the following sections.

In the next step, BUCs were identified. The identification of a BUC starts with the identification of a goal. If there is not a goal then possibly there is not a BUC as well. Also, the preconditions and postconditions are identified. The identification of the preconditions is valuable to the descriptions of the workflows as it is already described. During the first steps of building the BUC model the identification of BUC had focused in the identification of mainly the BAs and the BWs. This helped in the BAs and BWs identification and thus provided a first step in the specification and realization of BUCs.

Also, BUCs have been related with “include” and “extend” stereotyped relationships. The “include” stereotype is used to “partition out parts of a workflow for which the base use case only depends on the result, not the method for reaching the result” [Rational, 2003]. Thus, this relationship is used in the case of the course product development where the output of the BUC “course product development” is used from the base BUCs “get course information” and “register for a standard course program” and the way of producing the output is not relevant. This is shown in the following partition of the BUC diagram:

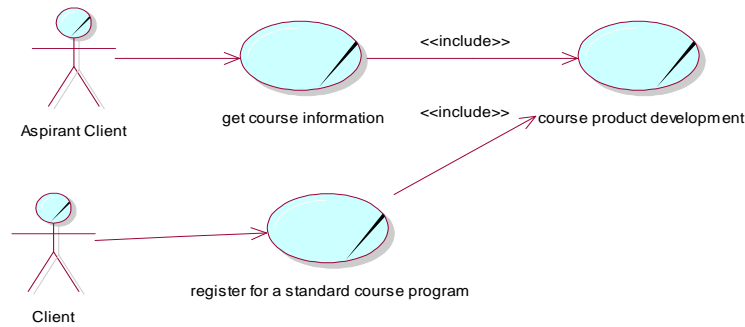


Figure 9 - 'include' stereotype usage

The “extend” relationship adds a flow to a business use case that is already complete in itself. This flow is conditional or optional and describes an exception or extension of the base BUC. This kind of stereotype has been used in the case of the “register for a standard course” BUC. This BUC can be followed by a “change date of training” BUC which is an extensional workflow that may follow or not the registration of a standard course program.

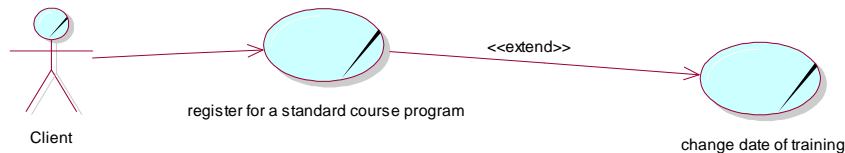


Figure 10 - 'extend' stereotype usage

For every identified BUC there is a specification document which contains the information of a BUC, among them its goals and the workflows, main and alternatives. The fields in the documents are: Introduction, Goals & Performance goals, Workflow, Category, Risk(s), Preconditions, Postconditions, Special Requirements and Extension Points. Also, we violated slightly the rule of workflow description which requires stating what is done; we stated what is done by whom. This approach is oriented towards the “how” of the business; the business analysis

model, and therefore acts as a connection link between these two models. It is also a first step to identify business workers. A UML activity diagram has been used where necessary to give a graphical explanation of the workflow. Also, swimlanes have been used to show the “who” of every step in the cases where this has been identified. This case was mostly preferred for workflow which contains a lot of branches and alternative workflows. The following activity diagram is the full workflow of “register for a standard course program” BUC.

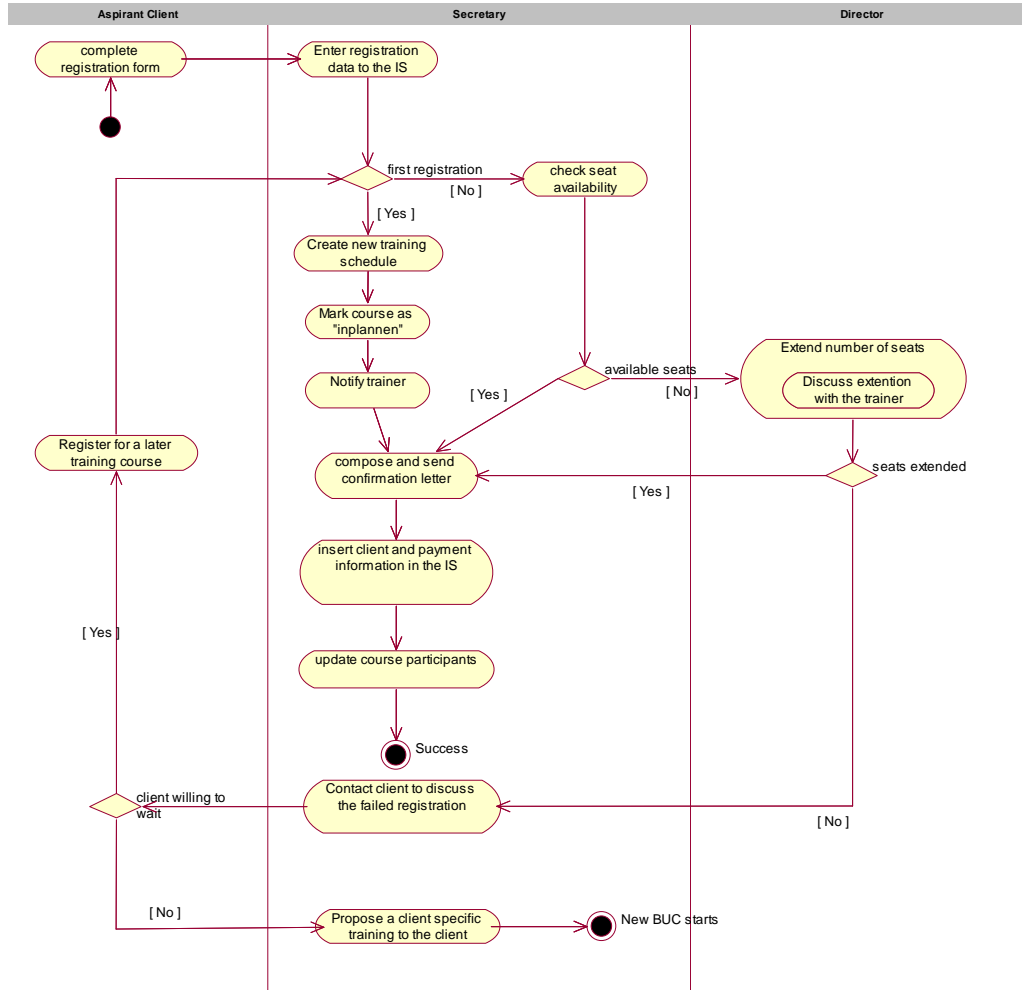


Figure 11 – Activity Diagram for a BUC specification

Finally, a goal hierarchy for Mprise has been created which shows the business goals of Mprise, following a specification hierarchy where the lower levels make more specific and measurable the higher level goals. Also, BUCs have been related with the goals that they support. A partition of this diagram follows.

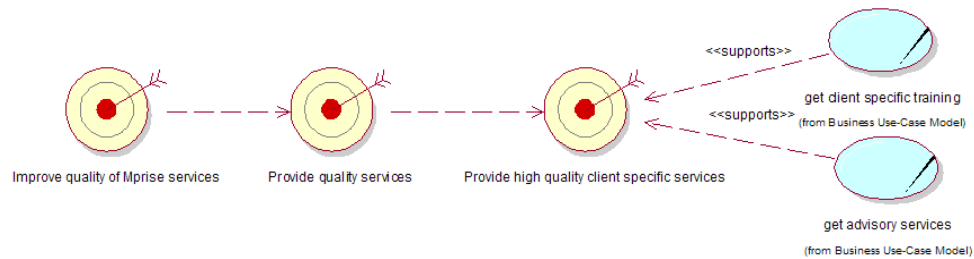


Figure 12 - A partition of the goal hierarchy

3.6.2. Building the Business Analysis Model

We will start the description of the business analysis model from the business class diagrams which also include the domain model of the business. The following diagrams extend the domain model by relating entities with each other as well as with BWs. During the development of the model we followed these rules from Behrens (2004):

- i. prefer redundancy to generalizations,
- ii. prefer attributes for simple types to relationships,
- iii. omit the types of attributes when they are obvious,
- iv. omit identifiers,
- v. use role name only when they help to clarify the context.

The whole domain model has been partitioned in two diagrams for improved readability. The main part of the business class diagram is shown below. The business analysis model report contains the description of every entity and worker of the following diagram.

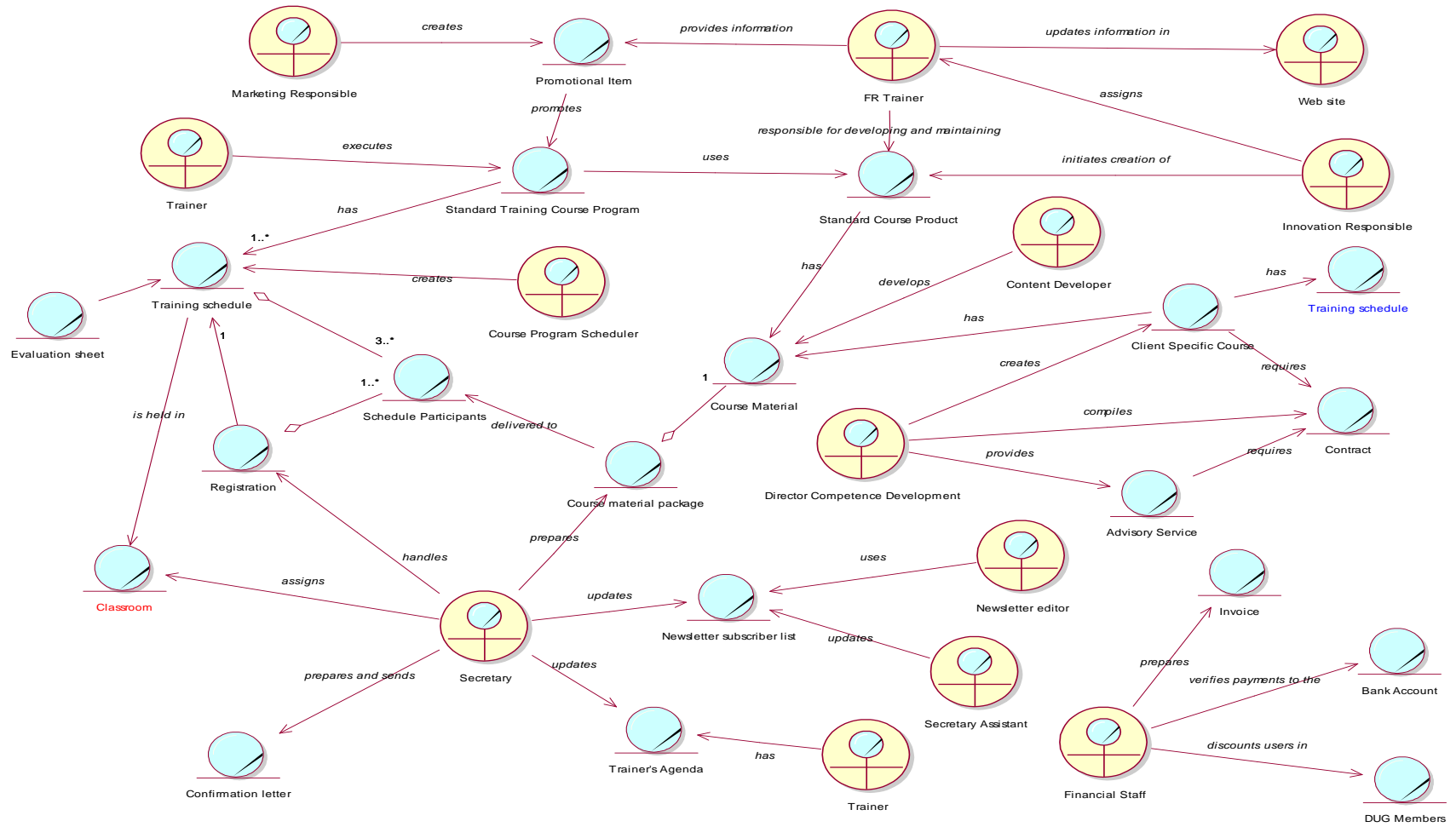


Figure 13 - The business class diagram

One suggestion of Behrens (2004) is to use lifecycles for the most important entities. UML state chart diagrams can be used for modelling the lifecycles. This state chart shows the manipulation of entities that changes their state across different BUCs. In the following diagram the lifecycle of “Training schedule” entity is shown. Rectangles represent states and arrows connecting them represent events that activate transitions from one state to another. Although Behrens (2004) suggests that the arrows should map to the BUC where these events occur, our case contains events that occur within the same BUC, depending on the workflow followed, and using this guideline will create a vague and ambiguous diagram.

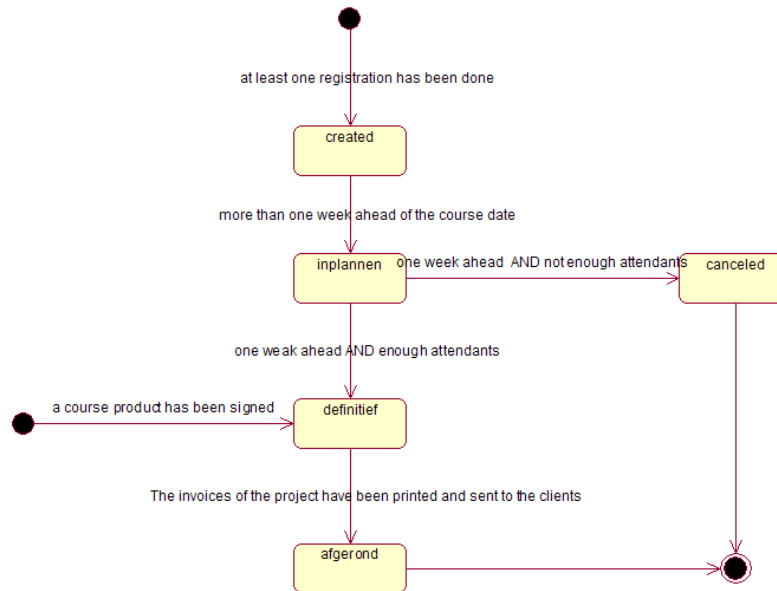


Figure 14 - The lifecycle diagram of “Training Schedule” entity

State chart diagrams have been used for BWs and BAs where applicable. These diagrams show the state transition in their attributes. Also, by convention the states show the transition of the undertaken roles by the BA or BW. Although this tailored technique can create over-assignment of roles in a single actor this potential problem has been recognized and taken into account. An example of using state chart diagram for the transition of roles is the BA “Secretary Assistant” which substitutes the role “Secretary” every Friday.

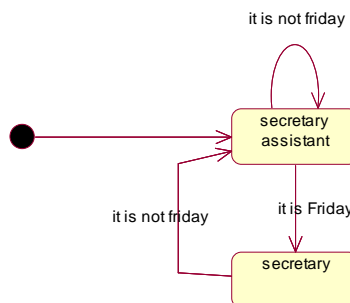


Figure 15 - Using state charts to denote assignment of roles

Finally, the Business Analysis model contains the realization of BUCs. These artifacts explain how the business realizes the BUCs. The approaches we chose, referring to the three approaches of Ng (2004) stated before, were to focus on work and information processes. Sequence diagrams including BWs, BEs and in some cases BAs have been used. Not all of the BUC have been realized with a sequence diagram.

Some BUCs have a small sequence of steps or the interaction among BWs either by communicating or exchanging information is simple; it does not contain a lot of interactions or information exchanging. Also, BAs have been included in the sequence diagrams. This is done in cases where Mprise is outsourcing some of the steps of a BUC. This technique allows showing that something is outsourced since this can not be done in the BUC model.

The sequence diagram of the realization of the “attend training” BUC is shown which also includes BAs. This sequence diagram shows the process of preparing and attending to a course, how BWs collaborate with each other, how they share information and what is the role of the BAs in the preparation and completion of the BUC realization.

Every arrow indicates a message, pointing to the receiver. This message indicates the request of an operation to the responsible business worker with the name of the operation being the label of the message. Also, any input information is indicated as argument of the operation. By using this technique we finally get the operations of every BW and BA since every incoming message is mapping to a BW’s responsibility [Ng, 2004]. Finally, using this technique we get a list of the responsibilities of every worker.

Ng (2004) also suggests that the labels of the arrows (thus the messages and the operations) should contain verbs such as “prepare” and “review” and should be avoided verbs such as “submit”, “approve” and “reject”. This is suggested because while the former refer to responsibilities and activities the later refer to events or actions. For example “submit” assumes the submission of a document which can be considered more correctly as an event or an action that causes a transition and not as an activity.

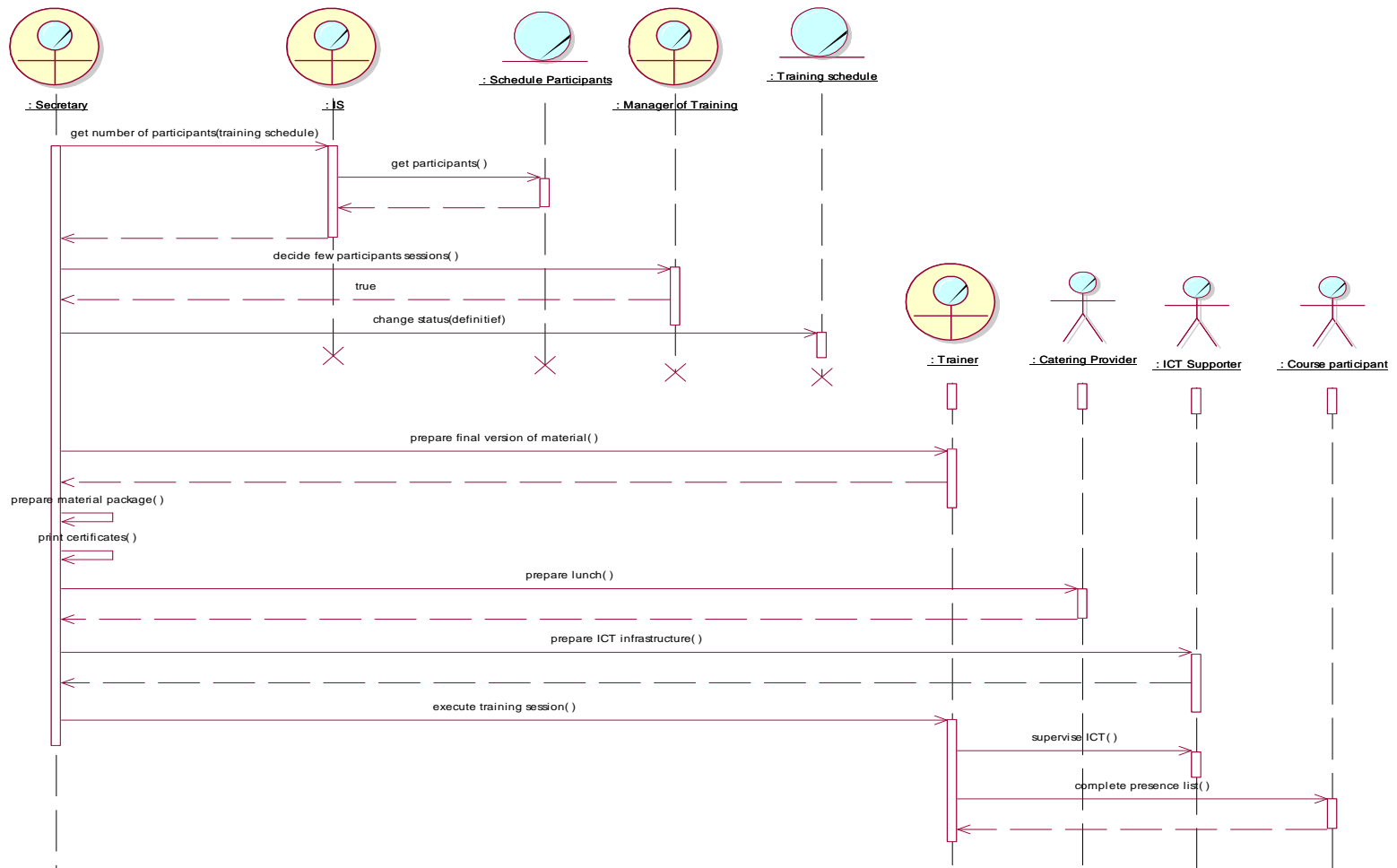


Figure 16 - The realization of a BUC with a sequence diagram

3.7. The Role of Business Model

The Business Modeling discipline and the artifacts created within this discipline are strongly connected with the software engineering process. First, they use the same modeling language, UML. Second, the business model provides the context where the software intensive system will operate.

Baker (2001) identifies the advantages that UML provides to the business modeling effort.

- i. UML provides a common language for business analysts and developers thus making the communication among them easier since they use the same symbols, diagrams and notations.
- ii. UML is visual and therefore the systemic aspects of business: who, how, what, why and when things are happening, are represented by visual elements which reduce complexity and ambiguities.
- iii. UML is object oriented and thus the modeled objects reflect real world entities very closely.
- iv. UML describes both the structural and dynamic aspects of business processes using the different diagrams of UML which model a system from different viewpoints.
- v. UML focus on the customer since it is use case oriented and therefore includes the client early in the process in an external, value driven view of the business.
- vi. UML helps derive better system requirements by identifying the business context where the system will operate. Furthermore, the business model can serve as a direct input to a requirements model. RUP defines direct mapping between the two models.

The business discipline can be done independently of a software development effort [Heumann, 2004]. However the last benefit mentioned before -help derive better system requirements- is the most important and we should examine this in more detail. The business model is a direct input to a requirements model and RUP defines a direct mapping between business models and requirement models [Heumann, 2004], [Kruchten, 2004], [Rational, 2003]. Furthermore, Rational (2003) mentions the relation of the Business modeling discipline to the other disciplines. These are [Rational, 2003]:

- *Requirements*: This discipline uses the business model as an input in deriving system requirements and building the requirement models.
- *Analysis and Design*: The business models are used as an input to define software systems that seamlessly fit into the organization.
- *Deployment*: Business models are an aid in planning the deployment of a software system.
- *Environment*: Guidelines are developed and supporting artifacts are maintained within this discipline, i.e. Business Modeling Guidelines.

Before examining the mapping that RUP defines for the creation of requirements model the Requirements discipline and its models and artifacts will be briefly described.

3.7.1. The Requirements Discipline

The main objective of the requirements discipline is to derive system requirements from stakeholder needs. These requirements describe what the system should do and facilitate the agreement between customers and developers [Rational, 2001]. To

achieve this, the system analysts elicit, organize and document the required functionality and constraints posed on the system by the stakeholders [Rational, 2001]. The goals of this workflow are [Kruchten, 2000]:

- To establish and maintain agreement with the customers and other stakeholders on what the system should do and why.
- To provide system developers with a better understanding of the system requirements
- To define the boundaries of (scope) the system
- To provide a basis for planning the technical contents of iterations
- To provide a basis for estimating cost and time to develop the system
- To define a user interface for the system, focusing on the needs and goals of the users

The main outcome of this workflow is a vision document, which “defines the stakeholders’ view of the product to be developed, specified in terms of the stakeholders’ key needs and features”, a use case model which is “a model of the system’s intended functions and its environment, and serves as a contract between the customer and the developers” and finally, a Supplementary Specifications document which “captures system requirements that are not readily captured in behavioural requirements artifacts” [Rational, 2003].

RUP defines requirement as “a condition or capability to which a system must conform” [Kruchten, 2004]. The requirements are categorized in two major categories: functional and non-functional. Functional requirements “specify actions that a system must be able to perform, without taking physical constraints into consideration” [Rational, 2003]. These types of requirements are described in the use-case model.

Actor and use cases are included in the use case model. An actor “defines a coherent set of roles that users of the system can play when interacting with it” [Rational, 2003] and it can be either an individual, a group of individuals or a system. A use case “defines a set of use-case instances, where each instance is a sequence of actions a system performs that yields an observable result of value to a particular actor” [Rational, 2003]. The steps that are performed in the interaction of the system with the actors are described in a use case specification which includes the basic and alternative flows of events, the preconditions and the postconditions as well as extension points and special requirements [Kruchten, 2004]. Thus the use case model provides a functional model of the system which relates user roles, functions and the results created by the system as described from an external point of view.

Non-functional requirements are further divided into: usability, reliability, performance, supportability; giving a categorization which is abbreviated as FURPS+. The plus (+) sign indicates that requirements about the design constraints, the implementation, the interface and the physical characteristic of the system should be included as well [Rational, 2003]. These types of requirements are described in the Supplementary Specifications and a software tool may be used for their management [Rational, 2001]. However, this categorization is only a template for requirements elicitation and for assessing the completeness of the understanding of the system [Kruchten, 2004]. It is not so important if a requirement is classified in usability or supportability as long as it is identified.

The process of capturing and managing requirements is described in Kruchten (2004) and Figure 17 explains it briefly.

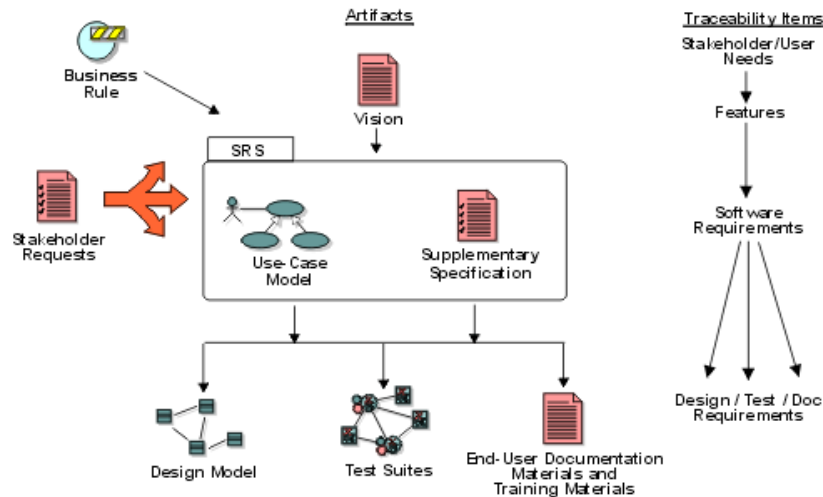


Figure 17 - The process of capturing managing requirements

First the stakeholders' requests and wishes are being captured by users, customers, marketing and other project stakeholders. These requests are used to develop the Vision document which contains the important stakeholder and user needs together with the high level functions of the system. A feature can also be included based on the cost of implementation and the return of investment. This kind of decisions is documented in the business case of the project.

The features documented in the Vision are translated into detailed software requirements at a level at which an actual system can be designed, built and tested. This translation is the use case model for the functional requirements and the supplementary specifications for the non-functional. These two artifacts constitute the Software Requirements Specifications (SRS) which captures the complete system requirements [Kruchten, 2004], [Rational, 2003].

Then, the SRS is translated into a design model that realizes the requirements, a test suite that tests if the requirements have been met correctly and the system/user documentation which documents the system from several perspectives.

Apparently there is a relationship between these artifacts which is called traceability. Traceability is "the ability to trace a project element to other related project elements, especially those related to requirements" [Rational, 2003]. These elements are called traceability items. These items and their relationships are shown on the right side of the above figure. The purpose of traceability is to [Rational, 2003]: understand the source of requirements, manage the scope of the project, manage changes to requirements, verify that all requirements of the system are fulfilled by the implementation and that the application does only what it was intended to do. Traceability exists in order to help management of requirements in an iterative process. The design model can change the use case model or flaws and inconsistencies may be that require the change of the initial vision and use case model [Kruchten, 2004]. Therefore questions should be given on how a requirement is derived, how this is realized, why a requirement has been removed from the initial list, how a requirement is derived from another etc. Anyway, iterative software development and management of requirements are two of the six RUP best practices.

3.7.2. The Analysis & Design Discipline

The Analysis & Design discipline transforms the requirements into a specification that describes how to implement the system [Kruchten, 2004]. The main goal is to

show how the system will be realized in the implementation phase [Rational, 2001]. The system should be designed such that it performs the tasks and functions specified in the use case descriptions, fulfils all its requirements, it is structured to be robust, adapt the design to match the implementation environment [Rational, 2001], [Rational, 2003].

The main artifacts created in this discipline are the analysis model and the design model.

The analysis model is an optional and a temporary artifact since it is a rough sketch of the system that can evolve to the design model by refining it further [Kruchten, 2004]. It is constructed in the elaboration phase and updated in the Construction phase of the project [Rational, 2003]. It is an “object model describing the realization of use cases, and which serves as an abstraction of the Design Model” [Rational, 2003]. It represents an early conceptual model of the system [Rational, 2003] and it is worth developing in cases of large, long living systems because the analysis model omits most of the details and provides an overview of how the system works [Kruchten, 2004].

The *analysis model* is composed of analysis classes and their associations in a class diagram. An analysis class “represents an early conceptual model for 'things in the system which have responsibilities and behavior’” [Rational, 2003]. They provide a first glance by identifying the prototypical classes of the system; they give rise to the major abstractions of the system but rarely survive into the design unchanged [Rational, 2003].

Also, RUP uses analysis mechanisms in order to reduce the complexity of analysis and improve the consistency of the analysis model. These mechanisms represent patterns that map a well tested solution to a common problem. They are “placeholders’ for complex technology in the middle and lower layers of the architecture” [Rational, 2003] which abstract the complexity of implementing an identified mechanism at the analysis level. They can also be viewed as a framework that match technological, problem domain unrelated, behavior to a domain related system or class. Examples can be error handling, notifications and object persistence. The *design model* is “an object model describing the realization of use cases, and serves as an abstraction of the implementation model and its source code” [Rational, 2003]. The design model is an abstraction of the source code [Rational, 2001] and adapts the analysis model to the constraints derived from non-functional requirements, the implementation platform and the performance of the system [Kruchten, 2004]. The design stops when the models provide enough information so the system can be implemented unambiguously. This statement is very general because the depth of design is strongly dependent on the specific project. The granularity of the specifications is dependent on the expertise of the implementer, the complexity of the design and the risk that the design might be misunderstood. The most important parts of the design model are the design classes with their relationships and the use case realizations.

A *design class* is “a description of a set of objects that share the same responsibilities, relationships, operations, attributes, and semantics” [Rational, 2003]. The class is the fundamental concept of object oriented paradigm. For the modern object oriented languages, such as Java, everything is represented as an object. Classes classify sets of objects in patterns. Furthermore, classes represent concepts [Fowler, 1997]. There are three perspectives which determine the concepts that a class represents [Fowler, 1997]. In the conceptual level classes represent concepts in the domain under study of the real world. In this level classes “should be drawn with little or no regard for the

software that might implement it” [Fowler, 1997]. The Analysis model maps to this level. The specification level of the classes is focusing in interfaces of the classes –the methods and operations- and not in the implementation. Finally, there is the implementation level which models the actual classes that implement the system.

The other part is the *use case realizations*. This artifact “describes how a particular use case is realized within the design model, in terms of collaborating objects” [Rational, 2003]. This description is done for every use case of the system, using various UML diagrams, such as class diagrams, sequence diagrams and communication diagrams which show the messages exchanged among classes while the system executes a use case. This allows the use case to be totally separated from its implementation and therefore changes in the implementation do not affect the use case [Rational, 2003].

3.7.3. Deriving System Models from the Business Model

RUP provides clear rules on how to derive system models from the business models. The business model of the business discipline is related to the use case model and the analysis model of the system. The derivation rules are depending on the goal of the system as will be shown. Two system development scenarios have been identified which denote the goal of the development process. The first scenario is to support business workers with software and the second is to replace business workers with software by automating them. Both hold some common guidelines for developing the analysis model of the system.

3.7.3.1. Supporting Business Workers

Every business worker who is undertaken by a person in the business analysis model is transformed to a system actor in the system use case model. The name of the actor remains the same. For each business use case in which the business worker participates, a candidate system use case is identified [Kruchten, 2004].

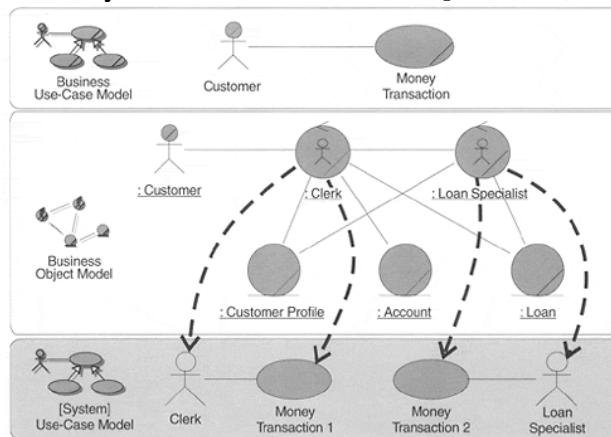


Figure 18 - Transforming BUC diagram into system use case diagram

Also, for every identified system actor the use cases that are related with him are identified in the business use case realization. The system use cases are the responsibilities of the business worker. Thus, every message in the BUC sequence diagram will possibly transform to a system use case diagram.

An example is the newsletter editor. The system actor “Newsletter Editor” has two responsibilities identified in the BUC realizations and thus he is related with two system use cases.

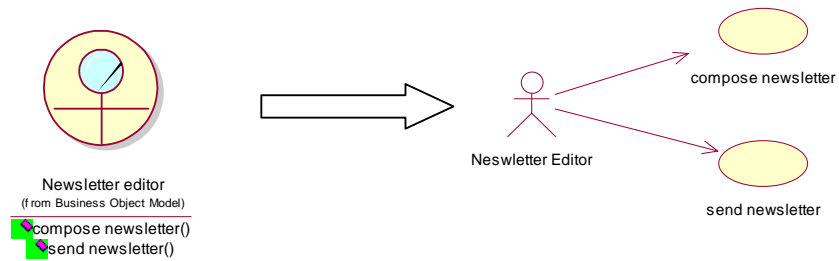


Figure 19 - Deriving system use case from the business model

3.7.3.2. Automated Business Workers

The system under development may automate some business workers where responsibilities from a business worker are transferred to a business actor. In such case business workers are removed so the business actor can initiate a use case. In such case the business actor is becoming a system actor.

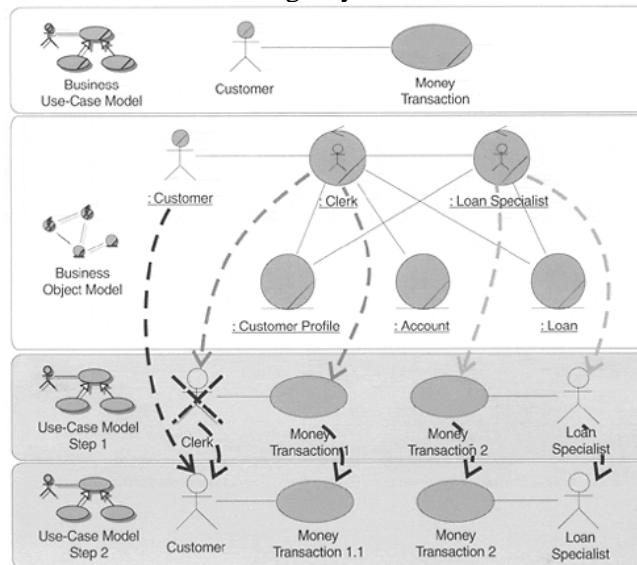


Figure 20 - Automating business processes

3.7.3.3. Analysis Model

A business entity in the business analysis model corresponds to an analysis class in the Analysis model. Also, some attributes may correspond to analysis classes. The relationships between business entities often indicate a corresponding relationship between the corresponding analysis classes. These guidelines apply to both scenarios.

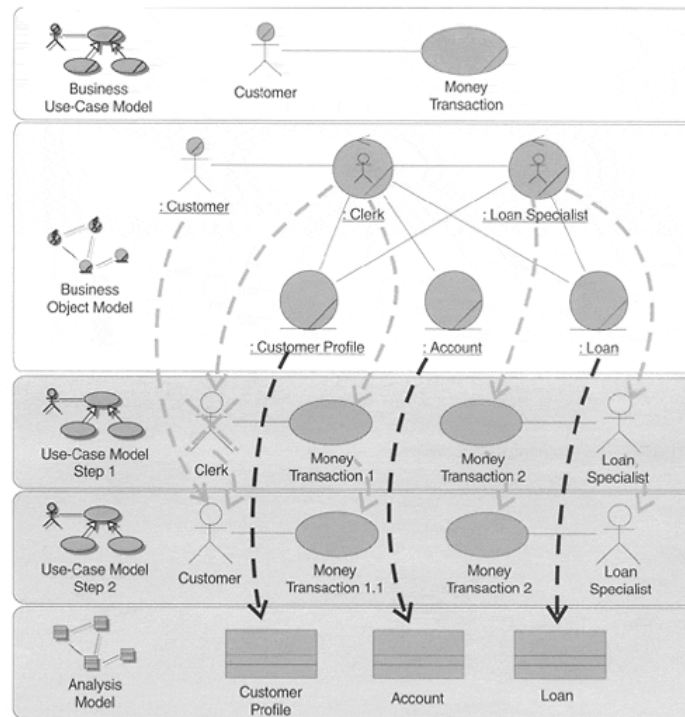


Figure 21 - Transforming BE into analysis classes

3.8. Contribution to the Research

This chapter provides an insight into the business modeling workflow of RUP. The artifacts, practitioner's roles, activities and guidelines on building the models have been mentioned. Most importantly the role of the workflow has been identified; what is the contribution of the business models to the software engineering process, what the role of the process is and how the models of the engineering process are transformed into software engineering process artifacts.

The outcome of this chapter will support us on how to combine the methodologies, how we can transform models and what are the advantages of DEMO against business modeling with UML. Most of the above will be obvious in the chapters that follow.

4. A Combination Framework

4.1. Theoretical Background

The combination of DEMO and RUP is inspired by and based on the previous work of Mingers and Blocklesby (1997). In their research paper they introduce their effort in developing a framework that will help practitioners to partition and combine different parts of methodologies, thus creating a multi-methodology that can be multi-paradigm and utilize the different parts of methodologies.

Firstly, they classify the distinct kinds of methodology combinations based on the interference of the methodologies within an intervention. Table 4 summarizes the different levels.

<i>Name</i>	<i>Description</i>	<i>Multi-paradigm?</i>
Methodological isolationism	Using only one methodology or technique from one paradigm	Single
Methodology enhancement	Enhancing a methodology with techniques from another	Single
		Multiple
Methodology selection	Selecting whole methodologies as appropriate to particular situation	
Methodology combination	Combining whole methodologies in an intervention	
Multi-methodology	Partitioning methodologies and combining parts	Single
		Multiple

Table 4 - Distinct classifications of methodology combinations

However in our case, the first level will provide us neither with a useful new methodology nor with a sound base for fruitful research; problem domains of successful usage of DEMO and RUP are distinct and known in advance; nothing new can be discussed about. The second level of methodology enhancement seems more interesting since the RUP could be enhanced with some techniques from DEMO (or contrarily DEMO could be enhanced with RUP techniques) especially in the business modeling discipline. The third level of methodology selection has gained some further research interest, like from Jackson and Keys (1984) and Jackson (1990). In these papers “a classification of problem contexts according to the nature of the systems(s) embedding the problem of concern and the relationship between relevant decision-makers” is being developed [Jackson, 1990]. Though, as already stated, the problem contexts of each methodology are quite distinct and well known thus a similar framework could be developed only in the case that we will require a configurable methodology depending on the problem context. Finally, the last two levels provide as with more interesting combination options since a combination of DEMO and RUP or a multi-methodology of them will create something new that utilizes the advantages of these two methodologies.

However, the title of this thesis may denote that we are restricting ourselves to the “Methodology Combination” level. Though, this is not the case but it was a deliberate option so not to restrict ourselves in the level of multi-methodology. We will not restrict now as well and we will leave this decision to the following steps. Here we only identify the influence of the above table to our research.

Their second step is to clarify and distinguish the terms of a methodology in order to be able to study and decompose methodologies in a more systematic way. These terms have been discussed in chapter Theoretical Background2.

The next step is to separate the world into three complementary worlds based on the work of Habermas¹. He assumes that real world situations of human activity will involve all three of the following worlds:

- *Material* world is outside of and independent of human beings. It existed before and would exist whether or not we did. Knowledge of this world is acquired through observation. Although, this world is objective (independent of the observer) the observations and descriptions of it are not.
- *Personal* world is our own individual thoughts, emotions, experiences and beliefs. This world is not observed but experienced. It is generated by and only accessible to the individual subject and thus it is subjective in contrast with the objective material world.
- *Social* world is a shared world between individuals which consists of a complex multi layering of language, meaning, social practices, rules, resources. It enables & constraints our actions but is reproduced through them. It is inter-subjective because it is a human construction as well as preexists of any particular individual.

The separation of the world can be graphically displayed in the following figure [Mingers & Brocklesby, 1997]:

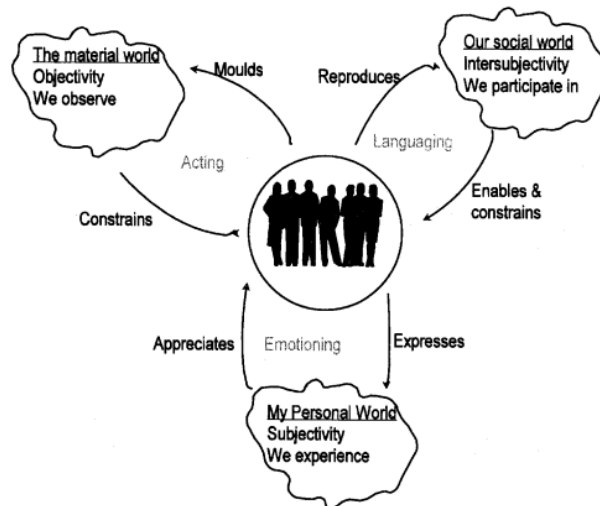


Figure 22 - The division of the world by Habermas

Also an intervention has been separated into phases as well. The authors, after making a short review of literature, separate an intervention into the following four phases. These phases should not be seen as discrete stages that follow each other but rather as aspects of an intervention that need to be considered throughout the intervention.

- *Appreciation* of the problem situation as experienced by the agents involved. In this phase the agent “gains a rich appreciation of the situation as possible”. This appreciation will always be dependent from the agent’s experience and their access to the problematic situation.
- *Analysis* of the underlying structure/constraints generating the situation as experienced. This phase tries to explain why the situation is as it appears to be,

¹ Habermas J. The Theory of Communicative Action Vols. I and II. Polity Press, Oxford, 1984.

A Combination Framework

the history that generated the current situation and the relations and constraints that maintain it.

- *Assessment* of the ways in which the situation could be other than it is; of the extent to which the constraints could be altered. The ways in which the problematic situation can be changed are examined by changing the structure and constraints of the situation.
- *Action* to bring about desirable changes.

The combination of the distinct worlds and the intervention phases creates a framework. This is a grid which has the three worlds in the rows and the four phases in the columns. It is used to “map the characteristics of different methodologies to help in linking them together” and is based on the logic that “a fully comprehensive intervention needs to be concerned with the three different worlds and the four different phases”. Thus, a methodology is examined depending on what extent addresses the aspects of every dimension of the cell. This can be done by mapping the methods (techniques) that address these aspects and using an assisting color scale depending on the extent that addresses these aspects. The framework is shown in Figure 23 and an example of the application of the framework to the Soft Systems Methodology that uses a 4-level color scale follows in Figure 24, both from Mingers & Brocklesby (1997).

	Appreciation of	Analysis of	Assessment of	Action to
Social	social practices , power relations	distortions, conflicts, interests	ways of altering existing structures	generate empowerment and enlightenment
Personal	individual beliefs, meanings, emotions	differing perceptions and personal rationality	alternative conceptualizations and constructions	generate accommodation and consensus
Material	physical circumstances	underlying causal structure	alternative physical and structural arrangements	select and implement best alternatives

Figure 23 - The multi-methodology framework

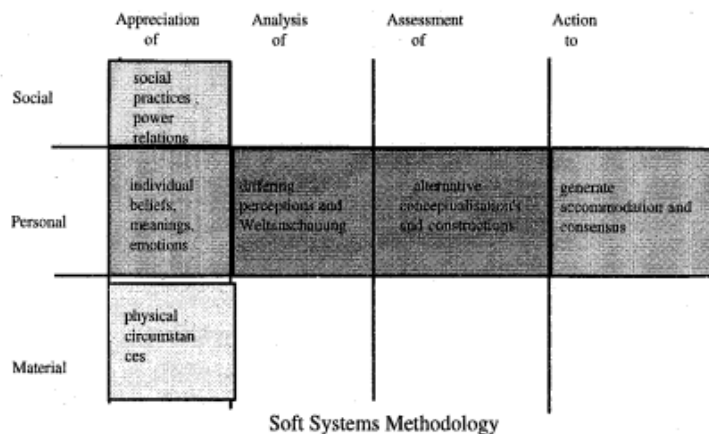


Figure 24 – An example of using the framework

Furthermore, this framework can also be used to study the methodologies and design a multi-methodology of them. A detailed study of the different methodologies has to be done in order to identify complementary techniques, detach them and link together

in a new (multi) methodology. Since multi-methodology states that phases and techniques of a methodology can be detached and used in another methodology, this should be done in a systematic way which will recognize detachable parts, their function and purpose within the intervention and how these parts can be combined together.

Thus, after the mapping of the methodologies to the grid a way to link the parts has to be devised. The framework guides and organises this linking by identifying complementary parts which address the different dimensions of every cell of the grid. Also, the framework can be used to design the new multi-methodology by attaching methodological elements (parts or techniques) in every cell and thus creating the new methodology's phases and techniques.

However, a multi-methodology has to be composed in a way that will recognize and transcend the paradigm incommensurability that may emerge when transferring one technique into a context which makes different paradigm assumptions. This requires the identification of detachable parts and if they can be transferred to another methodology with a different paradigm.

Thus, methodologies should also be studied in three levels in relation with the methodological elements mentioned earlier:

- *Philosophical principles (Why)*. Decomposed to:
 - ontological,
 - epistemological and
 - praxiological principles.
- *Methodological stages (What)*
- *Techniques (How)*

The primary element of a methodology is its stages; “a conceptual account of what needs to be done”. These are justified by the philosophical principles, and actualized by a set of activities or techniques. After studying the methodologies in this way, a detaching can be done, either at the level of technique or at the level of methodological stage. The later may be used in cases where moving a technique from one methodology to another with possibly different paradigms where the context and interpretation of the technique may change. As stated earlier, the framework can be used during the composition of the new multi-methodology by attaching stages and techniques into the cells.

Finally, the role of the agent has to be taken into account since these are the ones who will use the multi-methodology. The choice of the methodologies to combine must depend partially on the skills, knowledge, personal style and experience of the agent. Agents can not be efficient in the practice of a methodology if they are unfamiliar with it and it makes them uncomfortable; either because the paradigm conflicts with their personal beliefs, the cognitive skills needed are out of their interests or don't match with their existing skills.

4.2. Adapting the Framework

The unconditional adoption of the above framework appears to be neither possible nor acceptable. The main assumption that the writers are making implicitly is that this framework is concerned mainly with managerial or soft methodologies. This is evident in mainly two of its characteristics which can not be assumed in the case of DEMO and RUP and are explained in the following paragraphs.

We should mention here that on the one hand DEMO models a business or organization and on the other hand RUP is a software engineering process which

transforms the client's and user's requirements into software. Thus, by the combination of these two we intuitively expect to create a methodology that will go through the modeling of a business to the development of software and information systems that supports the business. Having this in mind we will examine the inappropriateness of the framework of Mingers and Blocklesby to our case and what should be done in order to adapt the framework in the specific requirements of our study.

4.2.1. The Separation of the World

The rows of the framework of Mingers and Blocklesby map to the division of the world into the social, material and personal world. The properties of the material world are that it is outside of and independent of human beings, it existed before us and will exist whether or not we did, we can only shape this world through our actions but are subject to its constraints and finally this world is described by observation and it is objective.

Apparently this world does not take into account two different things. The material world that is described obviously refers to the natural systems; chemical, physical and biological systems. Furthermore, systems like mechanical, constructional and electronic are included in this world since they can not fit to the other two.

However, this generalization is inappropriate, at least for our case. Although natural systems possess the previously mentioned characteristics, the systems developed by humans do not. A mechanical system didn't exist before humans and wouldn't exist if people had never appeared or humans' existence will be terminated in the future. These systems are indeed constrained by natural systems, however they are designed and engineered by humans, usually with one or more purposes in mind, and therefore they are not independent of the human beings and they can not always be described completely by observation.

In the theoretical background of the thesis we mentioned some efforts being done in order to divide and classify the world into a systemic way. Three different efforts have been identified: Jordan's system classification based on three dimensions, Boulding's system leveling based on system complexity and Checkland's system typology.

A candidate taxonomy for our case seems to be the framework of Checkland (1990). Checkland has some similarities with Habermas but contrarily to him he makes a clear distinction between the preexisting natural systems and the systems made by man. He goes on further and divides the man made systems into physical, abstract and human activity systems. Further on, he recognizes the relationship between human activity systems and designed systems by stating that "the fact that human activity systems form an entity is emphasized by the existence of other systems (often designed) which are associated with them" [Checkland, 1990].

Checkland also includes the transcendental systems; systems that are beyond our knowledge. In the case of a multi-methodology framework, the class of transcendental systems can be excluded. This exclusion can also be based on Checkland's conclusion: "the absolute minimum number of systems classes to describe the whole of reality is four: natural, designed physical, designed abstract and human activity systems".

If we keep in mind an intuitive example of a business which uses designed systems such as buildings, computers and machines as well as abstract systems such as organizational culture, terminology, rules and regulations, and furthermore that the designed physical systems are based on natural systems we can assume that this system classification of the world can describe the world we are focusing on.

Furthermore we focus on the human activity systems that are supported by an information system, a designed system which lays on a technological platform composed of hardware and software. The hardware, in respect with Checkland's system classes, can easily be classified as a designed physical system which uses the physical properties of its components; transistors, electrical circuits, electromagnetism and optoelectronics. The approach in reducing the complexity of the hardware is by creating layers which "hide" the complexity of the layer below. One of the layers is the operating system the purpose of which is to manage the physical devices and "provide user programs with a simpler interface to the hardware" [Tanenbaum, 2001]. The layering described briefly above can be seen in Figure 25 from [Tanenbaum, 2001].

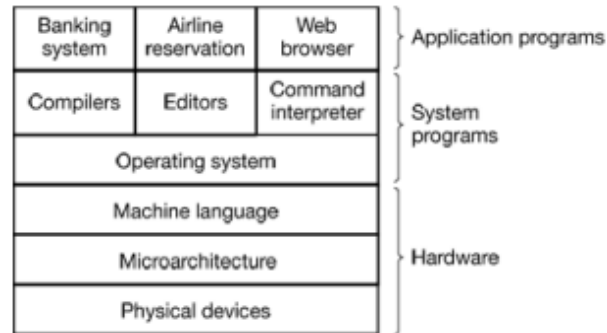


Figure 25 - The layering of computer systems

Therefore, for the purpose of the development of this framework we can claim that the natural system is "hidden" by the physical devices and consequently by the operating system which provides an "extended machine" or virtual machine [Tanenbaum, 2001], by building a designed abstract system, such as a programming language, to the application programmer that is easier to program than the underlying hardware and physical devices. Therefore, the class of natural systems is not necessary in the description of the business world we are focusing on.

4.2.2. The Intervention Phases

The framework of Mingers and Blocklesby contains the intervention phases: appreciation, analysis, assessment and action. The phase of 'action' brings about the desirable change.

However, we assume a world where designed systems² exist that are designed and engineered with a purpose in mind. The process of design and engineering appears to be complex enough not to fit in the phase of 'action' but to overlap into the others as well making difficult to delimit it clearly. Also, human activity systems are associated with designed systems; human activity systems "form an entity" because of the existence of a designed system. Often the case is that this designed system was developed having in mind the needs of the human activity system. Thus, the intervention phases should take into account this relationship.

A general intervention description that takes into account the above is the System Development Process (SDP) [Dietz, 2008, 2006]. The SDP is based on the distinction of the system function and construction (black and white box models).

There is a need of a system (Using System - US) for which the system under development (Object System - OS) is designed. More precisely the construction of the US (or its white box model) needs to use the function of the OS (or its black box

² We refer both to designed physical and designed abstract systems as designed systems.

model) since “function can not have a need for support; only construction can” [Dietz, 2008]. Thus, the function of the OS is designed to fit the construction of the US. Therefore function design of the OS starts from the construction of the US. The function of a system is determined by a “black box” model of the system. This model is the result of the function design which starts from the functional requirements provided by the construction of the US and ends with the functional specifications of the OS. This can be seen in the left hand part of the next figure.

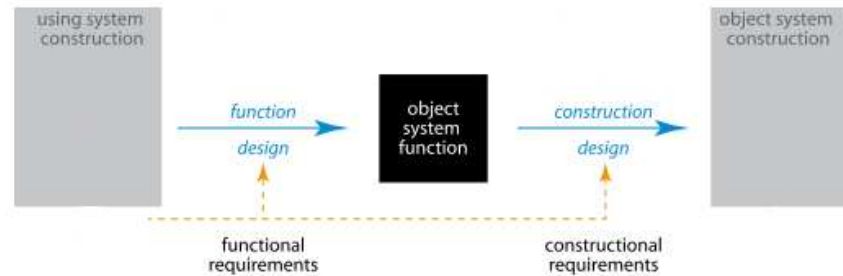


Figure 26 - The system design process

On the right side of the figure the construction design is seen as the following process of the function design. The construction design creates the construction of the OS, expressed in one or more “white box” models, based on the specified function of the OS. The main input in this phase is the constructional requirements provided by the US, also known as non-functional requirements, and regards characteristics such as performance, reliability, security etc.

However, SDP takes into account the iterative nature of the system design process. In the previous paragraph the functional requirements provided by the US were made distinct from the functional specification of the OS. This was done because “designing includes negotiation, in order that the end result is a balanced compromise between (reasonable) requirements and (feasible) specifications” [Dietz, 2008]. Thus, functional requirements are not identical to functional specification since they may be unfounded or/and not feasible.

Thus, SDP states that the design process is an alternation of synthesis and analysis steps which are performed incrementally and alternately. During an analysis step the problem is better understood by understanding the semantics, the completeness and the consistency of the function of the OS. During a synthesis step the solution becomes clearer by formalizing the semantic of the functional specifications and devising the construction of the OS.

The design phase is followed by the engineering and implementation phases. Engineering is defined as “the activity in which a series of constructional models are produced” while implementation is “the assignment of technological means to the constructional elements in the implementation model” [Dietz, 2008]. The engineering process starts from the ontological model of a system which is a model of its construction completely independent of the technology that is implemented. Then constructional models of increased detail are developed where each model is derived from the previous one until the system can be implemented. Based on the above, the reverse engineering refers to the construction of the ontological model from the implementation of a system. The following figure shows the role of ontology and technology and their relation with engineering and implementation.

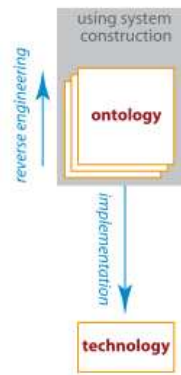


Figure 27 - Ontology, technology and implementation

By combining Figure 26 and Figure 27 we get the System Development Process (SDP) of Figure 28.

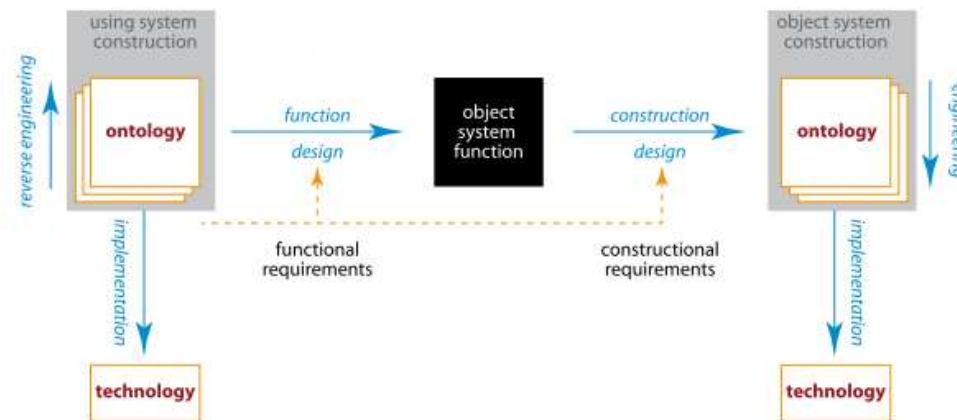


Figure 28 - The System Development Process

4.2.3. The World Division Incompatibility

So far we have identified an alternative prospective classification for the division of the world with 3 system classes and we adopted the SDP as a general theory that explains the system development process. According to the research paper of reference these two should be combined into a grid that will assist the creation of the multi-methodology.

However there is an inherent incompatibility that avoids the combination of these two. The SDP assumes the pattern of Using System – Object System in the process of system development; a system is developed because another system needs to use its functionality. The two systems can be anything, from a general class of systems to a specific system as long as there is always a system that uses some functionality and a system that provides this piece of functionality to it. In the general case systems should be layered such that the system on the upper level will use the functions of the system in the lower level.

However, the aspirant system classification of Checkland does not assume that the system classes are used in a clear way from other system classes. There is no layering assumed whatsoever. Thus, in the general case we can not assume that, for instance, a designed abstract system offers some functionality to a human activity system or to a designed physical system although there will be instances in every class that are layered. Even if Checkland states that a human activity system forms an entity because of the existence of a designed system, this is far from assuming that the

systems are layered in a US - OS pattern. Thus, it is impossible to combine Checkland's classification with the GSDP.

A systemic classification of the world, although narrowed to organizations, a human activity system in terms of Checkland, comes from DEMO itself and its organization theorem. The organization theorem states that "the organization of an enterprise is constituted as the layered integration of three systems: the B-organization (from Business), the I-organization (from Intellect) and the D-organization (from Document)" [Dietz, 2008]. The three organizations are layered; each organization supports its upper organization: D-organization supports I-organization while the latter supports the B-organization. This looks like figure 3.6.

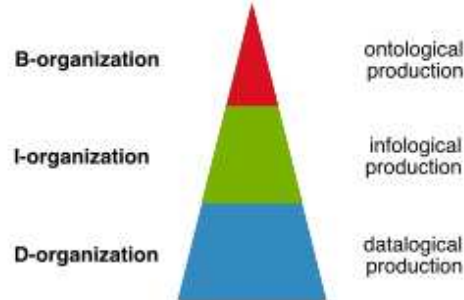


Figure 29 - The Organization Theorem

The classification is based on three distinct human abilities: forma, informa and performa. These human abilities are used during the execution of acts from the actors³ which DEMO theory divides in C- (from Coordination) and P-acts (from Production). We can separate actors based on the ability that they use during the execution of their P-acts and C-facts [Sandhyaduhita, 2009].

Concerning the coordination acts of actors, forma ability concerns the form aspects of communication and information, informa concerns the content aspects of communication and information while performa ability concerns the bringing about of new original things and is considered as the essential human ability of doing business of any kind [Dietz, 2008]. The following paragraphs are concerned with the production acts of actors.

The *D-organization* is concerned with the forma ability of the actors, who are called D-actors (Document) since they execute production acts in their forma ability. They are dealing with storage and retrieval of original or derived information items (called facts in DEMO terminology), data or documents [Sandhyaduhita, 2009].

The *I-organization* is concerned with the informa ability of the actors which are respectively called I-actors (Intellectual) since they execute production acts in their informa ability. This ability allows them to reason, compute, reproduce and remember original information and produce derived facts based on other derived or original or external facts [Sandhyaduhita, 2009].

Finally, the *B-organization* is concerned with the performa ability of the actors, the ability to establish original new things. These actors are called B-actors (Business). This ability is called ontological production act since it is the core act of doing business of any kind. B-actors are social actors and can only be undertaken by human persons.

The following figures show the layered integration of the organization theorem and how it is related with the SDP. Figure 30 from De Jong (2010) shows that every organization aspect has the function (F) and construction (C) perspective. Thus, in the

³ The notion of actor is explained in the next chapter where the framework is applied to DEMO.

layered integration of the organization of an enterprise the function of the lower organization supports the construction of the upper organization. Every organization offers a set of services to the upper level by initiating transactions that are executed using the respective human ability.

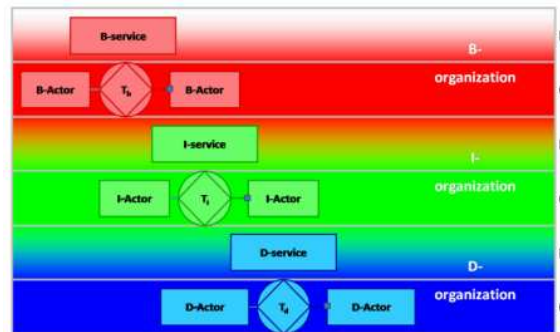


Figure 30 - The integration of an organization [De Jong, 2010]

Therefore the combination of the organization theorem with the GSDP depicts the following figure which shows how we can derive the functional requirements for the I-organization by building the constructional model of the B-organization.

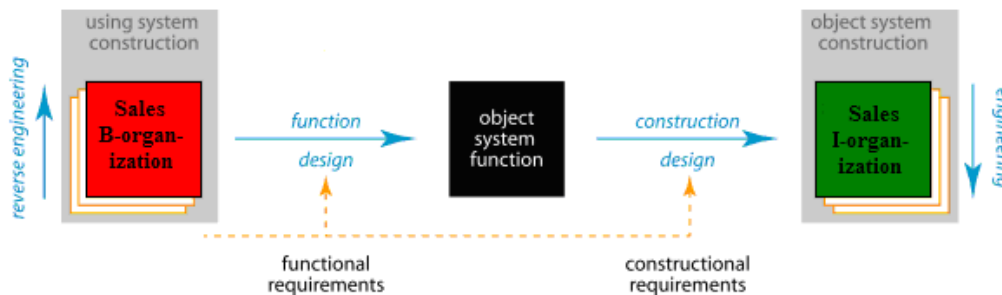


Figure 31 - Combining Organization theorem and SDP [Dietz, 2008]

In the following lines we should give a bit more attention to the I-organization in order to get a good understanding of it early enough. According to De Jong (2010) the I-services are of two types: remembering C-facts and P-facts and delivering original and derived facts which are defined both inside and outside the organization. I-services are requested and deliver facts to the B-actors.

B-actors can shape from a specific ability to any other ability. However, a B-actor can not immediately interact with an I-actor since they are not of the same category [De Jong, 2010]. Thus, a B-actor has to shape into his informa ability and then initiate an I-transaction with another I-actor as an external⁴ I-actor. Thus, “all subjects who fulfill B-actor roles and which need information fulfill external I-actor roles” [De Jong, 2010].

Thus, combining the P-acts that an I-actor can perform and the B-actor shaping mentioned we get the following figure [De Jong, 2010] that shows how B- and I-organizations are coordinating.

⁴ External in relation with the I-organization

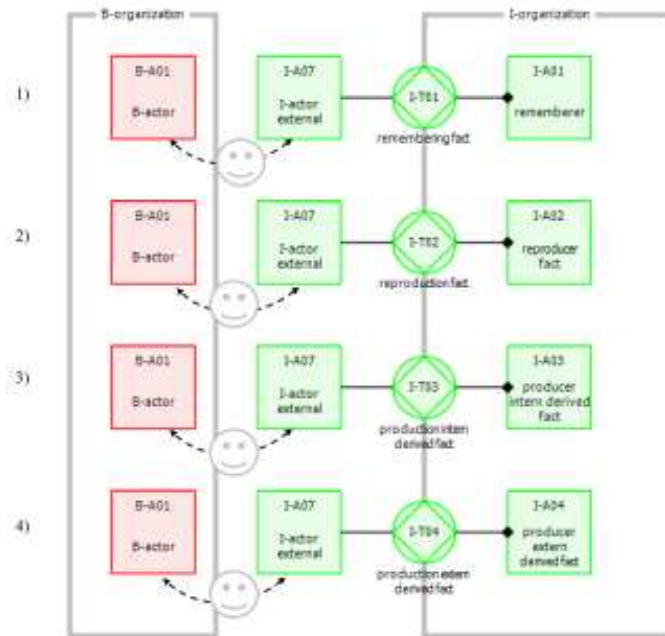


Figure 32 - The set of actor kind combinations between the B-organization and the I-organization [De Jong, 2010]

Using the same way of thinking in the levels of I-and D-organizations we can deduce a similar figure about the call of D-services by the I-actors.

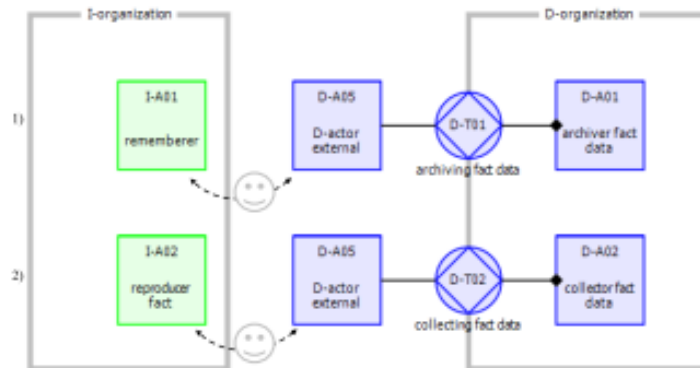


Figure 33 - The set of actor kind combinations in the D-organization [De Jong, 2010]

Thus, combining the above two figures we can visualize the complete sequence of initiating transactions and shaping ability that takes place for archiving a P-fact. This follows on the following figure from De Jong (2010) and we should note that the B-actor has shaped into his informa ability when initiating I-T01 transaction although not visible on the figure. The same holds for I-actor I-A01 who initiates D-T01 D-transaction. We should also mention that the actors of Figure 34 can be performed either by the same subject (human person) or by different subjects.

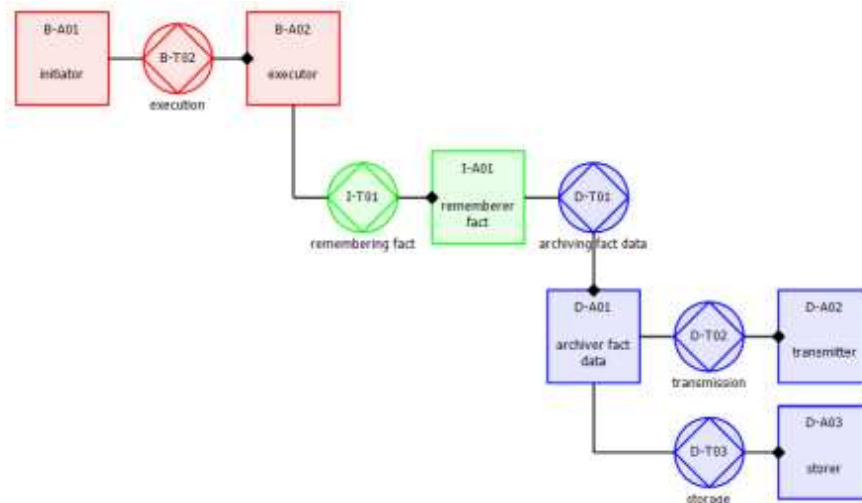


Figure 34 - Archiving P-facts [De Jong, 2010]

However, actors within the D-organization archive fact data into fact banks and collect fact data from fact banks without being conscious of the application of these facts in the I-organization [De Jong, 2010]. Similarly, I-actors remember and reproduce facts and also perform intellectual actions on facts without any understanding of the meaning of the derived facts for the B-actors [De Jong, 2010]. Based on the following we can define the scope of the framework and delineate the I-organization and D-organization which we will get involved. Before doing this we should quote the figure from [Dietz, 2006, 2008] which describes the role of ICT in relation with the organization theorem.

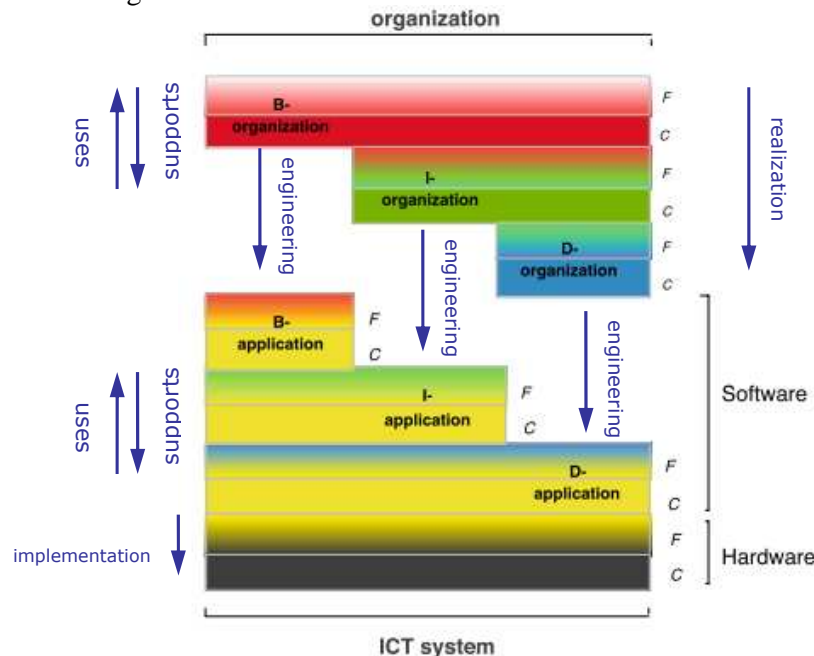


Figure 35 - The support of ICT in every aspect system

The upper part of this figure indicates the aspect organizations as defined in the organization theorem. By engineering the organizations their realization can be accomplished using ICT. ICT is presented in the lower part of the figure. ICT can support both production and coordination acts although the later may be more generic than the former.

On the lower level, the hardware exists which is used by the software in order to be executed, possibly using a layering similar to Figure 25. On the lower level of the software are the D-applications which consist mainly of operating systems, database management systems, communication protocols, programming languages and their libraries and general usage software such as text editors, document viewers, web browsers and e-mail systems. As the length of the bar of the D-applications denotes, the tasks of the D-organization can be completely taken over by the D-applications.

I-applications are built on top of the D-applications and they are information systems in the true sense [Dietz, 2008]. They provide information for monitoring the enterprise. Although they are by definition enterprise specific they include generic systems such as accounting systems, human resource applications and ERP. Similarly, the length of the bar denotes that the tasks of the I-organization can be taken over by I-applications but not completely.

Finally, the B-applications support the B-organization but they can not undertake their task in any way. “Software can not enter into and comply with commitments” [Dietz, 2008] only humans can and only humans can be held responsible for the performance of the machines. Thus, B-organization can not be substituted completely by ICT. However they can support actors or “mimic” them. Typical applications that support the production in this level are Business Process Management Systems, Decision Support Systems and ERP systems.

4.2.3.1. The Role of the ICT in the Operation of the Organization

Based on the previous section and more specifically on Figure 35 and Figure 32 we can build the case where an I-application has to be used in order to support the informability of an I-actor. This case will integrate the operation of the organization with the ICT. The picture that visualizes this scenario is Figure 36 on page 69.

We see in this diagram that the B-actor who wants to use an I-application has to shape into his informability and initiate a transaction with the “I-application user” actor of the I-organization. Concerning the subjects that undertake these two actor roles may be different humans or the same person. What we want to indicate here is that a B-actor always has to shape into his informability in order to use an I-application. Further on, when the “I-T01” transaction is initiated the “I-A02” actor role will be transformed in an “I-application user” system actor, who is displayed using the UML notation. The functions that the I-application user can call are related with Figure 32, the I-actors of this figure have been replaced by an I-application function (‘remember original fact’, ‘reproduce original fact’, ‘produce internal derived fact’, ‘produce external derived fact’). The use case specifications describe the steps that the I-actor has to go through in order to execute a function.

The actors are included into the B- and I-organizations respectively; therefore these actors have been modeled with DEMO as social systems as dictated by Ψ theory. However, I- and D-applications are rational systems, in our case ICT systems. Thus, RUP may be used in order to model both function and construction of the I-application, using the respective models and diagrams (see section 5.3 on page 78).

However, RUP builds information system based on an existing programming and implementation platform. Bloch (2007) defines *programming platform* as a programming language, its libraries and APIs (Application Programming Interface). Taking into account Figure 25 that shows the layers of a computer we can extend into the term of *implementation platform* which includes the layer of operating system, specialized software such as databases, network protocols and non-general purpose hardware such as barcode readers. We chose to separate the programming platform

from the implementation platform because modern languages, like Java and PHP, have eliminated the dependency of a programming language from the implementation platform. However there is always a need for a compiler (or interpreter), an operating system and hardware in order to run an application.

Furthermore, *the complexity of the implementation platform is “hidden” behind the programming platform*. For example, although the storage and retrieval of a file in a hard disk drive is indeed a very complex procedure the I-application programmer only has to call a function with few arguments of the programming language he is using. More examples are the networking protocol stack which reduces the complexity of sending messages by providing layers which offer services to the above layers and the SQL that provides a higher level language for manipulating data stored in a database. Thus, the programming and implementation platform of the ICT constitute the D-application, with the programming platform being the functional perspective and the implementation platform being the constructional perspective of the D-organization.

However, the D-application of Figure 36 is *only a subsystem of the D-applications of an organization*. This is only the subsystem that supports the I-application (see Figure 35). General use applications such as e-mail systems, text editors and document viewers are considered D-applications as well [Dietz, 2008].

Therefore there can be the case that a B-actor wants to call a function of a D-application, like send an e-mail, view or edit text. Doing so he has already shaped in his informa ability and he tries to accomplish an I-transaction by requesting a D-transaction. It will be more likely that this D-transaction is supported by a D-application. Thus, extending Figure 36 with an I-actor shaping in his forma ability we get Figure 37 which shows the role of ICT in the operation of the D-organization. We have modeled a scenario similar to Figure 34 where an I-actor wants to remember a fact, possibly in a text document. Therefore he has to change to his forma ability in order to utter information, store it and transmit it.

The programming and implementation platforms are missing in Figure 37. This of course does not mean that they are not required by the D-applications in order to run. It rather means that D-applications are more of a Graphical User Interface of the construction of the D-applications, a replacement for accessing the functions of the D-applications. They provide a more user friendly way to use the functions of a D-application such as file browser of an operating system or an e-mail client for using network protocol such as SMTP.

Also, the role of the programming and implementation platform in Figure 36 is to support the datalogical needs of an I-application and its user. Contrarily, a D-application supports the forma ability of a D-application user. Therefore, the user perspective separates them, not the implementation of the applications. Anyway, the implementation platform is general for these kinds of applications; no specialized software or hardware is required. Thus, there is no need for modeling them since their description may be fixed and known beforehand. However, from the application development point of view the ability that they support is unimportant; applications are not developed having in mind the human ability that they will support.

We could as well make one more diagram that shows the usage of a B-application by a B-actor. However, Dietz (2008) states that B-applications are simply I-applications that have been attributed the separate status of B-application in order to distinguish them from the pure I-applications that support the I-organization. So, a diagram like this won't provide something newer than Figure 36.

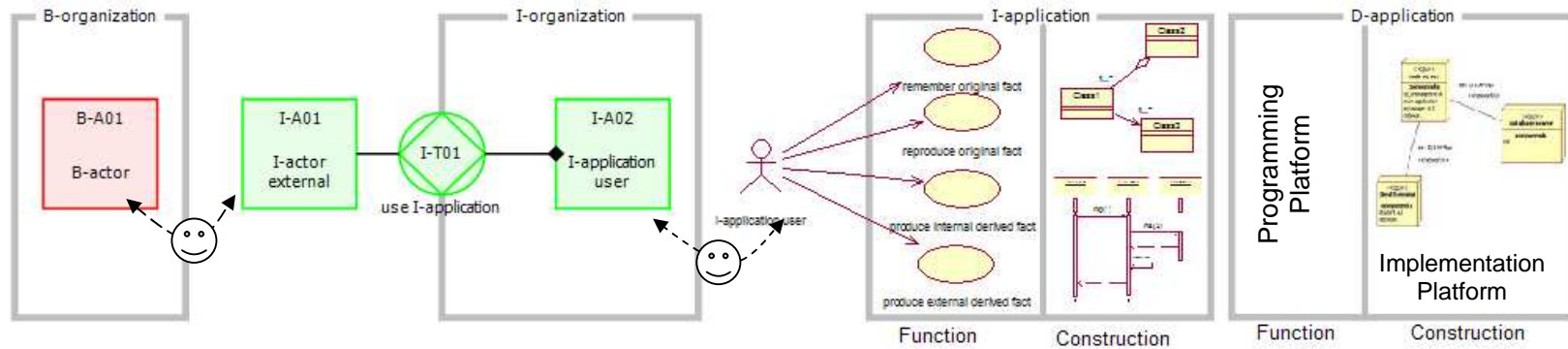


Figure 36 - The role of ICT in the operation of the I-organization

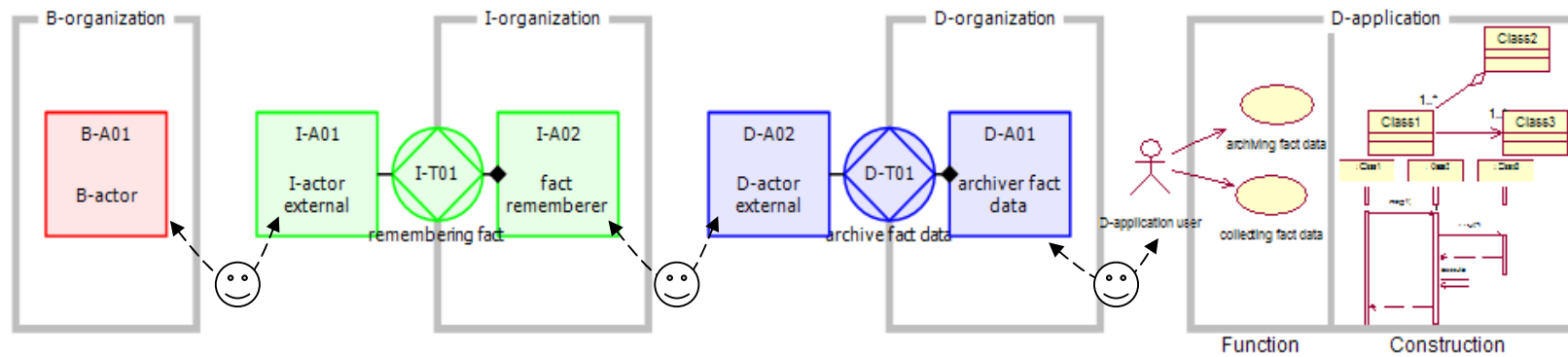


Figure 37 - The role of ICT in the operation of the D-organization

4.2.4. The Adapted Combination Framework

The adapted framework we create has the same structure and the same logic as the framework of Mingers & Brocklesby. It is a grid where the rows are the B, I and D organization as described in the organization theorem of DEMO and the columns are the phases as described in the SDP. Similar to the original framework it will be used to map the stages and the characteristics of different methodologies to help in linking them together. The grid is shown in Table 5.

	Function Design	Construction Design	Engineering	Implementation
B-Organization				
I-Organization				
D-Organization				

Table 5 - Adaptation of the combination framework

The above framework will be used to decompose the methodologies in a way that will identify in which aspect system and in which phase of the development process every methodological element applies. Thus, the composition of the new methodology will be concerned with as many cells of the framework as possible and will avoid the duplicate work by the practitioner.

In every cell we will try to map the elements of a methodology, such as methodological stage, technique, model or diagram, which has as an objective to model, accommodate or document the outcome of every phase of the SDP.

However, we should make an assumption clear at this point. The subsystem of the I-organization that we are focusing is that subsystem that can be substituted by I-applications. Consequently, the D-organization that we are focusing on is the subsystem of the D-organization which is substituted by the D-applications that support the I-applications. These subsystems are of more interest for the RUP practitioner. Thus, *we will improperly use the term I-organization and D-organization to refer to them*. These subsystems are delimited with black solid line in Figure 38.

Furthermore, we also consider B-applications as I-applications. We make the distinction in order to deal with the development of both categories in a uniform manner. However, we will still refer to B- and I-applications in order to be compatible with the theory. The same assumption holds for the D-applications that do not support the I-applications. These are also conceived as applications by the RUP practitioner, distinction is made by the DEMO practitioner.

To make it more clear we have drawn the line that delimits the scope of the framework using Figure 35 on page 66. The solid black line delimits the I- and D-applications we are focusing on. The black dashed line delimits the B- and D-applications since we deal with them with the same way as I-applications. However, in the context of RUP we may refer to I- and D-applications improperly with the term I- and D-organizations as well.

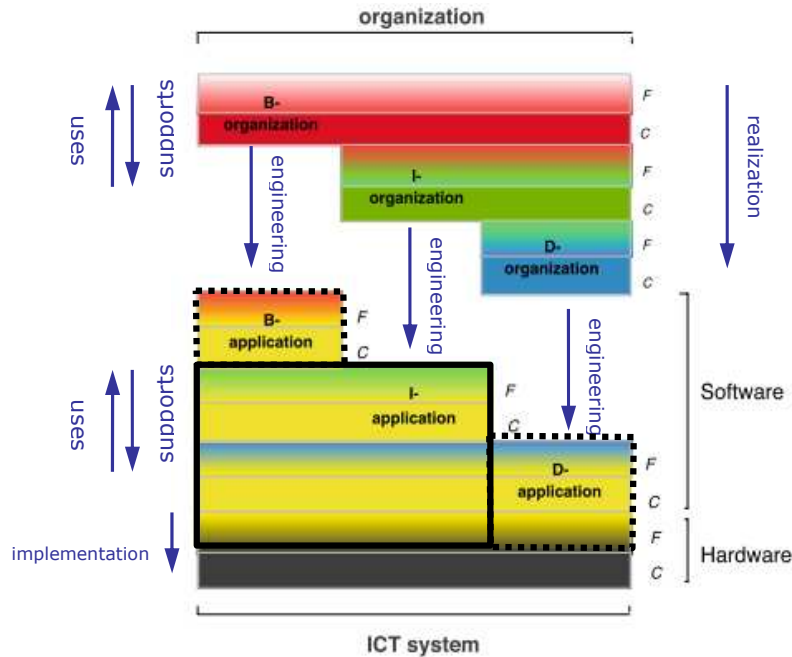


Figure 38 - The ICT systems that are of interest for the RUP practitioner

4.2.5. The Methodological Framework

The methodological framework used to decompose methodologies by Mingers & Brocklesby has a major consequence in the combination effort. The modelling language is not taken into account. Instead they take into account the ontology of the methodology as a dimension of the paradigm. Possibly, based on Guarino (1998), there will be a relationship between modelling language and ontology. But whether this relationship is straightforward or not and if this is bidirectional, namely ontology is affected by the modelling language and on the same time ontology determines the modelling language is a question that is outside the scope of this thesis.

Therefore, we choose another framework to decompose methodologies, the one introduced by Seligmann et al. (1989) and Wijers (1991) which can be seen in Figure 3. This framework takes into account the modelling language explicitly and therefore there is no need to investigate the above mentioned relationship.

Taking the modelling language of a methodology into account means that the relationship between the modelling languages of DEMO and RUP should be examined; their models and their roles in the process, the compatibility between models, a way to combine them, how to transform models of one methodology to models of the other or substitute and/or exchange models.

Furthermore, by changing the methodological framework the three levels of study of a methodology of Mingers & Brocklesby (philosophical principles, stages and techniques) are changing. Therefore, the levels that will help recognize detachable elements of methodologies should be:

1. Way of thinking
2. Way of working
3. Way of modelling
4. Way of controlling
5. Way of supporting

We see that we do not include the “way of communicating” in the decomposition. This is done because we prefer to stick to the official modelling language of every methodology. Anyway, DEMO does not give the freedom to the practitioner to change the semantics of the modelling language. Though, UML has some concepts without explicit semantics. For example, the stereotyped relationships, where the semantics of the stereotype (with few exceptions like “extends” and “includes” stereotypes of use case diagram) are determined by the practitioner, are labelled relationships with a tailored semantics for every project. It also provides a mechanism for extending the language (See Appendix C – UML profiles). However, we will choose not to interfere with the “way of communicating” since it is dependent from the project needs and team and not on the modeling language itself.

Furthermore, there may be some relationships between the two methodological frameworks. In Figure 3 of Seligmann’s framework we see that the ‘way of modelling’ and the ‘way of working’ dimensions constitute the operation of the methodology. Therefore we could assume that these dimensions are relative with the stages and techniques of the first methodological framework of Mingers & Brocklesby. Indeed, the techniques of I.S. development methodologies are mainly concerned with the construction of models, as RUP explicitly states [Kruchten, 2000]. They provide detailed guidelines and modelling languages with clear syntax and grammar rules which can be intuitively classified at the technique level. Also, by definition the ‘way of working’ describes the activities and can be related with the methodological stages. As we mentioned earlier that possibly there is a relationship between ‘ontology’ and ‘way of modelling’ dimensions and that the “way of thinking” is in the philosophical level. However, these analogies should be of no concern; since a new methodological framework is adopted in the combination framework the previous methodological framework does not affect it.

4.3. *Linking the Methodologies Together*

The above framework will be used in order to study the methodologies and help in linking them together. The steps that will be followed in order to derive a new methodology are:

- i. Study the methodologies
- ii. Map each one to the combination framework
- iii. Decompose methodologies based on the methodological framework
- iv. Combine methodologies based on step ii
- v. Identify “way of thinking” incommensurability of the combined methodology
- vi. Combine the “ways of working”
- vii. Combine the “ways of controlling”
- viii. Combine the “ways of modeling”
- ix. Combine the “ways of supporting”
- x. Apply to the case study

5. Applying the Combination Framework

5.1. Introduction

In this document we fit elements of DEMO and RUP in the framework presented in the previous chapter. First we choose as a methodological element the models and we complete the grid in such a way that every model of each methodology is placed in the right cell. We could as well attach methodological stages or techniques but DEMO provides detailed information mostly on the models that should be created and their supporting diagrams. Furthermore, RUP emphasizes in the development and maintenance of models as well [Rational, 2001] with the usage of UML. Though, its main focus is on the description of the methodological stages and their activities. However, the level of detail for the “way of modeling” has the same granularity in both methodologies, with easily identifiable elements and well defined diagrams thus providing uniform comparison when fitting methodologies into the grid.

Then we decompose the methodologies according to the adopted methodological framework of Seligmann et al. (1989). This will provide us with an insight on the “ways” that compose the methodologies. Thus during the combination phase we will be able to identify philosophical incompatibilities that can affect the new methodology, define a practical way of working, refine the way of controlling and connect the way of modelling by devising model transformations.

5.2. DEMO Grid

	Function Design	Construction Design	Engineering	Implementation
B-Organization		<ul style="list-style-type: none"> - Construction Model - Process Model - State Model - Action Model 		
I-Organization				
D-Organization				

Table 6 - Applying the combination framework to DEMO

5.2.1. Way of thinking

DEMO adopts the ontological system notion of Bunge, presented in chapter Introduction, and excludes the teleological. It also adopts the terms that Bunge uses to define a system, i.e. composition, environment, structure etc. DEMO describes only the construction of a system and excludes the function.

The way of thinking of DEMO is expressed in the Ψ theory [Dietz, 2006]; its four axioms and one theorem. These are [Dietz, 2006]:

- *Operation axiom*: “the operation of an enterprise is constituted by the activities of actor roles, which are elementary chunks of authority and responsibility, fulfilled by subjects”. The actors perform two kinds of acts, production and coordination acts (P-acts, C-acts), which have as a result production and coordination facts (P-facts, C-facts). The acts have effect into two distinct worlds, the production and the coordination world (P-world, C-world). The graphical representation of the operation axiom is shown in Figure 39 [Dietz, 2006]:

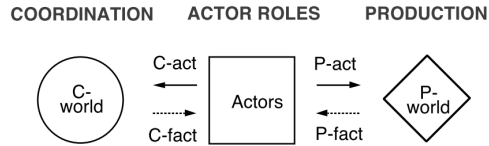


Figure 39 - Graphical representation of the operation axiom

- *Transaction axiom*: “C-acts are performed as steps in universal patterns” called transactions. They always involve two actors, the initiator and the executor, and produce a particular result. A transaction evolves in three phases: the order phase (O-phase), the execution phase (E-phase), and the result phase (R-phase) and they follow a basic patten of steps [Dietz, 2006].

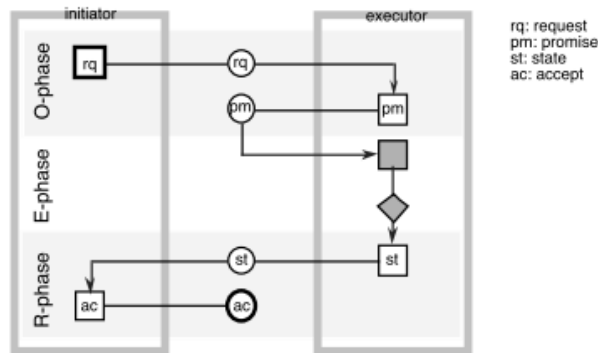


Figure 40 - The basic pattern of a transaction

- *Composition axiom*: “every transaction is enclosed in some other transaction, or is a customer transaction of the organization under consideration, or is a self-activation transaction”. Based on this axiom a business process can be defined as “a collection of causally related transaction type”.
- *Distinction axiom*: “there are three distinct human abilities playing a role in the operation of actors, called performa, informa, and forma”. The organization theorem is based on this axiom.
- *Organization theorem*¹: “The organization of an enterprise is a heterogeneous system that is constituted as the layered integration of three homogeneous systems: the B-organization, the I-organization, and the D-organization”.

5.2.2. Way of modeling

DEMO includes aspect models “in which the ontological knowledge of an enterprise is expressed, such that this knowledge is easily accessible and manageable” [Dietz, 2006]. The aspect models can be examined in relation with the representation of the organization theorem. Models belong to the B-organization and are expressed with diagrams.

The *Construction Model* (CM) visualizes information about the interaction structure of the organization [Dietz, 2006] by including the transaction types and the actor

¹ More information for the organization theorem can be found in section 4.2.3 on page 62.

roles. It provides “the most compact ontological model of the B-organization” [Dietz, 2006] and focuses on the coordination of the actors.

The CM specifies the identified transactions and the associated actor roles that participate in a transaction as initiator and executor, as well as the information links between the actor roles and the information banks. CM is divided into two models: the Interaction Model (IAM) shows the active influences between actor roles and the Interstriction Model (ISM) shows the passive influences between actor roles.

The *Process Model* (PM) expands the transaction pattern to the transactions of the CM by showing the sequence of steps and the causal relationships between the transactions. It provides “the set of lawful or possible or allowed sequences of states in the C-world” [Dietz, 2006].

By definition the *Action Model* (AM) is “the most detailed and comprehensive aspect model” [Dietz, 2006]. Therefore, its positioning in the grid is quite straightforward since it provides the most detailed knowledge about the construction of an organization. The AM specifies the action rules that serve as guidelines for the actors in dealing with their agenda in a form of pseudo-code.

The State Model (SM), in accordance with the PM, is “the specification of the state space of the P-world” [Dietz, 2006]. It is developed based on the concepts of object classes and fact types, the result types, and the ontological coexistence rules.

The following picture shows the ontological aspect models in relation with the diagrams that express these models [Dietz, 2006]. The red triangle represents the positioning of the models in the organization theorem of Figure 29.

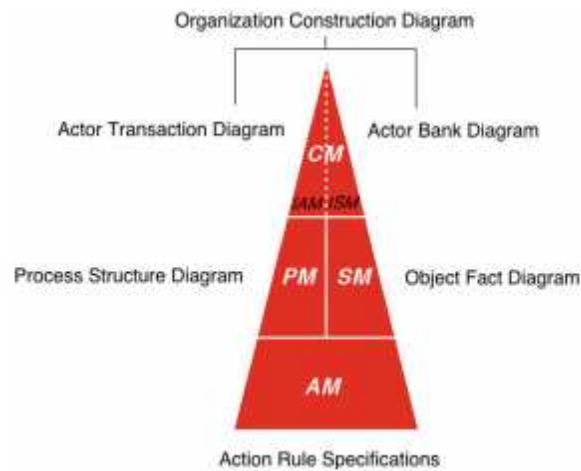


Figure 41 - The ontological aspect models and their diagrams

5.2.3. Way of working

DEMO does not describe the “way of working” in distinct and clearly visible phases in a manner similar to RUP. However, the development of every aspect model can be considered a distinct task of the process since it includes a clear objective with a clear output (developing a model), techniques on how to build the models and guidance on the sequence of model development (see “way of controlling”).

The techniques that DEMO uses for the development of models are the following six [Dietz, 2006]. These consist of three analysis and three synthesis steps. Briefly, these are:

- *The Perfoma-Informa-Forma Analysis.* Based on the distinction axiom the available knowledge, usually in a form of a narrative description, is divided into performa, informa and forma items. The coloring of the items, i.e. by using markers or changing font color, can help the practitioner during this step.
- *The Coordination-Actors-Production Analysis.* Based on the operation axiom the perfoma items identified in the previous step are divided into C-acts, P-acts and actor roles. Using the colored description of the previous steps the practitioner can mark performa items with an indicative shape or mark. Dietz (2006) suggests drawing a box, disk or diamond above every item or in the case of electronic text enclosing every item into [], () or < > in order to denote an actor, a C-act or a P-act respectively.
- *The Transaction Pattern Synthesis.* During this step the transaction pattern is used in order to identify transactions by clustering the C-act and P-act items identified so far. The Transaction Result Table is also completed during this step. Since the description may be incomplete a transaction can be identified when a P-act or a C-act has been found. Only if there is no hint at all about one of the steps of a transaction the practitioner can miss to model it [Dietz, 2006].
- *The Result Structure Analysis.* The composition axiom applies in this step. The end result of a transaction with an initiator in the environment of the system is decomposed in its components based on the transactions initiated by the executor and so on. The components can be identified by finding phrases which express dependencies between production acts or results. With this technique the causal and conditional relationships between transactions can be identified and modeled in the Process Model.
- *The Construction Synthesis.* This is the first step for the development of the Actor Transaction Diagram of the Constructional Model. It includes the identification of the initiator and executor of every transaction.
- *The Organization Synthesis.* This is a minor step that identifies the scope of the organization by selecting what parts of the construction are parts of the kernel of the organization or of its environment and what are the interfaces between them.

Finally, although not implicitly stated in the methodology book, DEMO models can be developed iteratively; i.e. the development of the PM is completed only after the SM has been developed. The example is derived by the role of the IUT cross model table and its positioning in Figure 42. More explanation follows in the next section. In DEMO-3, the process is designed to be recursive. Each concept is created in all the models before the next concept is produced [Sattari, 2010].

5.2.4. Way of controlling

DEMO does not provide a clear method or technique for controlling the execution of the methodology. Though, it provides guidance on the order of developing the models and the role of cross-model tables in the process. This is showed in the following figure [Dietz, 2006].

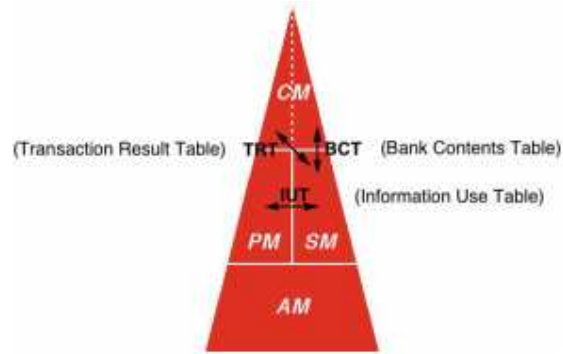


Figure 42 - The cross-model tables

The logical sequence of developing the aspect models is anti-clockwise, starting from the IAM². Therefore:

1. The first result of the method is a list of identified transaction types and the participating actor roles.
2. Based on them, the boundary of the enterprise can be identified. Based on this information the IAM³ of the CM can be made straightforwardly.
3. Further on the PSD of the PM is produced.
4. Based on this the ARS of the AM is developed.
5. Next, the SM is developed by making the OFD and the Object Property List (OPL).
6. When the SM is complete the Information Use Table cross-model table can be developed which will complete the PM.
7. Finally, the ISM of the CM is produced, which consists of an ABD and the BCT cross-model table, derived from the SM. Usually, the ABD and the ATD can be combined in the Organization Construction Diagram (OCD).

Thus, using the above guidelines, Figure 41 and Figure 42 the practitioner can go through the development of diagrams since he is aware of the relationships between the models.

From a project management perspective, the dependencies of the phases of the project can be implied from the above paragraph, thus they can be used by the project manager in order to plan the project by developing, i.e. a Gant chart or an activity network diagram.

5.2.5. Way of supporting

DEMO can be executed without the support of specialized tools. However, in practice the maintenance of diagrammatic models without the use of a software package is very difficult.

One solution for supporting DEMO is a Microsoft Visio shape stencil which is a template for displaying the graphical notation of DEMO to Microsoft Visio modeler. However, it lacks basic features like the complete DEMO models, support of the formal syntax of DEMO and the syntactical checking of diagrams.

Xemod⁴ is a software package which is developed by Mprise (former Xprise) and is a dedicated modeler for DEMO which supports the formal syntax, consistency checking of the models and use of a central repository. Also, it accommodates the

² The sequence of developing the aspect models can also be considered in the “way of working”

³ See Figure 41 and Figure 42 for the explanation of the acronyms

⁴ <http://www.xprise.com/>

communication between teams since it can create reports of the DEMO models in an interoperable way.

The way of controlling of DEMO does not require a specialized software tool. Thus, a general project management tool can be used such as Microsoft Project or Primavera Project Planner.

5.3. RUP Grid

	Function Design	Construction Design	Engineering	Implementation
B-Organization⁵	<ul style="list-style-type: none"> - Business Use Case Model - BUC Specification 	<ul style="list-style-type: none"> - Business Analysis Model - Domain Model - Business Rules 	<ul style="list-style-type: none"> - Business Analysis Model - Business Goals - Business Rules 	<ul style="list-style-type: none"> - BUC Realization
I-Organization	<ul style="list-style-type: none"> - Use Case Model - Use Case Specification 	<ul style="list-style-type: none"> - Analysis Model - Design Model 	<ul style="list-style-type: none"> - Analysis Model - Design Model - Navigation Map - GUI Prototype - Test Results 	<ul style="list-style-type: none"> - Use Case Realization - Navigation Map - Implementation Model - Design Model - Source Code
D-Organization		<ul style="list-style-type: none"> - Deployment Model (Descriptor Level) - Data model 	<ul style="list-style-type: none"> - Test Results 	<ul style="list-style-type: none"> - Deployment Model (Descriptor & Instance Level) - Programming & Implementation Platform

Table 7 - Applying the combination framework to RUP

Use case models provide a model of the environment and the functions of the system [Rational, 2003] by identifying actors and use cases of the system and associating them in use case diagrams. RUP defines use case models for the business system (business use cases) as well as the information system (system use cases).

A use case defines “a coherent piece of behavior without revealing the internal structure of the system” [Rumbaugh et al., 2005]. This piece of behavior contains a flow of events; the steps that should be followed during a use case from an outside perspective [Booch et al., 1998]. Furthermore, it contains the main flow as well as all the alternative flows that may arise due to exceptions. These workflows are included to the use case specification, usually in a textual and diagrammatic format, using an interaction diagram or an activity diagram [Booch et al., 1998]. Lastly, use case realization describes the implementation of a use case by describing how it is implemented [Rational, 2003] in terms of collaboration between elements of the construction model. The use case provides a “black box view” of the system, the flow of events provides a “gray box view” of the system and the use case realization provides a “white box view” of the system [Kruchten, 2005].

Both business use case models and system use case models serve the same purpose and have the same parts and semantics. The difference is the modeling system; system use cases are modeling the information system while business use cases are modeling the organization.

Concerning the rest cells of the B-organization, the business goals and their hierarchy guide the engineering process. Goals are requirements that must be satisfied by the business, thus the engineering process should be oriented towards their satisfaction. Also, the engineering process should take into account the business rules which are the declaration of the business policy [Kruchten, 2004].

⁵ See Business Modeling in RUP chapter for more detail in this mapping

However, business rules are expressed in several ways, thus they can be classified into model based and document based [Rational, 2003]. Model based business rules are captured in the models using stereotyped constraints of UML in natural language or the standard Object Constraint Language (OCL). The document based business rules are captured in a separate document either because the rules are distributed across different models or the number of rules that apply is large [Rational, 2003]. Thus, business rules, and more specifically model rules, can describe the construction of the organization as well.

The analysis model is an abstraction or a generalization of the design model, a rough sketch of the system which is refined further in design [Kruchten, 2004]. Therefore it provides a model that accommodates the engineering process by evolving itself in a more detailed model of the implementation. The role of the analysis model is to distinguish clearly the problem domain from the solution domain [Arlow, 2005]. They hold key attributes of the objects and very high level responsibilities only.

Analysis models can also give an overview of the elements that exist in the real world [Kruchten, 2004] by restricting to classes that are part of the vocabulary of the domain of the problem and map in a clear and unambiguous way to a real-world concept [Arlow, 2005]. In the B-organization the business analysis model describes the construction of the business using elements such as business workers and business entities [Rational, 2003] while in the I-organization these elements are classes that model things in the problem domain [Oestereich, 2001]. In the general case, the analysis classes are derived from business entities and evolve into design classes which act as the abstraction of the source code and the implementation model [Rational, 2003]. Thus, business analysis model and (IS) analysis model, despite the accommodation of the engineering process, also serve as an abstract constructional model of the respective systems.

Also, during the engineering process the navigation map and the graphical user interface (GUI) prototype are used as well [Kruchten, 2004]. The first one is a model of the structure and the navigation pathways of the GUI while the later is an early implementation of the GUI that does not include functionality, it is “a mile wide and an inch deep” [Kroll et al., 2003] which among others is used for involving the client in the process and thus gain support, understand requirements better and test the usability of the product [Kruchten, 2004]. Finally, Test Results although not a model, fine tune engineering by reporting mistakes during the process. The whole workflow of “Test” could also be included in the cell but this wouldn’t be comparable with the content of the rest of the cells.

The implementation model describes the physical composition of the implementation [Rational, 2003] in terms of components and interfaces. Thus it provides a way to model the implementation of the I-organization.

Finally, the construction of the D-organization can be modeled using the Deployment Model and the Data Model. The former, in the descriptor level, displays information about the physical deployment of devices, the processing nodes on these devices and the communication links between them [Rumbaugh et al., 2005]. This model, in the Instance level, can also reveal information about the implementation of the D-organization (application), such as special hardware devices, communication protocols or databases used.

The Data Model describes the logical and physical representations of persistent data used by the application [Rational, 2003] and therefore defines the static elements of the D-organization; how these elements are related and what their properties are.

The *Test Results* have the same effect as in the I-organization; they provide feedback in the engineering process and thus guiding towards a more robust implementation. Finally, the implementation of the D-organization contains, besides the Deployment Model, the Programming & Implementation Platform, the specific hardware devices (PC or Solaris) and software (communication protocols, database vendors, operating system, programming language, libraries) upon which the software will be developed.

5.3.1. Way of thinking

RUP is a *software development approach* that is iterative, architecture-centric, and use-case-driven [Kroll, 2003].

This approach can be briefly described in the following principles [Kroll, 2003]:

1. Attack major risks early and continuously or they will attack you.
2. Ensure that you deliver value to your customer.
3. Stay focused on executable software.
4. Accommodate change early in the project.
5. Baseline an executable architecture early on.
6. Build your system with components.
7. Work together as one team.
8. Make quality a way of life, not an afterthought

RUP approaches software development by deploying 6 best practises which are “commercially proven approaches to software development that, when used in combination, strike at the root causes of software development problems” [Kruchten, 2000]. These are [Rational, 2001]:

1. *Develop software iteratively*. The process supports an iterative approach in the development of software which faces and mitigates high risks in the first phases of the software lifecycle by creating prototypes of the system, involving the users [Rational, 2001].
2. *Manage requirements*. RUP practitioner should expect that the requirements are dynamic and will change during the software lifecycle. Thus, the process describes how to elicit, organize and document the functional and non-functional requirements of the system [Rational, 2001].
3. *Use component-based architectures*. Components are non-trivial modules, subsystems that fulfill a clear function [Rational, 2001]. They enable reuse of software, facilitate resilient architecture and separate clearly the elements of the system that are likely to change [Kruchten, 2000].
4. *Visually model software*. Visual models provide a better understanding of the system to be modeled by capturing the structure, behavior and architecture of the system. The complexity of the details of the system is hidden in the models and communication between team members is facilitated [Kruchten, 2000].
5. *Continuously verify software quality*. Software quality is reviewed and tested in correspondence with the requirements and is based on reliability, functionality, application and system performance. Quality testing is encompassed in the process and involves all of its elements (role, activities, artifacts), using the measurements and criteria [Rational, 2001].
6. *Control changes to software*. As the system evolves and is developed by a team, coordination, communication and change propagation should be managed in order to make team work as a unit and iterate successfully. RUP contains guidelines on how to control, track and monitor changes to enable successful iterative development [Rational, 2001].

The above best practices have implication on the way of working, way of modelling and way of controlling of the process. For example due to (4), RUP uses UML in order to model the system while 1, 2, 5 and 6 provide the basic way of working; working in iteration of evolving system prototype and requirements elicitation which will guide the next iterations into the completion of the final product. Also, 5) has an immediate result on the way of working by adding the “Test” discipline on the “way of working”.

However, contrary to DEMO, RUP does not explicitly states the way of thinking. We see above no assumptions about the basic constructional unit of the system or no explicit separation between function and construction (or teleological and ontological system notions). Thus, the above six practices do not illustrate the complete way of thinking. We should also consider the implications of the way of modelling, and especially the language of UML, to the way of thinking of the methodology. A description of the way of modelling of RUP exists in the following paragraph so the reader should refer there for further explanation.

5.3.2. Way of modeling

RUP activities focus mainly on the creation and maintenance of models since most of the artifacts are models [Rational, 2001]. RUP defines model as “a semantically rich representation of the software system under development” [Rational, 2001]. Models are expressed with the combination of several diagrams. The advantages that modelling adds to a software development process have been identified to [Rational, 2003]: aiding understanding of complex systems, exploring and comparing design alternatives at a low cost, forming a foundation for implementation, capturing requirements precisely and communicating decisions unambiguously.

The design activities are centred on the notion of (software) architecture [Rational, 2001]. RUP defines architecture⁶ in [Rational, 2003] as “the architecture of a software system (at a given point) is the organization or structure of the system's significant components interacting through interfaces, with components composed of successively smaller components and interfaces”. According to [Kruchten, 2000] the architecture of the software encompasses significant decisions about the following:

- The organization of a software system
- The selection of structural elements and their interfaces by which the system is composed, together with their behaviour as specified in the collaboration among those elements
- The composition of these elements into progressively larger subsystems

The architectural style that guides this organization, these elements and their interfaces, their collaborations, and their composition

The architecture of a software system is represented in multiple (architectural) views. Each view captures certain aspects of the system under development and it uses UML diagrams to do so. RUP models focus on a specific view.

The model of architecture that RUP suggests is based on the 4+1 architecture model of Kruchten (1995) and is divided into four views plus one more redundant which on the one hand “acts as a driver to help designers discover architectural elements during the architecture design” and on the other hand “validates and illustrates the architecture design”. As Kruchten (1995) also suggests “not all software architectures

⁶ In these paragraphs architecture has the meaning of software architecture

need every view in the 4+1 View Model. Views that are useless can be omitted.” This makes the process more configurable and thus suitable for various types of projects. The 4+1 architectural view can be schematically displayed in the following diagram which exists in various sources like [Kroll, 2003], [Kruchten, 1999, 2005], [Rational, 2003].

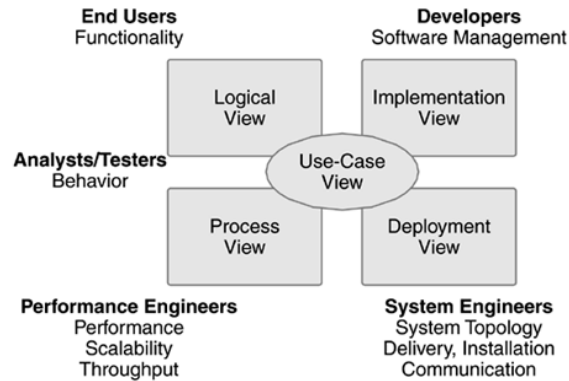


Figure 43 - Software architecture model of RUP

- The logical view (existing for any system), shows the various elements of the software and their structure: classes, packages, and so on.
- The process view shows the parallelism between various entities, and how communication and synchronization are achieved.
- The implementation view shows how the implementation elements (source code files, executable, and so on) are organized in the development environment.
- The deployment view, showing how physically at runtime the various runtime components are replicated and deployed, and how they communicate.
- The use-case view captures the most significant requirements: use cases or parts of use cases that have some great impact on the architecture, as well as significant non-functional requirements. This may include use-case realizations to illustrate how the system works.

RUP uses Unified Modelling Language (UML) as the language to create the models of the architecture of the software. The UML is a general purpose, visual modelling language that is used for visualizing, specifying, constructing and documenting the artifacts of a software-intensive system [Booch et al., 1998] and it is used to understand, design, browse, configure, maintain and control information about such systems [Rumbaugh et al., 2005]. UML is intended for a variety of software processes and methods. RUP is one of the methods that use UML.

UML captures information for the static structure and dynamic behavior of a system. The structural area defines the objects and the static relationships among them that are important to the system [Rumbaugh et al., 2005]. The dynamic behavior describes the history of objects over time and the communication among objects to accomplish goals [Rumbaugh et al., 2005]. Finally, the physical layout describes the computational resources and the deployment of components (mainly source code) on them while the model management describes the organization of the models themselves [Rumbaugh et al., 2005].

This gives UML the advantage to model a system from several viewpoints which when related with each other can give a complete understanding of the modeled

system. The relation of all the above aspects and terms of UML can be seen in the following table from [Rumbaugh et al., 2005].

<i>Major Area</i>	<i>View</i>	<i>Diagrams</i>	<i>Main Concepts</i>
structural	static view	class diagram	association, class, dependency, generalization, interface, realization
	design view	internal structure	connector, interface, part, port, provided interface, role, required interface
		collaboration diagram	connector, collaboration, collaboration use, role
		component diagram	component, dependency, port, provided interface, realization, required interface, subsystem
	use case view	use case diagram	actor, association, extend, include, use case, use case generalization
implementation view	component diagram	component, interface, dependency, realization	
dynamic	state machine view	state machine diagram	completion transition, do activity, effect, event, region, state, event, transition, action
	activity view	activity diagram	action, activity, control flow, control node, data flow, exception, expansion region, fork, join, object node, pin
	interaction view	sequence diagram	occurrence specification, execution specification, interaction, interaction fragment, interaction operand, lifeline, message, signal
		collaboration diagram	collaboration, guard condition, message, role, sequence number
physical	deployment view	deployment diagram	artifact, dependency, manifestation, node
model management	model management view	package diagram	import, model, package
	profile	package diagram	constraint, profile, stereotype, tagged value

Table 8 - UML views and diagrams relationship [Rumbaugh et al., 2005]

Another important classification of the UML diagrams is done in [OMG, 2009] where the UML diagrams are classified into two major categories: structure diagrams and behavior diagrams.

Structure diagrams describe the construction of the system while behavior diagrams describe the behavior of the system. On the one hand use case and activity diagram provide a functional model of the system. On the other hand interaction diagrams display how the different elements of the system communicate with each other in order to realize the use cases.

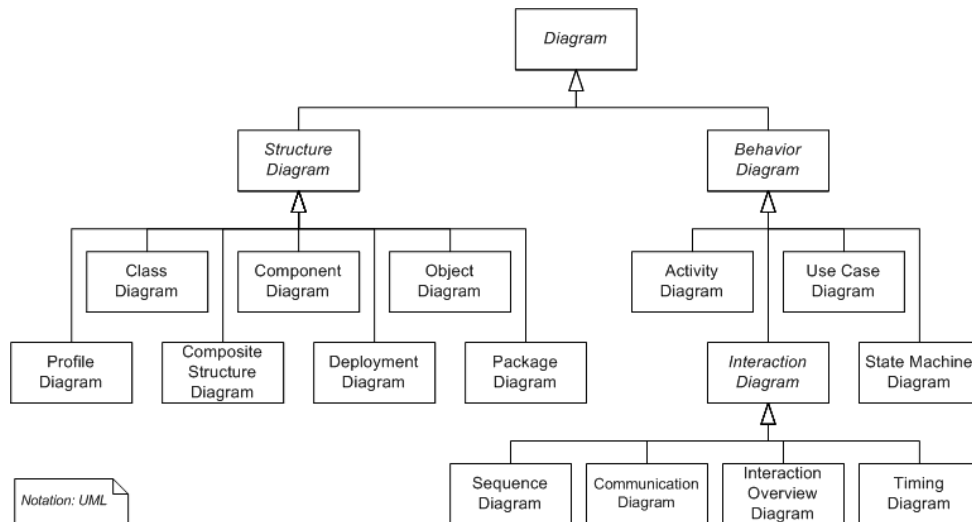


Figure 44 - UML diagrams categorization [OMG, 2009]

We see that the use case diagram is a special diagram. On Rumbaugh’s table it is classified as a structural diagram while on OMG’s classification as a behavior diagram. Although there seems to be a contradiction this is not the case. The functionality of the system, expressed in a use case diagram, apparently defines the behavior of the system. However, this behavior is not dynamic. It does not change over time since no new functions can be added in an existing system and they do not change dynamically over time. Thus it can not be classified as dynamic diagram on Rumbaugh’s table. Thus, we will accept both classifications, each one complementary of the other and not contradictory.

5.3.3. Way of working

RUP is well-defined and well-structured; it provides a disciplined approach to assigning tasks and responsibilities within a development organization. The structure of RUP is described in the following diagram [Rational, 2003].

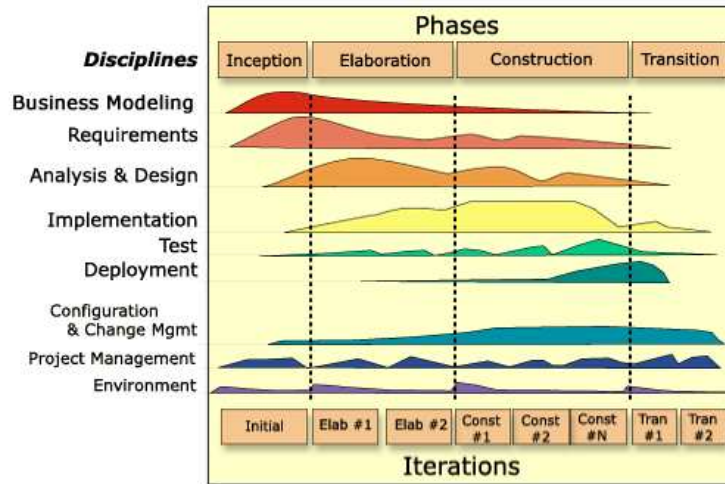


Figure 45 - 2dimensional structure of RUP

The horizontal axis of this diagram represents time, the dynamic structure of the process [Kroll, 2003]. It shows the lifecycle aspects of the process as it evolves, organized into iterations and phases [Kruchten, 2000]. The lower part represents the

iterations that divide the software lifecycle into cycles; each cycle is working on a new generation of the software.

The upper part divides each cycle into four consecutive phases, each phase is concluded with a well-defined milestone — “a point in time at which certain critical decisions must be made and therefore key goals must have been achieved” [Rational, 2001]. These phases repeat themselves in every cycle and they are [Kroll, 2003]:

- *Inception*. Establish a good understanding of what system to build by getting a high-level understanding of all the requirements and establishing the scope of the system
- *Elaboration*. Design, implement, test, and baseline an executable architecture, including subsystems, their interfaces, key components, and architectural mechanisms
- *Construction*. All remaining components and application features are developed and integrated into the product, and all features are thoroughly tested. The construction phase is a manufacturing process.
- *Transition*. Transit the software product to the user community. Test the product in preparation for release and make minor adjustments based on user feedback.

The vertical axis represents process workflows, the static structure of the process. It is a grouping of activities according to their nature [Kruchten, 2004]. The workflows are divided into core and supporting, with the bottom three being the supporting ones. A short description of every workflow follows from Kruchten (2004) while the description of the supporting workflows follows on the description of the “way of controlling” of RUP.

Business Modelling: RUP provides a common language and process for business modelling and software engineering, and guidelines on how to create and maintain direct traceability between business and software models. Many projects choose not to do business modeling [Rational, 2001]. More about this workflow can be found in Chapter 3.

Requirements: The requirements derived from this workflow describe what the system should do and facilitates the agreement between customers and developers. More about this workflow can be found in Chapter 3, paragraph 3.7.1.

Analysis & Design: The goal of this workflow is to show how the system will be realized [Rational, 2001]. In this workflow the requirements are translated into specifications that describe how to implement the system. The main artifacts are a design model which is an object model describing the realization of use cases, and serves as an abstraction of the implementation model and its source code [Rational, 2003] and a use-case realization which describes how a particular use case is realized within the design model, in terms of collaborating objects [Rational, 2003]. Also, an analysis model may be created which is a description of the realization of use cases, and which serves as an abstraction of the Design model [Rational, 2003].

Implementation: During this workflow the actual implementation of the design classes in a programming language is done. The purpose of this workflow is [Kruchten, 2004], [Rational, 2001]:

- To define the organization of the code in terms of implementation subsystems organized in layers
- To implement classes and objects in terms of components (source files, binaries, executables, and others)

- To test the developed components as units
- To integrate into an executable system the results produced by individual implementers or teams

The main artifact of this workflow is the implemented system.

Test: In an iterative process, testing is done throughout the project. This is the main advantages of an iterative process since you find defects and error as early as possible and thus the cost of fixing is reduced. RUP carries out tests along three dimensions: reliability, functionality, application and system performance. Also, it provides strategies and tools for automating testing [Rational, 2001]. The purpose of this workflow is [Kruchten, 2004]:

- Verifying the interactions of components
- Verifying the proper integration of components
- Verifying that all requirements have been implemented correctly
- Identifying and ensuring that all discovered defects are addressed before the software is deployed

Deployment: In this workflow, the product releases are produced and the software is delivered to the end users [Rational, 2001]. Depending on the project size this workflow can be trivial or extremely complex. The purpose of this workflow is [Kruchten, 2004]:

- Testing the software in its final operational environment (beta test)
- Packaging the software for delivery
- Distributing the software
- Installing the software
- Training the end users and the sales force (product roll-out)
- Migrating existing software or converting databases

If we combine the above framework with the iterative nature of RUP then we get a process that is a sequence of incremental steps. Every iteration includes all or some of the above workflows, depending on the project needs and development team. It also builds on the work of the previous iteration and has clear objectives; among them to produce a partial working implementation of the final product and refine the system until the final product is complete [Kroll, 2003].

The iterative development of RUP can be briefly described in the following picture [Rational, 2003].

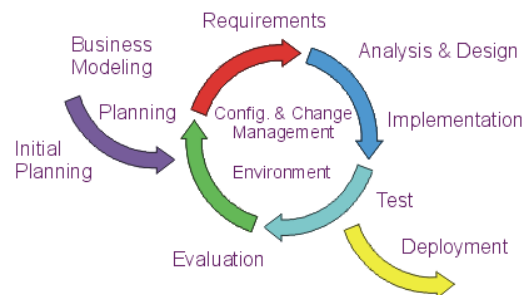


Figure 46 - An iterative process

Among the advantages of an iterative approach, as they have been identified in [Kroll, 2003] and [Kruchten, 2000] are:

- It accommodates changing requirements and makes change more manageable.
- Integration is not one "big bang" at the end of a project

- Risks are usually discovered and mitigated earlier; during early integrations.
- Management has a means of making tactical changes to the product
- Reuse is facilitated and has increased opportunities
- Defects can be found and corrected over several iterations
- The final product has a better overall quality
- It is a better use of project personnel
- Team members learn along the way
- The development process itself is improved and refined along the way

Every workflow describes *who* is doing *what*, *how*, and *when*. The RUP workflow details are represented using four primary modelling elements [Kruchten, 2000], [Rational, 2001]:

- Workers (Roles): The who. It defines the behaviour and responsibilities of an individual, or a group of individuals working together as a team.
- Activities: The how. A unit of work that a role may be asked to perform. It has a clear purpose and is assigned to a specific worker.
- Artifacts: The what. It is a piece of information that is produced, modified, or used by a process.
- Workflows: The when. It is a sequence of activities that produces a result of observable value. The most two common workflows are Workflows (or Disciplines in order to reduce confusion), which are the high level workflows, and Workflow Details, which are the workflows within a discipline.

The following picture represents the first three basic elements [Kroll, 2003] while Figure 4 shows an example of a workflow description detail of RUP.

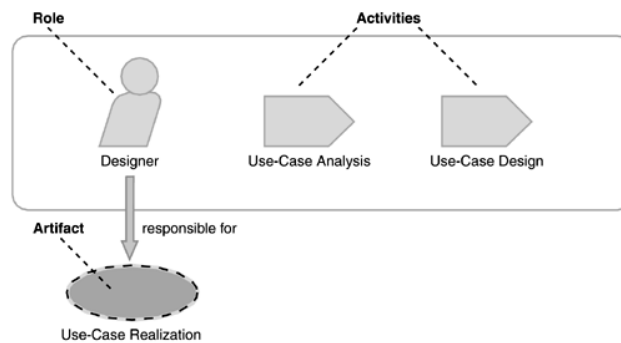


Figure 47 - Roles, activities and artifacts

5.3.4. Way of controlling

RUP provides a detailed way to control the process by dedicating three supporting disciplines for the control of the process (Figure 45). These are the last three of the 2 dimensional structure of the process: *Configuration and Change Management*, *Project Management and Environment*.

Configuration & Change Management: It describes how to control the numerous artifacts produced by the many people who work on a common project [Rational, 2001] so as to ensure that costly confusions are avoided and that the resultant artifacts are not in conflict. The purpose is to track and maintain the integrity of evolving

project assets [Kruchten, 2004]. The general actions that this workflow involves are [Rational, 2003]:

- identifying configuration items,
- restricting changes to those items,
- auditing changes made to those items
- defining and managing configurations of those items.

Project Management: This workflow tries to balance competing objectives, manage risk, and overcome constraints to deliver successfully the product [Rational, 2001]. The purpose of it is to [Kruchten, 2004]:

- To provide a framework for managing software-intensive projects
- To provide practical guidelines for planning, staffing, executing, and monitoring projects
- To provide a framework for managing risk

However not all of the aspects of project management are considered within this discipline. The management of people (hiring, training and coaching), of budget and of contracts are not covered. It only focuses on the specific project management aspects of the iterative process such as planning the whole lifecycle of the project and specific iterations, manage risks and monitor the progress of the project and its iterations.

Environment: This workflow provides the software development organization with the software development environment that is needed to support the development team and thus focuses on the activities to configure the process in the context of a project [Rational, 2001]. The main purpose is to provide a process configuration together with the appropriate tool to support it. Among these, this discipline results in [Kruchten, 2004]:

- Tool selection and acquisition
- Set up tools and configuration of tools to suit the organization
- Process configuration
- Process improvement
- Technical services to support the process: the information technology (IT) infrastructure, account administration, backup, and so on

5.3.5. Way of supporting

RUP is also a process product maintained by its development team in order to continuously improve it. The philosophy of RUP for this aspect of the process is that “software processes are software too” [Kruchten, 2004] and thus they have to be delivered to the practitioner not as a binder in a shelf or a book but as a software product. The characteristics of approaching RUP as a software product are [Kruchten, 2004]:

- Rational Software releases regular upgrades.
- It is delivered online using Web technology, so it is literally at the fingertips of the developers.
- It can be tailored and configured to suit the specific needs of a development organization.
- It is integrated with many of the software development tools in the Rational suite so that developers can access process guidance within the tool they are using.

Applying the Combination Framework

The above characteristics offer advantages in RUP which have been identified in [Kruchten, 2000]. These are:

- The process is never obsolete; companies get new releases at regular intervals, with improvements and up-to-date techniques.
- All project members can access the latest version of the process on an intranet.
- Java applets, such as a process browser and a built-in search engine, allow developers to reach instantaneously process guidance or policies, including the latest document templates they should use.
- Hyperlinks provide navigation from one part of the process to another, eventually branching out to a software development tool or to an external reference or guideline document.
- Local, project or company-specific process improvements or special procedures are included easily.
- Each project or department can manage its own version or variant of the process.

The RUP product framework can be viewed in the following figure [Rational, 2003] where it is divided in four integrated parts [Kroll, 2003].



Figure 48 - RUP Product framework

- *Best Practises*. This is the core of the cube and of the RUP as well and consists of the best practises and the approach adopted by RUP which are expressed in phases, workflows, roles, activities and artifacts. All this aspects were discussed previously.
- *Process Delivery Tools*. RUP is delivered to their practitioners using a web based knowledge base providing all team members with guidelines, templates and tool mentors. The knowledge base consists of extensive guidelines, tool mentors, templates and examples of the artifacts. This knowledge base has been used as a reference in this document.
- *Process Configuration Tools*. RUP is a configurable process which can be tailored from the practitioners in order to fit their project's specific needs. RUP product offers tools (RUP Builder) to accommodate and publish the tailored process [Rational, 2003].
- *Process Authoring Tools*. RUP allows the practitioners to capture and add their own best practices in their configured process. Rational Process Workbench is a tool to facilitate this and consists of RUP Organizer for managing content libraries, RUP Modeller for defining process models that

- extend the basic RUP process definition, and the RUP process engineering process. [Rational, 2003]
- *Community/Marketplace*. The Rational Developer Network (RDN) is an online community that allows users to exchange experiences and provide access to the latest updates of RUP in aspects like artifacts, articles and templates. [Kroll, 2003]

The RUP product consists of tools to support all activities in a system's lifecycle, to support the process and to develop, maintain and document the various artifacts that are created, especially models and UML diagrams [Rational, 2001]. Also, there are tools to configure and deploy the process, manage requirements, automate documentation editing, testing automation and team collaboration. Tool mentors provide step by step guidelines for implementing the RUP activities using the tools. There are also templates and examples for the artifacts that these tools develop. Some of the most useful tools are [Rational, 2003]:

- *The Rational Unified Process* is a flexible software development process platform. Through its configurable architecture, RUP enables practitioners to select and deploy only the process components that are required by the project's specific needs.
- *RUP Builder* enables project managers to select, configure, create custom views of and publish RUP-based processes for their projects.
- *Rational ClearCase* product family provides a configuration management solution.
- *Rational PurifyPlus* is an advanced debugging and diagnostic tool that pinpoints hard-to-find, run-time errors in your application; provides advanced application performance profiling; pinpoints areas of code that have not been exercised during runtime execution.
- *Rational Rose* is a graphical component modelling and development tool that uses the visual modelling language UML.
- *Rational SoDA* provides automatic generation of software documentation. SoDA templates support Microsoft Word 97, 2000, and XP.
- *Rational ClearQuest* is a defect tracking and change request management system.
- *Rational Test Manager* creates, maintains and executes automated functional tests.
- *Rational Suite Analyst Studio* facilitates the collection, management, and modelling of enhancement requests, requirements, and use cases in a comprehensive and integrated solution
- *Rational RequisitePro* helps teams organize, prioritize, track, and control changing requirements of a system or application.

6. Combining DEMO with RUP

6.1. The combined frameworks

In the previous chapter the two methodologies were decomposed based on a framework that divides organizations systemically into three aspect systems and recognizes four phases in system development. Based on this framework we combine the two methodologies in a manner that will utilize the complementary parts of every methodology so all the aspect systems of an organization are taken into account during all the phases of a system development process.

Thus, the following table (Table 9) makes a first intuitive combination of RUP and DEMO based on the idea that the complementary cells should be combined. However, this combination is intuitive; it does not take into account problems that may emerge due to the differences in the “ways” of each methodology.

	Function Design	Construction Design	Engineering	Implementation
B-Organization	<ul style="list-style-type: none"> - <u>Business Use Case Model</u> - <u>BUC Specification</u> 	<ul style="list-style-type: none"> - Construction Model - Process Model - State Model - Action Model 	<ul style="list-style-type: none"> - <u>Business Analysis Model</u> - <u>Business Goals</u> - <u>Business Rules</u> 	<ul style="list-style-type: none"> - <u>BUC Realization</u>
I-Organization	<ul style="list-style-type: none"> - Use Case Model - Use Case Specification 	<ul style="list-style-type: none"> - Analysis Model - Design Model 	<ul style="list-style-type: none"> - Analysis Model - Design Model - Navigation Map - GUI Prototype - Test Results 	<ul style="list-style-type: none"> - Use Case Realization - Navigation Map - Implementation Model - Design Model - Source Code
D-Organization		<ul style="list-style-type: none"> - Deployment Model (Descriptor Level) - Data model 	<ul style="list-style-type: none"> - Test Results 	<ul style="list-style-type: none"> - Deployment Model (Descriptor & Instance Level) - Product (H/W & S/W platform)

Table 9 - A combination of DEMO with RUP

Thus, the above table may not be feasible and it may change after considering the “ways” of each methodology. In the following paragraphs we will examine the possibilities of combining DEMO and RUP in the levels of the way of working, modeling and controlling. However, we should first recognize if the “way of thinking” of the methodologies is incommensurable something that can prevent their combination. Furthermore, we should remind to the reader that by saying I-organization and D-organization in the above cell we also mean I-application and D-application respectively. We only focus on the subsystems of I- and D- organizations that can be substituted by I- and D-applications.

6.2. Way of thinking

The way of thinking of the two methodologies requires an extensive study since an incommensurability appears, inherent from the different ontological commitment that is used by the methodologies to describe systemically the world they are modeling. By ontological commitment we mean the “vocabulary” that each methodology is using, the “repertoire of types of objects to the existence of which it is committed” [Smith, 2003]. Thus, in order for our effort to be successful we have to bridge the gap between the two ways of thinking carefully.

Both DEMO and RUP try to describe a system mainly with the usage of diagrams. Thus, by recalling chapter 1 and the first definition of system, this description should start from the elements (things) of the system and the relations between the elements. Furthermore, according to Bunge these relations should be active, namely elements should affect each other because of the existence of these relations.

DEMO assumes as the fundamental constructional element of the system the elementary actor role (Operation axiom). The interactions (or relationships) between the actors (either internal or external to the system) follow a specific pattern (transaction theorem). Actors perform C- and P-acts which have effect in the C-world and the P-world. These are systemic classifications of the B-organization, each referring to the relations and the elements of the system respectively. The need of such a classification is to describe the state of C- and P-acts which both are complex enough since every C-act can follow a pattern of 4 to 8 steps while the actors are assumed to be social elements, undertaken by humans, and thus the complexity of their production acts has to be modeled in more detail.

On the other hand RUP, as justified by UML, uses objects as the fundamental constructional element, which possesses attributes and provides functions to other objects. The combination of the value of the properties of an object denotes its internal state which changes over the lifecycle of the object based on events and/or the execution of functions [Rumbaugh, 2005]. The interaction between the objects is done with the exchange of messages, by calling a function and receiving a value, and by the encapsulation of objects; describing the attributes of an object in terms of other objects.

Figure 44 categorizes the diagrams of UML in two main categories, structure and behavior diagrams which visualize different perspectives of the system¹. This classification, although not directly related and semantically different, serves the same purpose as the C- and P-worlds of DEMO. Using the terminology of Bunge's ontological system notion [Dietz 2006], structure diagrams visualize the composition and some aspects of the structure of the system, mainly the simple and static influence bonds that do not change over time. Behavior diagrams visualize the structure (dynamic relations that change over time) and the production of the system; what is delivered to the environment and how is produced by the elements.

Furthermore, a classification of diagrams and their role in system modeling, similar to DEMO (Figure 41), is done with the 4+1 views of software architecture (Figure 43). Indeed, this architecture focuses on the description of a software system thus the views have meaning only when describing a software system. The two basic views that exist in every description are the logic view and the use case view [Kruchten, 1995, 2004]. The former describes the kernel of the system (in Bunge's system terms) since it does not include the interaction of the structure with the environment of the system. The later is captured in the use case view and provides the functions that the system provides to actors in the environment. It is the black box model of the system, developed with a use case diagram and its elements (actor, use case, flow of events).

Finally, the "Business Modeling" technique uses a UML profile (See Appendix C – UML profiles) to describe the function and construction of the B-organization in terms of business use cases for the functional model and business workers and entities

¹ For more detail see the relative paragraph

for the construction. Also, the implementation of the business is modeled in the “BUC Realization” artifacts.

We see from the above paragraphs that there is a fundamental incommensurability in the “ways of thinking” of the two methodologies. The fundamental elements and relationships used in the description of a system *are inherently different and can not be related or combined*. The elementary actors of DEMO can hardly be described in terms of UML objects or classes while it is impossible to transform transactions into UML relations. The transaction pattern can not be related to a concept of RUP or UML².

Also, the “Business Modeling” technique, although it defines business workers, a term similar to (internal) actor of DEMO, it *does not use them as the fundamental constructional unit*. It merely uses it for identifying the users and some functions of the information system by assigning them responsibilities over the entities and business processes. Also, business workers *are not per se social individuals*; they can be any kind of system, i.e. mechanical or software. Furthermore, *RUP technique is concerned with the implementation* of the business which is contradicted with the ontological notion that DEMO adopts and indeed constitutes one of its advantages.

However, this incommensurability can be avoided by *isolating methodologies in every aspect system*.

Thus, B-organization³ should only be described using either DEMO or RUP but not both. Therefore, the B-organization’s row on Table 9 should contain only the DEMO models of the “Construction Design” column. We have underlined the rest of the cells of this row for this reason. Therefore, the emerged methodology would belong to the “Multi-Methodology” level of Table 4 since RUP is partitioned and partially combined with DEMO.

Also, we have already made an assumption concerning the “way of thinking”. As was also discussed in paragraphs 4.2.3.1 on page 67 and 4.2.4 on page 70 in our case the terms I- and D-organizations are used interchangeably with the terms I- and D-application. Therefore in the RUP decomposition framework (Table 7) and the combined framework (Table 9) the green and blue rows can as well be named I- and D-applications respectively without contradicting Ψ theory.

Practically, this means that we are assuming that I- and D- organizations are not social systems per se; *they may be rational systems substituting the social system*. This assumption is made in order to avoid the impossible mission of defining a social system in terms of objects, message exchanging by function calling and object encapsulation. Thus, with respect to Figure 30 a feasible way of thinking for the combination of DEMO and RUP is that *DEMO describes the B-organization as a social system while RUP should describe the I- and D-organizations in terms of their supporting I- and D-applications perspectives*. However, by taking this assumption we deliberately and unavoidably include implementation details in their models, detracting one of the advantages of DEMO. Also, another side effect of including implementation details in the description of the x-organizations⁴ is that the limits between x-organization and x-application are blurring. However, B-applications can not substitute the B-organization, they can *only mimic elementary actors* [Dietz,

² Although we will see later that the transaction steps can be modeled with an interaction diagram this modeling element is not atomic.

³ DEMO and RUP do not interfere in the I- and D-organizations. See Table 9.

⁴ x=I or x=D

2006] thus the B-organization still remains a social system and RUP is used only for modeling the B-application.

6.3. Way of working

It is feasible to combine the “ways of working” efficiently, in a practical way. The option here is to *adapt the way of working of RUP* since it is a configurable process [Rational, 2001] and thus easier to adapt to the introduction of DEMO as the business modeling methodology.

As we see from Table 9, DEMO substitutes the artifacts of the “Business Modeling” discipline. Thus this discipline *should be removed during the RUP execution*. The fact that the “Business Modeling” discipline is not mandatory in every project of RUP makes it easier; practitioners will be familiar with executing a process instance without executing the “Business Modeling” workflow explicitly. The discipline of the “Business Modeling” is substituted by applying DEMO methodology.

Furthermore, since the “Requirements” discipline follows, *DEMO models can be used in order to elicit requirements*. Actually, RUP uses some transformation guidelines that use the “Business Modeling” artifacts as an input and creates the use case model of the software. Further research on this subject is done on the thesis of Puspa I. Sandhyaduhita (2009). How to derive functional requirements will be described in the paragraph of the way of modeling since RUP captures functional requirement in use case diagrams and it is actually a matter of transforming models.

Also, RUP is explicitly iterative but DEMO is not. However, *iterating during the development of DEMO models is possible* although not explicitly stated as explained in the “way of working” and way of controlling” analysis of DEMO. Thus, Figure 46 can be transformed to the Figure 49 by introducing the DEMO methodology for the business modeling of the organization.



Figure 49 - Introducing DEMO in the RUP iterative nature

However, if we take a deeper look into the iterative software lifecycle we will see that one of the advantages it offers is that it reduces the risk of developing the wrong software functions and reduces “the continuing stream of requirements changes” [Boehm, 1988]. These two advantages are achieved by going through a rough analysis of the problem domain to the design of a partial solution and the development of a prototype of the system that will help to better understand user needs and predict change propagation into the business. Thus, the total requirements can be elicited because the impact of the system to the business system can be predicted. Therefore, the business model of RUP can be altered during an iterative process by introducing the usage of the system under development by the business workers, something that affects the implementation of the business processes (business use case realization).

Also, the introduction of a software system may affect the other business models of RUP as well.

However, DEMO models are completely independent from the implementation of the system, providing only the essence of a business [Dietz, 2006, 2008]. Thus, a sound DEMO model does not have to be dramatically altered, if not at all, during an iterative process since the implementation details, i.e. if the business is supported by a software system or human, are abstracted from the models.

Practically, the practitioner of the combined methodology will have to develop the DEMO models based on the theory of DEMO during the inception phase. The models that should be developed in the first iteration are depended on the modeling guidelines. Later on he will have to develop the use case model of the I-organization. Also, the analysis and design of the I-organization and I-application should be done, as explained in “Analysis & Design” discipline of RUP. On the second iteration he will be able to complete the DEMO triangle of models and then complete the model of the I-organization and a prototype of the I-application.

DEMO models can support these two disciplines as will be shown in the “way of modeling” paragraph. Based on the model transformation guidelines the practitioner can decide what DEMO models are required as input for the activity of transforming into RUP models. Thus, when this input is available he can move on to the RUP artifacts without completing the rest of DEMO models. After completing, for instance, the IAM of the CM and the PM of DEMO the practitioner can create the use case model of the I-organization. Then, he can iterate and complete the DEMO triangle, finalize the use case model and continue to the analysis model based on the SM of DEMO.

However, iterating through DEMO and RUP *can be cognitively cumbersome* especially if there is only one person that undertakes the several roles of DEMO and RUP. The practitioner will be *required to switch several times into different “ways of thinking”* and therefore the quality of the developed models may be compromised. Thus, during the planning of the iterations this aspect should be taken into account and avoid the cognitive overload of the practitioner either by assigning the roles to different persons or by planning the execution of the activities of each methodology on different days or weeks.

6.4. Way of controlling

The “way of controlling” of the multi-methodology *adapts DEMO into the stages of RUP* for controlling the methodology execution. This is done due to the fact that DEMO only provides the cross model tables (Figure 42) that introduce some dependencies between the models and some guidance on the sequence of the development of the models. Contrarily, RUP provides three dedicated disciplines for the support of the process.

The guidance that the cross-model tables that DEMO offers can be used as an input for the “Project Management” discipline and its activities. It can be used by the project manager role to derive the dependencies of the project phases and the milestones that connect phases with each other. Thus, the Project Manager role of RUP has information about the dependencies of the project; namely the sequence that the models have to be developed. Then the overall and partial projects can be planned by developing a Gant chart or an activity network diagram.

6.5. Way of supporting

DEMO does not use an extensive set of specialized software tools to support the execution of the methodology. So far the most dedicated software that is used is the Xemod modeler developed by Mprise which supports the “way of modeling” of DEMO. Among others, Xemod provides the function of creating reports so people with different tools can communicate easily in an interoperable way.

Thus combining the “ways of supporting” of the two methodologies is possible since the practitioners of DEMO have a way to communicate with the practitioners of RUP. The practitioners of the former can create a report that can be used by the practitioners of the later without the need for an extra tool. Furthermore, we hope that this thesis will provide enough guidance to the practitioners of the multi-methodology on the value and utility of the Xemod report!

Though, this communication, although required, does not provide an added value since it lacks automation of tedious parts. The practitioner of RUP will have to do a lot of manual work in order to transform the models and enter every modeling element (actor, use case, class) into the modeling tool of RUP (Rational Rose). Also, he will have to apply the guidelines in a right way which can not be assumed that will always be the case.

However, Xemod may include the function of transforming DEMO diagrams into UML diagrams. The transformation algorithms can be based on the guidelines that follow but so far the granularity of the guidelines does not comply with the granularity of algorithmic steps. The steps of the guidelines can not be transformed immediately into algorithmic steps, executed by a programming language. Furthermore, the human judgment and experience of the practitioner is required during the transformations something that will require advanced techniques from disciplines such as artificial intelligence and human-computer interaction. Thus, although suggested the feasibility of adding this function to the program is not studied in more detailed.

6.6. Way of modeling

Based on Figure 44 there may be a *relationship between the two worlds of DEMO (C- and P-world) with the two main categories of diagrams of UML (behavior and structure diagrams respectively)*. Both C-world and behavior diagrams describe the interactions between the elements of the system. The C-world does this by keeping track of all the C-facts that have been created up to this time (PSD of the PM). UML behavior diagrams show the dynamic behavior of the objects in a system as a series of changes in their internal state and exchanged messages over time [OMG, 2009]. The state machine diagram shows the former and the dynamic behavior diagrams, such as interaction diagrams, of Table 8 show the later. Thus, there should be an unidentified correspondence between the PM of DEMO and behavior diagrams of UML that can be used in the mapping of the “ways of modeling”. We can make an intuitive mapping between the C-world and how it is modeled in DEMO (CM, PM) and the behavior models of UML as well as the P-world (SM, AM) and the structure diagrams of UML.

Transformation guidelines can be created that will guide the practitioner in the transformation of the models. Anyway, this is how RUP connects its “Business Modeling” artifacts with the rest of the process; by transforming them into the requirements discipline artifacts.

In the following paragraphs we provide some rough guidelines on how we can transform DEMO models into RUP models. However, we consider two transformation scenarios since each scenario requires different guidelines. We should mention here that these two scenarios are not disjoint. Within the execution of the multi-methodology each scenario can be used; the adoption of one scenario for an actor does not exclude the adoption of the other for another actor.

- i. *Mimicking B-Actors.* In this scenario a B-actor will be mimicked by ICT. It includes the development of a B-application, referring to Figure 35. We should keep in mind this figure and expect that this scenario can not apply to all B-actors.
- ii. *Supporting B-Actors.* In this scenario the intellectual needs of the B-actors are supported by ICT. This scenario could as well be named “Mimicking I-Actors” or “replacing I-actors” since I-Actors⁵ are supporting the B-Actors with the services they offer to them (see Figure 30). However, if we name this scenario with the two later options we imply that the I-actors have been identified. This will not be our option though. Identifying I-actors requires additional workload for the modelling of the I-organization with DEMO ([De Jong et al., 2010], [Sandhyaduhita, 2009]). Also, we assumed previously that the Operation Axiom does not hold for the I-organization in our case thus I-actors can not be identified.

For describing the guidelines we use some simple general examples in the following section which consist of simple diagrams of DEMO and how to transform them to UML diagrams. On the next chapter we apply the guidelines into the case study of Mprise that we used in chapter of Business Modeling in RUP. The narrative description can be found in Appendix D – The narrative description of the case study.

6.6.1. “Mimicking B-actors” scenario

This scenario has gained the most attention so far [Dietz, 2003], [Shishkov et al., 2004], [Mallens et. al., 2001], [Sandhyaduhita, 2009]. In this scenario the B-actors are replaced by ICT thus the transaction that is executed by him is replaced by a use case. The goal of this scenario is to develop the use case model and analysis model of the B-application.

The use case model that will be derived should contain two major components: use case diagram and use case description. The main components of the later are the flow of events and the goal of the use case. The flow of events describes the sequence of actions between the actor and the system [Kruchten, 2005] and contains all the exceptional flows that may arise as well. It is a textual description in natural language of the steps and usually it is supported by diagrams, mainly activity or sequence diagrams.

Based on [Dietz, 2003] and [Shishkov et al., 2004] we can immediately state that a B-actor is transformed to a use case actor. Furthermore, a DEMO transaction can be straightforwardly mapped to a B-application use case. The goal of a use case can be identified in the Transaction Result Table (TRT) of the IAM. The IAM is mainly used to derive the use case diagram.

Further information for the flow of events exists in the PM which shows the steps and the kind of relationships between the transactions. The AM contains the “agendum” of the actor, specific guidelines that he has to go through in every step of the transaction he is responsible for.

⁵ We refer to I-actors internal to the I-organization and not B-actors who had shaped their ability

Also, AM contains the object classes that the B-actor is manipulating during the execution of a transaction. The Information Use Table (IUT) of the PM contains this information in a compact format. These classes are described in the SM further. Every class that the replaced B-actor is using is transformed into an analysis class of the analysis model since the B-application will replace the B-actor and therefore the application should be aware of all the classes that are used by the B-actor.

6.6.1.1. Building the Use Case Model of the B-application

The simplest IAM is comprised of the initiator, the executor and the transaction. It can be shown in Figure 50:

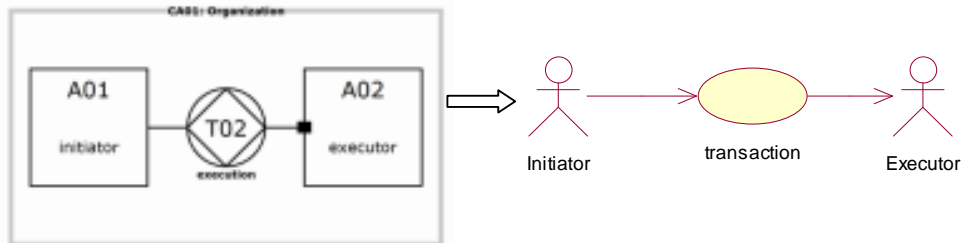


Figure 50 - Transforming the IAM into a use case diagram

We see that the executor is present in the use case. However, when the transaction is executed by the system the executor should not be modelled as an actor. Though, we keep this actor in the use case because it makes the following UML diagrams easier to understand.

Further on, the transaction pattern applies for the T02 transaction of Figure 51. This will provide the flow of events of the derived use case which can be modelled both in natural language and a sequence diagram. Below we chose a sequence diagram.

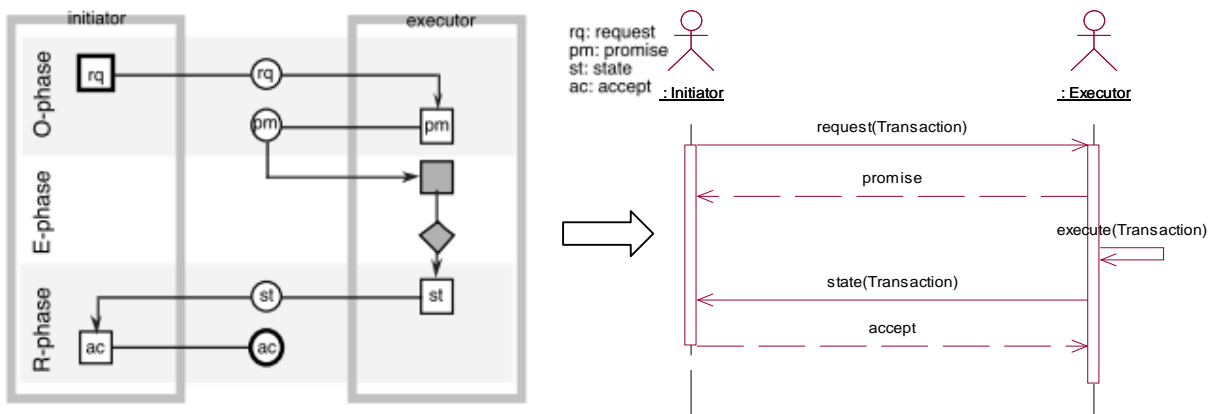


Figure 51 - Transforming PSD into sequence diagram

However, the above pattern is the simplification or the “shortest path to success” of the complete pattern. The complete pattern can be transformed in a UML activity diagram with swim lanes. Due to many alternative flows in the complete pattern a sequence diagram will become difficult to read.

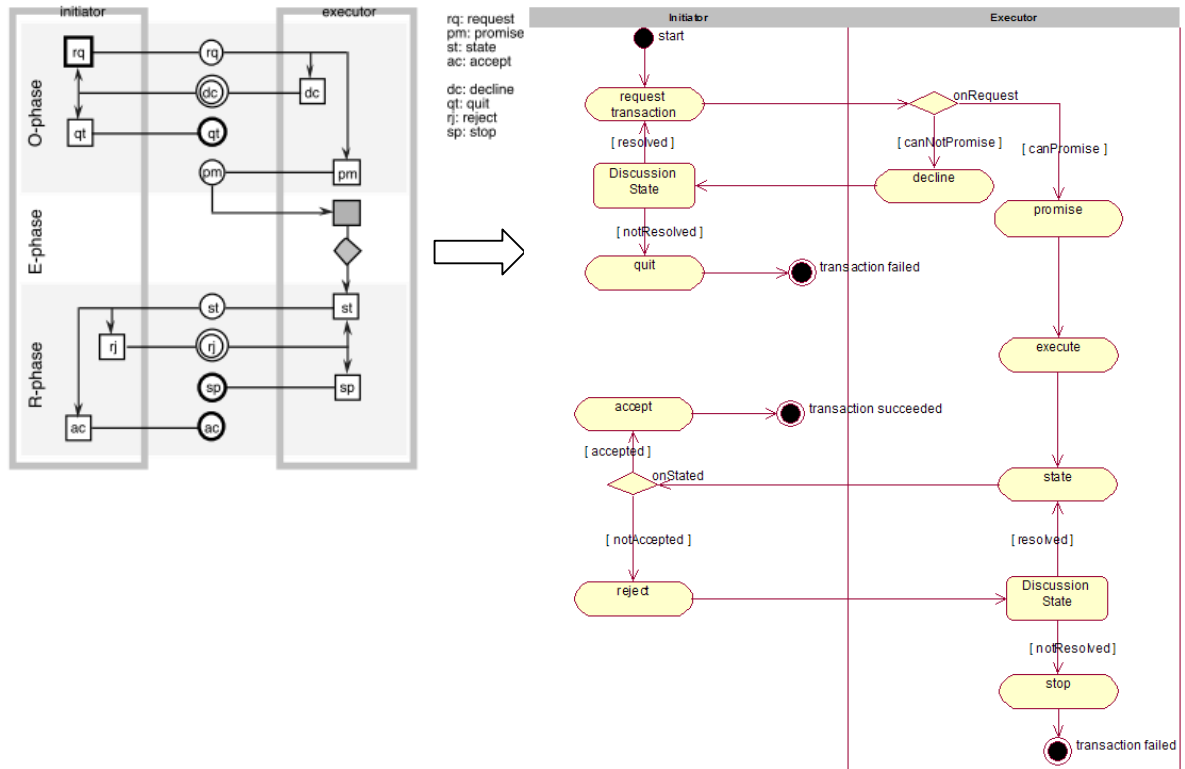


Figure 52 - Transforming the complete pattern into an activity diagram

We see in the activity diagram two decision nodes: onRequest and onStated. The outcome of the decision nodes is based on the agendum of the executor which is stated in the AM. The names of the nodes are indicative of the agendum that applies. We should include a slightly more complex example which includes the initiation of transactions from the mimicked B-actor. This example will show what happens to the use case model when the B-actor who is mimicked by a B-application has to initiate a third transaction. The DEMO diagrams that are used have been developed using Xemod 2010 beta version which uses the DEMO 3 notation for the PSD. First, based on [Dietz, 2003] the use case derived from two transactions of the IAM will be related with “extend” and “include” stereotyped relationships based on the PSD and the multiplicity of the causal link between them. If the multiplicity is 1, then the stereotype “include” is used while if the multiplicity is 0..n, $n \geq 1$, then the “extend” relationship is used. Thus, based on the following IAM and PSD we develop the following use case model.

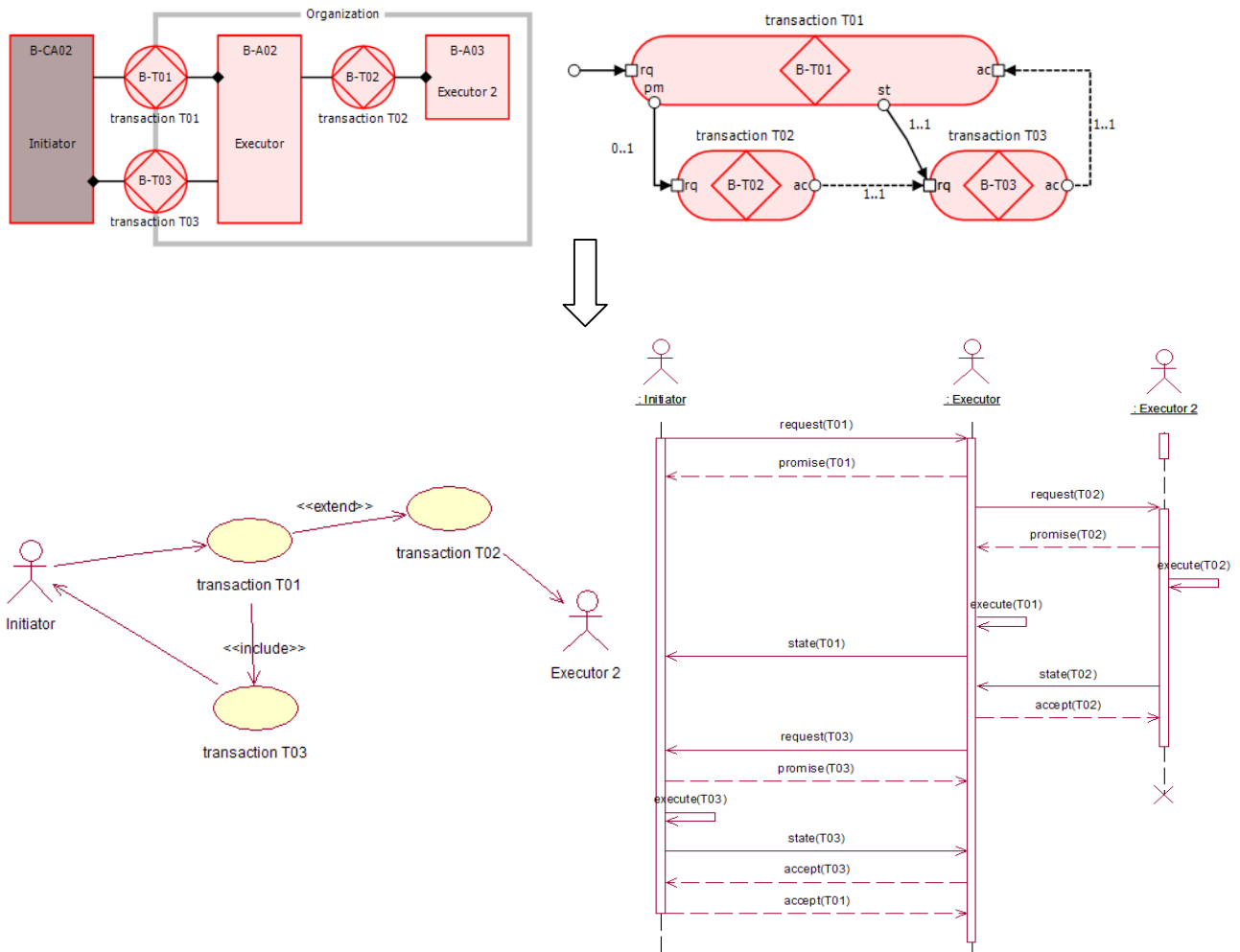


Figure 53 - A more complex case of transformation

We see that the sequence diagram can also include the information about the causal links of the PSD. We see in the sequence diagram that the “request(T03)” message is not sent before the “state(T01)” and “accept(T02)” messages have been sent in order to complete transaction T02. Also, “accept(T01)” message is sent in the end of the message exchanges, showing the same information as the PSD.

The use case model is completed with the use case specification. The most important parts of it are the use case goal and the flow of events. The goal of a use case can be identified in the TRT of the CM for the originating transaction. The flow of events of the use case can be found on the PSD which provides the steps that are taken during the execution of a transaction. Also, the AM contains the agendum of the B-actor, thus some more detailed steps he has to take. However, the natural language used in the description of the flow of events should be natural and not contain special terminology of DEMO since this may create confusion to the RUP practitioners. This should be taken more in account for the information that is contained in the AM since it can be confusing to a non-DEMO practitioner.

Finally, we should note here that we followed a different sequence of steps that [Dietz, 2003] and [Sandhyaduhita, 2009] have taken. Dietz (2003) first transforms the PM into a use case diagram, then determines the kind of relationships between the use cases and finally adds the infological components of the use cases. Finally, he chooses the use cases that are mimicked by ICT. However, in our research we first choose the

use cases that will be mimicked by ICT from the IAM, then we determine the kind of relationships based on the PSD and finally we determine the use case specification based on PM and AM.

Also, Sandhyaduhita (2009) uses the same approach but elaborates more on the definition of the infological components by defining the I-organization using DEMO. However, we chose not to follow this path because we assume that the Operation Axiom of DEMO does not hold for the I-organization thus no I-actors can be identified. The definition of the I-organization is considered in the next scenario using UML. Although, the technique we use is not fundamentally different from Sandhyaduhita, we use it in a different context than she does and we removed some intermediate steps. So far, the infological actions of a use case are stated in the use case specification since we will prefer to avoid adding DEMO terminology into the use case model. This may create problems on the communication of the (possibly) different teams that will use DEMO and RUP.

6.6.1.2. Building the Analysis Model of the B-application

The IUT table of the PM contains what object classes, fact and result types of the SM are used in every step of the PM. Based on this table, the object classes that the replaced B-actor is using during the execution of the transaction are *transformed into analysis classes of the analysis model of RUP*. This correlation is necessary because since the B-application is replacing the B-actor then it should hold a similar representation of the P-world; it should “know” the same “things” as the B-actor. The transformation is done following the bellow mentioned rules which are more rules of thumb than strict rules that always have to be followed:

1. Every object class used in the replaced transaction is transformed into an analysis class. There is no difference in the resulting analysis class if the object class is external or not.
2. Every original fact type of this object class in the OPL is a property of the analysis class. The scale denotes the type of the property but it may not relate directly to the types used for class diagrams.
3. Every derived fact is transformed into a method of the analysis class. The returned value of the method is of the same type as the scale of the derived fact. Though, not all of information has to be provided on the signature of the method since it is defined on a higher level.
 - a. The derivation rule of the derived fact describes the algorithm of the method of the analysis class. This may be modelled using an activity diagram or even the same rule if it is formally defined.
4. A unary result type associated with the object class defines a state of the analysis class. The transaction that creates this result type, based on the TRT, defines the event that initiates a transition to this state. Based on the PM, the TRT and the rest of the unary result types, the events that initiate transitions from a current state can be identified. Based on this knowledge a complete state machine diagram of the UML can be developed.
5. The unary unicity law of a binary factum defines the multiplicity of the relationship of the analysis classes. The object class that is under a unicity law in the SM has a multiplicity 1 in the corresponding relationship of the analysis model. On the opposite the related class has a multiplicity 0...*.
 - a. If there is a binary unicity law in two factums then the multiplicity in relationship of the analysis model may be 1...* for both roles.

6. A ternary fact type that does not define an extension creates an association class. The facts that are not under a unicity law are the properties of the class. The multiplicity of the association may be 1...* for both classes that participate in the relationship. Rule 5 may also apply.
7. If an object class is dependent (marked with a black dot) then the object classes participating in the relationship should also be included in the analysis model using the same above rules.
 - a. The dependency usually will denote a directed relationship between the analysis classes. The dependency dot denotes the start of the relationship.
 - b. The dependency may also denote (weak) aggregation where the dependent object class (container) creates the analysis class which is the container (white diamond on its side of the relationship).
 - c. Choosing the first or the second option is mainly dependent on the semantics of the domain and a rule can not be devised.

An example from the pizzeria case of [Dietz, 2006] is shown that uses all the above rules in order to transform the SM of the B-organization into the analysis model of the B-application.

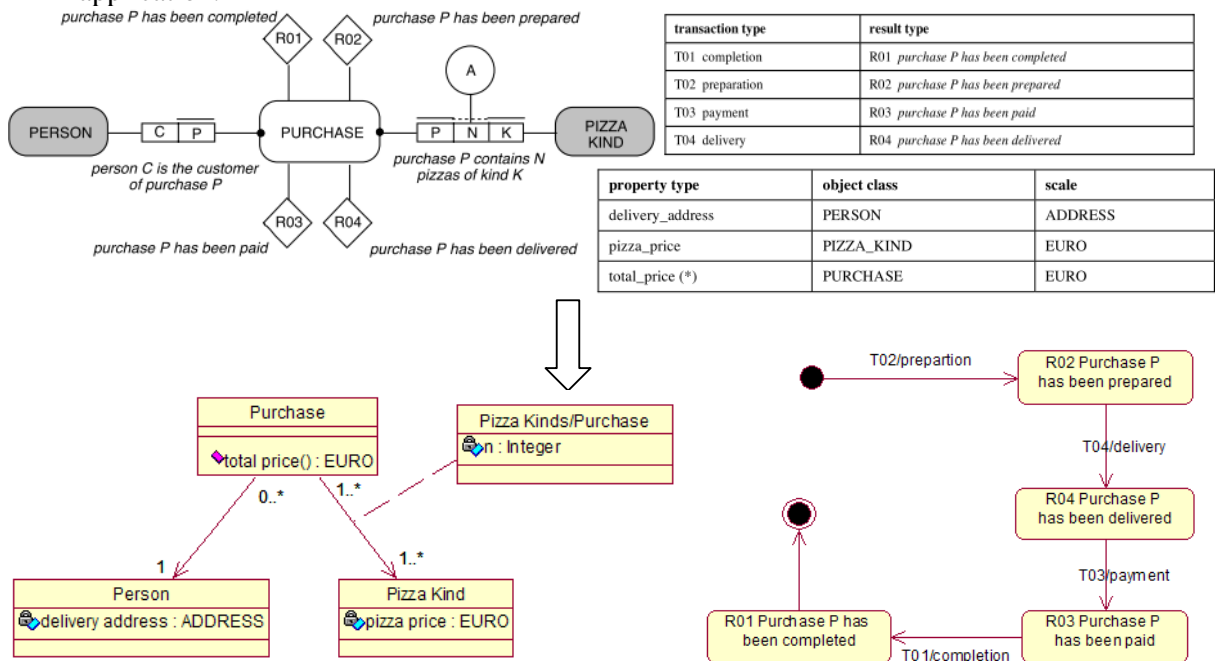


Figure 54 - Transforming a SM into an analysis model

8. The extension of a fact type can be either transformed into a new analysis class, an association class or a specialization class. Some rules of thump could be the following but they may not apply in every case.
 - a. An extension of a unary fact type of an object class transforms into a specialization class in the analysis model. The generalization class is the related object class. This is the case of specialization as presented in [Dietz, 2006, ch. 5].

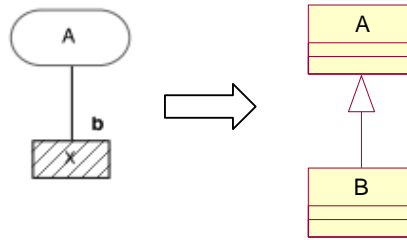


Figure 55 - Transforming a specialization

- b. An extension of a binary fact type that does not include both facts of the fact type introduces a specialization class in the analysis model. The generalization class is the object class related with the extension.
 - i. The multiplicity of the introduced class is determined using the above rules.

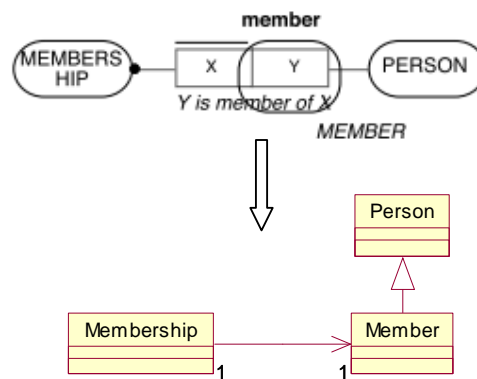


Figure 56 - An extension of a binary fact type

- c. An extension of a ternary fact type that includes one or two of the facts of the fact type is transformed in a relationship class. The rest of the facts may become properties of the new relationship class. This denotes the case of partition as presented in [Dietz, 2006, ch. 5].
 - i. The type of the properties of the relationship class can be derived from the associated scale if existent. However, there is no direct relationship between scales and property types.
 - ii. Also, there may be the case that the association class is redundant. Then the property of the association class can be included in the class which has multiplicity 1 in the relationship.

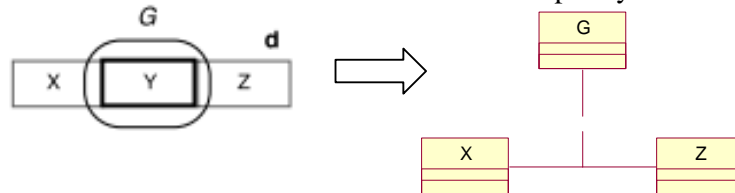


Figure 57 - An extension creating an association class

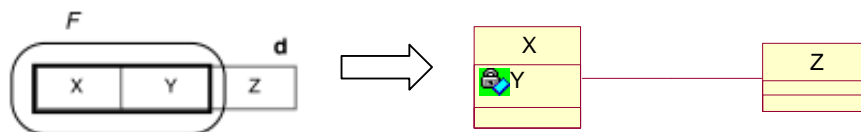


Figure 58 - An extension as the property of on class

- d. In a binary or ternary fact type where all facts are included in an extension a new analysis class is created. The participating object classes are transformed in analysis classes as well. The new class is related with a composition relationship. This denotes the case of aggregation in [Dietz, 2006, Ch. 5].
 - i. Aggregation in the analysis class diagram has two forms: strong and weak aggregation (or composition and aggregation). Composition is drawn with a black diamond on the container class while aggregation with a white diamond. Composition classes should always be present in an instance of the composed class.
 - ii. The selection depends on the unicity law. The object class under a unicity law is a composition otherwise is an aggregation.

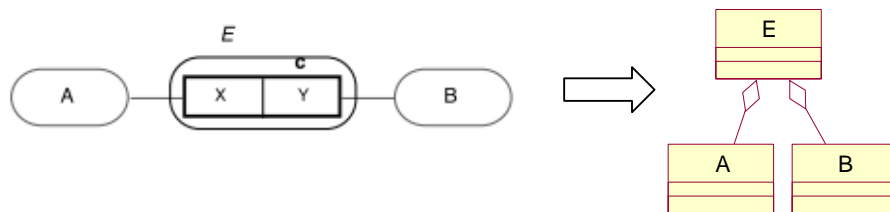


Figure 59 – Transformation of an aggregation

- 9. The union of two object classes creates a generalization analysis class associated with two analysis classes derived from the object classes participating in the union.

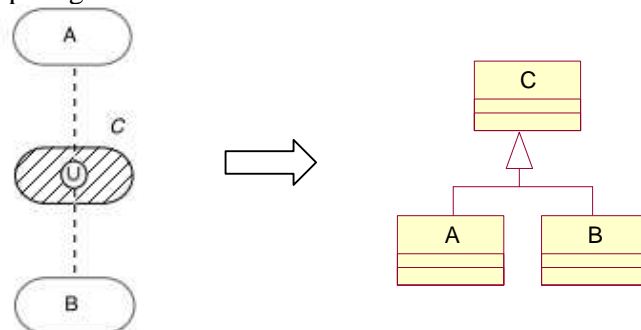


Figure 60 - Transforming a union into a generalization class

6.6.2. “Supporting B-actors” scenario

This scenario models the I-organization using the use case and analysis model. The development of the use case of the I-organization starts from the AM of DEMO which is the base of the triangle of the B-organization (see Figure 41) and also the most detailed and comprehensive model [Dietz, 2006]. The infological actions are identified using this model.

Also, the construction of the I-organization is the analysis model as well. The construction of the B-application and I-organization may be the same because B-application can be mimicked by the I-applications. Furthermore, by evolving the analysis model into the design model we can derive the construction of the I-application.

Thus, if we take an example from the source book of DEMO [Dietz, 2006] we can identify the following agendum for an actor that occurs every time the actor is requested the transaction T04:

```

on requested T04(L) with book_copy(new L) = C and membership(L) = M
    if #books_in_loan(M) ≥ max_books_in_loan(Y) → decline T04(L)
    ◇ #books_in_loan(M) < max_books_in_loan(Y) → promise T04(L)
    fi
no
    
```

We can derive infological actions from this agendum which are actions that are using the informa human ability in order to be performed. Thus, we can identify the following use case diagram for the I-organization.

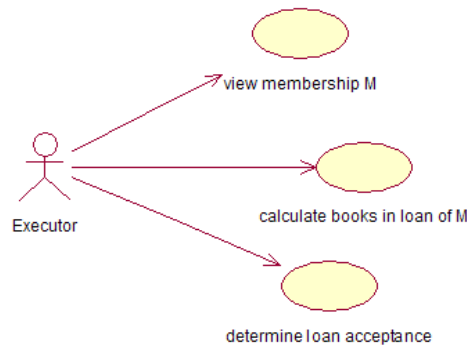


Figure 61 - Use case model of the I-organization

The use case specification can contain the steps that the actor has to do in order to view the membership and determine loan acceptance based on the example. The steps may as well contain the interaction with the I-organization or the I-application that supports the I-organization. Thus, they may not always be derived by a DEMO diagram.

Concerning the construction of the I-organization, modelled with the analysis model, this scenario uses the same guidelines in transforming the SM in the analysis model of the I-organization, using the information of IUT. The agendum of a B-actor contains the object classes that he is manipulating. Thus, it is also easy to determine which object classes are initially transformed into analysis classes.

7. Applying to the Case Study

7.1. Applying to the case study

The multi-methodology designed in the previous chapter will be applied in the case study of Mprise. The case study is based on the work of Sandhyaduhita (2009) and is based on the training division of Mprise which was a separate business entity until recently.

The narrative description of Mprise activities has been developed in [Sandhyaduhita, 2009] and for convenience is copied in Appendix D – The narrative description of the case study with permission. Also, Sandhyaduhita applied DEMO methodology to the case by developing the models. The DEMO diagrams have been approved for their validity thus they will be the base of applying the multi-methodology. They can be found in Appendix E – The DEMO models of Mprise [Sandhyaduhita, 2009].

In the application of the multi-methodology we will focus mainly on applying the “way of modeling” that was discussed in the previous chapter so we can validate our method. However, we should also mention the other ways wherever required.

7.1.1. The Case Study

The case study refers to the Mprise division of the former Dynaprise Group B.V. and not to the current organization of Mprise which is the merger of the former divisions of Dynaprise Group in one legal entity. Thus, from this point on when referring to Mprise we will refer to the former Mprise division. We should give a small introduction by quoting parts of the narrative description.

“Mprise is a company which has competence in ERP applications (Microsoft Dynamics-AX and Dynamics-NAV). It has the status of Microsoft Gold CPLS (Certified Partners for Learning Solutions) and therefore it has Microsoft Certified Trainers (MCT). Every year there is a certification renewal by Microsoft.

Furthermore, since Mprise is an independent training center (the only one in the Netherlands) of Dynamics-NAV and Dynamics-AX 95% of its business is to provide training and the rest is to provide advisory services in ERP applications (Microsoft Dynamics-AX and Dynamics-NAV). The business is addressed by two kinds of training services and an advisory service. Each of these services is supported by some business processes¹, as stated in the sub lists.

- a. *Standard course*
 - 1. *Course product development*
 - 2. *Establishment of standard courses*
 - 3. *Standard course enrollment*
- b. *Client specific course*
 - 1. *Client specific training development*
- c. *Advisory service*
 - 1. *Advisory service development*
- d. *Applied to all categories*
 - 1. *Course altering*
 - 2. *Course execution*
 - 3. *Course evaluation*

¹ This list is rather informal and conducted intuitively. The formal specification is expressed in the DEMO models.

4. *Provided service payment*

7.2. The “way of working”

In our case the DEMO model triangle is completed since we are based on a previous thesis conducted in Mprise. Thus we will only practice the guidelines of combining the “ways of modeling” of the previous paragraph and we will also practice the RUP workflows of “Requirements” and “Analysis & Design”, though we will focus mainly on the analysis part of the former.

Thus, iterating will be easier and less cumbersome since we avoid iterating in both methodologies. However, based on the guidelines of transforming DEMO models into use case and analysis models we could as well iterate between DEMO and RUP.

The use case model requires as input, mainly the IAM and the PSD of the PM of DEMO and during the completion of the use case model it also requires the AM. Thus, a first iteration could develop DEMO’s IAM, PM and part of the AM model. Then the use case diagram and some use case’s specification can be completed.

Then a second iteration can start by finalizing AM, building the SM and ISM of the CM and finally completing the PM with IUT. Based on the IUT, the complete AM and the SM the RUP practitioner will be able to complete the remaining of the use case specifications and the analysis model. With the completion of these two models a new iteration can start by adopting RUP completely and using it in order to devise the ICT system.

A brief description of the guidelines follows. These are not to be followed strictly since a project may have different needs. Their purpose is more to indicate a possible sequence of the tasks and reveal the dependencies of the models based on the guidelines provided earlier.

1. 1st iteration

- a. DEMO practitioner
 - i) Complete the IAM
 - ii) Complete the PSD of the PM
 - iii) Develop a part of the AM. The part can be 10-60% of the complete model.
- b. RUP practitioner
 - i) Identify use cases and system actors based on the IAM and the PSD
 - (1) Identify B-actors that can be mimicked by a B-application
 - (2) Identify B-actors that can be supported by an I-application. Not all of these actors can be identified during the 1st iteration since a complete AM is required.
 - ii) Develop the use case diagram based on the PSD
 - iii) Identify part of the flow of events of every use case
 - iv) Continue in designing a solution and a prototype if necessary following the techniques and guidelines of RUP.

2. 2nd iteration

- a. DEMO practitioner
 - i) Complete the AM
 - ii) Complete the SM
 - iii) Complete the PM with IUT
 - iv) Develop the IAM partially. The IAM is not used as input in the guidelines of transforming models thus is not required from the RUP practitioner.
- b. RUP practitioner
 - i) Complete Use case model by completing the use case specification

- ii) Develop the analysis model based on the SM, the IUT and the outcomes of steps 1.b.i.
- iii) Continue in designing a solution and a prototype if necessary following the techniques and guidelines of RUP.

The above guidelines on the “way of working” can be used for the “way of controlling” of the multi-methodology”. The previous list contains information about the phases of the project and the dependencies of the phases. Based on it, the phases, activities and tasks can be determined. By completing the list with an assessment of the duration of each part the project manager can develop a Gant or an activity network diagram which will assist him in the management of the project.

7.3. Transforming DEMO models to RUP models

In our case we have the complete triangle of the DEMO models completed. Thus, we do not have to iterate between DEMO and RUP. Therefore, we will follow only the steps under the “RUP practitioner” heading.

First, we examine the IAM, PSD and AM to identify B-actors that can be mimicked by a B-application and B-actors that can be supported by an I-application.

The B-actors that can be mimicked by a B-application are:

- A01 – Enroller (Figure 70)
- A07b – Client project completer (Figure 71)

Thus, based on the IAM diagram of the CM (see Figure 70 and Figure 71) we identify the following (system) actors and use cases

B-application actors	B-application use cases
CA01 – Person	T01 enrollment
CA02 – Client	T02 enrollment payment
CA03 – Debtor	T03 quotation
A24 – Approver	T04 quotation payment
A16 – Designer	T16 design
	T24 discount approval

Table 10 - The B-application actors and use cases

By identifying the causal and conditional links of the PSD we can complete the use case diagram with the stereotypes of the relationships between the use cases.

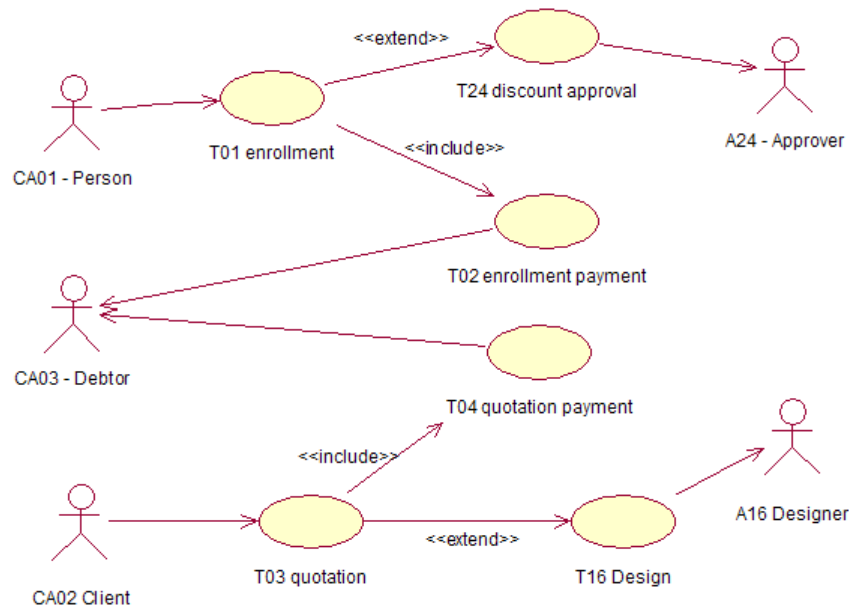


Figure 62 - Use case of the B-application

From the following use case diagram we will only focus on the use case “T02 enrollment payment” due to time and resource constraints. Thus, we will develop the use case specification only for this use case and later on we will develop the analysis model only for this use case. We present the goal of the use case, the preconditions, the basic and some alternative flow of events in a textual representation since it is easier to present; an activity or a sequence diagram could be used as well.

Use case goal

Enrollment E has been performed

Preconditions

1. There is a person who will be the participant in the enrollment
2. PA is the participant in E
3. The chosen standard course is currently offered
4. There is a client - representative of a company - who orders the enrollment for the participant
5. There is a debtor who will pay for the enrollment fee.

Flow of Events

1. If there are is no seats available then the enrollment is declined
 - a. Seats are available if the number of the participants of the chosen course is less than 10.
2. If there are available seats then it is examined if the enrollment should have discount.
 - a. This is examined by the “T24 discount approval” use case
3. After the determination of the discount the enrollment fee is calculated
 - a. The formula is $\text{Enrollment fee} = \text{price}(\text{Course Kind}) \times (1 + \text{tax}\%)$
4. If the enrollment is delayed, relatively with course date, then the delay fee has to be paid.
 - a. The formula is $\text{Delay_fee} = \text{price}(\text{CK}) (\text{delay_percentage}(\text{E})) \times (1 + \text{tax}\%)$, where the delay percentage is calculated based on the weeks before the course.

5. The enrollment is completed.

Alternative Flows

The person who enrolls has the option to cancel the enrollment. This can happen in any time before the execution of the enrollment. The steps that apply are:

1. The person requests the cancelation of an enrollment.
2. If the type of enrollment is normal and the standard course has not been given then the cancelation is possible and the cancelation can go to the next step.
3. The cancelation fee that the debtor has to pay is calculated
 - a. If the enrollment is not completed before cancelation is requested then there is no cancelation fee.
 - b. Otherwise the cancelation fee is dependent on the weeks before the scheduled course.

7.4. Building the Analysis Model

In order to build the analysis model we have to see what information is used by the transactions that the use cases are originating from. To find out, we check the IUT of the PM or the AM. The fastest information we can get is from the TRT which will give as the main outcome of every transaction.

Thus, the object classes that should be transformed into Analysis classes are:

- Enrollment
- Course Kind
- Course
- Standard Course
- Professional
- Person
- Company
- Period

Based on the OPL of the SM of Mprise (see Figure 81) and the guidelines provided earlier we can transform into the following analysis class diagram.

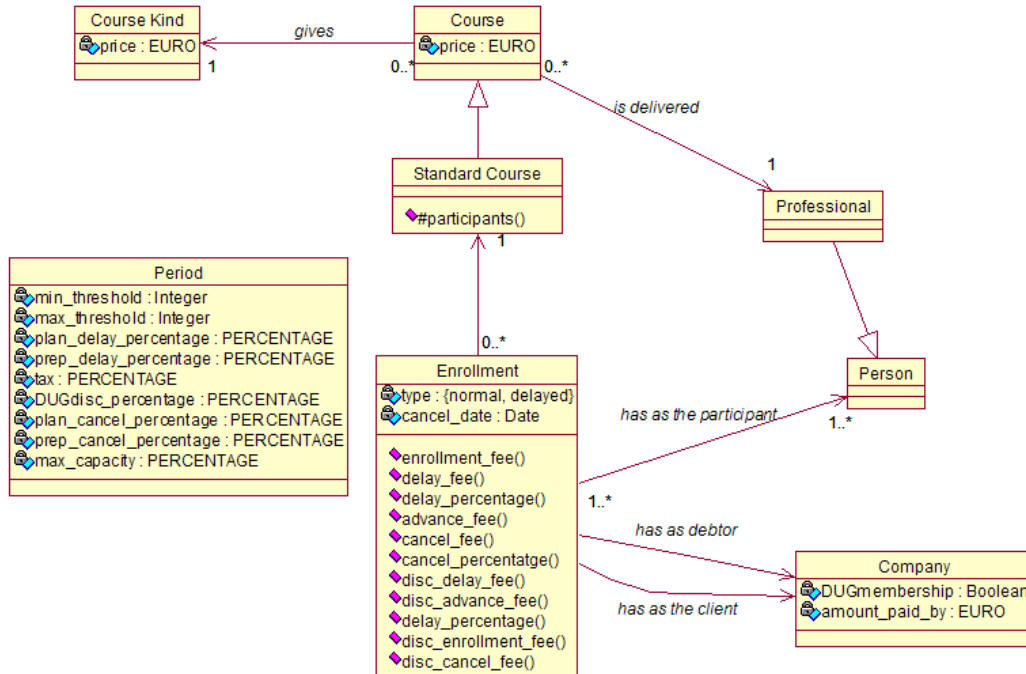


Figure 63 - Analysis model of the B- and I-application

The analysis classes of Enrollment, Course and Period need a state machine diagram to complete their description. We include only the state machine diagrams of Enrollment and Course since they are more meaningful.

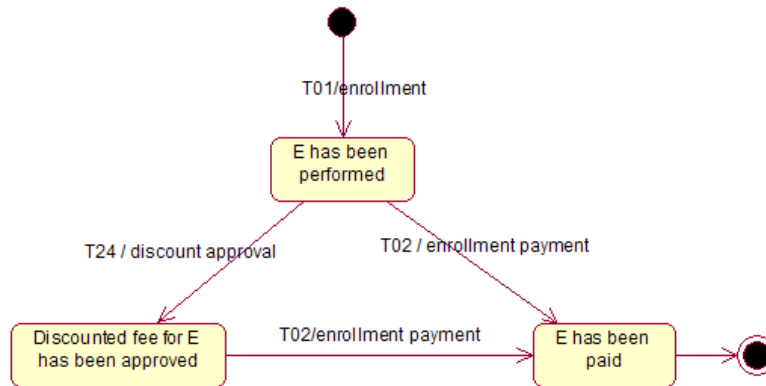


Figure 64 - State-machine diagram for the Enrollment class

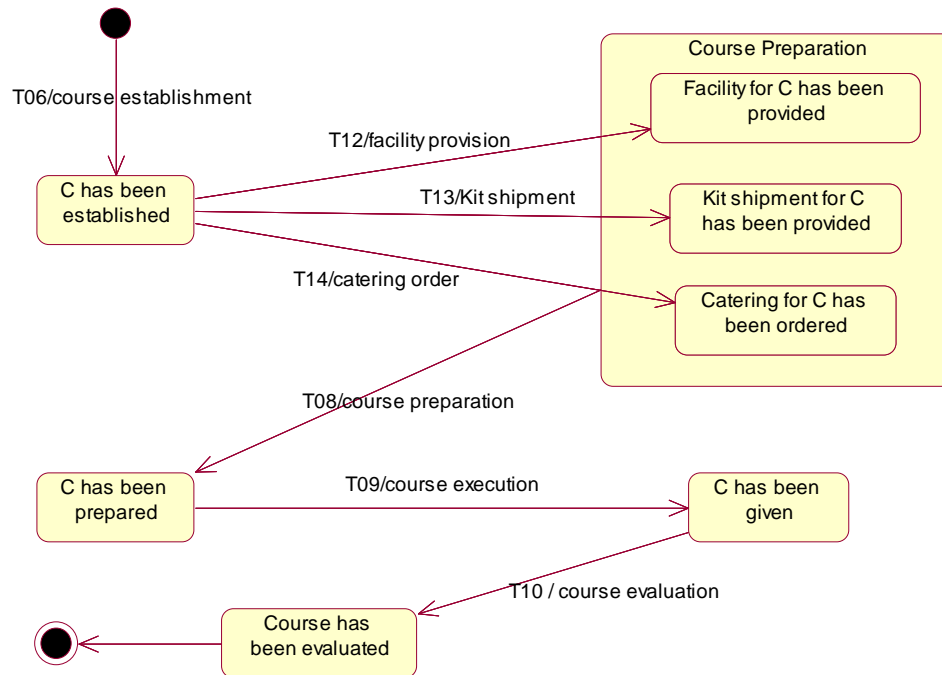


Figure 65 - State machine diagram for the Course class

7.5. The added value of the case study

Applying the multi-methodology to the case study has given an insight into the advantages of the emerged multi-methodology. These are discussed in the following chapter. Also, the multi-methodology was refined by applying the draft design versions into a real case problem. For example, it would be impossible to devise the rules of transforming the SM to the Analysis model without applying and verifying them in the case study.

However, the case study was not a complete execution of a RUP process, even as configured to match the multi-methodology. This is due to the fact that RUP requires an extensive set of documental artifacts and a diverse of undertaken roles that are difficult to be taken by one person. However, the configuration of RUP used was a possible configuration for simple projects. Also, the fact that the DEMO models were already developed has influenced the multi-methodology execution steps.

8. Discussion and Conclusions

In this thesis we tried to connect a business modeling methodology with a software engineering process. Our purpose was to exploit the advantages of both and create a multi-methodology that will be able to go through all the phases of developing large software systems.

We used DEMO methodology for modeling the business. DEMO has well identified advantages when modeling an organization but it can not be used for developing software. Contrarily, RUP is very popular among the software engineering practitioners for the development of software but it can not cope well with the business domain.

In this paragraph we will mention the advantages that DEMO offers against the RUP modeling technique and the advantages that the multi-methodology offers. For the former case we will a comparison medium. These are the 5 dimension of Dietz (2006) that are required in a business model. Let us start with the 5 dimensions after a small discussion about the RUP technique for business modeling.

8.1. Disadvantages of UML in Business Modeling

The business modeling in RUP uses a profile of the UML modeling language. It is rather a technique than a methodology and thus the lack of theoretical background can not give a sound basis which the practitioner will use to validate his results. The quality of the models is based on the experience of the practitioner. Also, the application of the technique varies among practitioners. This was obvious from the different assumptions that every writer made for the role, the identification and the description of a use case.

According to Dietz (2006) a business model should be coherent, comprehensive, consistent, concise and essential. *Coherent* means that the different models of a business should constitute a logical and integral whole. *Comprehensive* means that the different models are complete; that all relevant issues are covered. *Consistent* means that the models should not contradict each other. *Concise* means that no irrelevant information is included in the models. Finally *essential* means that the models show only the essence of the enterprise.

The *coherency of the models is not obvious to the practitioner*. After the identification of the business use cases the practitioner does not have other option than to observe the modeled business in order to identify business analysis classes. The correlation of use case and entities is done late, in the description of the business use case realization.

Also, UML business models are *not comprehensive*. There are diagrams that can be used for modeling all the issues of the business but the selection of them falls into the experience of the modeler. The technique suggests what diagrams should be used for every modeling issue but some diagrams may be optional. Furthermore, the selection of every diagram is dependent on the modeling case. I.e. the state machine diagram of a business analysis class is not mandatory for every class. The modeler should identify the states based on his experience and identify that these are states and not for example properties of the class.

This *affects the conciseness* of the models as well. The modeler can as well introduce superfluous information in some models by including irrelevant diagrams or diagrams without a clear role i.e. a state machine diagram of one state. One common mistake is the inclusion of business rules in the business use case specification.

UML business models *can be consistent but the technique itself does not guarantee consistency*. For example a business process can be described in the business use case specification and in the business use case realization. Although, the perspectives are different and complementary, a change only in one of them can make the models inconsistent. Another example is the business rules. Despite the modeling artifact of a business rule it is common that business rules are spanning in several diagrams, such as the decision nodes of activity diagrams, business entity diagrams and business use case realizations. Thus, by duplicating a rule or not making it clear, contradictions may emerge among the different instances.

Finally, RUP business models are not essential by default because *they include implementation details*. This suggested by the business modeling technique with the business use case realizations. Thus, changing something in the implementation i.e. introducing a new billing system or a new web portal, changes drastically the business model, i.e. a new business worker will have to be introduced.

Also, as it was discussed in paragraph 3.7 of page 48, two of the identified advantages of the UML business modeling profile are: it uses the same modeling language as the developers and therefore communication is accommodated and that the language is object oriented therefore objects reflect real world entities.

However, these two can not be considered an advantage. Firstly, the language may be the same but the semantics are different. This one the one hand may create misconceptions to the communication between analysts and developers. On the other hand the language does not provide a different way of thinking for describing the business world. This world is not object oriented, it is rather process oriented. Thus, if someone tries to describe this world using a vocabulary that contains object the chances that he will lose the essence and that will miss important notions are high. Also, during observation of the (business) world the modeler tries to identify wrong elements that are difficult to be observed. A business is more easily conceived and described in terms of business processes than objects.

8.2. The advantages of the multi-methodology

The advantages that this multi-methodology offers can be summarized by Mingers & Brocklesby (1997). First of all, it copes with the complexity of developing software for a business. This complex and multi-dimensional problem is now solved by adopting the right methodology for the right dimension of the world. Adopting a methodology means that you view the world “through a particular instrument” that reveals certain aspects of the world and omits others [Mingers et al., 1997]. Thus, we utilize two different paradigms for two different worlds.

Furthermore, the process of developing a system is supported in all aspect worlds and during most of the phases of the process. The process of developing a system, as shown, is comprised of several phases, each one having its own characteristics, purpose and workload. With this multi-methodology we can support most of the phases in the three aspect systems of the organization theorem. Thus the practitioner is better supported and guided, the complexity of the process itself is reduced and consequently the quality of the developed system is increased.

Finally, in practice DEMO and RUP practitioners may already combine and transform their artifacts. Work has already been done in relating DEMO diagrams with the UML diagrams. However, UML is not a process, it is a modeling language. There is neither a way of thinking to guide the practitioner nor specific guidelines on activities, tasks and roles as well as tools to support him. However, in this multi-methodology the models that each methodology produces are combined by taking into account their

role in the development process. Also, the tools, the management and the philosophy of each methodology are examined and combined.

Also, we should mention the advantages that the multi-methodology offers based on the three problems of Mallens (et. al., 2001). He identifies three problems that a system designer can face when there is no formal business modeling in the system development process: the *delimitation problem* which is the proper delimitation of the system boundary, the *identification problem* which is that all elements of the system have been properly identified, the *specification problem* which is what key model elements should be specified further and how this should be done unambiguously.

The *delimitation problem* requires the proper identification of the system boundary. In RUP this is done with the identification of the use case of the information system. RUP's business modeling technique is used in order to delimit the system by deriving the business functions. However, the identification of use case is based on the identification of a goal of a user which is not always clear to identify. DEMO on the other hand uses the transaction pattern and requires the identification of one of the transaction's steps.

Also, it is awkward to model use case steps using text, a sequence or an activity diagram. The problem is the granularity of the steps which may be atomic or complex since there are no principles on identifying an activity. Also, the use case specification does not include the informational needs of every step, especially in the diagrammatic representation. Anyway, the given guideline is to describe the interaction with the system from an outside perspective [Rational, 2003], so when followed the description of the informational needs should be avoided. DEMO can delimit a system formally by identifying transactions, the steps of every transaction, actors and facts used in every step of every transaction. Using these elements the scope of the information system contains the above crucial information.

The *identification problem* arises in RUP since the analysis model is not clearly related with the use case model until late in the process. Instead the analyst has to go through the system description or observe the real world in order to identify nouns that describe it. Furthermore, there is no principle on deciding when a noun can be a class or a property of a class. Also, the use cases themselves are difficult to be identified and difficult to verify that are complete. The guideline is that there is a use case wherever there is a goal that the user wishes to achieve, something that can be missed or omitted easily. Also, the identification of actors is done using scenarios, case studies and brainstorming about people or systems that will have interest on the functionality of the system.

However using DEMO as the business modeling methodology the system actors and use case can be identified straightforwardly in the elementary actors and transaction respectively. DEMO guarantees the completeness something that improves the requirements gathering of the system. Furthermore, DEMO relates the object classes of the system with the transactions that use them with IUT. Therefore there is a sound method that can guide through the identification of the analysis classes.

Finally, the *specification* of use cases and analysis classes can also be based on the DEMO models. The specification of a modeling element has two aspects: the unique identification and that it is easily identifiable and understood by all who deal with the modeling concept [Mallens et al., 2001]. Also, another aspect stemming from RUP is the traceability of the element; how this element is derived during the process [Kruchten, 2004] (see also Figure 17).

The transaction pattern and the PSD offer a good basis for specifying the use cases unambiguously. Moreover, the goal of the use case can be equated with the

transaction result of the TRT. Also, the pattern is more close to the real world, thus the flow of events is more likely to describe steps of the real world. Besides, the analysis model can reach a great level of specification since it can derive properties and states from the SM of the DEMO. Furthermore, these specification elements are derives from the real world, thus easily identifiable by the stakeholders of a project. Finally, the requirements can be traced in the DEMO models, yet satisfying the RUP aspect on requirement traceability.

Finally, DEMO is enhanced using methods of RUP. First of all, DEMO does not provide clear guidance for the “way of controlling” of the methodology. The three supporting workflows of RUP (Environment, Configuration & Change Management, and Project Management) can be used instead. Fortunately, RUP provides extensive guidance and tools that can be used in order to control a process.

Appendixes

Appendix A – Class diagrams in UML

A class represents a discrete concept within the application being modelled. A class is the descriptor for a set of objects with similar structure, behaviour and relationships. [Rumbaugh et al., 1999] A class diagram describes the types of objects in the system and the various kinds of static relationships that exist among them. Class diagrams also show the attributes and operations of a class and the constraints that apply to the way objects are connected. [Fowler, 1997] There are two principal kinds of static relationships:

- associations (for example, a customer may rent a number of videos)
- subtypes (a nurse is a kind of person)

In a class diagram a class is represented as a rectangle divided into three parts, the upper one is the name of the class, the middle one is the attributes that every object belonging to this class should have and in the bottom there are the methods and operations that the class should have. A specific object of interest that belongs to a class is represented as a rectangle with a semicolon (;) separating the name of the object with the name of the class it belongs to. An example of a simple class diagram and the respective object diagram follows from www.developer.com:

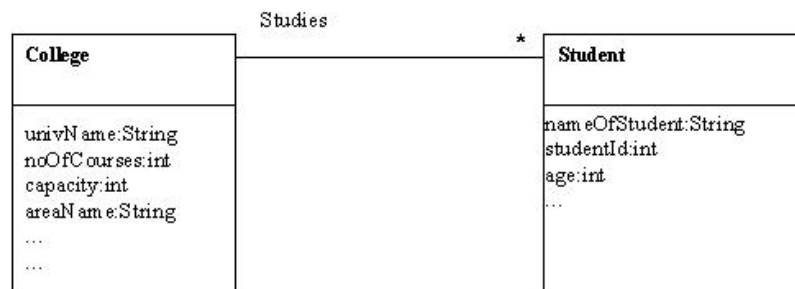


Figure 66 - Class diagram

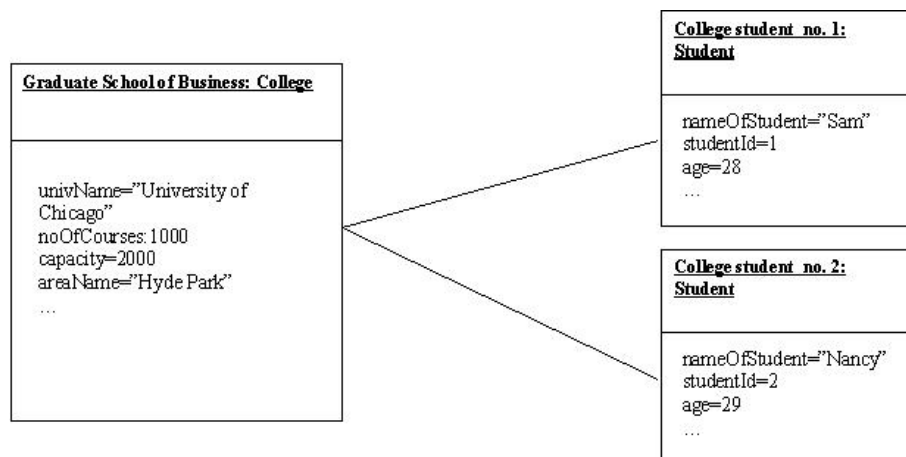


Figure 67 - Object Diagram

Classes can be related with several kinds of relationships. In [Rumbaugh et al., 1999] the various relations have been identified.








Relationship	Function	Notation
association	A description of a connection among instances of classes	 A solid line connects the two class boxes.
dependency	A relationship between two model elements	 A dashed line with an open arrowhead points from Class to Class2.
flow	A relationship between two versions of an object at successive times	 A dashed line with an open arrowhead points from Class to Class2, with a small arrowhead at the start.
generalization	A relationship between a more general description and a more specific variety of the general thing, used for inheritance	 A solid line with an open arrowhead points from Class to Class2.
realization	Relationship between a specification and its implementation	 A dashed line with an open arrowhead points from Class to Class2.
usage	A situation in which one element requires another for its correct functioning	 A dashed line with an open arrowhead points from Class to Class2.
aggregation	An association that represents a part whole-relationship.	 A solid line with an open diamond at the Class end connects to Class2.

Table 11 - Class relationships

Class diagrams can be classified according to their perspective, which is the perspective of the system under design. [Fowler, 1997] recognizes three perspectives that a class diagram can have: conceptual, specification and implementation.

Conceptual: A diagram that represents the concepts in the domain under study. These concepts will naturally relate to the classes that implement them, but there is often no direct mapping

Specification: The classes are more close to the software but the focus is at the interfaces, not their implementation.

Implementation: The classes that implement the system. This is probably the perspective used most often, but in many ways the specification perspective is often a better one to take

Appendix B – Discipline details in RUP

RUP contains detailed description for every discipline. Among the concepts used to describe them are: *role* (the “who”), *activity* (the “how”), *artifact* (the “what”) and *workflow* (the “when”) [Kroll et al., 2003]. These concepts are the backbone of the RUP but the RUP product also offers more detailed process elements which make the process easier to understand and practice by providing complete guidance to the practitioner [Kroll, 2003], [Rational, 2003].

The most important description is the workflow. Workflow in RUP is a sequence of activities that produce a result of observable value [Rational, 2001]. The most two common workflows are ‘Disciplines’, which are the high level workflows, and ‘Workflow Details’, which are the workflows within a discipline. A workflow can be diagrammatically expressed using a sequence diagram, a collaboration diagram or an activity diagram which shows the activities and the dependencies among them, although not all of the dependencies are shown because they are intended for humans and thus are not developed to follow them exactly and mechanically [Kroll, 2003].

These elements are:

Workflow Details: Workflow detail diagrams show groupings of activities that are often performed together in a workflow activity. These diagrams show roles involved, input and output artifacts, and activities¹ performed.

Steps: Activities are broken down into steps, which is not necessary to perform them every time an activity is executed. They classified into thinking, performing and reviewing steps.

Guidelines: Rules, recommendations or heuristics that support activities.

Templates: Models or prototypes for the artifacts to be developed.

Checkpoints: A quick reference to help assessing the quality of the artifact.

Tool mentors: They establish a link and guide on using the tools of the RUP product.

Concepts: Key definitions and principles.

Roadmaps: Guidance of the practitioner into RUP from the viewpoint of a specific role.

The above description can be diagrammatically described with the following diagram from [Passing, 2004]²:

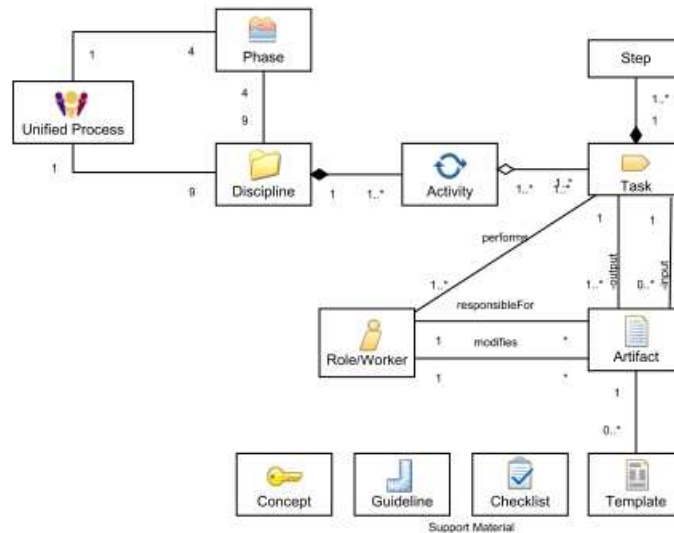


Figure 68 - The structure of RUP

For further details you can refer to [Kruchten, 2005] and [Kroll, 2003] for an introduction to this topic and to [Rational, 2003] for the complete details of RUP.

Appendix C – UML profiles

The business modeling discipline uses a profile of UML. UML profiles are a part of UML that are used as an extension mechanism to the basic language [Johnston, 2004]. More specifically UML contains three mechanisms that allow the designer to extend the basic elements of the language (classes, packages, associations etc.) in order to adapt it to a special application domain and meet its modeling requirements. These mechanisms are stereotypes, tagged values and constraints [Rumbaugh et al., 1999]. Stereotypes introduce new modeling elements, tagged values add modeling attributes to the elements and constraints represent the new modeling semantics [Booch et al.,

¹ Activities at this level of abstraction can also be referred as “tasks”.

² “Checklist” is used instead of “Checkpoint”, “Task” instead of “Activity” and “Activity” instead of “Workflow Details”

1998]. A stereotype is a new kind of model element defined in the model, a tagged value is a pair of a tag and a value that stores information about an element and a constraint is a rule that applies to a tailored model or element. The following picture from [Booch et al., 1998] briefly displays the above mechanisms.

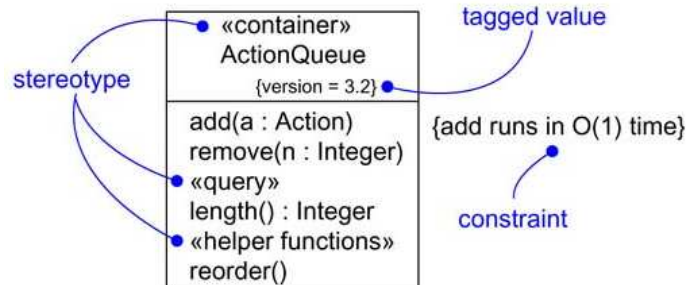


Figure 69 - The UML extensions mechanisms

Appendix D – The narrative description of the case study

Mprise B.V. is part of Dynaprise Group B.V (established in 1991). Mprise is a company which has competence in ERP applications (Microsoft Dynamics-AX and Dynamics-NAV). It has the status of Microsoft Gold Certified Partner for Learning Solutions and Microsoft Business Solutions. The status of Microsoft Gold CPLS (Certified Partners for Learning Solutions) means that Mprise has Microsoft Certified Trainers (MCT). Mprise facilitates and give supports for its trainers to acquire or maintain their Microsoft certification. Every year there is a certification renewal by Microsoft. Thus, Mprise provides time for its trainers to study for the exams and also pays for the exam fee.

Furthermore, since Mprise is an independent training center (the only one in the Netherlands) of Dynamics-NAV and Dynamics-AX its business is not to sell Microsoft licenses to companies but 95% of the business is to provide training and the rest is to provide advisory services in ERP applications (Microsoft Dynamics-AX and Dynamics-NAV). The business is addressed by two kinds of training services and an advisory service as follows:

- a. Standard course
- b. Client specific course
- c. Advisory service

As an education center, Mprise has four (4) classrooms for the standard courses. In each classroom, an ICT infrastructure which consists of a number of PCs for the course is provided.

The client of Mprise is either big or small companies. For the big company, the client is usually represented by a manager as the contact person. For the small one, e.g. a company with only one employee (a freelancer i.e. a consultant who runs his/her own business), the client is the freelancer his/herself. Thus, the participants of the courses offered by Mprise could be employees of a company or a freelancer.

Mirjam who is responsible for the marketing of Mprise groups the clients into the following:

1. Microsoft implementation partner companies

2. Companies which use Dynamics-AX or Dynamics-NAV
3. HBO (educational institution)

Consequently, the main target groups for the course i.e. the ones who will attend the course offered by Mprise can be classified into:

1. Consultants (of Microsoft implementation partner companies)
2. Key users and end users (from companies which are using Dynamics-AX or Dynamics-NAV)
3. Lecturers and students of HBO (educational institution)

In running its business Mprise can be seen as a coordination network of the following actors:

1. Mirjam Berntsen – director competence development; particularly responsible for sales and marketing
2. Joop de Jong – general manager, particularly responsible for innovation of new course products
3. Yvonne Leentfaar – secretary, responsible for administrative issues of Mprise
4. Dorien de Jong – financial staff, responsible for financial issues
5. Pieter Thijsse Claase – manager of training (also a senior trainer with competence in Dynamics-AX: all except logistic, production), responsible (as supervisor) for course preparation and execution and other trainers

Internal trainers:

6. Sicco Antuma – senior trainer (competence in Dynamics-AX: logistic, production)
7. Jan van Maanen – senior trainer (competence in Dynamics-NAV: installation, configuration; Dynamics-AX: administrator)
8. Berend Otten – senior trainer (competence in Dynamics-AX: development)
9. Marjan Baan – medior trainer (competence in Dynamics-NAV: logistic)
10. George Doorn – medior trainer (competence in Dynamics-AX: introduction, logistic, production)
11. Guido van Gemerden – medior trainer (competence in Business Intelligence, Dynamics-AX: installation, configuration)

Mprise also has several external trainers that are hired when needed. Mirjam is the one who decides to hire external trainers.

Course product development

Since the main business of Mprise is about courses or delivering (executing) courses, the content (material) for each course product must be developed beforehand. The developer i.e. the assigned trainer will be the responsible person ('owner' or can be called as the first responsible trainer (FR-trainer)) for the course product. Thus, he/she will be responsible for maintaining the course product during its existence and also responsible to provide information to promote the course via several marketing channels under Mirjam responsibility. Mirjam assigns the EV-trainer to keep the information about the course on the website up-to-date and to provide relevant information for course product promotion items (e.g. newsletter, direct mailing, presentation/seminar, knowledge session and advertisement product such as leaflet, poster, booklet, etc).

Mostly, the responsible trainer will also be the person who will give the course to the participants though not always. Therefore, if the responsible person leaves Mprise, the

responsibility of the particular course product will be given to other person.

The assignment of a trainer to a development project can vary. The trainer can be someone who has specialty corresponding to the new course material or someone who does not yet have enough knowledge about the material. In the latter case, he/she needs to learn beforehand in order to be able to develop the material for the new course product.

Generally, the content of the course product is bundled in the material which consists of:

1. main document: if the course is for consultant or key user, then the main document is usually downloaded from Microsoft (a book); however if the course is for end users, the main document is then made by Mprise (by its responsible trainer)
2. presentation slides: is made by Mprise (by its responsible trainer)
3. exercise: is mostly made by Mprise (by its responsible trainer)
4. additional material e.g. books needed to complement main material

Besides having a product code relevant to the content, every course product developed by trainers at Mprise also has a product version corresponding to that of Microsoft. The version is attached to a course product since Microsoft continues making improvement and development of its products (Dynamics-AX and Dynamics-NAV).

Joop who is responsible for the innovation observes whether there is a possibility for a new course product that needs to be developed. Thus, sometimes he visits a client to listen to his/her opinion to see if there is a prospect of a new course product. Moreover, Joop can also identify market needs either from the information given by the trainers or from Mirjam. If there is a new course product that needs to be developed he can initiate the development of a new course product. However, a discussion with the trainer of the corresponding specialty is sometimes needed to define the outline of the new course product i.e. the target group, description, preliminary knowledge, content, and duration (maximum five days, preferably one, two or three days). Nevertheless, this task might be done by the assigned trainer him/herself. To assign a trainer to do the development, Joop also needs to see the availability of the resource (the trainer). Thus, he needs to know the trainer's agenda. If the trainer is available Joop will assign the new course product development to the trainer. Next, the trainer will propose to Joop the estimated time for the development. Thus, the development project starts. After the trainer finishes producing the material, a colleague (other trainer) will review the material. If it is good then the assigned trainer will notify Joop that the new course product is ready. Otherwise, he needs to revise the material again.

New course product might also be developed when Microsoft launches new product of Dynamics-AX or Dynamics-NAV. Joop asks the corresponding trainer to develop a new course product. The development for this kind of course product can be straightforwardly assigned to trainer with corresponding specialties without too much discussion since almost all the material is available on Microsoft website. The procedure for this development is the same as the previous one.

Moreover, besides those two kinds of development, there is other trigger for course product development that is especially made due to client specific course project. This material is usually specific to a particular case of the client. Thus, this kind of development is initiated from a client specific course project. The developer (the assigned trainer) is usually involved in the intake process (analysis of client problem) thus he/she will have enough information to prepare the course material.

Sometimes, the development of a new course product can also be done by (delegated to) different trainers. However, the responsibility of the corresponding course product stays on the assigned trainer (the 'owner' of the course product / the FR-trainer).

Basically, the first two kinds of course product development are independent from the course project with the client. However, it can be the case a client specific course project started while the development of the corresponding material has not yet finished. Then, the arrangement for the course date should be made by considering the completion of the course material.

All course products (course materials) are stored in the shared folder where Yvonne has access to them. However, sometimes Yvonne does not print the material for the course from the shared folder since the trainer can also directly send the material to her to make it faster.

Usually, during the development of a new course product, Mirjam who is responsible for marketing will start a marketing action for the new course product.

There are several ways to promote course products i.e. by making direct contacts with clients, via website, newsletter, direct mailing, presentation/seminar, knowledge session by trainers, advertisement products (leaflet, poster, etc), booklet:

- Via the website: Mirjam assigns the responsibility to keep the content information for a course product up-to-date to the FR-trainer. Website is the main resource for marketing the course product.
- Via newsletter: Mirjam is the initiator and the editor of the newsletter. The content of the newsletter can be from Mirjam, Joop, Pieter and FR-trainers. The drafts are then given to Dorien who is responsible to compile and send it to the subscriber and upload it to the website. Any subscription or un-subscription request for the newsletter from an online form on the website will be directed to Dorien's email account and consequently will be followed up by her.
- Via direct mailing: Mirjam sends email containing new course product information to the relevant clients and participants.
- Via presentation/seminar and knowledge session: Mirjam herself or the trainers gives overview of the course products to potential clients.
- Via advertisement products (leaflet, poster): Mprise has a partner named Elevation Concept.
- For booklet: Mirjam assigns Yvonne to compile a new version of booklet. There is already a default design thus changes needed are only the content.

Establishment of standard courses

Mprise offers standard course programs which are valid for six months. The programs contain the description of each standard course. The programs and their schedules can

be found on Mprise website (most updated) and in the booklets of Mprise. Currently the programs are as follows:

1. Dynamics-AX for consultants
2. Dynamics-AX for users (key users / administrator and end users)
3. Dynamics-NAV for consultants
4. Dynamics-NAV for users

Twice a year, Mirjam creates training program and schedules (this year there are four (4) schedules as mentioned above) for the next 6 months. The courses in the training schedule can be either of existing courses from the previous semester or new courses. However, most time the standard courses will use the latest version of corresponding course materials. Mirjam chooses the courses from the available course product and sets date and the corresponding executor of those standard courses. Yvonne will do the rest by assigning the standard course to available classroom at Mprise education center. The standard courses are then offered to the market via several marketing channels mentioned earlier.

For the price Mirjam takes the market price for the standard courses. For standard course, some particular users can have discount (e.g. users who are registered as Dynamic user group – external group, partner discount).

Thus, each training schedule produced by Mirjam will consists of:

1. Course name (e.g. Dynamics-AX Financial 1)
2. Course code (e.g. X-FF1)
3. Duration – days (e.g. 3)
4. Fixed date or open date (e.g. fixed date, February 10-12)

Mirjam also compiles a description (for marketing) of each course which is provided by the corresponding trainer (the FR-trainer) which consists of:

1. Course name
2. Course code
3. Target group
4. Description
5. Prerequisite knowledge
6. Content
7. Exam information (available only if the course is part of the path/track for a specific exam to get certification from Microsoft)
8. Duration
9. Price

Other description that she needs can be acquired from the responsible trainer (the ‘owner’ of the course product). In each program of Dynamics-AX and Dynamics-NAV training for consultants, a diagram/schema on the website containing information of components (trainings and self-study) of certification requirements for Microsoft examination is provided. When a course symbol in the schema is clicked, it will lead to a page with description of the course. Likewise, a schema is also provided for the end users. As mentioned earlier the FR-trainer is responsible to update relevant information on the website according to his/her responsibility.

When Mirjam finishes the draft for both programs and schedule, she will ask someone to update the (the content of the) programs (usually the FR-trainer) and schedule on

the website and give the draft also to Yvonne. Yvonne will then print the booklets and the schedules. Yvonne will complete the tables by assigning the classroom to each course. She will then input into each trainer agenda (Microsoft Outlook) of the corresponding standard courses.

Standard courses

To have a standard training (standard course), a client (e.g. a manager of a company) can go to Mprise website and read all information about the standard training. Two online registration forms are available on the website for the client to register for Dynamics-AX course and Dynamics-NAV course. Besides those online forms, there is other registration form that can be downloaded from the website (Inschrijfformulier cursussen.pdf) thus can be sent by post or by fax to Mprise. In doing its business Mprise has general conditions the client should know. The client can read the general condition by downloading the file AlgemeneVoorwaarden.pdf from the website.

A client can order a standard course by filling a registration form on the website or filling a printed sheet of registration form then send it by post or fax it to Mprise. In most cases, a client uses the online registration form on the website.

When a registration form is submitted by a client through website, an email containing registration information is automatically sent to training@mprise.nl. If the order is received from Monday until Thursday, Yvonne who has access to that email account will further process the order. Otherwise, Dorien who substitutes Yvonne on Friday will handle it.

Since the website and Mprise information system is not yet connected, Yvonne and Dorien should input the information manually into the information system when a course request or other request is received in training@mprise.nl email account or in Dorien's email account. Likewise, the registration form sent by post or fax will be also processed by Yvonne or Dorien.

The registration form contains the following information:

1. course name and code
2. course date
3. name(s) of the participant(s) (along with sex and whether he/she has preliminary knowledge mentioned in the brochure or not)
4. company name
5. client name (surname) and initial of first name
6. email address
7. post address, post code and city
8. telephone

If the client wants the invoice to be sent to other party then he/she needs to fill the following information of the other party:

1. company name
2. name (surname) and initial of first name
3. postal address
4. post code
5. city

The information system used by Mprise is the Microsoft Dynamics-NAV added with add-ons provided by ABECON. The add-ons are for the course administration part while the financial part is from the standard package of Microsoft Dynamics-NAV. Thus, given the registration information, Yvonne will check into Mprise information system the available seats for the requested training. However, the course project for a standard course will be created by Yvonne in the information system only if there is participant request for the course. When a course project is created, Yvonne will also notify the corresponding trainer that the course has participant(s), thus most probably will be executed later on.

The default course capacity according to Mirjam is maximum 10 and minimum 3. However, the number is negotiable; Yvonne can ask Mirjam whether the class can have more than 10 participants. Mostly, Mirjam will forward the question to the corresponding trainer who will give the training. For normal case, if number of participants does not exceed the maximum number, Yvonne will directly compose and send a confirmation email to the client (if the client does not have email address - a very rare case - then Yvonne will send the confirmation by fax). The email will contain the following information:

1. course name and its code
2. date and time
3. name of participants
4. company name and address
5. course location
6. price
7. payment

If the requested course is not possible to be delivered, Yvonne will contact the client (the company) mostly by phone to discuss this matter. If he/she is willing to wait for the next schedule (2 months later) then Yvonne will insert the requested course to the next schedule. Other case, if the client does not want to wait, Yvonne will offer additional course, thus the client will have the client specific training. In this case, Yvonne will forward the request to Mirjam. The procedure for a client specific course is described later.

Besides making a confirmation email, Yvonne also inputs manually the corresponding information (the participant(s), the client (e.g. a manager of a company), the company (as the default debtor), and the debtor if the invoice will be paid by other company) of the requested standard course into the information system. When the course date of a standard course is still more than a week ahead, Yvonne will mark the course as 'inplannen'.

However, after receiving confirmation email sometimes the client wants to cancel or delay the date of the requested training. When a cancelation or a delay request comes Yvonne will look at the following rules (stated in the general condition) to process the request:

1. for delay request:
 - a. if the request comes longer than three (3) weeks before the course date, the client still has to pay 100% of the course fee for the canceled training, but in the next course in which he/she will attend, he/she does not need to pay any fee.

- b. if the request comes within two (2) and three (3) weeks before the course date, the client still has to pay 100% of the course fee for the canceled training, and another extra fee 25% of the course price in the next training.
 - c. if the request comes less than a week before the course date, the client still has to pay 100% of the course price for the canceled training, and another extra fee 50% of the course price in the next training.
2. for cancel request:
 - a. if the request comes longer than three (3) weeks before the course date, the client does not need to pay at all
 - b. if the request comes within two and three weeks before the course date, the client has to pay 50% of the course price
 - c. if the request comes less than a week before the course date, the client has to pay 100% of the course price; this rules also answer the question if a participant does not attend the course without notice or with notice but less than a week.

When the course date is already a week ahead, Yvonne starts to prepare the execution of the training. First she checks the participants of the training. If the participants are less than 3, Yvonne will ask Pieter whether the corresponding course should be delivered or not. If Pieter is not available, she will ask Mirjam instead. If the course is canceled, Yvonne will contact the client(s) to inform this cancellation and also notify the assigned trainer. Otherwise, the course will still be delivered, thus Yvonne marks the course in the information system as 'definitief'.

Then, Yvonne prepares a package of materials for each participant. The package consists of hard copies of:

1. course materials
2. presence list
3. evaluation sheets for participant

One week before course the responsible trainer for corresponding course should have submitted the final material to Yvonne so that she can start printing and binding it for each participant. Sometimes, Yvonne needs to order number of certain books as the part of the course material. Then, she will contact Ibis Hotel to prepare the lunch for the participants during the course days. Furthermore, she also needs to prepare certificates for the participants.

Regarding the ICT infrastructure in the classrooms for the course in Mprise, Mprise has SLA (service level agreement) with Dynaprise. In this case, Jan van Maanen is the contact person from Mprise. Thus, to prepare the course execution, Yvonne will directly contact the system administrator (in this case Dynaprise) to prepare a number of PCs with the software (VPC) as many as the number of participants in a particular classroom. She also gives the name of the trainer so that if the system administrator has questions he/she can directly ask the trainer.

Advisory service and client specific training

When a client wants to have an advisory service or a client specific training, he/she can send request through email to training@mprise.nl or make a phone call to Mprise. Or if he/she has had contact with Mirjam, then he/she can directly contact Mirjam to ask for an offering of specific course(s) or a consultancy for his/her problem.

When someone fills in a contact form on the website and asks a question, an email to training@mprise.nl is sent. Depending on the question; if question is quite simple or about the standard course then Yvonne will directly answer it, if the question is quite complex or it concerns with advisory service and client specific course then Mirjam will take over and give further response.

Mirjam is the one who will handle all requests and make contract proposal for advisory service and client specific training. When the client describes the problem to Mirjam, she can directly categorize the problem into three levels:

- simple case; Mirjam will make a phone call to the client and ask questions necessary to determine the solution; this can be called simple intake
- middle case; Mirjam creates a checklist to acquire information needed and sends to the client, then based on the answers a solution offered by Mirjam is made; this can be called middle intake
- complex case, Mirjam will propose an intake solution (advisory service) for the problem, later on other solution concerning the training can be offered based on the intake; this will be called complex intake

More detailed information for the three levels above is as follows:

In the intake, Mirjam will do inventory checking of the company to collect the information needed and to analyze what the company needs, in terms of training (of Dynamics-AX or Dynamics-NAV). In this step, Mirjam will sometimes need communication with Pieter and other trainers to discuss about the solution for the company problem and the possibility to deliver the content of the proposed training for the company. For the simple and middle case, Mirjam and/or the corresponding trainer do not need to visit the company to analyze the problem in order to determine the appropriate solution (i.e. what the company needs). She and/or the corresponding trainer can directly determine the solution for the problem and describe kinds of training and duration for the proposed training. However, for the complex case, she and/or the corresponding trainer will need to visit the company. In this case, Mirjam decides to propose a separate contract only for the intake process. This contract is referred to as advisory service contract. However, it can be the case a client requests for an advisory service only, without any initial intention to have any training afterwards. Before going to the client, the proposed solution i.e. the advisory service contract should be first signed by the client. The detail arrangement for the contract is explained later. Thus, after the intake process, Mirjam will propose a solution in a form of a client specific course contract to the client. If the client signs the contract and previously there is an advisory service contract, Mirjam will give discount for the training price 50% of the advisory service fee.

Both for advisory service and client specific course solution, Mirjam needs to determine the executor of the solution. Since she knows the area of competency of each trainer she can assign the trainer to deliver activity in the proposed solution.

For the price of client specific course(s) and advisory services, Mirjam has some guidance. There is a daily rate for an advisory service and also for a course with maximum three participants. For the training, if the number of participants is greater than three then extra fees are applied per additional participant. However, if course consists of many days, the client might ask some reductions.

In the proposed solution (contract proposal), it is preferable not to specify the date to make the offer flexible so that the client can sign the contract without worrying the date. However, if necessary the date is set by Yvonne, so she needs to take a look at the trainers' agenda and if needed Yvonne makes a phone call to negotiate the date with the client.

For all contract drafts Mirjam made, she will give it to Yvonne. The contract draft contains the name of the executor (e.g. the trainer) of the project. For the advisory service contract, Yvonne can just send directly to the client and wait for the reply. If the client signs, scans and sends it back to Yvonne by email, Yvonne will then record (create) the advisory service project into the information system and notify Mirjam or the corresponding trainer so that they know there is task for him/her. Dorien will then generate the invoice (on Friday) after the project is completed.

For the training contract, if the training solution is simple, Yvonne will see the agenda (Microsoft Outlook) of the corresponding trainer(s) and then set a date for the course on the contract and send it by email to the client. However, if the contract is complex, she can just directly send the contract by email and discusses the date later. If the client agrees, he/she will sign, scan and send it back to Yvonne. Cancellation for client specific course is not possible after the contract is signed.

For certain reason, the client can request to change the date of the course (i.e. delay the training). Mprise has the general condition for the delay as follows:

1. if the request comes longer than three (3) weeks before the course date, the client does not need to pay any extra fee
2. if the request comes within two and three weeks before the course date, the client has to pay an extra 25% of the course price
3. if the request comes less than a week before the course date, the client has to pay an extra 50% of the course price

However, according to Yvonne there has not been any case for change request for client specific training.

Thus, after a contract of client specific course is signed, Yvonne will insert the course(s) information (create a course project) into the information system (Microsoft Navision):

1. project number
2. course date (if there is more than one date, the date entered is the first date)
3. company name
4. client name (surname) and initial of first name
5. email address
6. post address, post code and city
7. telephone
8. location

If the invoice is to be sent to other party then Yvonne will insert the following information of the other party:

1. company name
2. name (surname) and initial of first name
3. postal address
4. post code
5. city

Yvonne also notifies the corresponding trainer's agenda (Microsoft Outlook) of the planned training. She gives the course the status 'definitief'.

Thus, after the contract is signed and inserted into the information system, the assigned trainer (the executor) will prepare the material. Since usually the executor of the course has involved during the intake phase, he/she would have had information about the training. However, Mirjam can also give additional information via email to the trainer.

The material for client specific course can be directly taken from the available standard course or just a combination with small modification of or addition to the material of standard training. However, it can be the case that the course material for a client specific course should be made specific only to a particular case. Thus, the assigned trainer should carry out the course material development beforehand.

The location of client specific course can be at client's site or at Mprise. If the location of the course is at Mprise, then Yvonne needs to also ensure about the facilities and to order the lunch to Ibis Hotel (a week before the course date), just like in the standard course course preparation. If it is at client's site Yvonne does not need to prepare the facilities and the lunch. Those facilities and lunch are taken care of by the client (e.g. contact person in the company). However, it is necessary for the trainer to have a communication with the contact person so that the trainer can make sure that the preparation is sufficient.

One week before training, the responsible trainer (a senior trainer) for corresponding course should have submitted the final material to Yvonne so that she can start printing and binding it for each participant. Sometimes, Yvonne needs to order a number of certain books as part of the course material. She also prepares course certificate for each participant which will be given at the end of the course by the trainers.

Course Execution

During the course, the partner who provides ICT support monitors the course to ensure the ICT support provision. At the end of the course, the trainer distributes evaluation sheets to the participants. He/she also has his/her own evaluation sheet over the course. A trainer should fill an evaluation sheet about the course including further follow up if needed. The participants are supposed to give evaluation over the trainer and the course.

Course Evaluation

The evaluation sheet is then submitted to Mirjam. Likewise, the evaluation sheets from the participants are also submitted to Mirjam. If the score for the course or the trainer from the evaluation sheets from the participants are low, Mirjam takes initiatives to contact the client to clarify and found out what went wrong and thus tries to fix it so that the relationship with the client is still good. This is done to ensure the client is happy about the project.

After the course has finished, usually there is a follow up to ask any progress or status of the post-course condition. Sometimes, it is the trainer himself who makes contact (usually a phone call) with the client. This activity is known as 'guarantee' meaning

Mprise cares for the post condition of the course and if necessary a follow up course for the client can be initiated when the client wants to have the benefit of the follow up training.

Other than the evaluation sheets, the participants also need to submit the presence list. In the presence list, there is an option to subscribe for a newsletter. If the participant wants the newsletter, Yvonne will turn on the automatic newsletter option in the information system so that the participant will receive newsletter automatically. Otherwise, she turns it off. Dorien will export all subscribers' emails and send them the newsletter when a newsletter is released.

It can be the case that a participant loses his/her course certificate. Thus, he/she can request a certificate by sending a request to Yvonne. She will look into the presence list to check the request and if it is valid i.e. the requester was truly the participant of the corresponding course she will prepare and send the requested certificate.

Payment

Every Friday, Dorien will print invoices for all projects executed within that week, the standard course and client specific course and also advisory service projects. Dorien looks into the information system of the training/project with the status 'definitief' and passed execution date. Then, she generates the invoice for the corresponding project. She calculates the tax and added into the invoice. Some clients who are members of Dynamics User Group (of Dynamics-AX) will get 10% discount for every training. For the client specific training, she also needs to check the price based on the file given by Mirjam. Thus, an invoice will contain the following information:

1. client name and address (debtor, the one who is responsible to pay the training)
2. number/id of the client (debtor number)
3. invoice number
4. invoice date
5. description: course name, starting date, duration, participant, discount (if applicable), etc
6. amount
7. tax
8. total
9. additional information for the payments (i.e. the invoice should be paid no longer than 14 days after the invoice date, number/id of client and invoice number should be mentioned in the bank transfer)
10. Mprise bank account, IBAN, BIC

When an invoice for a particular project has been printed, Dorien will set the status of the training/project in the information system as 'afgerond'. Then, all the invoices for that week are sent by post.

Then, every month when the account statement from the bank comes, Dorien enters the information of the money transferred into Mprise's bank account into the information system. Thus, the information system can just generate a report of clients who have not paid the invoices. Then, depending on the client's relationship with Mprise, either Mirjam or Dorien will remind him/her to pay e.g. first by letter then directly contact him/her by phone.

Appendix E – The DEMO models of Mprise [Sandhyaduhita, 2009]

The Interaction Model (ATD +TRT) of Mprise

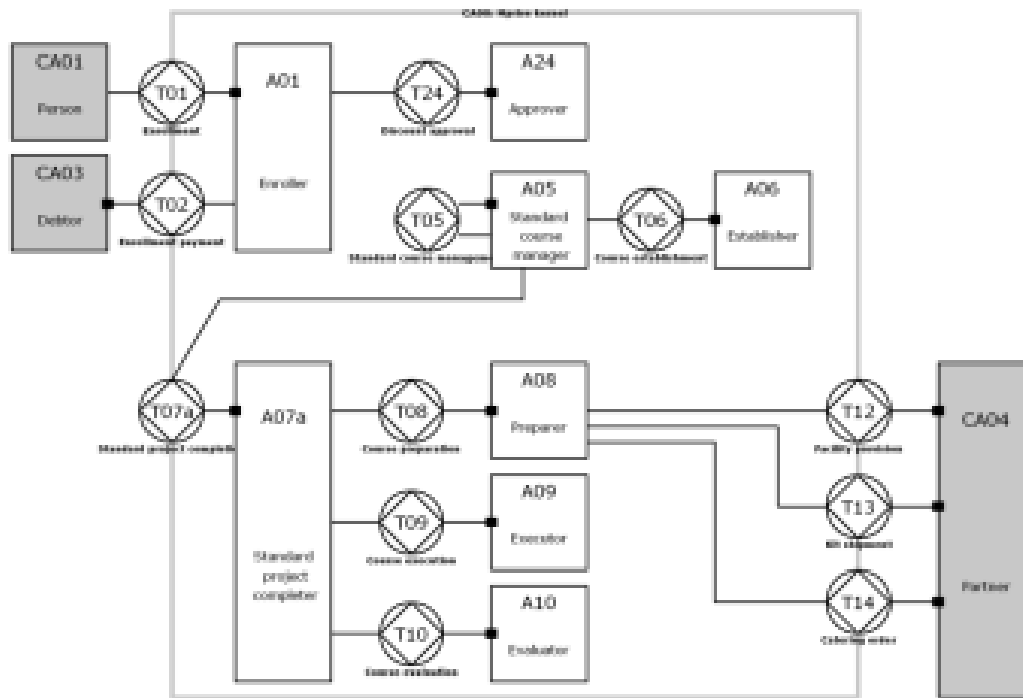


Figure 70 - ATD for the standard course

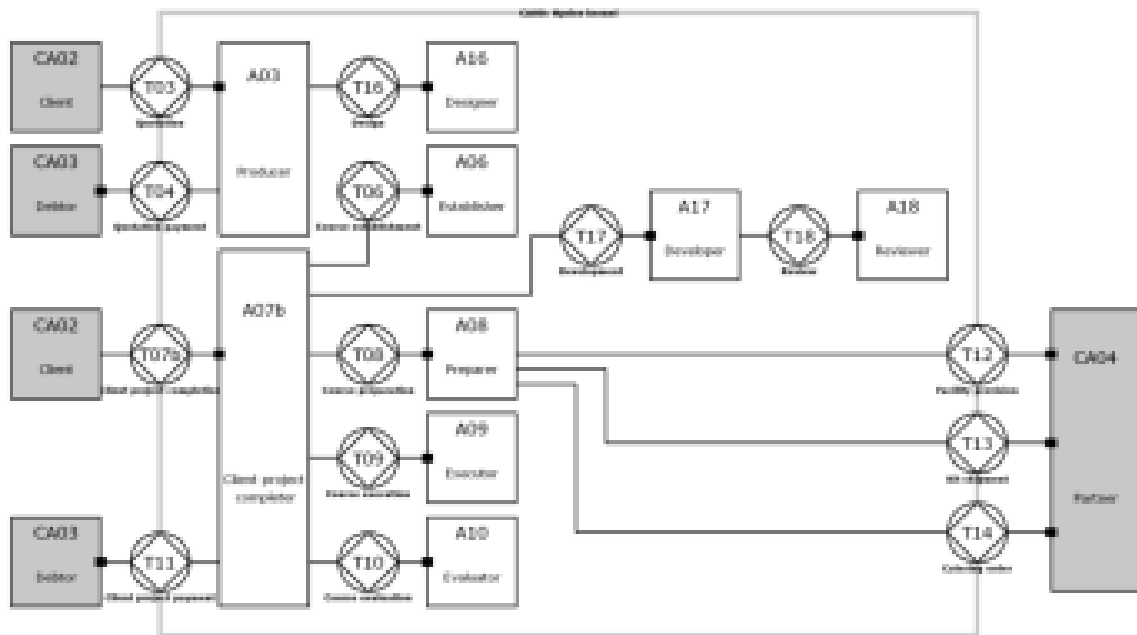


Figure 71 - ATD for client specific course

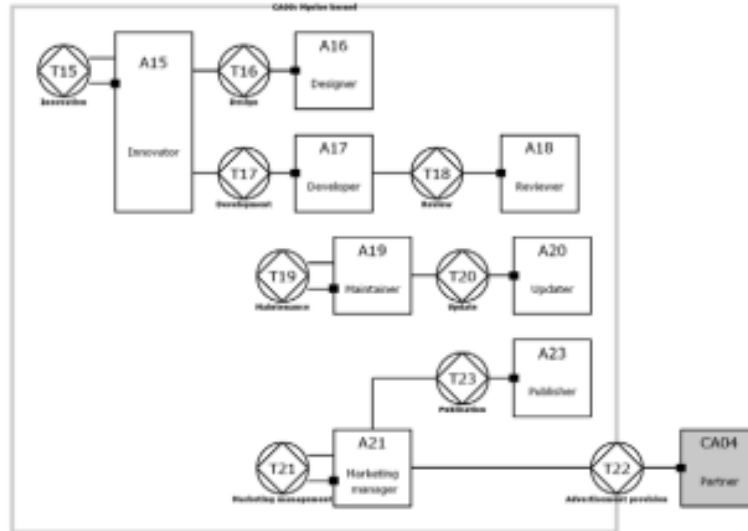


Figure 72 - ATD for development and marketing

transaction type	result type
T01 Enrollment	Enrollment E has been performed
T02 Enrollment payment	Enrollment E has been paid
T03 Quotation	Client specific course kind K has been produced
T04 Quotation payment	Client specific course kind K has been paid
T05 Standard course management	Standard courses for period P have been managed
T06 Course establishment	Course C has been established (a.k.a. planned, with date, trainer and/or classroom)
T07a Standard project completion	Standard project SP has been completed
T07b Client project completion	Client project CP has been completed
T08 Course preparation	Course C has been prepared
T09 Course execution	Course C has been given
T10 Course evaluation	Course C has been evaluated
T11 Client project payment	Client project CP has been paid
T12 Facility provision	Facility for course C has been provided
T13 Kit shipment	Kit shipment for course C has been provided
T14 Catering order	Catering for course C has been ordered
T15 Innovation	Course kinds innovation for period P has been performed
T16 Design	Course kind CK has been designed
T17 Development	Course kind CK has been developed
T18 Review	Course kind CK has been reviewed
T19 Maintenance	Course kinds maintenance for period P has been performed
T20 Update	Course kind CK has been updated
T21 Marketing management	Marketing for period P has been managed
T22 Advertisement provision	Advertisement for period P has been provided
T23 Publication	Course kind CK has been published
T24 Discount approval	Discounted fee for E has been approved

Figure 73 - TRT of ATD of Mprise

The Process Model of Mprise

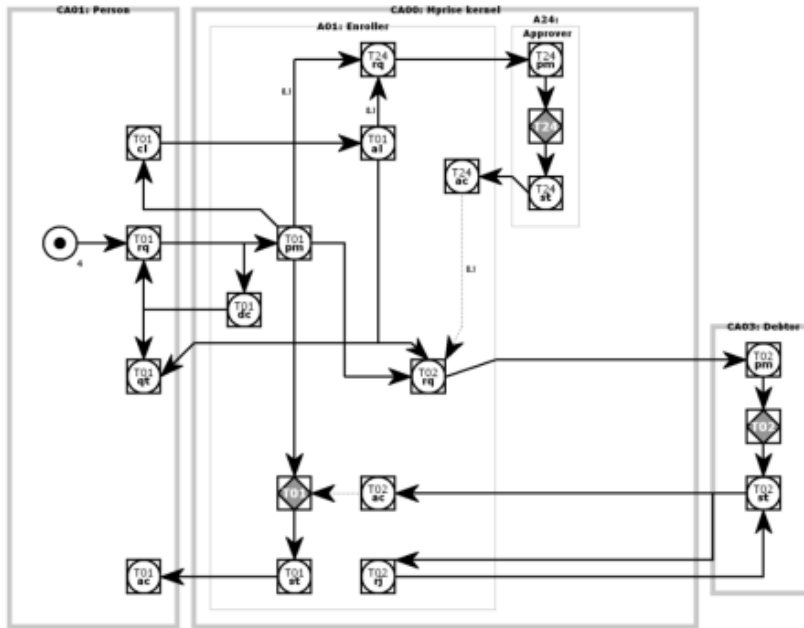


Figure 74 - PSD for the enrollment process

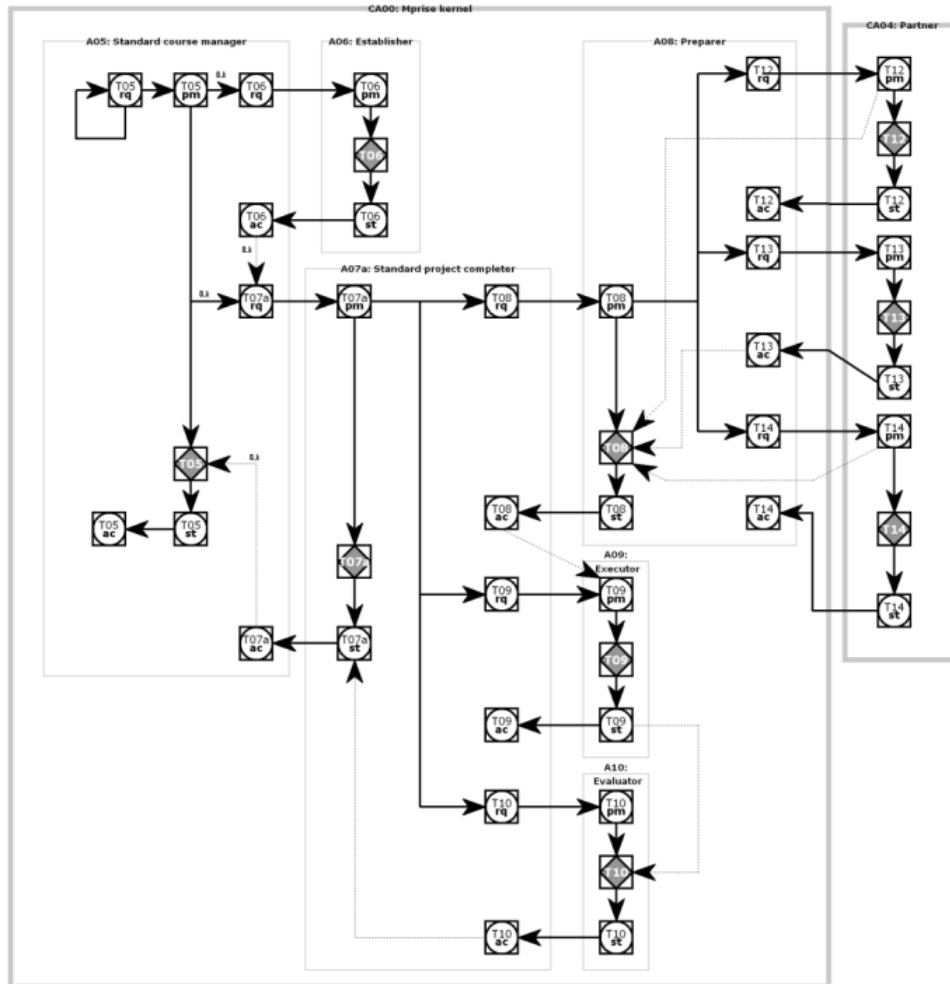


Figure 75 - PSD of the Standard Operation Process

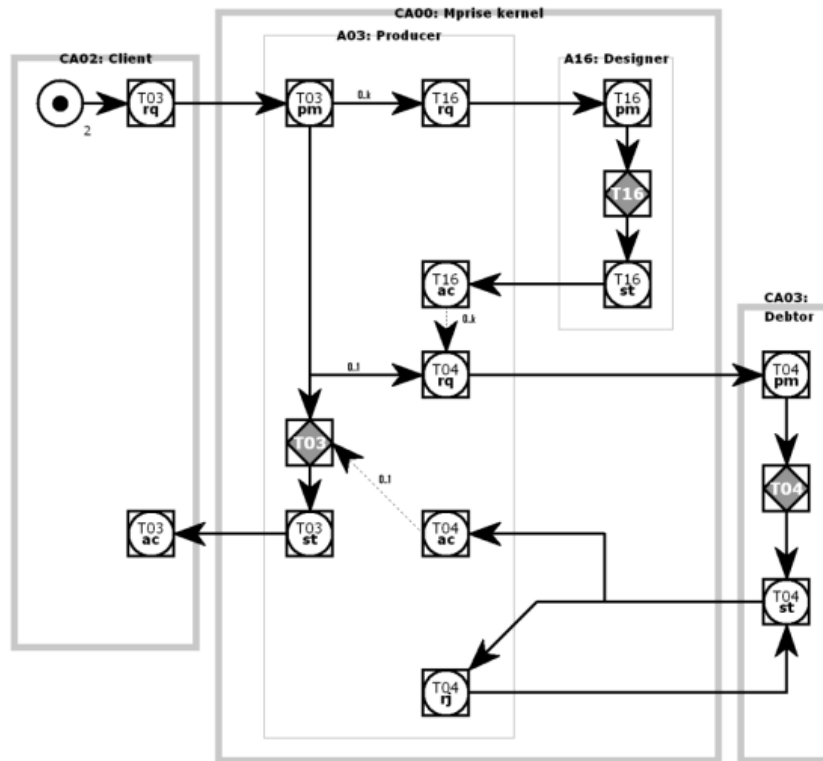


Figure 76 - PSD for the Client-Course-Kind Production Process

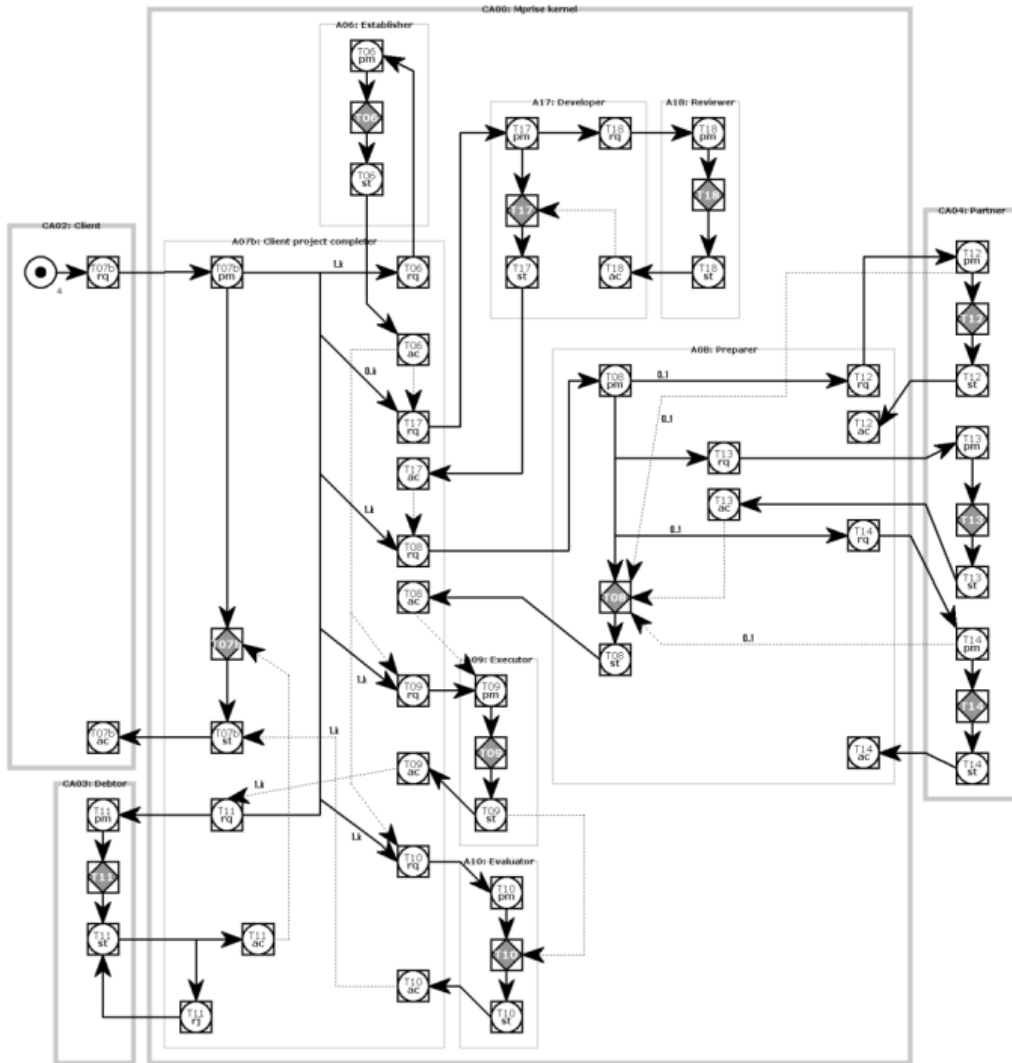


Figure 77 - PSD for the Client Specific Course Operation

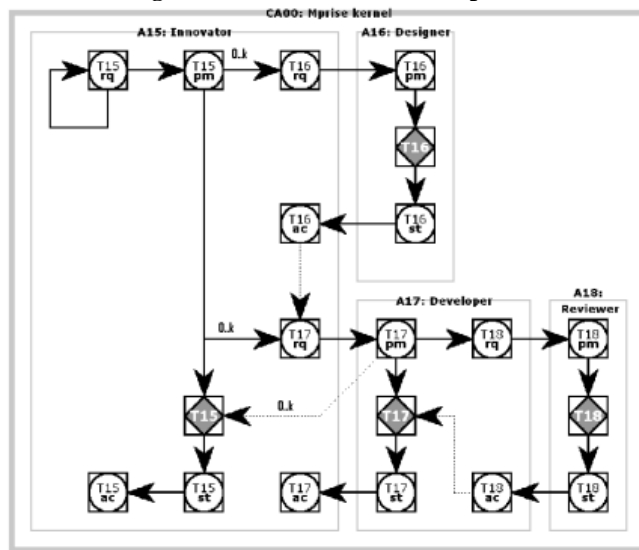


Figure 78 - PSD for the innovation process

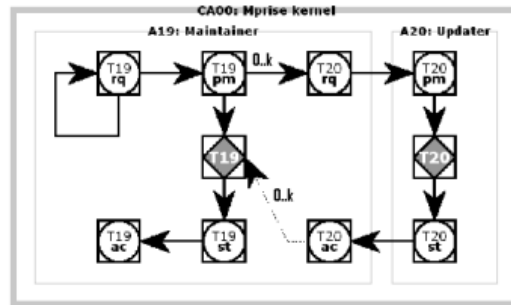


Figure 79 - PSD for the Maintenance Process

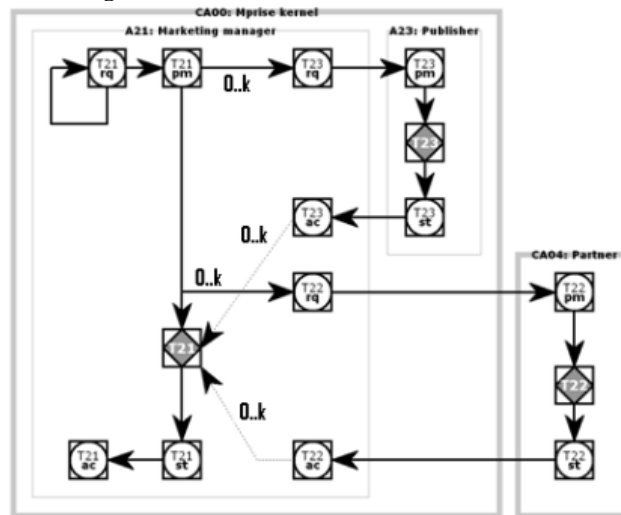


Figure 80 - PSD for the Marketing Process

The State Model of Mprise

The Interstricion Model of Mprise

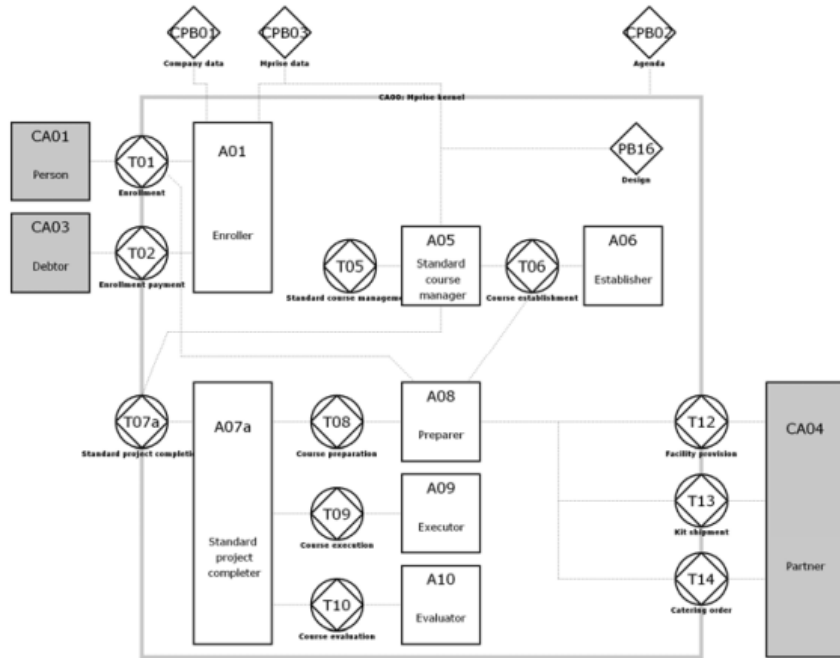


Figure 82 - ABD of the Standard Course

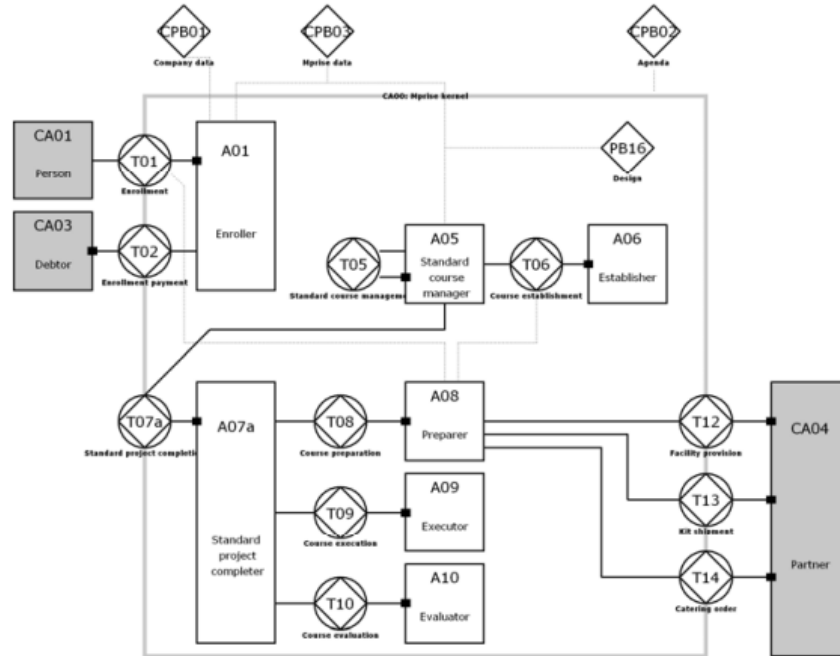


Figure 83 - OCD of the Standard Course

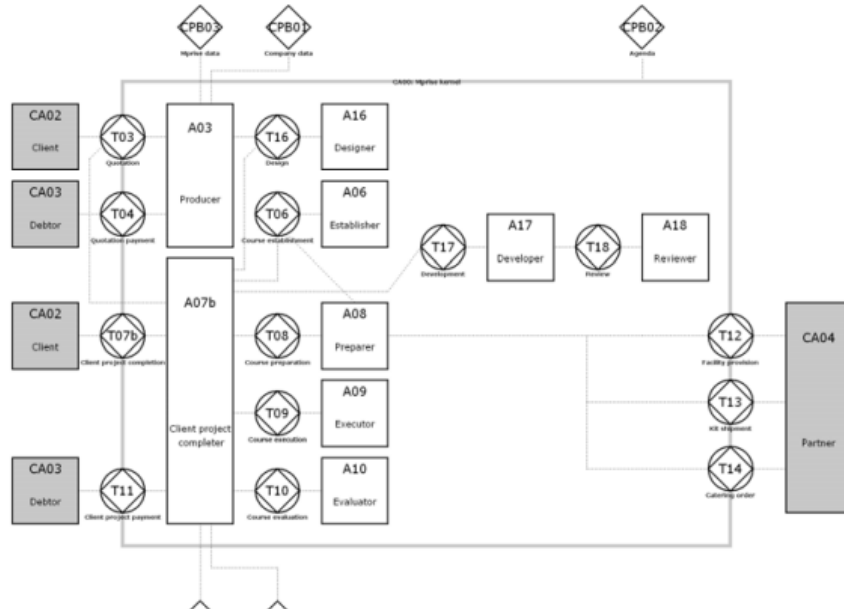


Figure 84 - ABD of the Client Specific Course

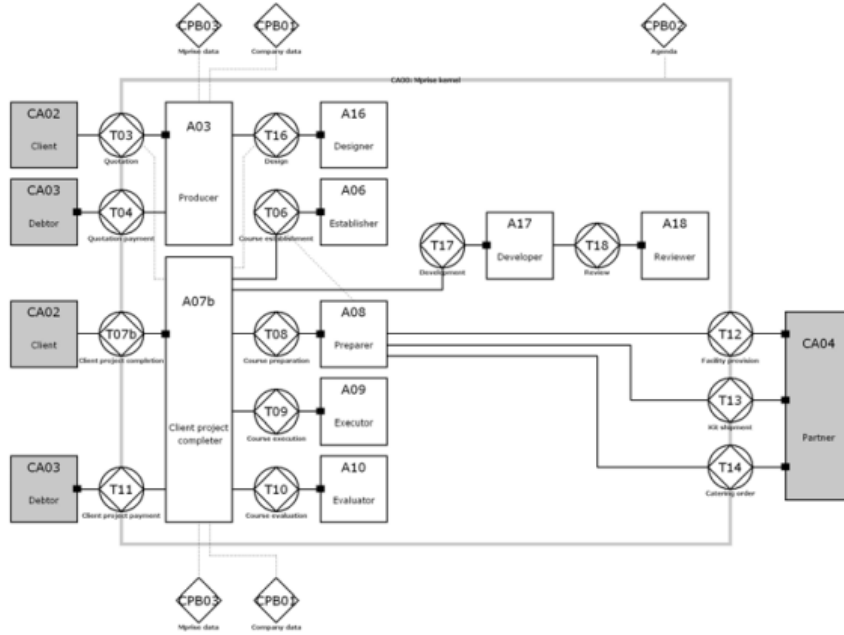


Figure 85 - OCD of the Client Specific Course

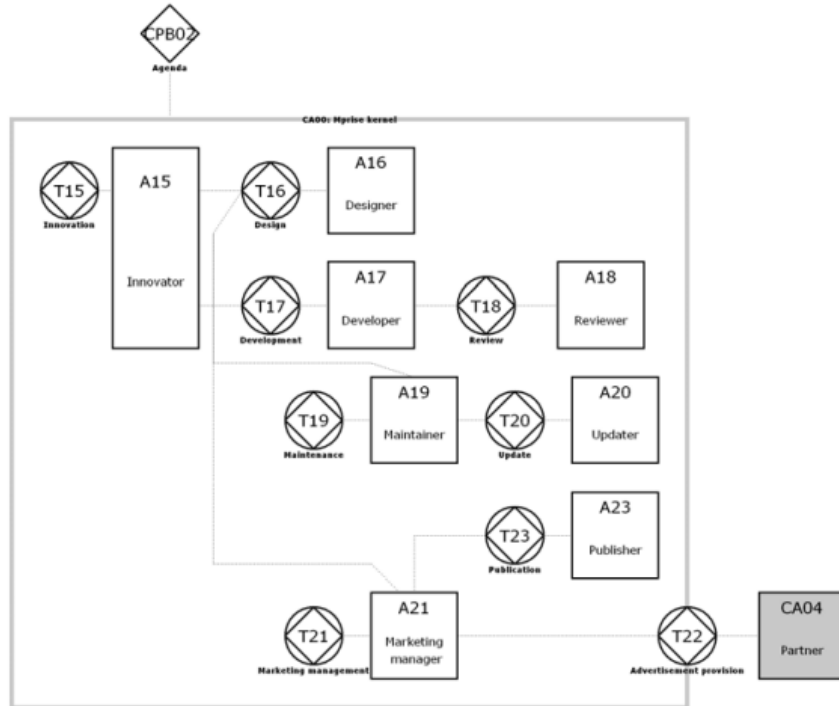


Figure 86 - ABD of the Development and Marketing

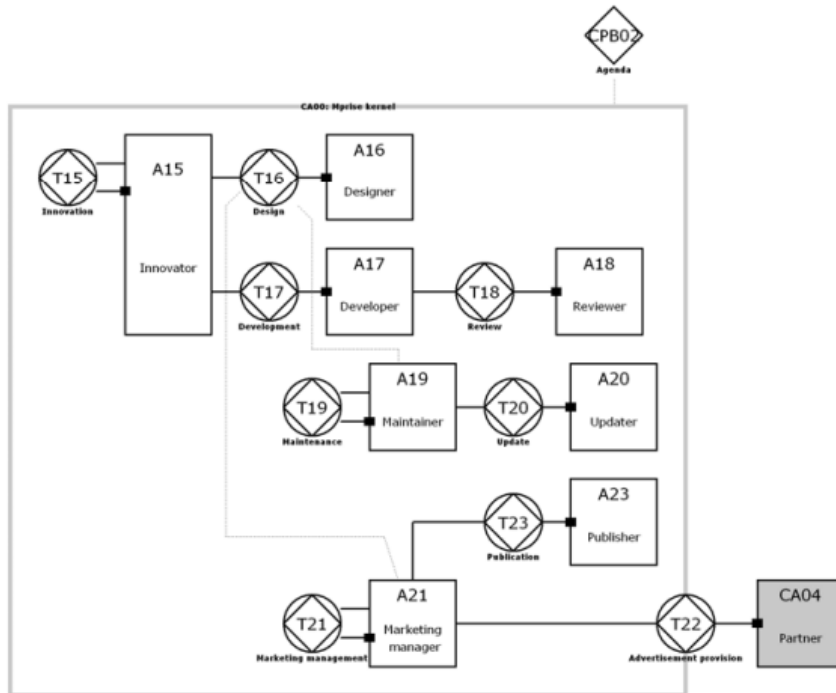


Figure 87 - OCD for the Development and Marketing

Abbreviations

ABD	- Actor Bank Diagram
AM	- Action Model
ARS	- Action Rules Specification
ATD	- Actor Transaction Diagram
B-actor	- Business actor
B-application	- Business application
BCT	- Bank Content Table
B-organization	- Business organization
B-transaction	- Business transaction
BUC	- Business use case
BW	- Business Worker
BE	- Business Entity
BUR	- Business Use Case Realization
C-act	- Coordination act
C-fact	- Coordination fact
C-world	- Coordination world
D-actor	- Document actor
D-application	- Document application
DEMO	- Design and Engineering Methodology for Organizations
D-organization	- Document organization
D-transaction	- Document transaction
GSDP	- Generic System Development Process
I-actor	- Intellect actor
IAM	- Interaction Model
I-application	- Intellect application
I-ATD	- I-organization Actor Transaction Diagram
ICT	- Information and Communication Technology
ISM	- Interstriction Model
I-organization	- Intellect organization
I-transaction	- Intellect transaction
IUT	- Information Use Table
OCD	- Organization Construction Diagram
OER	- Order-Execute-Result
OFD	- Object Fact Diagram
OPL	- Object Property List
OS	- Object System
P-act	- Production act
P-fact	- Production fact
PM	- Process Model
PSD	- Process Structure Diagram
RUP	- Rational Unified Process
SDP	- System Development Process
SM	- State Model
TRT	- Transaction Result Table
UML	- Unified Modeling Language
US	- Using System

References

- Ambler S.W., Nalbone J, Vizdos M.J. (2005). *The Enterprise Unified Process: Extending the Rational Unified Process*, Prentice Hall, 2005
- Arlow J., Neustadt I. (2005). *UML 2 and the Unified Process: Practical Object Oriented Analysis and Design 2nd Edition*. Addison-Wesley Professional 2005
- Baker B. (2001). *Business Modeling with UML: The Light at the End of the Tunnel*. The Rational Edge 2001
- Behrens T. (2004). *Capturing business requirements using use cases: Seven principles that can make a difference*. The Rational Edge, 2004
- Bertalanffy, L. von. (1976). *General System Theory: Foundations, Development, Applications*. New York, George Braziller, 1976
- Boehm B., (1988). *A Spiral Model of Software Development and Enhancement*. ACM Computer, Vol. 21, Issue 5 (p. 61-72), 1988
- Booch G. Jacobson I. and Rumbaugh J. (1998). *The Unified Modeling Language User Guide*. Addison Wesley Longman 1998
- Boulding, K. E. (1956). *General Systems Theory - The skeleton of science*. Management Science, Vol. 2
- Bloch J. (2007). *How to Design a Good API and why it Matters*. Google Talks, Jan. 24, 2007
- Bronts G.H.W.M., Brouwer S.J., Martens C.L.J. and Proper H.A. (1995). *A Unifying Object Role Modelling Approach*. Information Systems, 20(3), 213–235, 1995
- Checkland P. (1993). *Systems Thinking, Systems Practice*. Willey and Sons, 2003.
- De Jong J., Dietz J.L.G. (2010). *Understanding the realization of organizations*. CIAO Workshop, 2010
- Dietz J.L.G. (2003). *Deriving Use Cases from Business Process Models*. In Conceptual Modeling - ER 2003 (pp. 131-143). Springer Berlin / Heidelberg, 2003
- Dietz J.L.G. (2006). *Enterprise Ontology - Theory and Methodology*. Springer-Verlag Heidelberg, 2006
- Dietz J.L.G. (2008). *Architecture – Building strategy into design*. Sdu Uitgevers, 2008
- Dietz J.L.G. (2009). *Is it φτψ or bullshit?*. TU Delft, 2009
- Eriksson H.E., Penker M. (2000). *Business Modeling with UML: Business Patterns at Work*. John Wiley & Sons, 2000
- Fowler M. (1997). *UML Distilled: Applying the standard object modeling language 2nd Edition*. Addison-Wesley Longman, 1997

References

- Fowler M. (2004), *UML Distilled: Applying the standard object modeling language 3rd Edition*. Addison-Wesley Longman, 2004
- Flood R. L. (1990). *Liberating Systems Theory*. Plenum Press New York, 1990
- Flood R. L., Jackson M. C. (1991). *Creative Problem Solving: Total Systems Intervention*. Wiley & Sons, 1991
- Flood R. L., Carson E. R. (1993). *Dealing with complexity: an introduction to the theory and application of systems science*. Springer, 1993
- Jackson M. C., (1990), *Beyond a System of Systems Methodologies*. The Journal of the Operational Research Society, Vol. 41, No. 8, August 1990, (pp. 657-668)
- Jackson M. C., Keys P. (1984), *Towards a System of Systems Methodologies*. The Journal of the Operational Research Society, Vol. 35, No. 6, June 1984 (pp. 473-486)
- Johnston S. (2004). *Rational UML Profile for business modeling*. The Rational Edge 2004
- Jordan N. (1960). *Some Thinking about "System"*. The RAND Corporation, 1960
- Heumann J. (2001). *Introduction to Business Modeling using the Unified Modeling Language (UML)*. The Rational Edge, 2001
- Humphrey W.S. (1989). *The software engineering process: definition and scope*. ACM SIGSOFT Software Engineering Notes, Volume 14 Issue 4, June 1989
- Georgiou I. (2003). *The Idea of Emergent Property*. The Journal of the Operational Research Society Vol. 54. Palgrave Macmillan Journals 2003
- Guarino N. (1998), *Formal Ontology and Information Systems*. Proceedings of FOIS 1998, Trento Italy, June 1998 (pp. 3-15)
- Klir G.L., Elias D. (2003). *Architecture of systems problem solving*. Springer, 2003
- Kroll P., Kruchten P. (2003). *The Rational Unified Process Made Easy, A Practitioners Guide to the RUP*. Addison Wesley Longman 2003
- Kruchten P. (1995). *The 4+1 View Model of Architecture*. IEEE Software November 1995
- Kruchten P. (2000). *The Rational Unified Process, An Introduction Second Edition*. Addison Wesley Longman, 2000
- Kruchten P. (2004). *The Rational Unified Process, An Introduction 3rd Edition*. Addison Wesley Longman, 2004
- Laudan L. (1986). *Methodology's Prospects*. Proceedings of the Biennial Meeting of the Philosophy of Science Association, Vol. 2, 1986

References

- Maier M.W., Rechtin E. (2002). *The Art of Systems Architecting*. Boca Raton, CRC Press 2002
- Mallens, P., Dietz, J., & Hommes, B. (2001). The Value of Business Process Modeling with DEMO Prior to Information Systems Modeling with UML. EMMSAD 2001
- Mingers J., Brocklesby J. (1997). *Multi-methodology: Towards a framework for mixing methodologies*. Omega vol 25(5), 1997
- Ng P. W. (2002). *Effective Business Modeling with UML: Describing Business Use Cases and Realization*. The Rational Edge, 2002
- Oestereich B. (2004). *Developing Software with UML: Object oriented analysis and design in practice 2nd Edition*. Addison Wesley Longman, 2004
- Object Management Group [OMG] (2009). *OMG Unified Modeling Language, Superstructure Version 2.2*. OMG Press 2009
(<http://www.omg.org/technology/documents/formal/uml.htm>)
- Object Management Group [OMGb] (2009). *OMG Unified Modeling Language, Infrastructure Version 2.2*. OMG Press 2009
(<http://www.omg.org/technology/documents/formal/uml.htm>)
- Patel D., Cesare S., Lycett M. (2002). *Business Modelling With Uml: Distilling Directions For Future Research*. International Conference on Enterprise Information Systems Journal, 2002
- Passing J. (2004). *Requirements Engineering in the Rational Unified Process*. 2004
- Rumbaugh J., Jacobson I., Booch G. (1999). *The Unified Modeling Language Reference Manual*. Addison Wesley Longman 1999
- Rumbaugh J., Jacobson I., Booch G. (2005). *The Unified Modeling Language Reference Manual 2nd Edition*. Addison Wesley Longman 2005
- Rational (2001). *Rational Unified Process, Best Practices for Software Development Teams*. Rational Corp. 2001
- Rational (2003). *Rational Unified Process Version 2003.06.15*. Rational Corp. 2003
- Sandhyaduhita P. I. (2009). *Extending the DEMO methodology for determining information systems requirements*. Master's Thesis in Computer Science. TU Delft, Mprise B.V. 2009
- Sattari Khavas S. (2010). *The Adoption of DEMO in practice*. Master's Thesis in Computer Science. TU Delft, 2010
- Seligmann P.S., Wijers G.M., Sol H.G. (1989). *Analysing the structure of IS methodologies*. Proceedings of the 1st Dutch Conference on Information Systems, 1989

References

Shishkov B., Dietz J.L.G. (2004). *Deriving Use Cases from Business Processes; The advantages of DEMO*. Enterprise Information Systems V. Kluwer Academic Publishers. Netherlands, 2004

Smith B. (2003). *Ontology, chapter from Blackwell Guide to the Philosophy of Computing and Information*. Oxford Blackwell, 2003

Spirkin A. (1983). *Dialectical Materialism*. Progress Publishers. 1983

Tanenbaum, Andrew (2001). *Modern Operating Systems 2nd Edition*. Prentice Hall, March, 2001

Wijers G.M. (1991). *Modelling support in information systems development*. Thesis Publishers Amsterdam, The Netherlands, 1991.