BUSINESS PROCESS-BASED REQUIREMENTS SPECIFICATION AND OBJECT-ORIENTED CONCEPTUAL MODELLING OF INFORMATION SYSTEMS

Jose Luis de la Vara González

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

Advisor: Dr. Juan Sánchez Díaz

July 2011

Reviewers

Björn Regnell (Lunds Universitet, Sweden)

Jaelson Castro (Universidade Federal de Pernambuco, Brazil)

Paolo Giorgini (Università degli Studi di Trento, Italy)

Evaluation committee

Óscar Pastor (Universidad Politécnica de Valencia, Spain)

Xavier Franch (Universitat Politècnica de Catalunya, Spain)

Paolo Giorgini (Università degli Studi di Trento, Italy)

João Araújo (Universidade Nova de Lisboa, Portugal)

Nelly Condori (Universiteit Twente, Netherlands)

Arantza Illarramendi (Universidad del País Vasco, Spain) (subs.)

José Hilario Canós (Universidad Politécnica de Valencia, Spain) (subs.)

To my parents and siblings,
for their unconditional support and encouragement.
And to my nephew,
for the immense happiness that has brought to our lives.

Preface

"The only place where success comes before work is in the dictionary"

Donald Kendall

Even though it may sound pretentious, this dissertation represents much more that a common PhD thesis. It is a summary of around six years of work and research on development of a business process-based requirements engineering approach for OO-Method.

The subject initially seemed straightforward, but it turned not to be so. Much work has been necessary until defining the current state of the main contribution of this thesis: a methodological approach for business process-based requirements specification and object-oriented conceptual modelling of information systems.

The methodological approach has been modified, tuned, changed again, extended… several times, maybe too many times. Some times to mitigate some weakness, and others to take advantage of possible improvements. Some times a step forward was taken, and later two steps backward had to be taken. It has not been an easy journey.

As a result of the evolution of the work on the target subject, there exist a clear relation between the final project of my 5-year degree (Derivation of requirements models from organizational models), my MSc thesis (Requirements elicitation for information systems from business processes and goals) and this PhD thesis. Several mechanisms and guidance have been proposed and evaluated for the last six years in order to develop a business process-based requirements engineering approach for OO-Method, and some of them have been discarded in the journey.

Anyway, the lessons learned from the "mistakes" made until finishing this thesis have been very important to increase the quality and soundness of the current methodological approach, as well as to find the research path that I would like to follow.

Furthermore, research on business process modelling and on requirements engineering has evolved for the last six years, thus the methodological approach has also been affected by such evolution. In this sense, works from other researchers have influenced on the development of the methodological approach of the thesis, which has adopted and adapted ideas, mechanisms and principles from other works when necessary and possible.

The methodological approach presented must not be regarded as a finished and stable work. For sure it will continue evolving as new challenges and needs are found in business process-based requirements specification and object-oriented conceptual modelling of information systems.

Lastly, although developing and finishing a PhD thesis has been very hard sometimes, my family background on farming (including business processes) and effort to achieve difficult objectives has made me realize that I should not complain about such hardness. I just have to think about the work that many people (e.g., my grandparents) have and have had to perform to earn a living, especially in uneasy social and economic situations.

Valencia, May 2011 Jose Luis de la Vara González

Acknowledgements

Becoming PhD is a very long journey. Nonetheless, as any long journey, it will not be tiresome if you find the adequate people with whom to share the journey. In my case, I have found many people for the last five years (since I finished my degree in computer engineering) that have made this journey not only possible or more bearable, but also fun and exciting. I knew many of them before starting the journey, and others have appeared unexpectedly and fortunately.

The main guilty of this thesis is Juan Sánchez. If someone should be punished for the thesis, that is him. He gave the opportunity to join his research team in my last degree year and has guided me wisely (I hope) to the goal that this thesis represents. The thesis would have never been possible without his advice, lessons, support, trust, friendship and eventual patience.

Probably the main accomplice of Juan's crime is Óscar Pastor. His enthusiasm and passion have soaked into many colleagues and as well as into me, as if the "MDD umbrella" was not effective to cover people. His work makes the members of ProS feel proud to be part of it.

Furthermore, the crime has not only taken place in Spain. People from other countries have played a relevant role in this thesis. I have to thank Professors Björn Regnell and Paolo Giorgini for giving me the chance to stay in Lund University and University of Trento, respectively, during the development of this thesis. The experience and knowledge that I gained in Sweden and Italy have definitely had a very positive impact on my growth as a researcher and as a person.

The above people can be regarded as my research "bosses" during the last five years, thus their influence on this thesis is direct and evident. Nonetheless, becoming PhD is not only researching but also living and sharing experiences with other people. Therefore, I have to thank more people for their usually implicit and sometimes explicit support. Enumeration of all the "non-boss" people that have helped me to develop

this thesis would require writing another dissertation, and I really do not feel like it. I apologize if someone thinks that should be here but is not.

First of all (at least for closeness reasons), I have to thank all the colleagues in the lab104 of DSIC: David Anes, David Melo, Diane, Emanuel, Ignacio, Inés, Kevin, Luis, Marce, Mariam, Nathalie, Nelly, Paco, Raúl, Sergio, Urko and Yeshica. They have had to stay beside me much time, what can be difficult sometimes, and somehow I consider them co-advisors in many aspects of my research and of my life. I especially have to thank the rest of *Juanetes* for their work on development of tool support for the thesis, Ignacio for translating the abstract into *Valenciano*, and Paco for his help with MS Word.

At ProS, more colleagues have been important to develop this thesis. I have to express my thanks to Bea, Fani and Giovanni, without whom doctorate would have been much less fun and interesting. Clara, Isma, María, Mario, Miriam, Nacho, Pablo and Salva have been an important support to finish the thesis. It has been less hard (laughs have been usual at job), and Miriam and Nacho also advised me about tool support. The rest of current and former colleagues at ProS have had a direct or indirect impact on the thesis too. I thank them all for their comradeship.

I have also got to know many people in and from places different to Valencia and who have helped me to develop and finish this thesis, most of the times in non-research-related ways. These people appeared, for instance, in Trento (Amit, Fabiano, Michele and Raian) and Lund (Annika, Carl Johan, Christian, Emma, Flavius, Jesper, Kim, Krzysztof, Lars, Lhinn, Markus, Per, Raquel, Richard and the rest of members of *Datavetenskap*). Others have collaborated with me (Agnes, Alexandre, Claudia, Marcos and Michel).

Outside the work environment, I have to thank all my relatives and friends from several places (Albacete, Madrid, Valencia…). The time that I have spent with them has been important and necessary for me to relax, stop development of the thesis for a while and resume it more willingly. *No sois los mejores amigos, pero sois los que tengo.*

Finally, I am very grateful to the company CARE Technologies and the Spanish Government for their funding support, to the people that participated in evaluation, and to the reviewers for their comments.

Abstract

Two of the main needs when developing an information system for an organization are that system analysts know and understand the application domain and that the system properly supports the business processes of the organization. Consequently, elicitation of system requirements from business process models has been acknowledged as a suitable activity to deal with that needs. In addition, system requirements must be linked to subsequent development stages.

However, system analysts can face many challenges when performing these activities. They may have problems communicating with customer stakeholders and may need to analyse and operationalize the purpose of the information system. Furthermore, system analysts must bridge the gap between business and system domains for specification of system requirements, specify different types of system requirements and guarantee that their specification is precise, consistent and homogeneous.

In relation to object-oriented conceptual modelling-based information system development, system analysts must also avoid potential problems that may arise when a conceptual schema is created from system requirements as part of their link with subsequent development stages. For example, a conceptual schema can be incomplete or inconsistent if it is not properly managed.

As a solution, this thesis presents a methodological approach for business process-based requirements specification and object-oriented conceptual modelling of information systems. The approach consists of four stages: organizational modelling, purpose analysis, specification of system requirements and derivation of object-oriented diagrams.

By following the design research methodology for performing research in information systems, the methodological approach has been designed on the basis of many existing ideas and principles in academia

and industry. It also provides new mechanisms and guidance to address the challenges presented above.

The methodological approach mainly aims to help system analysts to elicit system requirements from business process models, adequately specify system requirements and derive the object-oriented conceptual schema of an information system from its system requirements. It does so by taking advantage of existing solutions and by modifying them to better tackle the associated challenges.

The methodological approach has been evaluated in laboratory and industrial contexts, especially focusing on its usefulness for practitioners. Thanks to evaluation, several lessons have been learned and many of them have driven definition of the methodological approach. Furthermore, the lessons learned can be very useful both in academia and in industry for identification of further research areas and for awareness of situations that may occur in information system development projects, respectively.

Resumen

Dos de las principales necesidades a la hora de desarrollar un sistema de información para una organización son que los analistas de sistema conozcan y comprendan el dominio de aplicación y que el sistema dé un soporte adecuado a los procesos de negocio de la organización. Como consecuencia, se ha reconocido a la captura de requisitos de sistema a partir de modelos de procesos de negocio como una actividad idónea para acometer dichas necesidades. Además, los requisitos de sistemas se deben enlazar con etapas de desarrollo posteriores.

Sin embargo, los analistas de sistema pueden afrontar muchos retos mientras desarrollan estas actividades. Pueden tener problemas para comunicarse con stakeholders cliente y pueden necesitar analizar y operacionalizar el propósito del sistema de información. Más allá de este problema y de esta necesidad, los analistas de sistema deben enlazar los dominios de negocio y de sistema para especificar requisitos de sistema, especificar diferentes tipos de requisitos de sistema y garantizar que su especificación es precisa, consistente y homogénea.

En relación al desarrollo de sistemas de información basado en modelado conceptual orientado a objetos, los analistas de sistema deben evitar problemas potenciales que pueden surgir al crear un esquema conceptual a partir de requisitos de sistema como parte de su enlace con etapas de desarrollo posteriores. Por ejemplo, un esquema conceptual puede ser incompleto o inconsistente si no se gestiona adecuadamente.

Como solución, esta tesis presenta una aproximación metodológica para la especificación de requisitos basada en procesos de negocio y el modelado conceptual orientado a objetos de sistemas de información. La aproximación consta de cuatro etapas: modelado organizacional, análisis del propósito, especificación de requisitos de sistema y derivación de diagramas orientados a objetos.

Siguiendo la metodología de investigación de diseño para investigar sobre sistemas de información, la aproximación metodológica se ha

diseñado a partir de varias ideas y principios que ya existían en el ámbito académico y en la industria. La aproximación complementa dichas ideas y principios suministrando nuevos mecanismos y directrices para encarar los retos presentados anteriormente.

La aproximación metodológica persigue principalmente ayudar a analistas de sistema a capturar requisitos de sistema a partir de modelos de proceso de negocio, a especificar adecuadamente requisitos de sistema y a derivar un esquema conceptual orientado a objetos de un sistema de información a partir de sus requisitos de sistema. Lo hace aprovechando soluciones existentes y modificándolas para encarar mejor los retos asociados.

La aproximación metodológica ha sido evaluada en contextos de laboratorio e industriales, centrándose dicha evaluación principalmente en la utilidad de la aproximación para profesionales. Gracias a la evaluación, varias lecciones se han aprendido y muchas de ellas han dirigido la definición de la aproximación metodológica. Además, las lecciones aprendidas pueden ser muy útiles tanto para la academia como para la industria para la identificación de nuevas áreas de investigación y el conocimiento de situaciones que pueden ocurrir en proyectos de desarrollo de sistemas de información, respectivamente.

Resum

Dos de les principals necessitats en el desenvolupament d'un sistema d'informació per a una organització és que l'analista conega i entenga el domini de l'aplicació i que el sistema suporte els processos de negoci de l'organització. Per tant, la captura de requisits des de models de processos de negoci és coneguda com una activitat adequada per tractar amb eixes necessitats.

Tanmateix, els analistes del sistema poden fer front a molts reptes quan s'utilitzen aquestes activitats. El modelat de processos de negoci pot tindre problemes de comunicació entre clients i stakeholders, i pot necessitar analitzar i operacionalitzar el propòsit del sistema d'informació. A més a més, l'analista del sistema ha de cobrir la distància entre el negoci i els dominis del sistema per especificar requisits del sistema, especificar distints tipus de requisits i garantir que les seues especificacions són precises, consistents i homogènies.

En relació al desenvolupament de sistema d'informació basats en el modelat orientat a objectes, l'analista ha d'evitar problemes potencials que apareixen quan l'esquema conceptual es crea a partir dels requisits del sistema, com a part de la seua connexió amb les següents fases de desenvolupament. Per exemple, un esquema conceptual pot ser incomplet o inconsistent si no es tracta de la manera adequada.

Com a solució, aquesta tesi presenta una aproximació metodològica per a l'especificació de requisits basats en processos de negoci i en models conceptuals orientat a objectes que representen sistemes d'informació. La proposta consisteix en quatre etapes: modelat organitzacional, anàlisi del propòsit, especificació dels requisits del sistema i derivació del diagrames orientat a objectes.

El disseny de la metodologia d'investigació s'ha dissenyat utilitzant moltes idees existents i principis tant de l'acadèmia com de la indústria, proporcionant noves mecanismes i guies per dirigir els reptes presentats anteriorment.

La aproximació metodològica té com a objectiu principals ajudar l'analista a capturar requisits des de models de processos de negocis, especificar requisits del sistema i derivar l'esquema conceptual d'un sistema d'informació des dels seus requisits. Açò s'aconsegueix traient profit de solucions existents i modificant-les per abordar els reptes relacionats de una manera més òptima.

L'aproximació metodològica s'ha avaluat en el laboratori i en entorns industrials, especialment centrat en l'utilitat per a professionals. Gràcies a aquesta avaluació, hem aprés diverses lliçons i moltes han conduit a la definició de l'aproximació metodològica. A més a més, les lliçons apreses poden ser molt útils en l'acadèmia i en la indústria per identificar àrees d'investigació futura i per prendre consciència de les situacions que poden ocórrer en el en el desenvolupament de projectes relacionats en el sistemes d'informació.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

"Common sense is the least common of the senses"

Anonymous

1.1 Research Area

An information system (IS) can be defined as a designed system that collects, stores, processes and distributes information about the state of a domain (Olivé, 2007). For example, it can distribute information about the state of an organization. The main goal of an IS is to provide users (people or other systems) with the information that they need, at the right time and in the right place (Pohl, 2010).

The role that information technology (IT) in general and ISs in particular play in organizations has evolved over time and their importance has increased considerably. ISs are nowadays expected to contribute to the competitiveness of an organization and to improve its performance. It is essential that they fit the needs of an organization and thus deliver value to the organization (McKeen, Smith, 2003). ISs should be targeted at concerns such as business productivity and cost reduction, business agility and speed to market, business/IT alignment, IT efficiency and agility, and business process reengineering (Luftman, Ben-Zvi, 2010).

As a consequence, IS development for an organization has become a more complex process. IS success not only depends on the resolution of technical problems or on the use of up-to-date technologies, but also highly depends on the requirements engineering (RE) process so that an IS fits actual business needs.

The RE process is the first stage of a software process. RE can roughly be defined as the branch of software engineering that is related to the discovery and documentation of the purpose of a software system. It is concerned with real-world goals for functions of and constraints on a software system, and with the relationship of these factors to precise specifications of software behaviour and to their evolution over time and across software families (Zave, 1997). Although the RE process may vary depending on the characteristics of a project, its basic activities are requirements elicitation, analysis, specification (aka documentation), validation, negotiation and management (Sommerville, 2005).

The requirements of a software system correspond to the activities, capabilities or conditions that the system must support, possess or meet, respectively, to fulfil stakeholders' needs. Requirements represent external characteristics of a software system (Davis, 1993) and can be specified at different abstraction levels (Aurum, Wohlin, 2005b). For example, requirements can be specified at the business level (business requirements) and at the software system level (system requirements).

Requirements can be considered the main indicators of the success (Nuseibeh, Easterbrook, 2000) and of the quality (Finkelstein, 1994) of a software system, as well as of project success (Procaccino, Verner, Lorenzet, 2006). The RE process is also interrelated with many processes of a software development company and with other development stages and strongly influences on them (Pohl, 2010). As a result, the RE process has been recognised as the most important software development stage (Davis, 1993). Disregarding it can lead a software development project to fail and significantly increase its duration and cost.

When performing the RE process of an IS, it is essential that system analysts have a deep knowledge and understanding of the application domain to fulfil organizational needs (Jackson, 1995). Organizational concerns must drive requirements elicitation (Zave, Jackson, 1997) and requirements must be specified in terms of phenomena that occur in the organizational environment (Sommerville, Sawyer, 1997).

Consequently, the convenience of performing a stage of organizational modelling during the RE process of an IS (Loucopoulos, Karakostas, 1995; Bridgeland, Zahavi, 2009) and the need of system analysts to play a business analyst role (IIBA, 2009; Rubens, 2007) have been widely acknowledged during the last two decades. Organizational models depict the goals, structure and behaviour of an organization, and help system analysts to understand the application domain, the organizational activity and needs, and the requirements of an IS.

Business process modelling is part of most of the organizational modelling-based RE approaches. Business process models reflect the activity of an organization and facilitate understanding of the application domain. They can also be used for elicitation of system requirements (Alexander, Stevens, 2002; Lauesen, 2002). Furthermore, business process modelling can be regarded as essential for IS development.

Any IS for an organization should manage and execute business processes involving people, applications and/or information sources (Dumas, van der Aalst, ter Hofstede, 2005). ISs must support the business processes of an organization (Kirchmer, 1999) and thus contribute to the achievement of its goals. Otherwise, an IS may hamper the activity of an organization and fail. These problems can be avoided by specifying system requirements from business process models.

Nonetheless, proper specification of system requirements from business process models does not imply that a RE approach is adequate for IS development. Any RE approach must be linked to subsequent development stages so that it is appropriate and useful for the software process into which it is integrated (Verner, et al., 2005).

For IS development on the basis of object-oriented (OO) conceptual modelling, a RE approach must be linked with conceptual models. Therefore, it is necessary to determine how OO conceptual schemas, which specify the knowledge that an IS needs to know (Olivé, 2007), can be created from system requirements.

The research area of this thesis is IS development for organizations. More specifically, the thesis focuses on business process-based requirements specification and its link with OO conceptual modelling as a subsequent development stage. Business process modelling and OO modelling are among the practices most frequently used for conceptual modelling in industry (Davies, et al., 2007).

1.2 Motivation

When addressing business process-based requirements specification and subsequently creating an OO conceptual schema for the development of an IS for an organization, system analysts face different challenges. Several needs must be considered so that requirements are adequately elicited, analysed, specified and integrated into a software process. Furthermore, some problems may arise. If the needs and problems are not addressed, then a RE process and the associated IS development project may fail.

Communication with customer stakeholders is one of the main challenges that system analysts face for IS development (Zave, 1997). Good communication between system analysts and customer stakeholders is necessary during the RE process (Sommerville, Sawyer, 1997) and for organizational modelling (Stirna, Persson, Sandkuhl, 2007). Customer stakeholders are the main source of information of the activity of an organization and of the requirements of an IS, and they have to validate the information that system analysts gather and the models that system analysts create in order to guarantee that they are correct, i.e., that they contain the data that they must contain.

Communication between system analysts and customer stakeholders can be difficult as a consequence of their different vocabularies and professional backgrounds (Berenbach, et al., 2009) and of the existence of a culture gap between business and system domains (Taylor-Cummings, 1998). These conditions can cause mismatches between what customer stakeholders say or want to say and what system analysts understand or believe that have understood.

One reason for miscommunication is that the models that system analysts create during the RE process for interacting with customer stakeholders usually focus on representation of the system domain instead of on requirements of the business domain. Customer stakeholders usually lack a deep software background, thus the models can be hard to understand and validate for them. Consequently, models and notations that do not just focus on the system domain and thus facilitate communication between system analysts and customer stakeholders should be used during the RE process.

Another problem that system analysts face is that business process models may not be enough to analyze an organization and understand the application domain of an IS. Organizations usually decide to introduce or modify an IS in order to solve some problems or meet some needs (Nuseibeh, Easterbrook, 2000; Pohl 2010). These problems or needs correspond to the goals that the system must (help to) fulfil (Pinheiro, 2003), i.e., the system purpose. Therefore, it is important for system analysts to explore the goals of different customer stakeholders in addition to their activity so that problems are solved, needs are met and purposeful requirements are defined (Rolland, Salinesi, 2005).

When system purpose is not very complex, it can be directly analyzed on business process models. An example is when system purpose is mainly related to activity automation. However, this case does not always happen in IS development projects and system purpose may require a deeper analysis. In this case, the use of a goal-oriented RE approach helps system analysts to model, understand and analyse the purpose of a system, to relate requirements with system purpose and, consequently, to better respond to the needs of an organization.

In addition, fulfilment of system purpose may only be possible by changing (reengineering) the business processes of an organization so that they fit the needs of the organization. ISs should aim to support new ways of running a business that would not be possible without them (Alexander, Bider, Regev, 2003), and the new ways of running may involve a different execution of business processes in order to improve, for instance, the efficiency and competitiveness of an organization. This situation must be considered when analysing business processes and system purpose during the RE process.

Once the business processes that an organization executes or wants to execute after IS development have been modelled as part of the RE process, they can be used for elicitation of system requirements. However, it is necessary to bridge the gap between business and system domains during specification of system requirements. This gap is the consequence of characteristics such as different terminology, levels of granularity and models between the domains (Arsanjani, 2005; Castro, Kolp, Mylopoulos, 2002). The business requirements that are specified in business process models must be analysed in order to determine the functional system support that the business processes need.

In practice, inadequate functionality is the most common cause of system failure to meet expectations (Lauesen, 2002). As a consequence, the functional requirements of an IS (which specify what the system shall do) might be considered the most important type of requirements and usually receive most of the attention during the RE process (Borg, et al., 2003). Nonetheless, functional requirements are not sufficient to completely and precisely specify the system requirements of an IS, and disregard for other types of system requirements may result in an inadequate and unsatisfactory system for customer stakeholders (Firesmith, 2005; Lawrence, Wiegers, Ebert, 2001).

The behavioural perspective of an IS that is provided by functional requirements must be complemented with a data perspective (Siau, Lee, 2004) and a quality (aka non-functional) perspective (Loucopoulos, Karakostas, 1995). Data requirements (which specify what the system shall show, store and manage; (Lauesen, 2002)) and quality requirements (which specify restrictions on the system and how well the system shall perform its functions; (Berntsson-Svensson, 2009)) of an IS must be considered when specifying its system requirements too.

Furthermore, it is important that system analysts properly manage functional, data and quality requirements so that their specification is consistent. The specification of these requirements should also be precise enough so that, for instance, they can be interpreted easily by other supplier stakeholders that have to deal with them (e.g., programmers).

An additional problem that may arise when specifying system requirements of an IS is lack of homogeneity. Homogeneity is achieved if all the system requirements that are specified in a given style and thus are at the same abstraction level have the same granularity. Although the problem of homogeneous granularity has not been widely addressed by the RE community, some authors have acknowledged it (Dutoit, Paech, 2002; España, et al., 2009; Gorschek, Wohlin, 2006). System analysts may mix and misinterpret granularity and abstraction levels of system requirements, and as a result application of a RE approach can be hindered, the quality of a system requirements specification (SyRS) can be negatively affected and a SyRS can be inconsistent.

After system requirements have been specified and customer stakeholders have validated them, they can be used as basis for creation of the OO conceptual schema of an IS, i.e., for creation of a conceptual

schema that meets system requirements and consequently meets the needs of an organization. This means that the RE approach followed is integrated into the software process of an organization in the case of OO conceptual modelling-based IS development.

This thesis has been developed in the context of OO-Method (Pastor, Molina, 2007). OO-Method is an approach for automatic software generation that is based on OO conceptual modelling and that is supported by the OlivaNova industrial tool[1]. OO-Method consists of several models and diagrams. For example, it includes a class diagram (object model) and state transition diagrams (dynamic model).

System requirements that are specified from business process models must be linked to OO-Method models so that business process-based requirements specification is integrated into the approach. Consequently, the RE approach followed would meet the needs of the organizations that use OO-Method and would be useful for them. However, problems in integration may arise.

System requirements and OO conceptual schemas are usually specified and created, respectively, separately or by different people (Yourdon et al., 1995), and system analysts may have difficulties creating an OO conceptual schema and creating it from system requirements (Batra, 2007; Phalp, Cox, 2007; Svetinovic, Berry, Godfrey, 2005). These problems are especially frequent when system analysts are novice.

As a result, problems such as inconsistency and incompleteness in an OO conceptual schema may appear (Glinz, 2000) or an OO conceptual schema may not meet system requirements (Insfrán, Pastor, Wieringa, 2002). OO conceptual schemas and system requirements must be properly managed to avoid these problems, and specification of system requirements should aim to make creation of OO conceptual schemas possible and straightforward (Yue, Briand, Labiche, 2009).

Despite the importance of ISs in organizations and thus of business process-based requirements specification and its link with OO conceptual modelling, and as shown in Chapter 2, no existing RE approach properly deals with all the challenges described. Since disregard of one of the needs or problems may result in failure of an IS development project, research targeted at tackling these challenges is necessary.

[1] http://www.care-t.com. Accessed July 13, 2011.

1.3 Problem Statement

As discussed in the previous section, business process-based requirements specification for IS development and its link with OO conceptual modelling can be difficult for system analysts. They have to carefully perform these activities so that several needs are met and some problems are solved or avoided. Otherwise, an IS development project may not be satisfactory for its stakeholders and thus may fail.

No existing RE approach addresses all the challenges presented. Therefore, business process-based requirements specification for an IS and its link with OO conceptual modelling are not closed research topics. New research efforts should be performed to adequately develop these activities, and they must provide perspectives and approaches in the form of new mechanisms and guidance to help system analysts face the associated challenges.

The above statements are also in line with problems and challenges explained and justified in recent works. Support for business process enactment, automation or execution based on business process models is a top issue and a top challenge in industry and in academia (Indulska, et al., 2009), and focus on business processes during the RE process is a challenge for future research on RE (Jarke, et al., 2010). Integration of approaches and provision of guidance for this purpose is also one of the main needs of RE (Cheng, Atlee, 2007).

In summary, the challenges that this thesis addresses can be stated by the following research questions:

Research question 1. How should the business processes of an organization be modelled and analysed for elicitation of system requirements of an IS?

Research question 2. How should system requirements of an IS be specified from business process models?

Research question 3. How can business process-based system requirements be linked to OO conceptual modelling?

These challenges can be faced by achieving the objectives that are enumerated in the next section.

1.4 Objectives

The main goal of the thesis is to develop a methodological approach for specification of system requirements of an IS for an organization from business process models and for link of the system requirements with OO conceptual modelling. This goal can be reached and the associated challenges can be faced by achieving several objectives, and such objectives are related to the three research question presented in the previous section.

The objectives that are related to the research question 1 are:

1. To identify an appropriate notation for business process modelling in the RE process;
2. To identify a suitable approach to model the purpose of an IS, and;
3. To define mechanisms and guidance to systematically analyse the purpose of an IS and determine how it can affect the business processes of an organization.

The objectives that are related to the research question 2 are:

4. To define mechanisms and guidance to systematically bridge the gap between business and system domains when specifying system requirements from business process models, and;
5. To develop a style for SyRS that
    a. determines support for business processes;
    b. integrates specification of functional, data and quality requirements, and;
    c. addresses consistency, precision and homogeneous granularity of system requirements.

The objectives that are related to the research question 3 are:

6. To define guidance to systematically derive a class diagram from business process-based system requirements, and;
7. To define guidance to systematically derive state transition diagrams from business process-based system requirements.

1.5 Proposed Solution

The objectives of the thesis are achieved by developing a methodological approach for business process-based requirements specification and OO conceptual modelling of ISs. The approach consists of four stages:

- **Organizational modelling**

  This stage aims to model and understand the behaviour of an organization prior to development of an IS for it in order to gain knowledge about the application domain. Business process models are the main artefacts of the stage, and models that facilitate involvement of and communication with customer stakeholders are used.

- **Purpose analysis**

  This stage aims to analyse the goals that an IS should allow an organization to achieve and to specify how their achievement would affect the business processes of the organization. Goal models are used as basis for the development of the stage and involvement of customer stakeholders is promoted.

- **Specification of system requirements**

  This stage aims to specify system requirements of an IS whose implementation would adequately support the business processes of an organization. The stage also addresses the gap between business and system domains, homogeneous granularity of system requirements, and complete, precise and consistent specification of functional, data and quality requirements.

- **Derivation of OO diagrams**

  This stage aims to create OO diagrams (i.e., an OO conceptual schema) of an IS that meet its system requirements and are complete and consistent. The OO diagrams correspond to a class diagram and a state transition diagrams. As a result of the derivation, specification of system requirements and the rest of previous stages are integrated into OO conceptual modelling-based IS modelling and development. For example, they can be integrated into OO-Method-based IS development.

This four-stage approach is presented in detail in the next chapters and corresponds to the overall contribution of the thesis. Thanks to the approach, system analysts can more easily face the challenges presented when specifying system requirements from business process models and subsequently linking them to OO conceptual modelling.

1.6 Research Methodology

The thesis has been developed by means of a research project that has followed the design research methodology for performing research in ISs (Vaishnavi, Kuechler, 2008).

Design research involves the analysis of the use and performance of designed artefacts to understand, explain and very frequently improve the behaviour of aspects of ISs. Examples of such artefacts are system design methodologies and languages. In the case of this thesis, the designed artefact is a methodological approach for specification of the system requirements of an IS for an organization from business process models and for OO conceptual modelling from the system requirements, which corresponds to the main goal of the thesis.

Figure 1.1 shows the general process of design research. It consists of five stages:

1. **Awareness of problem**

   It may come from multiple sources (new developments in industry or in a reference discipline, reading in an allied discipline, etc.). Its output is a proposal for a new research effort.

2. **Suggestion**

   It follows the proposal and is intimately connected with it. The tentative design (output of this stage) is an integral part of the proposal and must targeted at the proposal. The suggestion stage is an essentially creative step wherein a new artefact is envisioned based on a novel configuration of either existing or new and existing elements.

3. **Development**

   The tentative design is implemented in this stage, and the techniques for implementation will vary depending on the

artefact to be constructed. The implementation itself may not be very novel, but the novelty may be primarily in the design, not in the construction of the artefact.

4. Evaluation

Once the artefact has been constructed, it is evaluated according to criteria that are always implicit and frequently made explicit in the proposal. This stage contains an analytic activity in which hypotheses are usually made about the behaviour of the artefact. The results of the evaluation stage and additional information gained in the construction and running of the artefact are brought together and feedback to another round of suggestion (circumscription arrow in Figure 1.1).

5. Conclusion

This stage is the finale of a specific research effort. The results of the effort are not only consolidated and documented at this stage, but the knowledge gained in the effort is categorized as either firm (facts learned) or as loose ends (behaviour that serves as basis of further research). Awareness of the problem changes after conclusion (operation and goal knowledge arrow in Figure 1.1).

For this thesis, the outputs of each stage of the general methodology for design research are summarised in Table 1.1. Some of the outputs have already been presented in this chapter, and most of them are presented in the next chapters of the thesis.

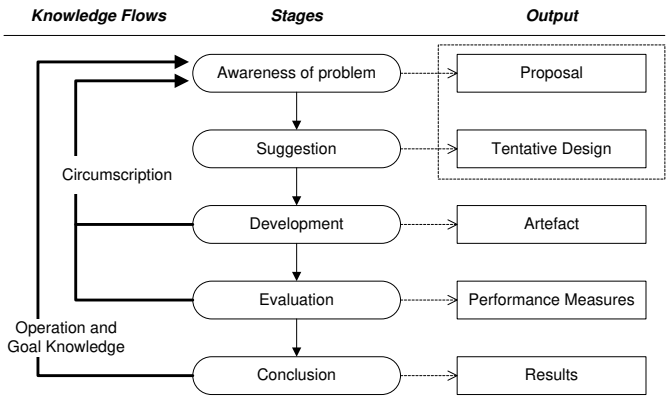

Figure 1.1 General process of design research

Table 1.1 Outputs of the stages of design research in this thesis

Stage of design research	Outputs
Awareness of problem	Motivation, research questions and review of state of the art
Suggestion	Objectives, proposed solution and fundamentals of the proposed solution
Development	Methodological approach
Evaluation	Lessons learned
Conclusion	Publications, thesis, contributions and future work

1.7 Thesis Context

This thesis has been developed at the research centre *Centro de Investigación en Métodos de Producción de Software (ProS)* of the *Universidad Politécnica de Valencia* (formerly known as OO-Method research group of the *Departamento de Sistemas Informáticos y Computación* of the same university), and in the context of the following regional, national and international research projects:

- "Producción automática de software a partir de modelos organizacionales". Regional project with industry.
- "DESTINO: Desarrollo de e-Servicios para la nueva sociedad digital". National CYCIT project referenced as TIN2004-03534.
- "Automatización de Procesos Orientado a Servicios a Partir de Modelos Organizacionales". Regional project with industry.
- "Generación de infraestructuras de tecnologías de información a partir de modelos organizacionales". National FPU Project referenced as AP2006-02324.
- "SESAMO: Construcción de Servicios Software a partir de Modelos". National CYCIT project referenced as TIN2007-62894.
- "ITEI: Information Technologies supporting the Execution of Innovation projects". International ITEA 2 project referenced as TSI-020400-2008-117.
- "From Business Objectives to Information Systems / De los Objetivos de Negocio a los Sistemas de Información". International project referenced as HI2008-0190.

- "ITEI_AVANZA (Information Technologies Supporting the Execution of Innovation Projects)". International AVANZA project referenced as TSI-020400-2009-8.
- "ProsREQ - Producción de Software Orientado a Servicios basada en Requisitos: La parte Funcional". National CYCIT project referenced as TIN2010-19130-C02-02.

1.8 Thesis Structure

The rest of the thesis is organized as follows.

Chapter 2: state of the art.

This chapter reviews the most relevant existing works related to the thesis. Their analysis has been necessary to determine the current state of research and practice and to adopt existing principles and mechanisms in the proposed solution.

Chapter 3: fundamentals of the proposed solution.

This chapter describes the main ideas and principles on which the thesis is based. Their presentation is important for awareness of many of the decisions that have been made and of the rationale behind them, as well as and for understanding of the design and development of the proposed solution.

Chapter 4: organizational modelling.

This chapter presents the first stage of the proposed solution. It describes a business process-driven approach for organizational modelling as a first step for modelling and understanding of the application domain and for determination of organizational needs.

Chapter 5: purpose analysis.

This chapter presents the second stage of the proposed solution. It describes an approach for analysis of the goals of an IS and determination of their impact on the business processes of an organization. It is a second step for understanding and analysis of organizational needs.

Chapter 6: specification of system requirements.

This chapter presents the third stage of the proposed solution. It explains how to specify the system requirements of an IS from the

business process models of the organization for which the system is going to be developed.

Chapter 7: derivation of OO diagrams.

This chapter presents the fourth stage of the proposed solution. It describes how to link the business process-based system requirements of an IS to OO conceptual modelling (as a subsequent development stage) in the form of a class diagram and state transition diagrams.

Chapter 8: evaluation.

This chapter describes the evaluation that has been performed for the proposed solution. A survey has been performed on the basis of application of different methods for validation of RE approaches, and several lessons have been learned.

Chapter 9: conclusions.

This chapter summarises the main conclusions that can be drawn as a result of the development of this thesis. It describes the contributions that have been made, discusses the impact of the thesis and presents the future work that could be performed.

Appendix A: conceptual framework

This appendix presents a conceptual framework that graphically shows the concepts that are used in the stages of the proposed solution and the relationships between the concepts. It complements the presentation of the stages in Chapters 4, 5, 6 and 7.

Appendix B: tool support.

This appendix outlines the tool support that has been developed for the proposed solution. It corresponds to several prototypes whose main purpose has been to show feasibility of automation of the proposed solution.

Chapter 2
State of the Art

"It is said that if you know your enemies and know yourself, you will not be imperilled in a hundred battles; if you do not know your enemies but do know yourself, you will win one and lose one; if you do not know your enemies nor yourself, you will be imperilled in every single battle"

Sun Tzu

This chapter reviews the state of the art related to this thesis. It embraces different fields, disciplines and techniques that are related to business process modelling, RE and OO conceptual modelling. For its review, state of the art is divided into: 1) notations for business process modelling; 2) goal-oriented RE approaches; 3) business process-based (RE) approaches for organizational modelling; 4) approaches for specification of system requirements, and; 5) approaches for link of system requirements with OO conceptual modelling.

The following sections present the works that can be considered the most relevant ones in each category. After their review, discussion about the works is performed on the basis of their support for achievement of the objectives of the thesis. Finally, a summary of the chapter is presented. It must be indicated that the definition of business process adopted in this thesis is presented and discussed in Chapter 3.

2.1 Notations for Business Process Modelling

Business process modelling is a very common practice in organizations nowadays. It plays a major role in many fields both in industry and in academia (Indulska, et al., 2009), helping organization to be competitive and achieve their goals. Therefore, the importance of business process modelling is undeniable.

Business process models allow people to communicate and understand the running of an organization. They are used for multiple initiatives, such as business process reengineering, business process management or software development. In summary, business process models are used within organizations for learning, for decision support about process development and design, for control and decision support during process execution, and for analysis of information technology support (Aguilar-Savén, 2004).

A sign of the relevance of business process modelling is the high number of existing notations for this purpose (Dumas, van der Aalst, ter Hofstede, 2005; Weske, 2007), each one of them with specific purposes, strong points and weaknesses. Some notations have appeared in academia and have been adopted later in industry, whereas others have been initially defined for industrial purposes and have later been refined and improved by means of academic research.

Another characteristic of current work on notations for business process modelling (and on languages for business process execution) is the existence of standardization efforts (Ko, Lee, Lee, 2009). Nonetheless, a "universal" notation for business process models does not exist. The reason behind this fact can be considered obvious. As said above, each notation has been developed with specific purposes, thus they focus on those aspects that have been regarded as more important by their designers. Furthermore, existence of a unique "best-for-all" notation would even be counterproductive. Such a notation may become too complex to understand or to be handled for many purposes.

Since review of all the existing notations in this thesis is not possible, the next subsections present the notations that can currently be considered the most relevant ones in academia and industry. Section 2.1.5 analyses the notations and discusses selection of one of them for the methodological approach of the thesis.

2.1.1 BPMN

BPMN (Business Process Model and Notation; (OMG, 2009)) is a graph-based standard notation whose specification was developed by BPMI (Business Process Management Initiative) in 2004 and that is now an OMG (Object Management Group) standard. The notation has evolved since its creation, and its version 2.0 has been published recently. Evolution of the notation has mainly focused on improvement of its expressiveness and broadening of its modelling purposes.

The main goal of BPMN has always been to provide a notation that is easy to understand by all business process users. It also aims to provide a standard that fills the gap between business models and their implementation, and is closely related to WS-BPEL (Web Service Business Process Execution Language). As a result, BPMN can be used both for business modelling and for system modelling. Nonetheless, BPMN can be regarded as more business-oriented than system-oriented.

BPMN can help organizations to understand their procedures by means of a graphical notation. The notation allows these procedures to be communicated in a standard way by means business process diagrams (BPD, i.e., business process models in BPMN terminology), which consists of different types of graphical objects:

- Flow objects, which are the main graphical objects for business process modelling; they are events (start, intermediate and end events, which can have triggers and be "catching" or "throwing"), activities (sub-processes and tasks) and gateways (exclusive, inclusive, complex and parallel gateways).
- Connecting objects, which allow other graphical objects to be connected; they are sequence flows (normal and default), message flows and associations.
- Swimlanes, which allow participants of a business process to be represented; they are pools and lanes.
- Artifacts, which provide additional information about a business process model; they are data objects, groups and annotations.

In addition, BPMN allows more graphical objects to be defined according to the needs of business process users. Finally, the notation is supported by many tools and vendors.

2.1.2 UML 2.0 Activity Diagrams

UML (Unified Modelling Language; (OMG, 2005)) is a notation for modelling of object-oriented software systems. It is considered de facto standard for high-level descriptions of this kind of systems, and consists of 13 diagrams. Among them, activity diagrams are a behavioural diagram that aims to model business processes and flows in a software system.

Activity diagrams were added to UML rather late and initially they were poorly integrated, lacked expressiveness and did not have an adequate semantics. As a solution, several concepts and graphical elements have been introduced in UML 2.0. In addition to significant syntax modifications, the main difference is switching from state machine-based semantics to token flow (Petri net-like) semantics.

The fundamental units of behaviour in an activity diagrams are:

- Activities, which are the highest-level units of behaviour, and;
- Actions, which are the fundamental units of executable functionality in an activity; they take a set of inputs and converts them into a set of outputs.

Graphically, an activity diagram consists of a set of nodes and edges. They can correspond to:

- Action nodes, which receive and manage control and data values and provide control and data for other actions.
- Control nodes, which coordinate flows between other nodes and route control or data tokens through the graph; examples of control nodes are decision points and forks.
- Object nodes, which hold data tokens temporarily as they wait to move through the graph; they also indicate an instance of a particular object.
- Control flow edges, which start an activity node after the completion of the previous one by passing a control token.
- Object flow edges, which provide inputs to actions; they also model the flow of values to or from object nodes by passing objects or data tokens

2.1.3 EPC

EPC (Event-driven Process Chain; (Dumas, van der Aalst, ter Hofstede, 2005; Scheer, 2000)) was developed from a project between SAP AG and the University of Saarland. It is based on stochastic networks and Petri nets. However, use of the notation does not require a formal framework. For example, the notation does not rigidly distinguish between output flows and control flows or between places and transitions.

The initial notation has been extended several times in order to provide support for modelling of more aspects of an organization. This led to what has been called the extended EPC notation (eEPC). Nonetheless, the notation is usually referred to just as EPC. Its diagrams are called EPC too.

Adoption of EPC is very high in Germany, Austria and Switzerland (Becker, et al., 2010), both in industry and in academia. The most likely reason is that EPC was originally invented in Germany, what may have caused that many German companies and researchers initially adopted the notation. The companies may also operate in Austria and Switzerland, and, therefore, EPC may have easily found users in the three countries. In addition, EPC is a key component of SAP modelling concepts for business engineering and customization.

EPC consists of the following main graphical elements:

- Functions, which are active elements and model the activities within an organization.
- Events, which are created by processing functions or by actors outside a model; they act as preconditions or postconditions of functions.
- Logical operators, which connect functions and events; they can be AND, OR and XOR.
- Organization units and roles, which depict the people responsible for executing a function.
- Information objects, which represent input and output data of the functions.
- Deliverables, which represents results (services or products) that functions require or produce.

2.1.4 YAWL

YAWL (Yet Another Workflow Language; (ter Hofstede, et al., 2010)) was developed by researchers from Eindhoven University of Technology and from Queensland University of Technology. Its purpose is to overcome some limitations of contemporary workflow management systems and of Petri nets when modelling workflows. More concretely, YAWL is an extension of Petri nets whose main purpose is to support more workflow patters (multiple instances, advanced synchronisation and cancellation patterns).

During the last years, YAWL has considerably advanced and evolved. The research efforts related to the notation have been extensive, and well-known members of the business process management community have work on its improvement. These works can be considered quite fruitful, and different and important contributions have been made.

A design and runtime environment supports YAWL and enactment of workflows modelled with the notation. The environment is open source, and it is freely available and can be extended. The environment also allows interaction with other systems on the basis of a service-oriented architecture. In this sense, the environment has been linked to the ProM environment for simulation of YAWL models and creation by applying process-mining techniques.

YAWL consists of five main graphical elements:

- Conditions, which represent a state of a process; input and output conditions are special types of conditions and represent where a process starts and finishes, respectively.
- Tasks, which represent actions that are performed by a human or an external application; they can be atomic, composite or multiple instance tasks.
- Splits, which represent the division of a path; they can be AND, OR or XOR splits.
- Joins, which represent the union of several paths; as splits, they can be AND, OR or XOR joins.
- Cancellation regions, which represent element deactivation upon activation of a task.

2.1.5 Analysis and Discussion

This section discusses the selection of the notation that can be regarded as the most suitable one in the RE process of an IS (first objective of the thesis). Selection is based on the purposes of the notations, their use in industry, their evaluations, their comparisons and their strong points.

The notation that has been selected is BPMN (version 1.2). Figure 2.1 shows its graphical objects. BPMN explicitly aims to be understandable by all business process users, and selection is also in line with its use in industry. Use of BPMN for the RE process is its most frequent application for technical purposes in practice (Recker, 2010).

BPMN has been extensively evaluated on the basis of different criteria such as the Bunge-Wand-Weber model (Rosemann, et al., 2006), the workflow patterns (Wohed, 2006), quality frameworks (Nysetvold, Krogstie, 2005; Wahl, Sindre, 2005), the perspectives of users and developers (Recker, Indulska, Green, 2007), its actual use (zur Muehlen, Recker, 2008) or its cognitive effectiveness (Genon, Heymans, Amyot, 2011). The main conclusions that can be drawn are the following ones:

- BPMN is the notation that supports a higher percentage of elements of the Bunge-Wand-Weber model.
- BPMN supports most of the workflow patterns; no notation for business process modelling supports all of them.
- BPMN is very expressive and easy to use and understand, although it can be improved.
- BPMN does not properly support business rule specification and has graphical objects that are redundant or overloaded.
- Just 20% of BPMN elements are regularly used, and some of them are hardly ever used.
- BPMN could modify representation of some graphical elements to improve cognitive effectiveness.

In addition to some of the above works, others have compared BPMN to the rest of notations reviewed (e.g., (Becker et al., 2010; Birkmeier, Klöchner, Overhagen, 2010; Decker, et al. 2010; Figl, et al, 2010; List, Korherr, 2006; Recker, et al. 2009; Rodríguez, et al., 2009)). Some conclusions from the comparisons are the following ones:

- Notations for business process modelling can be and are combined in practice.
- YAWL is more focused on modelling at an execution level than BPMN; nonetheless, BPMN can be mapped into YAWL.
- BPMN, activity diagrams and EPC properly support functional, behavioural, organizational and informational perspectives.
- Both BPMN and activity diagrams can become complex for stakeholders, thus they are usually adapted in practice to reduce complexity; BPMN may not be easier to use for all stakeholders; nonetheless, BPMN can be mapped into activity diagrams.
- BPMN is widespread in industry, and stakeholders' satisfaction is higher with BPMN than with EPC or activity diagrams.

Some original weaknesses of BPMN have already been mitigated. For example, lack of formal semantics (e.g., on the basis of Petri Nets (Dijkman, Dumas, Ouyang, 2008)). Others do not affect the thesis because correspond to issues that are not addressed when using BPMN. For example, representation of system structure (Rosemann, et al., 2006)

In summary, although BPMN has weak points, it can be regarded as the best suited notation for business process modelling currently. It can be considered the most expressive and easy to use and understand notation, and it is receiving strong support from academia and industry. As a result, it is recognised as the de facto standard notation for business process modelling. Nonetheless, awareness of the weaknesses of BPMN when using it is important in order to try to avoid them or mitigate them.

Finally, other analyses of BPMN can be found in (Aagesen, Krogstie, 2010; Recker, 2010; Recker, 2011). These works have analysed in more depth some of the issues that have been discussed in this section.

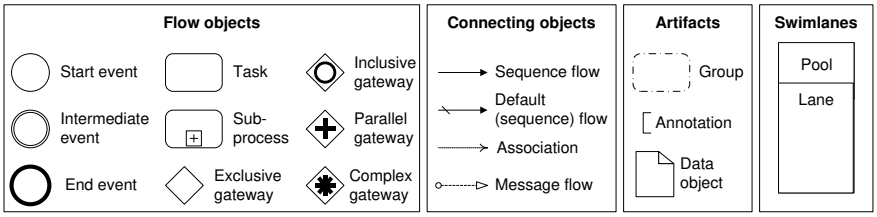

Figure 2.1 BPMN graphical objects

2.2 Goal-Oriented RE Approaches

Goals have long been recognized to be essential components of the RE process (van Lamsweerde, 2001). They can be defined as objectives that a software system should achieve in order to meet stakeholders' needs. Goals can be formulated at different abstraction levels, ranging from high-level (strategic concerns) to low-level (technical concerns).

Goal-oriented RE approaches emerged as a means to overcome a major drawback of traditional RE approaches. They lead to systems technically good but unable to respond to the needs of users. Goal-oriented RE approaches consider that requirements should initially focus on the why and how issues of a software system rather than on the issue of what needs to be implemented.

More specifically, traditional RE approaches have focused on the functionality of a system and its interactions with users. Instead of asking what the system needs to do, goal-oriented RE approaches ask why a given functionality is necessary and how it could be implemented. Therefore, goals give a rationale for system functionality.

In addition to this purpose, goal modelling and analysis in the RE process aims to: better understand a system; facilitate requirements elicitation; identify and evaluate alternative implementations; detect irrelevant requirements; obtain complete requirements specifications; identify and resolve requirements conflicts, and; define stable goals (Pohl, 2010). Goal-orientation is usually targeted at organizational change (Kavakli, Loucopoulos, 2006).

Nonetheless, goal-oriented RE approaches also present some weaknesses that can hinder their application (Rolland, Salinesi, 2005). For example, the approaches should better address goal abstractness, elicitation, fuzziness and operationalization, and better guide discovery of alternative goals.

The next subsections review the goal-oriented RE approaches that can be considered the most relevant ones. Section 2.2.4 analyses the approaches and discusses selection of one of them for the methodological approach of the thesis. A relevant work on goal-oriented RE that is not reviewed is GBRAM (Antón, 1997). This approach focuses on goal discovery, not on goal modelling and analysis. EKD goals model (Section 2.3.1) is neither reviewed in this section.

2.2.1 The i* Framework

The i* framework (Yu, 1995) was developed at University of Toronto to model and reason about organizations and their ISs. It focuses on modelling of the dependencies that exist between the business actors in order to achieve organizational goals. The framework consists of two models: the strategic dependency model and the strategic rationale model.

The strategic dependency model shows the dependencies that exist between actors to achieve their goals, to perform tasks and to provide or request resources. A dependency describes an intentional relationship between two actors. It is composed by a depender (the actor who is dependent on another actor), a dependee (the actor on whom another actor depends) and a dependum (the task, goal, resource or softgoal on which the relationship is focused).

The strategic dependency model is composed by four types of dependencies. Goal dependency represents that an actor depends on another to achieve a goal. Resource dependency represents that an actor depends on another to deliver a resource, which can be either material or informational. Task dependency represents that there exists a dependency for performing a task. Softgoal dependency is similar to the goal dependency, but with the difference that the goal and how it can be achieved are not precisely defined.

By means of the strategic rationale model, a deeper analysis of the reasons that exist behind each dependency relationship is performed. This is useful for representing tasks that have to be performed by the actors to achieve their goals, as well as for thinking about new ways of working. This model is based on the elements of the dependency model, but it also adds task decomposition links (to represent the combination of the necessary tasks to achieve a goal) and mean-ends links (to present the diverse options that can be taken to fulfil a task or goal).

The i* framework is also closely related to the NFR Framework (Chung, et al., 2000). The frameworks share creators, several principles and concepts. For example, both frameworks address analysis of softgoals. Nonetheless, the NFR framework focuses on identification, analysis and operationalization of non-functional requirements.

2.2.2 KAOS

KAOS (Knowledge Acquisition in autOmated Specification; (Dardenne, van Lamsweerde, Fickas, 1993)) was developed by researchers from University of Oregon and University of Louvain. Its purpose is to support requirements elicitation and specification from high level goals that system requirements must fulfil. In this sense, in KAOS goals are refined until they are assigned to individual agents.

KAOS consists of four complementary models: the goal model, the object model, the agent responsibility model and the operation model. For review of state of the art, the KAOS goal model is described.

The goal model is the driving model of KAOS. It declares the goals of a software system. A goal defines an objective that the system should achieve, usually through the cooperation of multiple agents. Goal-refinement links (AND/OR links) relate a goal to a set of sub-goals. A set of sub-goals refines a parent goal if the satisfaction of the sub-goals is sufficient for satisfying the parent goal. In addition to goal refinements, conflicts between goals can also be captured.

The goal model has a two-layer structure. An outer semantic net layer is used for declaring goals and goal links, and an inner textual layer is used for defining goals. Goals are defined in natural language and may optionally be defined formally in real-time temporal logic.

The goals at the top of a model usually represent strategic or business goals, whereas the goals at the bottom represent system requirements. Therefore, characteristics of both the business domain and the system domain can be represented in a goal model and business characteristics are refined until system ones are specified. As mentioned above, the agents responsible for achievement of the bottom goals are determined.

KAOS distinguishes among four patterns (achieve, cease, maintain and avoid) and five categories (satisfaction, safety, security, information and accuracy) of goals. The categories are organized into specialization hierarchies. For example, the category of security goals is specialized into subcategories such as confidentiality and authentication goals. Nonetheless, determination of a category for a goal is optional.

It must also be indicated that KAOS is usually regarded as targeted at embedded software systems, not at ISs (Pohl, 2010).

2.2.3 Map

The Map approach (Rolland, 2007) was developed at University of Paris 1 (Panthéon Sorbonne). It aims to capture the intentions (goals) of an enterprise or system and to determine the strategies that can contribute to the fulfilment of these intentions.

The emphasis on the concept of strategy as a way to achieve a goal distinguishes Map from other goal-oriented RE approaches. This emphasis is motivated by the fact that stakeholders do not naturally make the distinction between goals and strategies. As a consequence, pitfalls can arise. The size of a goal model can unnecessarily increase when strategies are expressed as goals, alternative ways to run an organization can be more difficult to discover, and recognizing stable elements in an organization (intentions) versus more versatile ones (strategies) can be more difficult. In addition, Map promotes variability analysis at the requirements stage.

Map diagrams consist of a graph whose nodes are intentions and whose edges are strategies. An edge entering a node identifies a strategy that can be used to achieve the intention of the node, so a map shows which intentions can be achieved by which strategies. The aggregation of a source intention, a target intention and a strategy is called section. A section can be refined in another Map diagram.

Given the low number of main constructs that Map propose, it can be considered that its diagrams must not be difficult for stakeholders to understand, or that at least that they must be easier to understand than the models of other goal-oriented RE approaches.

A relevant characteristic of Map is that it has not only be used as an approach for goal modelling, but also as an approach for business process modelling. By lowering their abstraction level, nodes can be used for modelling of activities and the edges for modelling of ways and of sequences to perform the activities.

However, modelling of business processes with Map can be regarded as not completely adequate. Although the use of few constructs is an advantage for understanding of the diagrams, it is also a disadvantage for business process modelling. Common and necessary information that should be included in a business process model is not supported by Map. For example, the roles that perform the activities cannot be specified.

2.2.4 Analysis and Discussion

This section discusses the selection of the approach that can be regarded as the most suitable one for modelling of the purpose of an IS (second objective of the thesis). Selection is based on the purpose of the approaches, on their strong points and on their weaknesses.

The approach that has been selected is Map (Figure 2.2, adapted from (Rolland, 2007)). It focuses on strategies to achieve goals and has only two main concepts (goal and strategy). The low number of concepts is positive to facilitate its use and understanding, thus it can be considered that the Map approach facilitates customer stakeholders' participation and communication with them. It also does not deal with tasks and roles that execute them, which can be modelled in business process models.

The i* framework has been discarded because several weaknesses have been identified (e.g., (Estrada, et al., 2007; Franch, 2010; Maiden, et al., 2004; Moody, Heymans, Matulevicius, (2010)). i* diagrams might be too complex and difficult to understand for stakeholders and should better support aspects such as granularity, scalability and refinement.

KAOS has been discarded because ways to achieve goals (strategies) have to be modelled as goals, what may increase the size and thus the complexity of a goal model. In addition, the Map approach uses fewer concepts, thus it is easier to use and understand.

Finally, there exist works that have compared and analysed i* and KAOS (e.g., (Kavakli, 2002; Matulevicius, Heymans, 2007; Matulevicius, Heymans, Opdahl, 2007)) and have found weaknesses. Finally, the relationship between i* and KAOS was studied in (Monteiro, et al., 2010).

Nonetheless, the value of the i* framework and of KAOS cannot be denied. They can be useful for development of many software systems, and their use will depend on the purpose of goal modelling (Pohl, 2010). For example, the i* framework is suitable to analyse dependencies among actors. In fact, many RE approaches are based on it, as shown below.

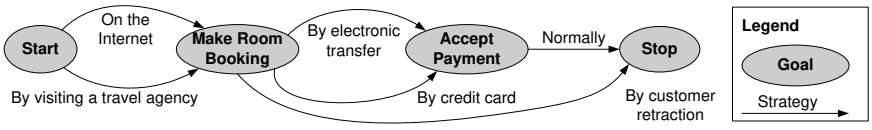

Figure 2.2 Example of Map diagram

2.3 Business Process-Based Approaches for Organizational Modelling

An organizational model (aka enterprise or business model) is a consistent set of special-purpose and complementary models that describe various facets of an organization to satisfy some purpose of business users (Vernadat, 1996). Organizational modelling is the set of activities that are used to develop the various parts of an organizational model. An organizational model usually consists of different sub-models for representation of the facets (activity, information, constraints, etc.).

Organizational modelling-based RE approaches aim to represent and understand the organization for which a software system is going to be developed. Requirements need to be articulated in the framework of "real-world" knowledge, which provides the purpose of the intended system as well as the knowledge about the phenomena that are common to the business and system domains (Loucopoulos, Karakostas, 1995).

Most of the existing organization modelling-based RE approaches focus on creation of business process models and elicit system requirements from them. A type of approaches that has been considered not to belong to this category are the i*-based approaches.

The success of the i* framework in academia is clear. There exist many approaches for IS development that are based on it, such as Tropos (Bresciani, et al., 2004; Castro, Kolp, Mylopoulos, 2001), RESCUE (Jones, Maiden, 2004) and PRiM (Grau, Franch, Maiden, 2008). Nonetheless, these approaches are not considered business process-based approaches because business process modelling is just partially addressed and plays a secondary role. For example, they do not address activity sequence.

BPMN-based approaches for requirements specification have been presented in (de Castro, Marcos, Vara, 2010; Rodríguez, et al., 2009) Nonetheless, these approaches are not reviewed in detail because they do not properly support any of the objectives of the thesis (apart from the first). This weakness is also present in most of the existing approaches for specification of system requirements from business process models (e.g., (Coskuncay, et al., 2010; Odeh, Kamm, 2003))

The next subsection reviews the approaches that can be regarded as the most relevant ones. Section 2.3.5 analyses them.

2.3.1 EKD

EKD (Enterprise Knowledge Development; (Bubenko, Persson, Stirna, 2001)) was developed at Royal Institute of Technology. It provides a systematic and controlled way of analysing, understanding, developing and documenting an organization and its components.

The purpose of EKD is to provide a clear and unambiguous picture of how an organization operates at a given moment, what are the requirements and the reasons for change, what alternatives could be devised to meet these requirements, and what are the criteria and arguments for evaluating these alternatives.

EKD consists of six models:

- Goals model, which focuses on describing the goals of an organization, i.e., what the organization and its employees want to achieve or to avoid and when; this model analyses goals, problems, causes, constraints and opportunities, and uses AND/OR decompositions for goals.
- Business rules model, which is used to define and maintain explicitly formulated business rules, consistent with the goals model; business rules may be seen as operationalization or limits of goals.
- Concepts model, which is used to define the "things" and "phenomena" that are present in the other models and includes organizational concepts, attributes, and relationships.
- Business process model, which is used to define organizational processes and the way they interact and handle information (as well as material); a business process is assumed to consume inputs (of information or material) and produce outputs.
- Actors and resources model, which is used to describe how different actors and resources are related to each other and how they are related to components of the goals model and of the business processes model.
- Technical components and requirements model, which defines requirements for the development of an information system; this model focuses on the technical system that is needed to support the goals, processes and actors of an organization.

2.3.2 ARIS

As EPC, ARIS (ARchitecture of ntegrated Information Systems; (Scheer, 2000)) arose from a project between SAP AG and the University of Saarland. ARIS aims to provide a framework spanning the gap between business requirements and ISs, i.e., to provide a precise way of expressing business processes and to allow effective communication and detailed analysis of them. It also aims to provide an unambiguous basis for the development of the necessary IS to support business processes.

ARIS provides a structure to organise different types of model and objects of an organization and to define their relation to each other. ARIS structure consists of five views (i.e., models):

- Organization view, which represents static models of the structure of an organization; it includes departments, people resources and roles in hierarchical organisation charts, technical resources (e.g., equipment and transport) and communication networks.
- Data view, which represents static models of business information; it includes data models, knowledge structure, information carriers, technical terms and database models.
- Function view, which represents static models of process tasks; it includes function hierarchies, business objectives, supporting systems and software applications.
- Product/service view, which represents static models of the structures of products and services; it includes product trees, products and services.
- Process (control) view, which depicts dynamic models showing the behaviour of processes and how they relate to the resources, data and functions of the business environment; it includes EPCs, information flow, materials flow, communications diagrams, product definitions, flow charts and value chain diagrams.

The first four views focus on the structure of an organization, while the process view focuses on the dynamic behaviour of the business process and brings together all the different elements of the other views. Use cases can be specified as a link between the organization and the function views. Class diagrams can also be used in the data view.

2.3.3 UML-Based Approaches

Many organizational modelling-based RE approaches that focus on business process modelling are based on UML. These approaches adopt and adapt UML diagrams and principles for creation of organizational models. Some of them are the following ones.

(Eriksson, Penker, 2000) is probably the most cited UML-based approach for organizational modelling. It proposes an approach that is based on the extension of UML by means of stereotypes. Extensions are targeted at modelling of business processes, resources, goals, business rules and relationships. The approach uses four different views (i.e., models):

- Vision view, which describes a goal structure for an organization and illustrates problems that must be solved in order to reach goals.
- Process view, which represents the activities and value created in an organization and illustrates the interaction between the processes and resources in order to achieve the goal of each process.
- Structure view, which shows the structures among the resources in an organization.
- Behaviour view, which represents the individual behaviour of each important resource and process.

With regard to other UML-based approaches, (García Molina, et al. 2002) propose an approach that is based on the OOram three-model architecture and the IDEA method for elicitation of use cases and creation of OO conceptual schemas from organizational models. As RUP (Kruchten, 2003), this approach mainly focuses on identification of business processes and organizational actors and addresses specification of business use cases. The approach also addresses determination of business rules and information objects.

In (Marshall, 2000), UML is used for modelling of purpose, processes, entities and structure of an organization. Finally, other works have addressed derivation of use cases and class diagrams from activity diagrams (e.g., (Rodríguez, et al., 2009)).

2.3.4 Communication Analysis

Communication Analysis (España, González, Pastor, 2009; España, et al., 2011; González, et al., 2011) is an approach that has been published recently. Differently from other organizational modelling-based RE approaches, business processes are modelled and analysed from a communicative perspective, not from a behavioural perspective.

In Communication Analysis, ISs are means to support organizational communication. Consequently, it focuses on communicative interactions that occur between an IS and its environment. In addition, this approach does not only consider ISs from a software perspective, but also considers them from an organizational and social perspective.

Communication Analysis proposes a requirements structure that allows successive refinements for ISs description though five levels:

- System/subsystems level (L1), which refers to an overall description of an organization and its environment; it also involves decomposition of the problem to reduce its complexity.
- Process level (L2), which refers to business process descriptions both from a dynamic viewpoint (by identifying flows of communicative interactions, aka communicative events) and a static viewpoint (by identifying business objects).
- Communicative interaction level (L3), which refers to the detailed description of each communicative event (e.g., the description of its associated message) and each business object.
- Usage environment level (L4), which refers to capture of the requirements related to the usage of a software-based IS, the design of the user interfaces and the modelling of object classes that will be stored in the IS memory.
- Operational environment level (L5), which refers to the design and implementation of the software components and architecture of a software-based IS.

Levels L1, L2 and L3 belong to the problem space. They do not presuppose the computerisation of an IS and aim to discover and describe the communicational needs of users. In contrast, Levels L4 and L5 belong to the solution space. They specify how the communicational needs are going to be supported by a software-based IS.

2.3.5 Analysis and Discussion

Business process-based approaches for organizational modelling are analysed on the basis of their support for achievement of the objectives 3 to 7 of the thesis (Section 1.4). The objectives 1 and 2 have not been considered because they are out of the scope of the approaches, although it must be noted that they influence on the objective 3.

Table 1 summarises the analysis. The support that the approaches provide for achievement of each objective is represented by means of symbols. If an approach (or a set of approaches) is considered to properly support achievement of a goal (i.e., it provides mechanisms or guidance that allow the objective to be achieved without problems), then the corresponding cell contains the symbol "+". If it is considered that the support that is provided should be improved, then the corresponding cell contains the symbol "+/-". Finally, if it is considered that improper or no support is provided, then the corresponding cell contains the symbol "-".

This kind of analysis has also been performed for approaches for specification of system requirements and for approaches for link of system requirements with OO conceptual schemas (Sections 2.4 and 2.5). It must also be indicated that some readers may not agree on the results of the analyses, i.e., on the rating of the support of the approaches as proper, not completely proper or improper. Explanation of the reasons for rating is provided to try to reduce disagreement.

EKD provides support for achievement of all the objectives except for derivation of state transition diagrams. Nonetheless, most of the support that provides should be improved. Effect of system purpose on business processes is weakly addressed, and EKD does address issues such as bridging the gap between business and system domains, specification of quality requirements and homogeneous granularity of system requirements. Finally, EKD data models (conceptual schemas) are less detailed than class diagrams.

The main problem of ARIS is that it does not deal with too many objectives of the thesis. In addition, it only properly supports one objective (specification of system requirements that determines support for business process). ARIS should also provide more guidance to bridge the gap between business and system domains, and the degree of detail of the system requirements and of the class diagrams should be higher.

UML-based approaches present an important weakness that can hinder their application: its diagrams might be difficult for stakeholders to use, understand and validate (Dobing, Parsons, 2000; Siau, Cao, 2001). This is a result of its focus on the system domain. UML-based approaches should also improve their mechanisms and guidance to bridge the gap between business and system domains. For example, direct map of business task to use cases assumes that they have the same granularity, which it is not always the case. Finally, the degree of detail of the class diagrams that are derived should also be improved. More and more specific information should be included.

The support that Communication Analysis provides for the challenges that it tackles (and are common to this thesis) can be regarded, in general, as good. It is especially important (in relation to this thesis) its focus on homogeneous granularity of requirements to increase their quality (España, et al., 2009). Among its weaknesses, Communication Analysis does not address analysis of system purpose for requirements elicitation and of its effect on the business processes of an organization. It may also improve specification of system requirements by addressing quality requirements.

Despite the weaknesses that have been indicated for the business process-based approaches for organizational modelling and thus their lack of completely proper support for achievement of the objectives of the thesis, the contributions that the approaches have made and their importance cannot be denied. Furthermore, the ideas and mechanisms that they propose could be modified or combined with other techniques to mitigate most of the weaknesses.

Table 2.1 Analysis of business process-based approaches for organizational modelling

Objective	EKD	ARIS	UML-based	Communication Analysis
3	+/-	-	+/-	-
4	+/-	+/-	+/-	+/-
5a	+	+	+	+/-
5b	+/-	+/-	+/-	+/-
5c	+/-	-	-	+
6	+/-	+/-	+/-	+
7	-	-	-	+

2.4 Approaches for Specification of System Requirements

Specification (aka modelling) of system requirements is the activity of the RE process that is related to documentation of the requirements of a software system that belong to the system domain. The output of this activity is a SyRS.

Specification of system requirements can be considered the main activity of the RE process because all the others are influenced by or are targeted at it. Requirements elicitation aims to discover the system requirements that will be specified. Requirements analysis aims to examine the system requirements that have been discovered and may be specified and thus implemented. Requirements validation is concerned with the adequacy of the system requirements for fulfilment of stakeholders' needs. Requirements negotiation is performed on the basis of the system requirements that should be implemented. Finally, requirements management is related to the control of the changes that may occur in the system requirements of a SyRS.

There exist many and very different styles and approaches for specification of system requirements (Davis, 1993; Kotonya, Sommerville, 1998; Lauesen, 2002). As notations for business process modelling, each style for specification of system requirements has specific purposes and different strong points and weaknesses. Therefore, their use will depend on the purpose of specification of system requirements. For example, a purpose can be specification of user requirements.

The approaches that have been considered to belong to the state of the art of this thesis are those that address determination of support for business processes (objective 5a). In addition to the approaches that are reviewed in this section, the styles and approaches of the organizational modelling-based RE approaches that have been reviewed in Section 2.3 are also considered approaches of the state of the art for specification of system requirements.

The most relevant approaches related to this thesis for specification of system requirements and the styles proposed for specification are presented in the following subsections. Section 2.4.4 analyses the approaches.

2.4.1 Scenario-Based Approaches

Scenarios (Alexander, Maiden, 2004) aim to find possible ways to use a software system to accomplish some desired function. They are based on the idea of a sequence of actions that have to be performed by a user and by a software system.

Within scenario-based approaches, different styles exist. Probably the most famous one is use cases. Other well-known styles are user stories, misuse cases and storyboards.

Scenarios are related to this thesis because they usually use the language of the application domain and aim to facilitate agreement upon system support for business processes and to interrelate system functionality and business processes (Weidenhaupt, et al., 1998). In addition, the simplicity of scenarios facilitates communication between system analysts and customer stakeholders.

Scenarios usually deal with three parts of a software system. The system context refers to descriptions of the broader environment in which the system is embedded (e.g., an organization in the case of an IS). System interaction covers how the system interacts with its environment (e.g., users). Finally, internal system refers to internal interactions among system components.

Many and different templates for specification of scenarios-based system requirements can be found in literature. A typical template contains this information:

- Name of the scenario
- Actors that participate
- Goal that should be achieved by executing the scenario
- Main story
- Variations of the main story
- Exceptions of the main story
- Preconditions for execution of the scenario
- Postconditions after execution
- Non-functional requirements that constrain or affect the scenario

2.4.2 Task and Task & Support Descriptions

Task descriptions (Lauesen, 2002) are based on a simple but important idea: a software system must support user tasks. They focus on requirements at the domain (business) level, and aim to specify adequate support for business tasks and for what users and software systems should achieve together.

For this purpose, the work areas that will be affected by a system are determined and task descriptions for each area are specified. A task description has a specific goal, and a user performs the task and either achieves the goal or cancels the whole activity.

The main difference with the scenario-based approaches is the focus on the collaboration between a user and a system, in contrast to the focus on the definition of the actions of a software system and the interactions with it. The information that is specified in a task description is:

- Name of the task
- Purpose of the task
- Trigger/Precondition for execution
- Frequency and critical situations of execution of the task
- Sub-tasks and their sequence
- Variants during execution of the task

As a natural following step, task & support descriptions address specification of the software-based solutions that can be provided for task descriptions. Each sub-task is analysed so that its existing problems and possible solutions (system support) are determined.

The main advantages of task and task & support descriptions are that stakeholders find them easy to validate, focus on understanding of the application domain and facilitate validation. An initial weakness was that they did not address data requirements. However, this problem was solved by linking them to Virtual Windows, in which the pieces of data that a system has to show so that a user performs a task are determined.

Finally, a study on effectiveness of task descriptions and use cases is presented in (Lauesen, Kuhail, 2011). Task descriptions better address problem analysis and link of requirements with the application domain.

2.4.3 Business Transactions-Based Approaches

Approaches that focus on specification of business transactions for software systems have appeared recently (Chalin, Sinnig, Torkzadeh, 2008; Correa, Werner, 2004). Their authors argue that most of the works on business transactions deals with them from a design perspective instead of from a requirements perspective. However, modelling of business transactions and concurrency management can be considered domain activities and thus should be analysed during the RE process.

These approaches aim to provide proper means for an integrated specification of functional and business transaction requirements. They are related to this thesis because they focus on analysis of the application domain, system support for business (transactions) and precision in SyRS. They also deal with documentation and analysis of business information for elicitation of system requirements (e.g., business rules).

For specification of business transactions, the approaches adapt and extend use cases. Since use case descriptions mainly focus on the sequence of interactions between actors and the system, it is common to see analysts trying to specify interaction details before having a precise knowledge of the underlying transaction results that should be achieved. Three types of use cases are considered: transactional use cases, support use cases and data extraction use cases.

The information that is included in business transactions-based use cases is:

- Business transactions to be supported by the use case
- Input for the use case
- Expected results
- Main scenario for execution of the use case
- Scenarios for abortion situations on the basis of actor's decision
- Scenarios for detection of accesses to transactional resources by a software system
- Scenarios for failures in access to transactional resources
- Response time out for a system
- Policies for management of failures in data storage

2.4.4 Analysis and Discussion

The approaches for specification of system requirements have been analysed on the basis of the support that they provide to achieve the objectives of the thesis. The objectives 1, 2, 3, 6 and 7 have not been considered in the analyses because they are out of the scope of the approaches. Therefore, just the objectives that are related to the second research question of the thesis have been considered.

Although all the approaches that have been reviewed address all the objectives that are related to the second research question, none of them provides adequate support to achieve the objectives. The main problems of the approaches are that: 1) they focus on definition of the information to specify but do not explain how to obtain the information (e.g., from business process models); 2) they do not pay too much attention to quality requirements, and; 3) the guidance for homogeneous granularity should be more detailed, specific and objective to facilitate its application.

The definition of criteria for assurance of homogeneous granularity of system requirements is missing in most of the RE approaches in general and in those reviewed in this section in particular. There exist some criteria, but most of them need to be more precise so that they are easy to apply. Clear examples are goals of use cases (Cockburn, 20001) and closure and "coffee break test" (Lauesen, 2002). If precise and objectives criteria for homogeneous granularity are not defined for system requirements, then it could be difficult to achieve and validate.

The validity of the approaches for specification of system requirements should not be doubt in spite of their weaknesses. The approaches are sound and have been and are successfully applied in many software development projects. Their problem in relation to this thesis is that they are not targeted at some of its objectives.

Table 2.2 Analysis of approaches for specification of system requirements

Objective	Scenario-based	Task and task & support descript.	Business transact.-based
4	+/-	+/-	+/-
5a	+/-	+/-	+/-
5b	+/-	+/-	+/-
5c	+/-	+/-	+/-

2.5 Approaches for Link of System Requirements with OO Conceptual Modelling

Creation of OO diagrams (e.g., OO conceptual schemas) from or in conjunction to specification of system requirements has been considered a necessary step in software development projects since the appearance of the first approaches for OO development approaches (e.g., (Jacobson, et al., 1992)). The most common practice has been the combination of use cases and class diagrams, which can be considered to complement each other for system modelling (Siau, Lee, 2004).

Although link of system requirements with OO diagrams is clearly important for software modelling and development, most of the approaches that deal with this step present an important weakness: they do not provide means to avoid problems such as incompleteness of OO diagrams and inconsistency of OO diagrams and SyRSs. As a solution to this weakness, many approaches have focused on provision of mechanisms and guidance in order to try to avoid potential problems. Among these approaches, those considered to be most related to the thesis are reviewed in this section.

All the approaches that are reviewed share a common characteristic: they specify system requirements from a scenario-based perspective. In this sense, use cases are used as style for SyRS and as starting point for the derivation of OO conceptual schemas. Depending on the approach, details of use cases are specified in a given way so that an OO conceptual schema is created from their analysis.

In addition to the approaches that are reviewed in this section (which focus on link of system requirements with OO conceptual schemas), other four kinds of approaches exist for creation of class diagrams from system requirements.

The first kind corresponds to approaches that deal with design-level class diagrams (e.g., (Cox, Phalp, 2007)). These approaches are not reviewed because they address derivation of design characteristics of a software system. On the basis of conceptual modelling principles (Olivé, 2007), conceptual schemas must specify the knowledge that an IS needs to know, not the internal characteristics of the system. They correspond to analysis models about the application domain and do not include descriptions of software components (Larman, 2005).

The second kind of approaches correspond to linguistic-based approaches (e.g., (Overmyer, Lavoie, Rambow, 2001)), which address creation of class diagrams from analysis of textual specifications. They are not reviewed because they do not address specification of system requirements for support of business processes.

The third kind of approaches correspond to i*-based approaches that address derivation of class diagrams (e.g., (Castro, et al., 2001)). As in Section 2.3, these approaches are not reviewed because they focus on support of goals instead of on support of business processes when specifying system requirements and thus when deriving OO conceptual schemas. Another goal-based approach (although non-i*-based) is presented in (Liang, 2003). This approach analyses use case goals for identification of classes and of their properties, and subsequently for modelling of a class diagram.

The last kind of approaches corresponds to those that derive OO conceptual schemas from business process-oriented organizational models. They have been reviewed in Section 2.3.

There exists another type of works related to link of system requirements with OO conceptual models. Such works have studied business processes modelling from a data-centred perspective. They have not been presented as RE works, but as business process management works, thus they focus on analysis and design of business processes instead of on requirements elicitation and specifications.

These works have addressed issues such as the notion of business artefact (Nigam, Caswell, 2003), design of product-based workflows (Reijers, Liman, var der Aalst, 2003), detection of data flow anomalies (Shun, Zhao, Numaker, 2005) and document-driven workflows (Wang, Kumar, 2005). These works are not reviewed in detail in this section because they do not regard business process models as a means for understanding of the application domain and for elicitation of system requirements.

In (Kumaran, Liu, Wu, 2008), the correspondence between activity-centric business process models and information-centric ones is discussed. This work also shows how an activity-centric business process model can be transformed into a data-centric model.

With regard to state transitions diagrams (as a part of the OO conceptual schema of an IS), approaches that address the link of system requirements with OO conceptual models do not usually deal with them. Nonetheless, works related to this issue can be found in literature. Although they are not reviewed in depth because they do not aim to specify support for the business processes of an organization, some of the works are the following ones.

There exist works that have addressed derivation of state transition diagrams from scenarios in general (e.g., (Uchitel, Kramer, Magee, 2003; Whittle, Schuman, 2000)) and from use cases in particular (e.g., (Ratcliffe, Budgen, 2005)). Scenarios are represented by means of message sequence charts or sequence diagrams, and the derivation of state transition diagrams from them is usually called synthesis. For synthesis, formal descriptions and algorithms are provided by the works. However, these works address design aspects of a software system and are not oriented towards conceptual modelling, as discussed above for the works that deal with derivation of design-level class diagrams.

Another stream of related work includes works that have combined business process models and state transition diagrams. However, such a combination has been targeted at model checking (e.g., (Bhattacharya, et al., 2007; Eshuis, 2006)) and at consistency between the models (e.g., (Küsters, Ryndina, Gall, 2007)). They do not aim to link system requirements to OO conceptual models as part of IS development, but to guarantee that business process models are correct.

Last but not least, two systematic reviews related to the link of system requirements with OO conceptual modelling have been published recently. They reviewed RE approaches for model driven development (Loniewski, Insfrán, Abrahão, 2010) and transformation approaches between user requirements and analysis models (Yue, Briand, Labiche, 2010). Nonetheless, both works have focused on review of technical issues when deriving class diagrams from system requirements, not on provision of mechanisms and guidance to avoid potential problems when addressing the derivation.

The next subsections present the approaches for link of system requirements with OO conceptual schemas that have been regarded as the most suitable ones for achievement of the objectives of the thesis. Section 2.5.5 analyses the approaches.

2.5.1 RETO

RETO (Insfrán, Pastor, Wieringa, 2002) was developed at Universidad Politécnica de Valencia. As this thesis, it aims to provide a RE approach for OO-Method.

The approach addresses the problem of software engineers that do not know if an OO conceptual schema meets user requirements. The intended solution is based on the provision of guidance to link user requirements (represented through the TRADE framework) to an OO-Method conceptual schema in a traceable way.

System requirements are specified by means of three complementary parts. A mission statement describes the purpose of the system in one or two sentences. A function refinement tree deals with partition of external interaction according to the different business areas or business objectives of an organization. Finally, a use case model includes two parts: 1) a use case diagram to show the communications between the actors and the system, and; 2) a use case specification in the form of textual templates for scenario-based specification in order to determine the composition of external interactions. Therefore, each use case is analysed at two levels: at the use case diagram level and at the use case specification level.

Functionality to support use cases is allocated in the classes of an IS by analysing the use case diagram. For each step described or implied in a use case specification, a responsibility (or a set of responsibilities) is identified and a system analyst has to allocate it to a class. Such a class may have been previously identified or it may be necessary to define a new one.

Sequence diagrams are used in order to deal with the activity of identifying responsibilities and allocating them into class components of an IS. These diagrams show how classes participate in and are affected by the execution of a use case, and specify the interactions between the classes.

Sequence diagrams are also enriched by specifying the messages that classes send to others. Several types of messages are defined, and UML stereotypes are used to differentiate them. Once the sequence diagrams have been modelled, a set of rules allow derivation of part of an OO-Method conceptual schema in the form of a class diagram.

2.5.2 ADORA

ADORA (Analysis and Description Of Requirements and Architecture; (Glinz, Berner, Joos, 2004)) is an approach for OO modelling of software systems that was developed by researchers from University of Zurich. It provides a lightweight approach for consistency between a scenario model and a class model (Glinz, 2000).

This approach is based on the fact that OO requirements specifications typically combine a scenario (or use case) model and a class model for expressing functional requirements. With such a combination, the problem of consistency between the two models arises. Therefore, ADORA aims to provide mechanisms and guidance that ensure that the information in these models is neither contradictory nor partially incomplete.

Nearly all requirements modelling techniques that use more than one model have no systematic approaches to combine the models consistently. The consistency problem is simply ignored and thus left to the system analysts and to the users, who have to validate a requirements specification.

A scenario model and a class model are considered to be consistent if: 1) there are no contradictions between the information in the scenario model and the information in the class model (both where information is shared and where the models interact), and; 2) there is no partial incompleteness with regard to the other model. A partial incompleteness is a situation where information that is present in one model requires corresponding information in the other model.

Consistency is achieved in ADORA by minimizing overlaps between the two models and by systematically cross-referencing corresponding information. A set of model construction and checking rules is provided both for developing a consistent specification and for checking the consistency of a completed specification.

In summary, this approach allows systematic identification of information in a class model that corresponds to information in a scenario and vice-versa. It provides elementary conformance rules that can be checked automatically, have rules for inspecting corresponding information and can systematically detect both contradictions and information that is missing on either side (partial incompleteness).

2.5.3 SCORES

SCORES (Kösters, Six, Winter, 2001) was developed at University of Hagen. Similarly to ADORA, it is based on the fact that inconsistency and incompleteness may arise between use cases and class models. For SCORES, the reason is that the models are based on different modelling techniques and aim at different abstraction levels.

SCORES proposes a method for coupling of use case and class models. Since the class model provides a finer granularity and more rigorous semantics compared to the use case model, use cases are refined to achieve more precise specifications. For this purpose, activity graphs are used. Therefore, granularity and semantics of the refinement allow transition of use cases via activity graphs to a class model

An activity graph is a variation of a state machine in which the states represent the development of actions or sub-activities and the transitions are triggered by the completion of the actions or sub-activities. An activity graph focuses on a single modelling element (e.g., an operation, a class or an entire system). Consequently, it cannot model the behaviour of more than one interacting modelling element. In particular, it cannot cope with interaction information (differently from sequence diagrams). Furthermore, an activity graph is not able to capture associations to actors and «include» or «extend» relationships between use cases.

In SCORES, the use cases are refined by specifying actions. Actions together with basic control flow information derived from narrative descriptions of business tasks are composed into activity graphs.

The class model obtained comprises the most important classes of the domain with their responsibilities, root operations, some initial attributes and relationships. In an incremental, iterative process, analysts explore the activity graphs in more detail in order to extract all the modelling elements involved in the execution of the corresponding use cases.

In summary, SCORES addresses precise modelling of use case behaviour in terms of refined activity graphs covering internal, interaction and contextual information. It also addresses seamless, traceable transition of use cases to a class model via activity graphs. In addition, SCORES addresses validation of a use case model and the verification of a class model against a use case model.

2.5.4 Info Cases

Info Cases (Fortuna, Werner, Borges, 2008) was developed at Federal University of Rio de Janeiro. As ADORA, it addresses the problem of the joint use of use case models and class (domain) models.

There are difficulties in this use, mainly when trying to obtain a class model from a use case model or when trying to maintain consistency between them. This approach proposes a specialization of use cases called info cases, from which a class diagram can be derived by means of semi-formal rules.

Two steps must be taken to solve the difficulties of the joint use of use case models and class models. First, the elements of the class model must be systematically captured while modelling use cases. Second, these elements must be precisely represented within the description of the use cases. As a solution, this approach provides an integrated model capable of capturing, in a single conceptual framework, the elements involved in use cases and in the class model.

Use cases must have value for a stakeholder and make achievement of some of his goals possible. This means that a change of state in the system and in its environment is made. The state of the system when the goal is achieved must be a steady state, that is, consistent with the state of the environment in which the system is introduced. Therefore, the state is free of any need for rollback to a previous state, even if no other use case is activated subsequently. This partitioning criterion defines a level of abstraction for the elicitation of use cases, which is called informational level of objectives.

The flows of information exchanged between the actors and the system in each use case are used to capture the elements of a class diagram. These flows are called information flows, and are exchanged through the informational interface of the use cases.

An info case is a use case of the informational level of objectives, with its informational interface specified by means of information flows. Information flows, which are capable of capturing elements of a class diagram, have two parts: a specification of the composition of flow and a dictionary of elementary items of information. For determination of the elements of a class diagrams from information flows, several rules are provided.

2.5.5 Analysis and Discussion

The three first objectives of the thesis have not been considered for analysis of the approaches for link of system requirements with OO conceptual schemas because they are out of the scope of the approaches. The objectives that are related to the second and third research question have been considered because most of the approaches deal both with specification of system requirements and with derivation of OO conceptual schemas.

A common weakness has been found in all the approaches that have been reviewed: none of them address derivation of state transition diagrams. Both static and dynamic properties are important for IS conceptual modelling (Olivé, 2007; Pastor, Molina, 2007). They neither address the gap between business and system domains nor determination of support for business processes.

In addition, they do not explicitly address specification of quality requirements and do not provide guidance for determination of all the parts of a class diagram (classes, attributes, methods and associations). In this sense, their class diagrams are incomplete. Finally, just info cases deals with homogeneous granularity of use cases.

As for the rest works reviewed in the previous sections of this chapter, the existence of weaknesses in the approaches for link of system requirements with OO conceptual modelling does not imply that they do not have strong points. Furthermore, identification of both the strong points and the weaknesses of the approaches has been vey important for development of the proposed solution of the thesis, as well as for awareness of the problems that may arise and the approaches address.

Table 2.3 Analysis of approaches for link of system requirements with OO conceptual schemas

Objective	RETO	ADORA	SCORES	Info Cases
4	-	-	-	-
5a	-	-	-	-
5b	+/-	+/-	+/-	+/-
5c	+/-	+/-	+/-	+
6	+/-	+/-	+/-	+/-
7	-	-	-	-

2.6 Summary

This chapter has presented the state of the art of the thesis. Five categories of works have been reviewed, and the adequacy of the works of each category for achievement of the objectives of the thesis has been analysed and discussed.

The main conclusions of the review of the state of the art are that:

1. BPMN and Map are probably the notation for business process modelling and the goal-oriented RE approach, respectively, that best fit the challenges, needs and objectives of the thesis, and;
2. No existing RE approach (business process-based approach for organizational modelling, approach for specification of system requirements or approach for link of system requirements with OO conceptual modelling) allows achievement of all the objectives of the thesis, i.e., no approach properly addresses all the challenges (research questions) of the thesis.

Although some objectives could be achieved by using some approaches, other objectives would require changes on the approaches or combination with other approaches. In fact, this is the line that has been followed in this thesis for development of the proposed solution. Instead of proposing a completely new approach, the methodological approach presented is based on many ideas, mechanisms and guidelines proposed in previous works. This fact is further explained in the next chapters.

Chapter 3
Fundamentals of the Proposed Solution

"Those are my principles, and if you don't like them... well, I have others"

Groucho Marx

Before describing the stages and the evaluation of the methodological approach of the thesis, this chapter introduces several aspects on which the proposed solution is based or that have been analysed or defined as part of its design and development. References to these aspects are made throughout the thesis, thus it is necessary to present them in order to understand the explanation of the methodological approach.

First, a definition of business process is proposed and the design of the methodological approach is presented and discussed. Next, a stakeholders taxonomy and a requirements taxonomy for this thesis are described. The top ten principles of the proposed solution are then explained, and finally the correspondence between business process models and goal models is discussed. To conclude, a summary of the chapter is presented.

3.1 Definition of Business Process

Definitions of what a business process is are abundant, and many authors have their own definitions. Some of the most widely known and used are the following ones:

- A (business) process is a specific ordering of work activities across time and place, with a beginning and an end, and clearly identified inputs and outputs: a structure for action (Davenport, 1993).
- A business process is a collection of activities that take one or more kinds of input and creates an output that is of value to the customer (Hammer, Champy, 2001).
- A business process is a set of one or more linked procedures or activities which collectively realise a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships (WfMC, 1999).
- A business process is the complete and dynamically coordinated set of collaborative and transactional activities that deliver value to customers (Smith, Fingar, 2002).
- A business process consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations. (Weske, 2007).

Definitions are usually very similar, thus it could be argued that the notion of business process is straightforward. However, some authors (Lindsay, Downs, Lunn, 2003; Melão, Pidd, 2000) have studied the nature and definitions of business process in depth and have identified different perspectives for their modelling (deterministic machine, complex dynamic system, feedback loop and social constructs). They argue that it is essential to know and understand the nature and possible perspectives of business process modelling in order to properly define what a business process is and thus to model them adequately.

On the basis of these works and previous definitions, the following definition of business process is proposed and adopted in this thesis:

A business process is a set of structured and ordered activities that are performed in an organization to achieve some business goal. A business process takes inputs from the business environment and creates outputs, and is executed coordinately and dynamically by people and/or technical components that exchange information.

3.2 Design of the Methodological Approach

The methodological approach of the thesis, which corresponds to the proposed solution, has been introduced in Chapter 1. It consists of four stages: organizational modelling, purpose analysis, specification of system requirements and derivation of OO diagrams. In this section, the existing approaches and ideas on which its design has been based are discussed.

As explained in Chapters 1 and 2, no existing RE approach properly addresses all the challenges of the thesis. Nonetheless, it does not mean that they are useless for development of the thesis. In this sense, the strategy followed for the design of the methodological approach has been to try to use (i.e., adopt, adapt, modify, extend…) as many existing ideas and mechanisms as possible, instead of developing and proposing completely new artefacts. If different existing works can make achievement of specific objectives of the thesis possible, then the combination of the works will make achievement of several or of all of them possible.

Although BPMN (OMG, 2009) and Map (Rolland, 2007) have been the reference works for design of the organizational modelling and purpose analysis stages of the methodological approach, EKD (Bubenko, Persson, Stirna, 2001) has had a great influence on them too. Many of its models have been adopted, adapted and extended, especially focusing on modelling business processes according to the definition proposed. The main weakness of EKD is probably the lack of focus on the purpose of an IS for understanding of the application domain and for elicitation of system requirements. This weakness has been addressed by including a Map-based purpose analysis stage.

For specification of system requirements, a new style has been defined. It is called Extended Task Description (ETD), and it is influenced by previous works. ETDs are mainly based on Lauesen's Task & Support Descriptions (Lauesen, 2002). Among other changes, Lauesen's style has

been extended by describing user-system interaction by means of essential use cases (Constantine, Lockwood, 1999), by including information flows based on those of the Info Case approach (Fortuna, Werner, Borges, 2008), and by specifying quality requirements on the basis of the ISO 9126-1 standard (ISO, 2001).

For derivation of OO diagrams, some existing ideas and mechanisms have been used. For example, consistency rules from ADORA (Glinz, 2000) have allowed identification of relevant potential problems that may arise. When limitations were found in other works, ways to overcome them were studied and included.

Figure 3.1 shows the stages and artefacts of the methodological approach. It is based on the assumption that an organization has a problem or need that could be fulfilled by an IS.

The first stage depicts the current organizational environment (As-Is), in which the problem or need exist. The organization will change to solve the problem (To-Be), and the change will have an effect on its business processes. Once the new business processes are modelled, systems requirements are specified from them and OO diagrams are derived from system requirements.

Participation and involvement of customer stakeholders in the process is essential. Apart from being the source of information from which the activity of and organization and system requirements are discovered, they must validate that the BPDs of an organization are correct (i.e., they properly depict the organizational activity), agree on the effect of an IS on the business processes and validate the system requirements.

It must be noted that Figure 3.1 depicts and ideal sequential process for execution of the methodological approach, what does not exactly correspond to reality. In actual executions, the process can be iterative (in fact it is the usual way), and incomplete or insufficient information in a stage can be discovered in subsequent stages.

In addition, although the information and models of the As-Is part are not included in the To-Be part (in order to keep Figure 3.1 as small as possible), they will also be represented and maintained in the To-Be situation of an organization. For example, changes in business processes may imply changes in other artefacts of the organizational modelling stage (roles model, business rules, etc.), and they must be reflected.

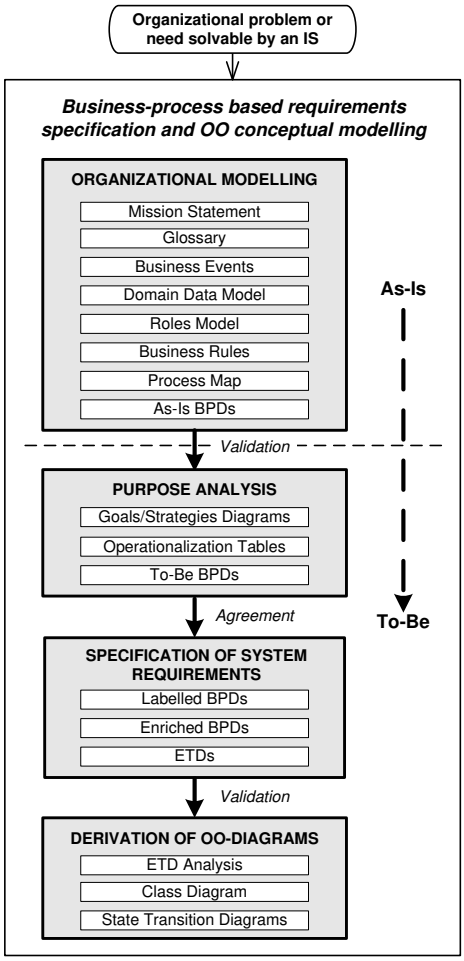

Figure 3.1 Stages and artefacts of the methodological approach

In summary, and as explicitly stated as possible in the research methodology followed (Vaishnavi, Kuechler, 2008), stages and artefacts of the thesis themselves may not very novel, but the contributions are primarily in the design and construction processes of the artefacts. Therefore, an overall contribution of the thesis is to show how combination of existing works can make achievement of its objectives possible. This contribution is made by proposing extensions to these works, new mechanisms and new guidance, and is also in line with the needs of integration and combination of existing RE approaches and of provision of systematic guidance to apply them (Cheng, Atlee, 2007).

When reviewing literature, approaches whose design is similar to the methodological approach can be found. Modelling and analysis of As-Is and To-Be situations of an organization is common when addressing the RE process of an IS (Pohl, 2010), business process management (Becker, Kugeler, Rosemann, 2003) or business process reengineering (Carr, Johansson, 1995). For example, the PRiM approach (Grau, Franch, Maiden, 2008) models both situations. In (Berenbach, et al., 2009), a model-driven RE process is presented on the basis of five models: business model, feature/goal model, use case (analysis) model, design model and implementation model.

The stages of the methodological approach of the thesis and the creation processes of their artefacts are presented in the next chapters, as well as further details about the combination of existing RE approaches.

3.3 Stakeholders Taxonomy

This section presents and defines a stakeholders taxonomy for the thesis. The types of stakeholders of the taxonomy correspond to people and roles that may be affected by business process-based requirements specification and OO conceptual modelling of an IS.

Even though other stakeholders taxonomies or terminologies can be found in literature (e.g (Alexander, Beus-Dukic, 2009; Berenbach, et al., 2009; Lauesen, 2002)), it is important to explicitly determine what types of stakeholders are mainly addressed in this thesis and what people and roles are referred to when mentioning a type of stakeholders. Otherwise, misconceptions may appear.

Figure 3.2 shows the taxonomy in the form of a class diagram, and Table 3.1 shows the correspondence among several taxonomies and the one proposed. The types of stakeholders of the taxonomy of the thesis are defined as follows.

- Stakeholder: a stakeholder is a person (or group of people) that has an interest in an IS and whose opinions, needs or preferences are likely to be relevant to the success of the system.
- Customer stakeholder: a customer stakeholder is a stakeholder that is part of the customer side of an IS development project and corresponds to a person that works for the organization for which an IS is going to be developed.

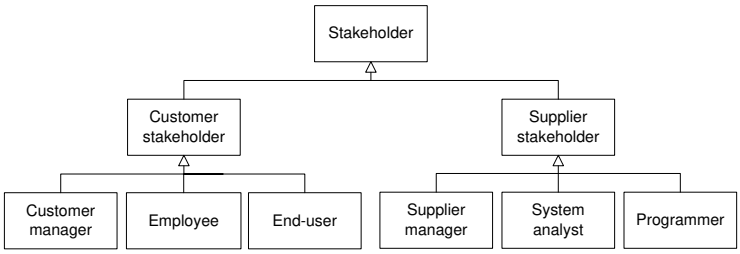

Figure 3.2 Stakeholders taxonomy

- Customer manager: a customer manager is a customer stakeholder that corresponds to a person that is in charge of the management, control and decision making of some part of an organization (e.g., a plant manager).
- Employee: an employee is a customer stakeholder that corresponds to a person that performs the operational work of an organization and is not a customer manager.
- End-user: an end-user is a customer stakeholder that represents the people that will need to use an IS to perform their work.
- Supplier stakeholder: a supplier stakeholder is a stakeholder that is part of the supplier side of an IS development project, i.e., a person that works for the software development company that is going to develop an IS.
- Supplier manager: a supplier manager is a supplier stakeholder that corresponds to a person that is in charge of the management, control and decision making of some part of a software development company (e.g., a project leader).
- System analyst: a system analysts is a supplier stakeholder that corresponds to a person that interacts with customer stakeholders during the RE process in order to specify the system requirements of an IS; this type of stakeholder is sometimes referred to as requirements engineer (see Table 3.1) or business analyst (e.g., (IIBA, 2009)).
- Programmer: a programmer is a supplier stakeholder that corresponds to a person that is in charge of the actual development and implementation (i.e., coding) of an IS.

Table 3.1 Correspondence among stakeholders taxonomies

This thesis	(Alexander, Beus-Dukic, 2009)	(Berencbach, et al., 2009)	(Lauesen, 2002)
Stakeholder	Stakeholder	Stakeholder	Stakeholder
Customer stakeholder	Functional beneficiary	Customer, business stakeholder	Customer, sponsor
Customer manager	Sponsor, champion	Buyer	Manager of the departments
Employee	Expert	Expert	-
End-user	Normal operator	User	Daily user
Supplier stakeholder	Manufacturer	Supplier, technical stakeholder	Software supplier
Supplier manager	Product manager	Development/project manager	-
System analyst	Operational support	Requirements engineer/analyst	Analyst, requirements engineer
Programmer	Developer, maintenance	Developer-architect-designer	Programmer

3.4 Requirements Taxonomy

In addition to the stakeholders taxonomy, a requirements taxonomy for this thesis is defined too. The types of requirements of the taxonomy correspond to aspects and characteristics of an IS that must be considered for business-process based requirements specification. Nonetheless, the taxonomy could be used for requirements elicitation and specification of ISs in general.

As with the stakeholders taxonomy, there exist other requirements taxonomies and classifications (e.g., (Aurum, Wohlin, 2005b; Glinz, 2007; Lauesen, 2002)). Nonetheless, it is again important to present and define the requirements terminology that is used in this thesis.

The purpose of the definition of a requirements taxonomy is that readers know what a requirements-related term refers to when used so that ambiguity and misinterpretation are avoided. These problems may appear when different people use different semantics for common terms. For example, different definitions and interpretations exist for very common terms such as system requirement (Davis, 2003) and non-functional requirement (Glinz, 2007).

It must also be indicated that other authors may refer to the types of requirements of the taxonomy with other terms or use the terms of the taxonomy with other semantics. For example, some authors use the term "software requirement" to refer to the term "system requirement" of the taxonomy or the term "goal level requirement" to refer to "strategic requirement" (e.g., (Lauesen, 2002)).

Figure 3.3 shows the requirements taxonomy in the form of a class diagram, and Table 3.2 shows the correspondence among other requirements taxonomies and the one proposed. The types of requirements of the taxonomy are defined as follows.

- Requirement: requirements are activities, capabilities or conditions that an IS must support, possess or meet, respectively, to fulfil stakeholders' needs.
- Business requirement: business requirements are requirements that belong to the application domain (i.e., to the organizational/business environment).
- Strategic requirement: strategic requirements are business requirements that specify goals or objectives that must be achieved so that an organization succeeds.
- Operational requirement: operational requirements are business requirements that specify actual activity (daily work) of an organization; operational requirements support strategic requirements.

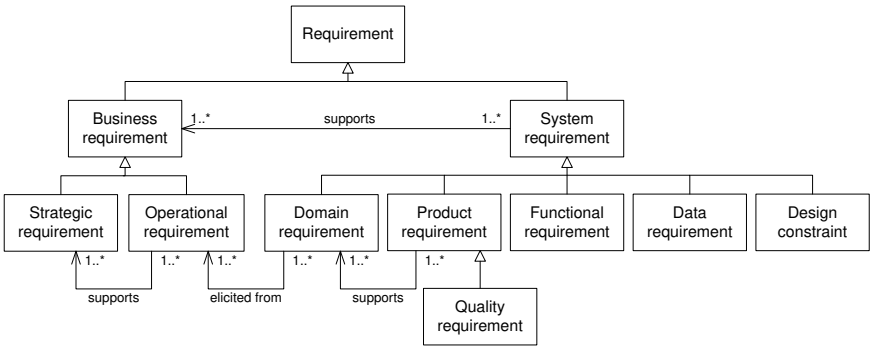

Figure 3.3 Requirements taxonomy

Table 3.2 Correspondence among requirements taxonomies

This thesis	(Aurum, Wohlin, 2005b)	(Glinz, 2007)	(Lauesen, 2002)
Requirement	Requirement	Requirement	Requirement
Business requirement	Business requirement, primary requirement	-	Domain level requirement
Strategic requirement	Goal level requirement	-	Goal level requirement
Operational requirement	-	-	-
System requirement	Product level requirement, technical requirement	System requirement	Product level requirement
Domain requirement	Domain level requirement, derived requirement	Functional requirement	-
Product requirement	Product level requirement, design level requirement	System requirement	Design level requirement
Quality requirement	Non-functional requirement	Performance requirement, specific quality requirement	Quality requirement
Functional requirement	Functional requirement	Functional requirement	Functional requirement
Data requirement	-	Functional requirement	Data requirement
Design constraint	-	Constraint	Design constraint

- System requirement: system requirements are requirements that belong to the system domain; system requirements support business requirements.
- Domain requirement: domain requirements are system requirements that are elicited from operational requirements.
- Product requirement: product requirements are system requirements that are not elicited from operational requirements; they are elicited from stakeholders, usually from end-users; product requirements support domain requirements.

- Quality requirement: quality requirements are product requirements that specify quality characteristics that an IS must possess.
- Functional requirement: functional requirements are system requirements that specify actions that an IS shall do or restrictions on these actions.
- Data requirement: data requirements are system requirements that specify the pieces of information than an IS shall store and manage or restrictions on these pieces of information.
- Design constraint: design constraints are system requirements that specify global restrictions (affect the whole system) on how an IS has to be designed.

Although the expressions system requirements specification (SyRS) and specification of system requirements are not part of the requirements taxonomy, and their interpretation may be straightforward for many readers, it must be indicated the difference that is made between them in the thesis. SyRS is the document in which system requirements are specified and documented, whereas specification of system requirements is the process performed to create a SyRS. This difference is not always clear in literature. For example, sometimes authors use the expression requirements specification to refer both to the document and to the process. In this thesis, ETDs are proposed as a style of SyRS.

3.5 Top Ten Principles of the Proposed Solution

This section presents the main principles on which the proposed solution is based. They represent issues that should be addressed in the RE process of any IS and thus are addressed in the methodological approach of the thesis. Although most of them have already been introduced in the previous chapters, more and new emphasis on their importance for the thesis is considered important.

There are top ten principles, which have been adopted from both literature and practice, i.e., from literature review and from evaluation of the methodological approach. There exist more principles that have had influence on the proposed solution and are important, but the following ones can be regarded as those that most strongly have influenced on the methodological approach.

1) Understanding and knowledge of the application domain are preconditions for requirements elicitation and specification

When developing any software system and performing the RE process, understanding and knowledge of the application domain are keys for success and practically preconditions for adequate requirements elicitation and specification (Jackson, 1995) and subsequently for business/IT alignment (Reich, Benbasat, 2000).

In the case of ISs, the application domain corresponds to the organization and the organizational environment in which a system will be deployed and used. Lack of knowledge of the application domain may cause an IS not to fit the actual needs of the environment and of the customer stakeholders. Repair of problems derived from this fact may be highly costly for a software development company, both in time and in money.

2) ISs must support the business processes of an organization

The need and importance of business process modelling during the RE process of an IS for an organization has been largely justified in Chapter 1 and, therefore, is one of the bases of the thesis. The business processes of an organization must be considered during the RE process so that system requirements support them, and domain requirements must be part of any SyRS of an IS. Nonetheless, business process modelling may not be sufficient.

Business process models must be analysed so that the gap between business and system domains is bridged and correct system support for business tasks is specified. In addition, it may be important to delay the splitting of work between an IS and its users. Before proposing solutions, the problem to be solved by an IS (organizational needs to be fulfilled and business processes to be supported) must be understood (Loucopoulos, Karakostas, 1995).

3) System analysts must be aware of the purpose of an IS

The need of a new IS in an organization is the consequence of the existence of some business need that should be fulfilled or of some business problem that should be solved by (using) the IS. For example, an organization may need an IS to remain competitive or to increase competiveness. If no need or problem existed, there would not be reason for developing the system.

These needs and problems correspond to the purpose of the IS, and system analysts must be aware of it so that clear understanding of system objectives exist (Alexander, Beus-Dukic, 2009) and the system addresses them (Rolland, Salinesi, 2005). If system analysts disregard its purpose, then an IS may not fit the actual goals that an organization pursues.

Furthermore, modelling and detailed analysis of the purpose may be necessary, for instance, for determination of the ways to achieve the goals of an organization and of how these ways affect the organization (Yu, 1995). Such an analysis may not be straightforward (Antón, 1997), thus provision of mechanisms and guidance to perform it is important.

4) Customer stakeholders' involvement during the RE process must be promoted

Customer stakeholders' involvement is recognised as very positive and necessary for the RE process in general (Sommerville, Sawyer, 1999) and for organizational modelling in particular (Stirna, Persson, Sandkuhl, 2007). It is also important for business/IT alignment (Reich, Benbasat, 2000). Its lack can lead an IS to failure.

One of the issues that must be properly addressed to promote customer stakeholders' involvement is communication with them. Communication can be difficult because of differences in vocabularies and backgrounds between customer stakeholders and system analysts (Berenbach et al., 2009), and misunderstandings may appear. Therefore, models and notations that facilitate communication between system analysts and customer stakeholders must be used in the RE process.

5) System requirements must be specified using an external view

Although it may be regarded as surprising, the notion of system requirement might be considered a recurrent problem both in academia and in industry. Different people usually have different perceptions of what a system requirement is and what it is not, and abstractions based on criteria such as the difference between what and how a software system shall do can be confusing or impractical (Davis, 1993; Kotonya, Sommerville, 1998).

As a solution to this problem, the use of the criterion of external view is followed in the thesis: system requirements correspond to external observable characteristics of a software system that are required by stakeholders (Davis, 2003).

6) Different types of system requirements must be specified

Most of the RE approaches that are presented in academia usually just focus on a type of requirements (product, functional, data, quality…) and disregard other types. However, it is essential to consider several types of requirements so that a SyRS meets one of its necessary characteristics: completeness (Firesmith, 2005).

Specification of the types of requirements that are not addressed in a RE approach may not be trivial (from or in conjunction to specification of other requirements), and incompleteness and inconsistency may appear between system requirements in a SyRS.

7) System requirements must be specified homogenously

System requirements that belong to a same abstraction level (e.g., ETDs or use cases) must be specified homogeneously. Non-ambiguous criteria that assure this condition must be used in RE approaches.

Homogeneity is achieved in a SyRS if all its units of system requirements at a given abstraction level have the same granularity. When criteria and thus homogeneity do not exist, a specification can be regarded as inconsistent and problems may arise. For example, application of a RE approach may be hindered (Dutoit, Paech, 2002), comparison of system requirements may be difficult (Gorschek, Wohlin, 2006) and the quality of a SyRS may be negatively affected (España, et al., 2009).

8) SyRSs must be structured

The need of structuring a SyRS has been acknowledged as basic for any RE approach (Alexander, Stevens, 2002; Sommerville, Sawyer, 1999). Well structured specifications facilitate understanding, specification, validation and management of system requirements.

For structuring SyRS, the use of standard templates for specification is very useful and thus advisable. Templates indicate the information that must be gathered and described, allow system analyst to focus on such information and to be able to locate it, and allow customer stakeholders to more easily distinguish among the types of information that are included in the template.

9) System requirements must be linked to subsequent development stages

Once the system requirements of an IS have been elicited and specified as a result of the RE process, just a part of the work is finished. The purpose of most of the software development companies is to develop software systems, not just to determine its requirements.

As a result, the way to perform subsequent development stages from system requirements must be determined. Furthermore, problems such as inconsistency or incompleteness in artefacts of subsequent stages must be addressed to try to avoid them. Otherwise, the usefulness of the system requirements would clearly decrease, and even may cause a RE approach not to be used by a software development company.

10) Detailed guidance must be provided

Detailed guidance for elicitation and specification of all types of necessary requirements is essential so that a RE approach can be applied (Dutoit, Paech, 2002). If an approach does not provide detailed guidance, then different people from its designers[1] may have difficulties when applying or trying to apply it. For example, system analysts would know what information they would have to compile, but they may not know how to compile it.

In addition, results from application by different people may not be the same. Companies may also not use or may stop using a RE approach if they found difficulties or problems in their application because of lack of guidance.

3.6 Correspondence between Business Process Models and Goal Models

This section presents and discusses the correspondence between business process models and goal models. On the basis of this correspondence, combination of goal models (for specification of strategic requirements) and business process models (for specification of operational requirements) in the thesis is discussed and justified. Conclusions from

[1] When referring to the designers of the methodological approach of the thesis in this thesis, the PhD candidate and his advisor are referred to.

discussion are considered to apply to the RE process of an IS and to organizational modelling in general.

The following subsections explain the correspondence in detail. First, background and preliminary concepts are explained. Next, a running example and guidelines for derivation of goal tress from business process models are presented. Finally, the correspondence is discussed.

3.6.1 Background: Operational Goals

Business processes have goals that must be fulfilled during or after their execution (Kueng, Kawalek, 1997). There are sub-goals that denote important milestones within a business process and whose fulfilment is possible due to the actions of all the participants involved (Ould, 1995). These sub-goals are called operational goals, and indicate when the instance of a business process (model) can be considered completed (Bider, 2003). Therefore, an operational goal can be defined as an objective or state that must or may be reached in a business process and that indicates its completion.

In most of the existing notations for business process modelling (e.g., BPMN), the operational goals of a business process are implicitly declared in the structure of a business process model and the states of its resources and data entities. These entities and resources are input or output of the activities of a business process model, and their states can change and evolve during execution of a business process.

Since operational goals are implicitly part of a business process model, then it can be assumed that a business process model is equivalent to a goal tree (or model) and thus a goal tree can be derived from a business process model. Nonetheless, the correspondence between a business process model and a goal tree must be determined. If such a correspondence was found, then a business process model could be mapped into a goal tree from patterns of the business process model.

In addition to a business process model, a domain data model may be necessary for derivation of a goal tree from it. This model is a simplified class diagram that includes the entities that are used in a business process model and whose states change as a result of the execution of the business process. Entities and the relations between them (associations and aggregations) must be modelled.

3.6.2 Preliminary Concepts

This section defines several concepts on which derivation of goal tress from business process models is based. The concepts also aim to facilitate explanation and understanding of the derivation process.

A goal tree consists of operational goals that are decomposed into other goals or tasks by means of 'AND' and 'OR' decompositions. A task is an atomic activity that is performed to fulfil a goal. The contributions of other goals or tasks are necessary to fulfil an operation goal.

The semantics of an 'AND' decomposition is that all the descendant elements have to be fulfilled (for goals) or performed (for tasks) in order to fulfil the decomposed goal. For an 'OR' decomposition, the decomposed goal will be fulfilled when some of the descendant elements are fulfilled or performed. Therefore, 'OR' decompositions depict alternative ways to fulfil a goal.

Several concepts are defined to specify the guidelines for derivation of a goal tree from patterns of a business process model. These concepts might be complicated, but they are necessary to simplify the explanation of the guidelines. Figure 3.4 shows some patterns that are used to explain the concepts. The figure has been modelled with BPMN.

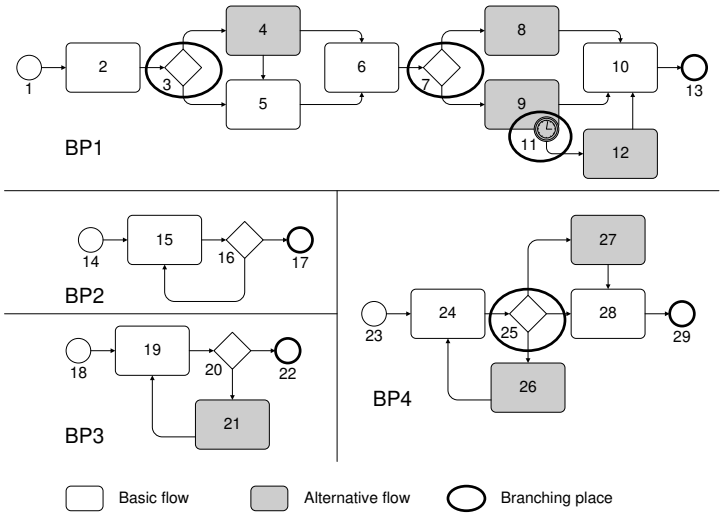

Figure 3.4 Patterns in business process models

- The basic flow of a business process model is the set of elements that are executed in all the instances of the business process.

  In Figure 3.4, the basic flow of BP1 is the set of elements {1, 2, 3, 5, 6, 7, 10, 13}.

- An alternative flow in a business process model is a set of flow objects that is not part of the basic flow of the model and does not have more than one connection to another flow (regardless whether the flow is basic or alternative).

  In Figure 3.4, the alternative flows of BP1 are the sets of elements {4}, {8}, {9} and {11, 12}. The set {9, 11, 12} is not an alternative flow because it would have two connections with the basic flow (9 and 12 with 10)

- A loop in a business process model is an iteration of a sequence of the elements of the model.

  In Figure 3.4, the sequence of elements {16, 15} is a loop in BP2.

- A loop with alternative executions in a business process model is a loop that contains elements that are part of the basic flow of the model as well as elements that are not.

  In Figure 3.4, the loop {20, 21, 19} in BP3 is a loop with alternative executions.

- An alternative execution of a loop in a business process model is each one of the possible executions of a loop with alternative executions. The sequence of elements of the loop that are part of the basic flow of the model is an alternative execution of the loop too.

  In Figure 3.4, the sequences of elements {19, 20} and {21, 19, 20} in BP3 are the alternative executions of the loop.

- A branching place of a business process model is a place in the model where:

  (a) an alternative flow begins, and;

  (b) not all the alternative flows that begin from it are part of a loop whose end condition is checked in the place.

In Figure 3.4, the branching places of BP1 are (3), (7) and (11). In BP4, (25) is a branching place too. However, place (20) in BP3 is not a branching place because it does not fulfil the second condition.

3.6.3 Running Example: The Garment Company

A garment company is used as a running example to show the derivation of goal trees from business process models. Figure 3.6 shows a BPD for the company, whereas Figure 3.5 shows a domain data model. The BPD corresponds to a business process for order processing, which is described as follows.

A secretary selects the next order that must be processed in the company. Each order contains the garments ordered by a client and the shipment destinations of the garments. The packing lists show the decomposition of an order by shipment destination. The garments are sent to the clients with a delivery note, and store managers select the destinations that must receive the garments first, i.e., they prioritise the destinations. Store operatives receive the delivery note and place the garments in the boxes to be shipped.

If there are not enough garments in stock to cover an order, then just the garments that are available are placed in the boxes. The store manager has to decide whether to wait for the rest of garments or to send the partial shipment (with fewer garments than ordered) to the destinations. The packing list is modified in either case and sent to a secretary, who creates the final version of the packing list. Then the delivery note is placed into the box and the shipment is prepared for delivery.

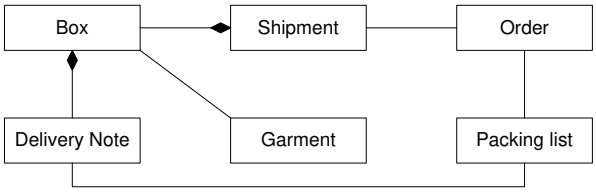

Figure 3.5 Example of domain data model

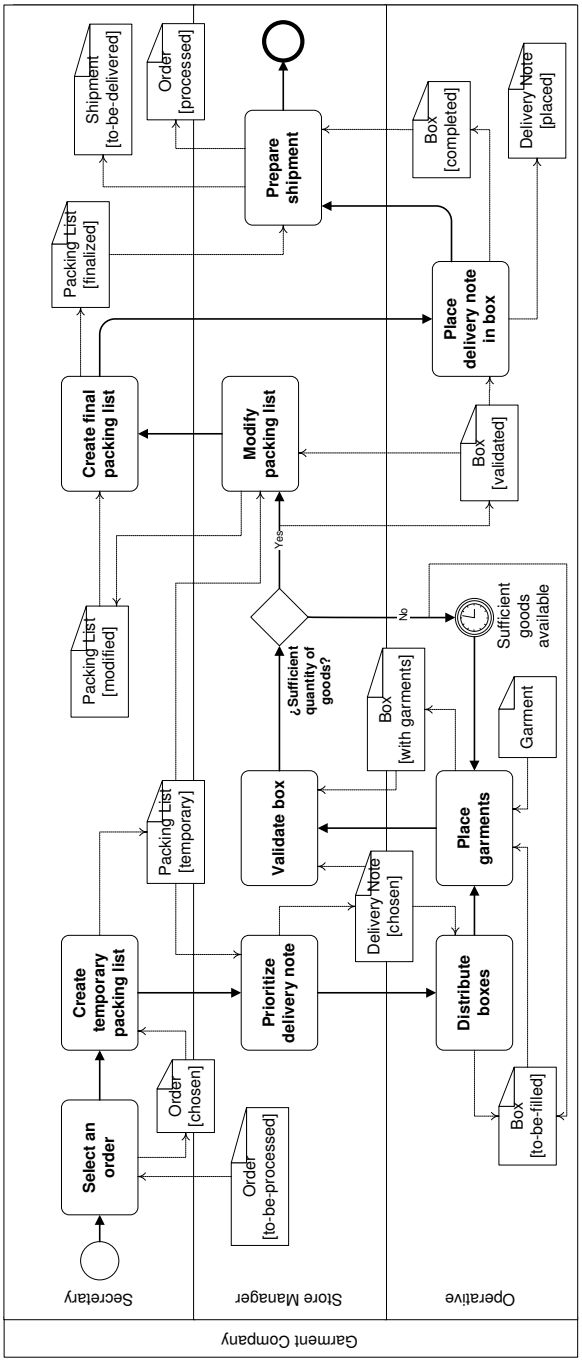

Figure 3.6 Example of BPD

3.6.4 Guidelines for Derivation of Goal Trees from Business Process Models

Possibility of derivation of goal trees from business process models was discussed and justified in Section 3.6.1 on the basis of the implicit (or explicit, depending on the notation) existence and modelling of operational goals in a business process model. This section presents the guidelines for derivation of goal trees, which are divided into four groups of guidelines: derivation, refinement, contribution and completion guidelines. For definition of the guidelines, BPMN terminology is used (BPD, sub-process, event, etc.).

Derivation guidelines allow goals and tasks to be defined and named. Refinement guidelines allow the type of decomposition of a goal to be determined. Contribution guidelines allow contributions of goals and tasks to the fulfilment of other goals to be determined. Finally, completion guidelines allow a goal tree to be finished.

The contribution guidelines and the refinement guidelines are applied together. For example, the refinement guideline R.1 needs a contribution guideline (guideline C.1) in order to define the descendant elements of the goal that refines it.

Table 3.3 shows a summary of the guidelines. It presents the mapping of BPD elements and patterns into elements of a goal tree (goals and tasks), as well as the type of decomposition and the elements of a goal tree that contribute to the fulfilment of each goal.

Figure 3.7 shows the goal tree derived from the BPD of the running example. The goal tree can be considered similar to a Tropos (Bresciani, et al., 2004) or a KAOS goal model (Dardenne, van Lamsweerde, Fickas, 1993). In relation to this fact, a combination of the notations of the i* framework for modelling of goals and tasks and of the structure of the KAOS goal model is used in the goal tree.

Table 3.4 shows the guidelines that have been applied to derive the goal tree of Figure 3.7. For each element of the goal tree, the guidelines applied for its derivation, refinement and contribution are specified. It must be noted that completion guidelines are not applied in the running example.

The next subsections present the guidelines of each group defined.

Table 3.3 Summary of guidelines to derive a goal tree from a BPD

BPD element	Element of a goal tree	Decomposition	Descendent element
BPD	Goal	AND	- Goals and tasks that do not contribute to another goal in the goal tree
Sub-process	Goal	-	-
Task	Task	-	-
Event with a trigger	Task	-	-
Loop with no alternative executions	Goal	AND	- Goals and tasks derived from the BPD elements of the loop
Loop with alternative executions	Goal	OR	- Goals derived from the alternative executions of the loop
Alternative execution of a loop	Goal	AND	- Goals and tasks derived from the BPD elements of the alternative execution
Branching place	Goal	OR	- Goals derived from the branches that follow the branching place
Branch that follows a branching place	Goal	AND	- Goals and tasks derived from the BPD elements of the branch
Data object	Goal	AND	- Goals and tasks derived from BPD elements that change the state of the data object and are not in a loop - Goals derived from loops that change the state of the data object - Goals derived from other data objects that are related to the data object by means of an inclusive aggregation relationship

3.6.4.1 Derivation Guidelines

There exist nine derivation guidelines, which are defined as follows.

Guideline D.1 (BPDs)

A BPD depicts a goal that corresponds to the root of a goal tree and is fulfilled when the business process ends. The name of the goal in the goal tree is the same as the name of the BPD.

Guideline D.2 (sub-processes)

A sub-process in a BPD depicts a goal in a goal tree that is fulfilled when the sub-process ends. The name of the goal in the goal tree is the same as the name of the sub-process in the BPD.

Guideline D.3 (tasks)

A task in a BPD depicts a task in a goal tree. The name of the task in the goal tree is the same as the name of the task in the BPD.

Guideline D.4 (events)

An event with a trigger in a BPD depicts a task in a goal tree (except link triggers, which are only used to link BPDs). The name of the task in the goal tree will depend on the criterion of the creator, but it has to refer to the event type (start, intermediate, final) and the event trigger (message, timer, cancel…).

Guideline D.5 (loops)

A loop in a BPD depicts a goal in a goal tree that is fulfilled when the loop ends. The name of the goal will depend on the criterion of the creator, but it has to refer to the condition that is fulfilled when the loop ends.

Guideline D.6 (alternative executions of a loop)

An alternative execution of a loop in a BPD depicts a goal in a goal tree that is fulfilled when the alternative execution is executed. The name of the goal will depend on the criterion of the creator.

Guideline D.7 (branching places)

A branching place in a BPD depicts a goal in a goal tree that is fulfilled when all the branches that follow the branching place end or merge into basic flow. The name of the goal will depend on the criterion of the creator.

Guideline D.8 (branches that follow a branching place)

A branch in a BPD that follows a branching place depicts a goal in a goal tree that is fulfilled when the branch ends or merges into basic flow. The name of the goal will depend on the criterion of the creator.

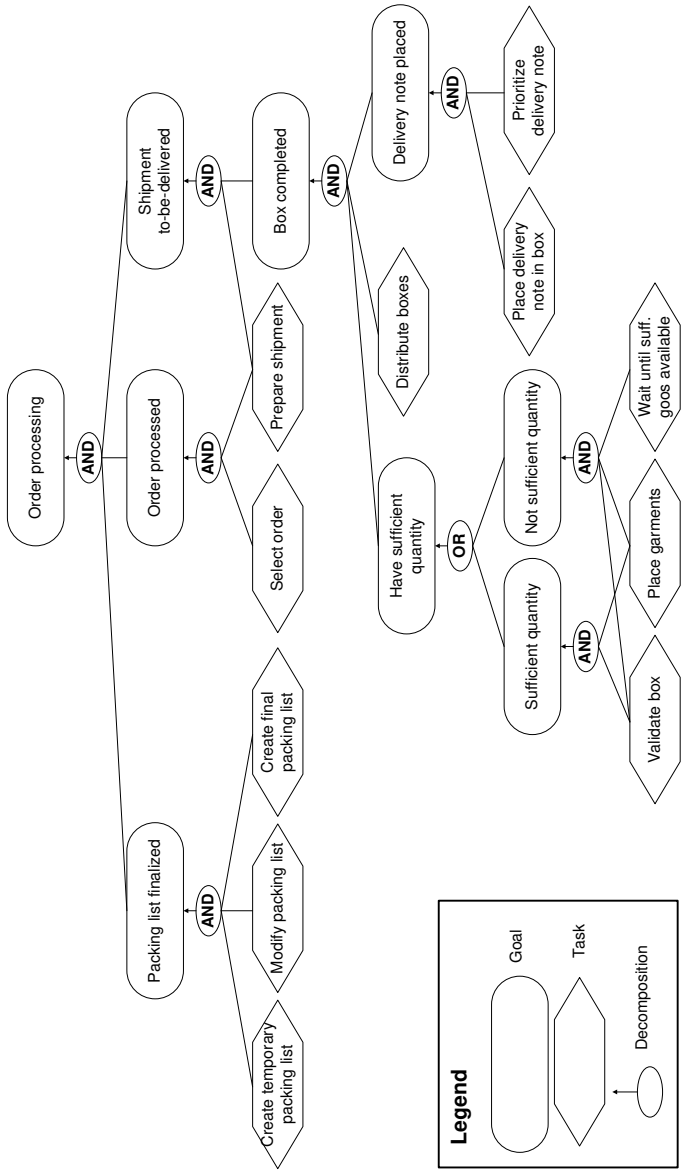

Figure 3.7 Example of goal tree

Guideline D.9 (data objects)

A data object in a BPD whose state changes during the execution of the business process depicts a goal in a goal tree that is fulfilled when the data object reaches the last of its states in the BPD. The name of the goal is the name of the data object in the BPD followed by the last state that the data object reaches.

3.6.4.2 Refinement Guidelines

There exist two refinement guidelines, which are defined as follows.

Guideline R.1 (BPDs, loops with no alternative executions, alternative executions of a loop, branches that follow a branching place and data objects)

A goal that is defined from a BPD, a loop with no alternative executions, an alternative execution of a loop, a branch that follows a branching place and whose first flow object belongs to an alternative flow, or a data object whose state changes during the execution of a business process, is refined in a goal tree by means of an 'AND' decomposition.

Guideline R.2 (loops with alternative execution and branching places)

A goal that is defined from a loop with alternative executions or a branching place is refined in a goal tree by means of an 'OR' decomposition.

3.6.4.3 Contribution Guidelines

There exist nine contribution guidelines, which are defined as follows.

Guideline C.1 (elements of a loop with no alternative executions)

The goals and tasks that are derived from the elements that are executed in a loop with no alternative executions contribute to the fulfilment of the goal of the loop in a goal tree.

Guideline C.2 (alternative executions of a loop)

The goals that are derived from the alternative executions of a loop contribute to the fulfilment of the goal of the loop in a goal tree.

Guideline C.3 (elements of an alternative execution of a loop)

The goals and tasks that are derived from the elements that are executed in an alternative execution of a loop contribute to the fulfilment of the goal of the alternative execution in a goal tree.

Guideline C.4 (branches that follow a branching place)

The goals that are derived from the branches that follow a branching place contribute to the fulfilment of the goal of the branching place in a goal tree.

Guideline C.5 (elements of a branch that follows a branching place)

The goals and tasks that are derived from the elements of a branch that follows a branching place and whose first flow object belongs to an alternative flow contribute to the fulfilment of the goal of the branch in a goal tree.

Guideline C.6 (data objects)

The goals and tasks that are derived from tasks and sub-processes of a BPD, are not executed in a loop and change the state of a data object contribute to the fulfilment of the goal of the data object in a goal tree.

Guideline C.7 (data objects in loops)

The goals that are derived from loops whose execution changes the state of a data object contribute to the fulfilment of the goal of the data object in a goal tree.

Guideline C.8 (inclusive aggregation relations between data objects)

The goals that are derived from a data object that is related to another data object in the domain data model by means of an inclusive aggregation relation (component data object) contribute to the fulfilment of the goal of the latter data object (composed data object) if defined in a goal tree.

Guideline C.9 (goals and tasks with no contribution)

The goals or tasks in a goal tree that do not contribute to the fulfilment of some goal contribute to the fulfilment of the root of the goal tree.

3.6.4.4 Completion guidelines

There exist two completion guidelines, which are defined as follows.

Guideline T.1 (goals with no descendants)

The goals that do not have descendants in a goal tree and that have not been derived from a sub-process are changed into tasks.

Guideline T.2 (goals with only one descendant)

The goals that have only one descendant are removed from a goal tree. The descendant will contribute to the fulfilment of those goals to which the parent goal contributes in the goal tree.

Table 3.4 Guidelines used in the running example

Element of the goal tree	Guidelines
Order processing	D.1 / R.1 / C.9
Packing list finalized	D.9 / R.1 / C.6
Create temporary packing list	D.3 / - / -
Modify packing list	D.3 / - / -
Create final packing list	D.3 / - / -
Order processed	D.9 / R.1 / C.6
Select order	D.3 / - / -
Prepare shipment	D.3 / - / -
Shipment to-be-delivered	D.9 / R.1 / C.6, C.8
Box completed	D.9 / R.1 / C.6, C.7, C.8
Have sufficient quantity	D.5 / R.2 / C.2
Sufficient quantity	D.6 / R.1 / C.3
Not sufficient quantity	D.6 / R.1 / C.3
Validate box	D.3 / - / -
Place garments	D.3 / - / -
Wait until sufficient goods available	D.4 / - / -
Distribute boxes	D.3 / - / -
Delivery note placed	D.9 / R.1 / C.6
Place delivery note in box	D.3 / - / -
Prioritize delivery note	D.3 / - / -

3.6.5 Discussion

Once background, guidelines and an example of the correspondence between business process models and goal models have been presented in the previous sections, this section discusses the implications that this correspondence has in RE in general and in this thesis in particular.

In general, most of the goal-oriented RE approaches (see Section 2.2) model and analyse the application domain of a software system with focus on the goals and agents (or actors) of a system. In contrast to this approaches, most of the business process-based approaches for organizational modelling (see Section 2.3) model and analyse the application domain with focus on the activities that are performed in an organization, their sequence and coordination, and the roles that perform the activities.

These perspectives could be regarded as distinct or even opposite because of the explicit focus on different aspects of the application domains (goals vs. activities). However, on the basis of the possibility of derivation of goal trees from business process models, the perspectives should not be regarded as distinct, but as complementary or even equivalent in some aspects (e.g., for modelling of operational goals).

The existence of a correspondence between business process models and goal models implies that goal models (or at least part of them) are implicitly created when modelling business process, and vice versa. Therefore, business process models allow specification of part of the information that is gathered and analysed in goal-oriented RE approaches, and goal models allow specification of part of the information that is gathered and analysed in of business process-based approaches for organizational modelling.

In relation to the thesis, the existence of such a correspondence allows to further justify the combination of BPMN and Map for business process modelling and purpose analysis, in addition to its adequacy for modelling and understanding of the application domain of an IS.

On the one hand, their combination allows all types and abstraction levels of goals of an organization and of an IS to be addressed. Strategic goals are modelled and analysed on the basis of Map, whereas operational goals are modelled and analysed on the basis of BPMN.

On the other hand, Map complements BPMN by allowing system analysts to analyse the purpose of an IS on the basis of the strategic goals of an organization, whereas BPMN complements Map by allowing system analysts to model details of organizational activity that cannot be modelled with the goal-oriented RE approach or whose modelling presents limitations.

The combination of BPMN and Map is the way proposed in this thesis for modelling and analysis of the application (business) domain and thus for specification of business requirements. Map allows specification of strategic requirements and BPMN allows specification of operational requirements.

In summary, the combination of BPMN and Map allows the proposed solution to embrace most of the information that is usually modelled in goal-oriented RE approaches. It also includes further information such as activity sequence. The combination also allows the proposed solution to embrace all the information that can be part of a business process model and represents the application domain, extending such information with details about the purpose and the goals of an IS. Finally, the combination is used for specification of business requirements, and subsequently for elicitation of system requirements.

Nonetheless, it must also be noted that business process models and goal models are similar and equivalent in some aspects, but not in all. Therefore, the selection of one of the types of models instead of the other should be justified and explained when modelling and analysing and organization or an IS so that the decision and the rationale behind it are clear.

The use of a type of models or of others when developing a software system will depend on the part or aspect of the application and of an IS with which system analysts and other stakeholders are mainly concerned. For example, business process models should be used instead of goal models when development mainly aims to support organizational activities and their sequence. In contrast, goal models can be considered better-suited when an organization is aware of its goals but not of its business processes. This situation may happen when new goals (needs) arise in an organization and they need to be analysed for development of a new IS or for definition a new business process to fulfil the goals.

3.7 Summary

This chapter has presented the fundamentals on which the proposed solution of the thesis is based. These fundamentals represent a summary of and a justification for many decisions that have been made during the design and development of the methodological approach of the thesis.

By explicitly and precisely defining what is meant by business process in the thesis, the information that should be depicted in business process models and that thus should be gathered for their creation is determined.

By introducing the design of the methodological approach, its relation with previous works has been determined, as well as the influence of the works on the thesis.

By presenting a stakeholders taxonomy and a requirements taxonomy, understanding of the terminology used in the thesis is facilitated. In addition, the taxonomies could be used as basis for presentation and explanation of other RE approaches for IS development.

By listing the top ten principles of the proposed solution, more emphasis on their importance for the development of an IS in general and for this thesis in particular has been placed. These principles could also be used for assessment and comparison of existing RE approaches.

Finally, by studying, presenting and discussing the correspondence between business process models and goal models, their equivalence, combination or use of one of the types of models for a modelling and analysis purpose can be justified.

Chapter 4
Organizational Modelling

"If you can't describe what you are doing as a process, you don't know what you're doing"

William Edwards Deming

The previous chapters have presented the main background work on which the methodological approach of the thesis is based. Once this work is known, presentation in detail of the approach starts in this chapter.

The chapter presents the first stage of the methodological approach (Figure 4.1): organizational modelling. The stage aims to model the structure and behaviour of an organization prior to development of an IS in order to gain knowledge and understanding about the application domain. The stage especially focuses on modelling the business process of the organization and is a first step for awareness of organizational needs and thus of requirements. In this sense, business process models depict operational requirements.

The chapter is organized as follows. First, an overview of the stage and a running example for its explanation are presented. Next, the artefacts created as a result of the development of the stage (Figure 4.1) are described. The descriptions include guidelines for creation of the artefacts. Finally, a summary of the chapter is presented.

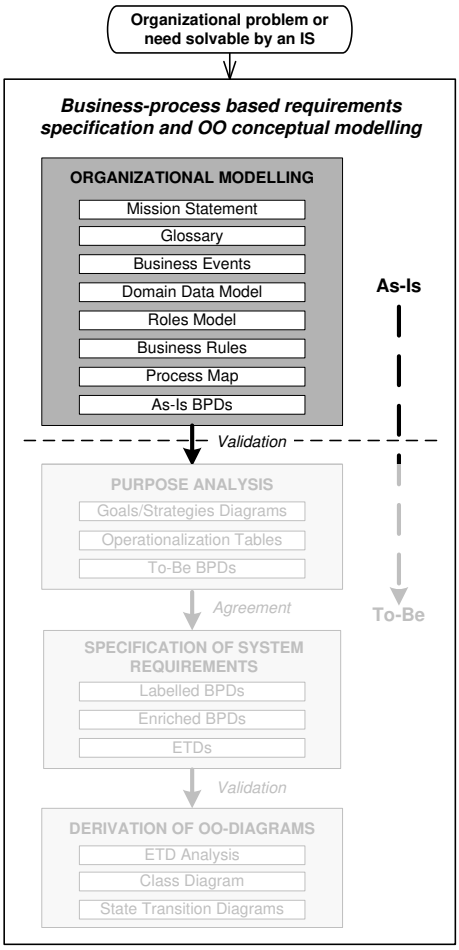

Figure 4.1 Stage and artefacts presented in Chapter 4

4.1 Overview of the Stage

As explained in Chapter 3, the main characteristic of the organizational modelling stage is the influence of EKD (Bubenko, Stirna, Persson, 2001) and the use of BPMN (OMG, 2009) for business process modelling. The stage is targeted at obtaining BPDs, as well as facilitating communication with customer stakeholder and thus their involvement. As a result, and as justified in Chapter 2, BPMN has been selected for business process modelling because it has strong points that can facilitate communication among all the stakeholders.

Figure 4.2 shows the activities that are performed in the organizational modelling stage. The artefacts that are created and managed are shown in Figure 4.1.

For modelling of the organization for which an IS is going to be developed, the information gathered is a mission statement, a glossary, the business events, a domain data model, the business rules, a roles model and a process map. This information is discovered by interviewing the employees of the organization so that they describe their work. Workshops with several employees can also be performed. In addition, it is advisable to look through the available documentation related to the organizational activity and the business policies.

The purpose of the information collected is to understand the business environment (i.e., the application domain) and to be able to model the business processes of the organization correctly (according to the definition proposed in Chapter 3). Missing or lack of information may imply that BPDs do not reflect the actual organizational activity. Nonetheless, system analysts may decide to directly model the As-Is BPDs of an organization or not to create all the other artefacts. What is considered important is that system analysts are aware of the information that a BPD must reflect and thus contain.

Although presented sequentially, creation of the artefacts related to the information gathered is usually performed in parallel. As employees are interviewed and documentation is checked, new details of the artefacts are discovered. In this sense, creation of the artefacts non-in parallel can be ineffective (Stirna, Persson, 2009). This also applies to modelling of As-Is BPDs.

As-Is BPDs are created from the information gathered, and a set of guidelines are provided to facilitate this activity. The diagrams must be validated by the customer stakeholders in order to guarantee that the organizational activity has been properly understood and thus modelled. Several iterations are usually necessary to get the final version.

With regard to the models and artefacts of this stage, other and more artefacts may be created during an organizational modelling stage. For example, business strategy (Bleistein, et al., 2006) or interactions with suppliers and customers (Gordjin, Akkermans, 2003) could be modelled. The proposal and use of more models is common in enterprise architecture frameworks (Lankhorst, 2009).

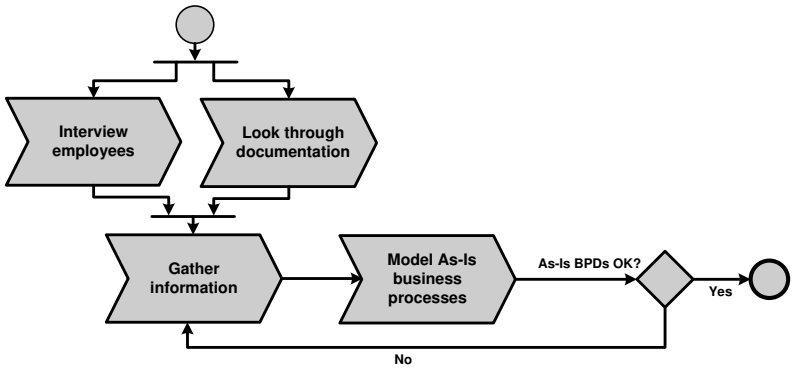

Figure 4.2 Activities of the organizational modelling stage

The use of some models or others depends on the purpose of organizational modelling and the perspective adopted for understanding of an organization and the application domain. Disregard of this fact can lead a modelling project to fail, for instance, if too much information or irrelevant information is gathered (Becke, Kugeler Rosemann, 2003). In this stage, the models and artefacts that have been considered more adequate and sufficient for business process-based organizational modelling have been included. Nonetheless, more models could be added, but their relationship with the ones defined should be defined.

Finally, although modelling of the As-Is situation of an organization is quite common and recommended, it may have negative points and thus people who are against it can be found (Becker, Kugeler, Rosemann, 2003). For example, it may be said that As-IS modelling is a very time consuming activity. In this thesis, the advantages of modelling As-Is situations (e.g., understanding of the current application domain and identification of shortcomings and possible improvements) are considered much more important than the possible negative points.

4.2 Running Example: The Software Development Company

When presenting the stages of the methodological approach of the thesis in Chapters 4, 5, 6 and 7, running examples are used to show how the stages are performed and how their artefacts are created. This is similar to the use of the garment company to show correspondence between business process models and goal models in Chapter 3.

The running examples correspond to illustrations (Wieringa, 2008). They do not aim to show validity of the methodological approach, but to facilitate its explanation and understanding. Although they could be regarded as simplistic, they are considered to be good to show the application of the methodological approach.

Actual (complete) examples or cases could have been used for explanation of the stages of the methodological approach. However, they would have implied the need to present more information about an organization (i.e., about the application domain) that is not relevant to show application of the methodological approach. Furthermore, their complexity may hinder understanding of the approach. As a solution, use of straightforward running examples has been decided.

In this chapter, a software development company is used as a running example. More concretely, the organizational activity related to product development will be used to explain organizational modelling and show its application.

The company (SwDepCo) develops software products that are provided to several customers. The products are standard, so no customer has a customized version of them. Nonetheless, customers can request improvements on the products, and they are included in future versions.

The following sections describe the artefacts of the organizational modelling stage and use the software development company to show examples of the artefacts.

4.3 Mission Statement

For understanding of an organization, the first thing that must be known is the main goal of its activity. The mission (statement) of an organization states why the organization exists and its basic purpose. A clear mission is also vital for discovery and understanding of the needs of an organization and the requirements of an IS for the organization (Alexander, Beus-Dukic, 2009).

Figure 4.3 shows a generic framework to show the relationship between the business processes of an organization and its mission. Business processes contribute to the achievement of the mission of an organization by making achievement of strategic goals possible, which

usually correspond to long term goals. That is, strategic goals are achieved by executing business processes, and achievement of such goals implies achievement of the organization mission.

In addition, it is common that indicators are defined in an organization for assessment of the achievement of strategic goals. In this sense, business processes are assigned the indicators that can be measured thanks to their execution. The mission and the strategic goals of an organization are considered intentional features, whereas indicators and business processes are considered operational features.

The above issues (strategic goals and indicators) are part of the definition of the strategy of an organization. Different models and approaches exist for this purpose, e.g., Porter's value chain (Porter, 1985) and the Balanced Scorecard (Kaplan, Norton, 1996). Nonetheless, strategy definition for an organization and its analysis are out of the scope of this thesis.

For the software development company, its mission statement can be defined as "provide customers with software products". This example is very abstract and could be used for any software development company, but it may be more concrete and detailed if specialisation of the company on development some type of software system or for some application domain was assumed. For example, a mission statement may be defined as "provide banks with information systems for investment management".

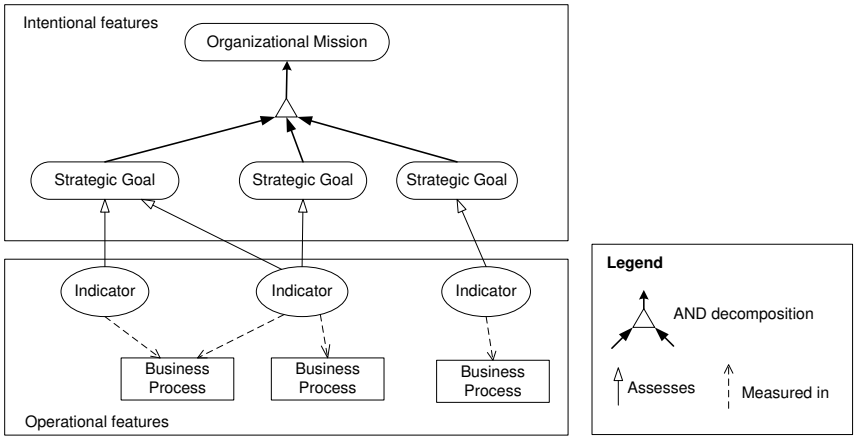

Figure 4.3 Relationship between organizational mission and business processes

4.4 Glossary

When developing the RE process of an IS for an organization, it is usual that the organization uses some terms that are unknown by system analysts or whose semantics is different to system analysts' perception. For example, the semantics of such a wide-spread term as invoice can vary among organizations and people.

In addition, it is common that a specific application domain has its own terms, which are not used in other domains and thus are unknown. For example, the term packing list for the garment company used as running example in Chapter 3 to show derivation of goal trees from business process models.

A glossary (aka as dictionary) is an artefact that contains the set of definitions of organizational terms that system analysts must know in order to understand organizational activity and to properly communicate with customer stakeholders (and possibly with other supplier stakeholders). If a term is unknown, then system analyst may have problems to communicate with or to understand customer stakeholders. Therefore, the need of a glossary is directly influenced by the need of understanding the application domain.

The purpose of a glossary is to define the organizational terms in an unambiguous and concise way, especially those that are hard to understand or whose semantics may be misinterpreted. A glossary is a very common and necessary artefact not only for organizational modelling, but also for requirements specification in general (Alexander, Beus-Dukic, 2009; Berenbach, et al., 2009; Lauesen, 2002).

The terms of a glossary are arranged by means of a list in which new terms are added in alphabetical order. The list may also be numbered, and acronyms and relationships between the terms may be specified (Alexander, Beus-Dukic, 2009; Berenbach, et al., 2009). For example, synonyms and homonyms may be specified in a glossary, or specialization hierarchies.

For the software development company, a term that may be part of the glossary is work unit. A work unit is a set of activities that must be performed so that a product is modified as a response to a customer request.

4.5 Business Events

Organizational activity is performed in an environment in which different happenings (or "things") can occur. Some of these happenings do not affect the organizational activity, but others are important and must be detected because they can trigger some specific behaviour and necessary response within the organization.

Business events are recurrent and significant happenings that occur in the environment (aka context) while the organization activity goes on and to which the organization must respond, i.e., happenings that are perceived in the environment and are pertinent to the organization (Olle, et al., 1991).

Business events are usually triggered by external agents of an organization, such as suppliers and customers. Nonetheless, they can also be triggered within an organization as a result of the development of its activity. For example, failure in a machine in a factory is a business event. The business events impact the organization and affect its behaviour.

In addition to for organizational modelling, business events are important for the RE process of an IS. They can be later translated into and thus correspond to events that must be captured or thrown by the system (Alexander, Beus-Dukic, 2009). Therefore, they must be discovered and specified.

In the organizational modelling stage, business events are specified by means of a table in which the name and a brief description about them are provided. This way, business events can be better understood, what also facilitates understanding of the application domain. For the software development company, Table 4.1 lists three business events: customer request, unable to finish on time and problem detected.

Table 4.1 Examples of business events

Name	Description
Customer request received	A customer submits an expected improvement on a product
Unable to finish on time	A developer is unable to finish an activity before the planned date
Problem detected	The product manager discovers a situation that may cause problems in version development

4.6 Domain Data Model

As specified in the definition of business process proposed in Chapter 3, the business processes of an organization take inputs from the business environment and create outputs. In addition, the participants (either human or technical) exchange information.

In the context of an IS for an organization, such inputs, outputs and information correspond to information about the domain (data) entities, i.e., the domain entities that are necessary for execution of the business processes. Furthermore, precise references to data models are important when modelling business processes as part of the RE process of an IS (Becker, Kugeler, Rosemann, 2003).

In the organizational modelling stage, a domain data model is created in order to facilitate understanding of the application domain and specify the domain entities that are necessary for execution of business processes. This model allows a system analyst complement the behavioural perspective of an organization with a data (or resource) perspective.

As explained in Chapter 3 when presenting background work for derivation of goal tress from business process models, a domain data model is a simplified class diagrams that includes the domain entities (aka classes or entity types) that are used in a business process model and whose states change in the business process. This model just includes entities and the relationships between, and the types of relationships considered are associations (aka binary relationships), aggregations (aka Part-Of relationships) and generalization/specialization (aka Is-A or inheritance relationships). Associations must be named.

It must be noted that more types of relationships can exist between and among domain entities (Olivé, 2007). Their consideration for creation of a domain data model will depend on the purpose of the model and on the degree of detail determined as necessary or satisfactory. For the organizational modelling stage, the above types of relationships are considered sufficient.

Since creation of a domain data model may be difficult (Batra, 2007; Svetinovic, Berry, Godfrey, 2005), especially for inexperienced system analysts, some explicit definitions and principles for identification of domain entities and of relationships between them are presented. They are based on the descriptions and recommendations provided in (Olivé,

2007; Parsosn, Wand, 1997; Parsons, Wand, 2000). Some reader may regard the definitions and principles as well-known or simple, but it is considered important to provide system analysts with as much guidance as possible in order to facilitate their work.

Discovery of the domain entities of an organization and their relationships is a classification activity that aims to structure a perception of the world and the knowledge about it. Classification assumes the existence of a concept and of an object to be classified. The classification operation consists in determining whether or not the object is an instance of the concept. For a domain data model, such concepts represent the domain entities and their relationships.

Domain entities are abstractions created by humans in order to describe useful similarities among things. Consequently, a domain entity can be defined an abstract description of a set of properties shared by a set of instances of the organizational environment.

A relationship exists between two domain entities if a mutual property is found. For a given relationship, each domain entity plays a role in the relationship, instances of both entities are necessary for the existence of the relationship, and the entities must correspond to different instances.

The relationships that are considered in a domain data model are the following ones:

- Association: it is the most generic relationship between two domain entities, and corresponds to mutual properties for which no specific patter (aggregation of generalization/specialization) is discovered.
- Aggregation: it is a type of relationships in which one domain entity plays the role of a part (P) and the other play the role of the whole (W); this relationship implies that:
    a) P is part of W;
    b) W is a composite formed by P (and possibly other entities), and;
    c) existence of P is only relevant and possible if related to W (i.e., if no instance of W exist, then no instance of P can exist).

- Generalization/Specialization: a domain entity S is a specialization of an domain entity G if all the instances of S are also instances of G; this implies that:
    a) S has the defining properties of G and some others;
    b) G is a generalization of S, and thus the defining properties of S include the defining properties of G;
    c) in any situation on which G is used, S can be used too;
    d) S is a subtype of G, and;
    e) G is a supertype of S.

For determining that a domain entity corresponds to a concept that represents a set of instances of objects of an organization and that its definition is cognitively correct, the domain entity must fulfil the following conditions:

- Abstraction from instances: a domain entity can be defined only if there are instances in the organization (relevant universe of instances) possessing all properties that define the domain entity.
- Maximal abstraction: a relevant property possessed by all instances of a domain entity should be included in its definition.
- Completeness: Given a relevant universe of instances and properties, every property should be used in the definition of at least one domain entity.
- Non-redundancy: A domain entity that is a subtype of another must have at least one property that is not in its supertype.

Cardinality is not specified in the relationships of the domain data model. Nonetheless, and on the basis of experience reviewing data models, another important issue for distinguishing different domain entities is to check the cardinality of their relationships. In this sense, the maximum and minimum cardinality of a relationship between two domain entities cannot be 1 (1:1). If so, both domain entities would represent the same concept and thus would correspond to a same domain entity.

An exception to this rule is shown in Figure 4.4. A country has just one capital and a capital belongs to just a country. Nonetheless, the

domain entity "capital" is a specialization of the domain entity "city", which is related to country via a non-1:1 relationship. In summary, a 1:1 relationship between two domain entities can exist if one of them is the specialization of another entity that is related to the other entity too and the cardinality of that relationship is not 1:1.

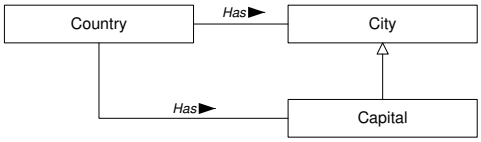

Figure 4.4 Example of domain data model in which the cardinality of a relationship can be 1:1

Finally, it must be noted that there is no a single "correct" set of domain entities to model the instances and properties of a given organization, i.e., no unique domain data model exists for an organization. The particular choice of domain entities (a view of an organization) depends on the purpose of a domain data model.

For the organizational modelling stage, its purpose is to find the set of domain entities (and the relationships between them) that are necessary for execution of the business processes of an organization because are inputs or outputs of its activities and represent information that the participants need and may exchange. Other domain entities such as physical elements of materials (that do not correspond to information necessary for execution of business processes) are not considered.

For the software development company, Figure 4.5 shows its domain data model.

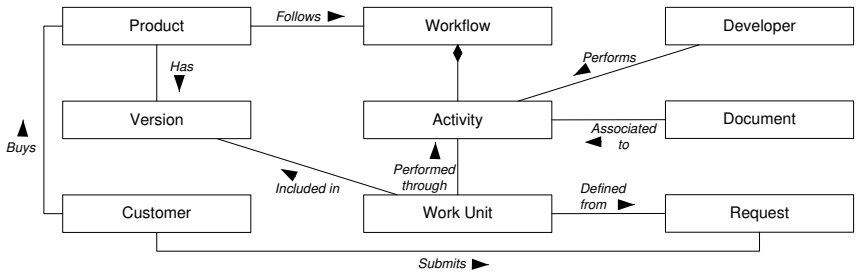

Figure 4.5 Example of domain data model

4.7 Roles Model

The activity of an organization is performed by its employees, who participate in the execution of its business processes. All employees do not perform the same activities or can perform any activity, but they are usually in charge of a part of the activities performed in the organization.

In addition, it is common that organizations are divided into different parts (i.e., organizational units), such as departments or sections. Nonetheless, on the basis of business process orientation in organizations (e.g., (Hammer, Champy, 2001)), employees from different organizational units usually participate in the execution of the business processes, thus they interact and share information and resources. Such employees play different roles for execution of the business processes. Therefore, a role represents a responsibility that a person (i.e., an employee) assumes when he holds a position in an organization (Bridgeland, Zahavi, 2009).

In the roles model, the organizational units and the roles that the employees that are part of them can play are specified. For each role, the set of activities in which the role is in charge is specified too.

An activity in a roles model represents an atomic action that the employees that play the corresponding role can perform. Action atomicity implies that:

a) once the action is started, its execution is completed or cancelled, and;

b) no alternative executions exist within the action (e.g., an activity cannot have alternative outputs).

These criteria define the granularity for an activity of the roles model. If development of an activity may be stopped and later resumed, then this fact should be specified by means of another activity. If alternative executions where found, then the activity should be decomposed in several actions.

It must be noted that execution of an activity may be stopped and cancelled because of happening of some event. In this case, the event would interrupt execution of the activity, and some response to the event would follow its happening. The above criteria must not be confounded or regarded as conflicting with the fact that an activity can be executed

several times (multiple instances or loop tasks and sub-processes in BPMN).

The roles model is created by means of a table that lists the activities, roles and organizational units of an organization. If considered necessary, a description of the activities may also be included. Table 4.2 shows an example of a roles model for the software development company. With regard to product development, the organizational unit (i.e., department) that participates in that organizational activity is "development". Among its roles, product managers and developers perform activities.

Table 4.2 Example of roles model

Organizational Unit	Role	Activity
Development	Product manager	Define product workflow
		Assign activities to developers
		Create product version
		Assign version to work item
		Check version development
		Solve problem
	Developer	Define work item
		Estimate activity
		Carry out activity

4.8 Business Rules

Another aspect of an organization that systems analysts must know to understand organizational activity and the application domain is the set of restrictions under which the organization operates. Such constraints correspond to the business rules of the organization.

A business rule is a statement that defines or constrains some aspect of an organization (The Business Rules Group, 2000). Business rules define the structure and behaviour of an organization and guide the activity of its employees (Bridgeland, Zahavi, 2009). They also explain what is allowed and what is not in an organization, and the consequences of violation of a business rule. A business rule can be classified as (The Business Rules Group, 2000):

- Derivation: it is a statement of knowledge that is derived from other knowledge of an organization.
- Structural assertion: it is a defined concept or a statement of a fact that expresses some aspect of the structure of an enterprise.
- Action assertion: it is a statement of a constraint or condition that limits or controls the behaviour of an organization.

It must be indicated that other classifications exist and may have been used for explanation of business rules (e.g., (Wan-Kadir, Loucopoulos, 2004)).

For the organizational modelling stage, just those business rules that do not correspond to information of other artefacts of the stage are specified. For example, the terms of the glossary or the assignation of activities to roles in the roles model can be regarded as business rules. In addition, other business rules are not considered important for the organizational modelling stage and thus are not specified at this stage of the methodological approach. For example, the attributes of the domain entities are not addressed.

As a result, derivations and structural assertions are not considered, and just action assertions are specified at the organizational modelling stage. Among them, the following types of action assertions must be discovered and specified by system analysts:

- Obligation: this type of business rules makes employees to behave in a given way; it is usually specified by means of "must-do" or "must-do-if" statements.
- Prohibition: this type of business rules prevents employees from performing a given behaviour; it is usually specified by means of "cannot-do" or "cannot-do-if" statements.
- Permission: this type of business rules allows employees to perform some behaviour under certain circumstances, e.g., violation of a business rules; it is usually specified by means of "can-do-if" statements.
- Response to event: this type of business rules can be regarded as a specialization of obligations, and defines what employees must do in case of happening of a business event; it is usually specified by means of "if-then-must" statements.

These types of action assertions are based on and defined from (Bridgeland, Zahavi, 2009). Differently from that work and other works (e.g., (Bubenko, Stirna, Persson, 2001)), no specific structure for specification of business rules is proposed or imposed in this thesis. Nonetheless, it may be defined. What is considered important is to classify business rules as belonging to some of the above types and that they refer to information of the other artefacts of the organizational modelling stage. For example, a response to an event must refer to a business event and to an activity of the roles model.

Business rules are specified textually by means of a bulleted list. For each business rule, its type must be specified. For the software development company, some business rules are the following ones:

- Customer requests must be managed by development in a week time (obligation)
- A developer cannot perform an activity that has not been assigned to him (prohibition)
- The product manager can define a work item for a customer request if it has not been managed by development in a week time (permission)
- If the event "unable to finish on time" happens, then the product manager must solve the problem (response to event)

4.9 Process Map

An organization performs a set of business processes to fulfil its mission. Nonetheless, the importance of all its business process is not the same. The influence of some of them on fulfilment of the mission is higher and thus their importance. This means that it is very important for a company that the most important business process are adequately executed, what implies activities such as control, monitoring and management.

As explained in Chapter 3, business processes have goals that must be achieved after their execution, and these goals are called operational goals. They also have inputs and outputs, and people participate in their execution. In addition, the business processes of an organization can be classified into three types (Ould, 1995):

- Core (aka operative) business processes: this category focuses on satisfying external customers of an organization; they directly add value in a way perceived by the customers, respond to customer requests and generate customer satisfaction.
- Support (aka supporting) business processes: this category focuses on satisfying internal customers of an organization; they might add value to the customers indirectly by supporting a core business process, or they might add value to the business directly by providing a suitable working environment.
- Management (aka strategic) business processes: this category focuses on managing core and support business processes, on planning and on key, success factors at the strategic level.

For the software development company, a core process is version development, a support process is definition of workflow, and a management process is agreement with customers. As can be observed, business processes are named in a noun-oriented way, similar to many existing works (e.g., (Harrington, 1993; Lankhorst, 2009; Smith, Fingar, 2002))

It must be indicated that other classifications for business processes exist and could have been used. For example, business process can be classified as enterprise, cross-process, process-specific and technical business processes (Silver, 2009), or as organizational and operational business processes (Weske, 2007).

In addition, although the organizational modelling stage is mainly targeted at discovery of intra-organizational business processes (Weske, 2007), shared business processes (with partners, suppliers or customers) would be considered if they were going to be supported by an IS. In this case, the problem or need solvable by an IS would affect several organizations, not only one.

The approach followed in the organizational modelling stage for determination of the business process of an organization is line with those works that considered that many business processes exist in an organization, and not just a small set of basic and high level business processes in which a customers triggers a business process and it finishes when the customer receives the product or service requested. Examples of first type of works are (Harrington, 1991; Ould, 1995; Smith, Fingar,

2002), which also provide (long) lists of possible business processes in an organization. Such a list can facilitate their identification.

Nonetheless, adoption of an approach or other in practice is flexible, and its criteria for decision are straightforward: 1) use the approach with which customer stakeholders feel most comfortable; 2) If no preference exist in customer stakeholders, then use the approach with which you (as system analyst) feel most comfortable.

The process map of an organization consists of the business processes that are identified on the basis of the information previously gathered and the perception of the customer stakeholders on the organizational activity. A process map is represented by means of two diagrams. First, a BPD is created and each business process discovered is included in the BPD in the form of a sub-process. Then the sub-processes are grouped on the basis of their type (core, support or management business process). Second, another BPD is created to specify the execution sequence of the business processes. It must be indicated that a different approach may be used (e.g., use of a list for each type of business processes instead of the first BPD).

In addition, each business process is initially described in a template. An example for the software development company is shown in Figure 4.6. The template includes the name of the business process and its type, responsible, participants, operational goals, inputs and outputs.

The responsible of a business process is an individual who is in charge of the correct and efficient execution of the business process. He is also in charge of detecting inefficiencies and of improving the business process, in close collaboration with the participants and (possible) process designers (Weske, 2007). Participants correspond to the roles of the employees that execute activities of the business process.

Inputs and outputs correspond to the domain entities that are necessary for execution of the business process and that are generated or modified, respectively. It is also common that the operational goals are related to the outputs, and more operational goals can be discovered and thus specified once a business process has been modelled (e.g., on the basis of the guidelines for identification of goals of business process-based goal trees presented in Chapter 3).

Business Process: **VERSION DEVELOPMENT**	
Type: **Core Process**	**Responsible**: Product Manager
Participants: Product manager, Employee	
Operational Goals: Work unit completed, Version finished	
Inputs: Product, Work unit	
Outputs: Version, Work unit	

Figure 4.6 Example of template of a business process

It must be indicated that different templates for description of business process exist and thus could be used. The one proposed is considered to fit the needs of business process modelling in the methodological approach of the thesis, but it could be modified. For example, if the strategy of an organization was determined or analysed too, then the template may include a section for specification of the indicators that can be measured in the business process.

4.10 As-Is BPDs

The last activity of the organizational modelling stage is to model the As-Is BPDs of an organization, which represent the current situation and way of working of the organization. As-Is BPDs are the artefacts targeted at when modelling an organization, and correspond to the first important representation about the application domain (combination of several perspectives and artefacts) in the methodological approach.

As-Is BPDs allow system analysts to represent the structure and behaviour of an organization (from a business process-based perspective) in a single model, and correspond to the style for specification of operational requirements (see Chapter 3) in the methodological approach. To guarantee that they are correct, customer stakeholders must validate them and agree upon the part of real world that they represent.

In case of disagreement (because of incorrect or lack of information), modifications may be necessary both in the BPDs and in the artefacts previously created in the organizational modelling stage. For example, an employee may realise that an activity is missing in an As-Is BPD. Therefore, the activity should be introduced in the BPD as well as in the roles model so that consistency between the models is guaranteed. An example of modification that would not affect other artefacts is a change

in execution sequence of some activities. It would not affect other artefacts because execution sequence of activities is specified only in BPDs.

The As-Is BPDs of an organization may also correspond to its To-Be BPDs. This is the case when an organization does not aim to change its business processes when an IS is going to be developed, for instance, because the main purpose of the system is just task automation and the organization considers that the design and execution of its business processes is satisfactory. In this case, and as discussed in Chapter 3, development of the purpose analysis stage would not be necessary, and thus As-Is BPDs would represent the future wanted situation of an organization.

The following subsections outline the mapping of the previous artefacts of the stage into BPDs and present a set of guidelines for their modelling. The guidelines are based both in literature review and in application and evaluation of the methodological approach of the thesis.

4.10.1 Mapping of Previous Artefacts into BPDs

Once the artefacts related to the glossary, the business events, the domain data model, the roles model, the business rules and the process map of an organization have been created, As-Is BPDs are modelled from them. As discussed above, this does not imply that business process modelling cannot start until all the previous information is gathered (i.e., system analyst consider that the information is complete), but that this information is mapped into and correspond to details of the As-Is BPDs of an organization.

The information of the artefacts must be consistent. The information that different artefacts share must be present in all the artefacts (e.g., a participant in the template of a business process must appear as a role in the roles model). The consistency needs are explicitly shown in Appendix A, in which the relationships between the concepts (i.e., information) of the artefacts is determined.

Table 4.3 shows a summary of the correspondence between BPMN elements (i.e., elements that can be modelled in a BPD) and previous elements (i.e., information of the rest of artefacts of the organizational modelling stage). When the correspondence is multiple (i.e., a previous

element can be mapped into different BPMN elements), system analysts have to select the one that they prefer. In addition, other correspondence may be defined. In summary, selection and definition of a mapping will depend on the criteria and preferences of the system analysts and the customer stakeholders.

Nonetheless, the guidelines of the following section suggest selection of the correspondence depending on the characteristics of the previous elements and on the basis of the quality aspects of a business process model. These aspects are explained in the following section. For example, use of annotations is not suggested in order to keep BPDs as small as possible and thus try to facilitate their understanding. Nonetheless, system analysts or customer stakeholders may like using them, and thus the corresponding guideline would not be followed.

In must be noted that business process models in the methodological approach consists of a BPD and of associated documentation (e.g., with a description of the business process, its business rules and the inputs and outputs of its activities).

Finally, the correspondence shown in table 4.3 justifies the need of the creation of other artefacts (i.e., of gathering their associated information) to model BPDs according to the definition proposed in Chapter 3. Correspondence for mission statement does not exist, but its definition is necessary for understanding of the application domain.

Table 4.3 Correspondence between BPMN elements and elements of the artefacts created in the organizational modelling stage

Previous element	BPMN element
Business process	BPD
Organization	Pool
Participant	Lane
Business event	Event with a trigger
Activity	Task, event with a trigger, decision or no-element
Obligation, Prohibition and Permission	Decision, event with a trigger, documentation or annotation
Response to event	Sequence flow between an event and a task
Domain entity	Documentation or data object
Definitions	Documentation or annotation

4.10.2 Guidelines for Modelling of BPDs

Definition and use of guidelines for business process modelling are important concerns both in industry and in academia. Guidelines are related to the issues of following a methodology when modelling business processes (Bandara, Gable, Rosemann, 2005; Indulska, et al., 2009) and of obtaining quality business process models (Becker, Rosemann, von Uthmann, 2000). Guidelines are especially important for novice modellers in order to facilitate and help them with business process modelling (Mendling, Reijers, van der Aalst, 2010).

Among the existing works on business process modelling, many have focused on provision of advice and guidelines for its development. The guidelines can be targeted at six aspects of a business process model (i.e., there exist six types of guidelines) and its creation process (i.e., modelling of a business process) (Becker, Rosemann, von Uthmann, 2000):

- **Correctness**

  This aspect is related to the syntax and semantics of a business process model. A model is syntactically correct if it is consistent and complete against the metamodel on which the model is based. A model is semantically correct if it postulates that the structure and the behaviour of the model are consistent with reality. The consistency between different models is also part of the correctness of the models.

- **Relevance**

  This aspect is related to the selection of a relevant object system (universe of discourse) in order to take a relevant modelling technique or to configure an existing metamodel adequately and· to develop a relevant (minimal) model. A business process model includes elements without relevance if they can be eliminated without loss of meaning for the model user.

- **Economic efficiency**

  This aspect is a constraint to all the other guidelines. For example, it restricts the correctness or the clarity of a business process model. Economic efficiency is related to the cost/benefit and feasibility of modelling a business process. Approaches to

support the economic efficiency are reference models, appropriate modelling tools or re-use of models.

- **Clarity**

  This aspect is related to the need of a business process model to be readable, understandable and useful. It is extremely subjective and postulates exactly that a model is understood by the model user. It is not sufficient if a model designer regards the model as understandable. Construct overload is an example of missing clarity as additional knowledge outside the modelling technique is required. Mainly layout conventions put clarity in concrete terms.

- **Comparability**

  This aspect is related to the need of the use of all the aspects within a modelling project. It aims to increase the comparability between businesses and periods, and includes, for instance, the uniform application of layout or naming conventions. Otherwise, two models would follow certain but different rules. The necessity to compare business process models is obvious and especially important if As-Is and To-Be business process models or enterprise-specific and reference business process models have to be compared.

- **Systematic design**

  This aspect is related to the need of well-defined relationships between models that belongs to different views, e.g., the integration of business process models with data models (every input and output data within a business process model has to be specified in a corresponding data model). A possible solution is to use a metamodel which integrates all relevant views. In this thesis, such a metamodel corresponds to the conceptual framework presented in Appendix A.

The first three aspects are considered basic for a business process model, whereas the last three are considered optional.

When reviewing literature, many concrete guidelines for business process modelling in general and for BPMN-based modelling in particular can be found. They address different aspects of a business

process model. Adoption of some guidelines or others will depend on the criteria of system analysts (who model business processes) and of the customer stakeholders, and even they may define and propose their own guidelines.

For BPMN-based business process modelling in the methodological approach of the thesis, 35 guidelines are presented in order to address the above aspects of a business process model. The guidelines have been adopted from literature or determined from definition, application and evaluation of the methodological approach, and most of them address several aspects of a business process model.

In addition, some guidelines have been implicitly presented in previous sections of this chapter when explaining how to discover and specify the information of the rest of artefacts of the organizational modelling stage. For example, the criteria for action atomicity in the roles model are related to correctness, relevance, comparability and systematic design of business process models. It must also be indicated that the conceptual framework presented in Appendix A addresses some of the aspects (correctness, relevance and systematic design) and thus is related to guidelines for business process modelling.

Another set of implicit guidelines for modelling of BPDs is to comply with BPMN specification, which is related to correctness. Guidelines for this facet are not presented, but the restriction on business process modelling with BPMN (version 1.2) can be consulted in (OMG, 2009).

Other type of guidelines that are not presented are general graphical (aesthetic) conventions for creation of diagrams (e.g., (Becker, Kugeler, Rosemann, 2003; Bridgeland, Zahavi, 2009; Effinger, Jogsch, Seiz 2010; Grosskopf, Decker, Weske, 2009; Lankhorst, 2009; Schrepfer, et al., 2009)), which address clarity of business process models. Some examples are:

- use of a common size for the elements of a same type;
- alignment of the elements;
- avoidance of overlaps of elements and of line crossing, and;
- modelling of diagrams from left to right;

Table 4.4 shows the main works (there exist more) on which definition of some guidelines is based (e.g., the work has identified weaknesses on modelling with BPMN that are mitigated by means of the guidelines) or

that include similar guidelines. Many guidelines are based on metrics for business process models (Sánchez, et al., 2010) so that they address quality aspects of a model.

All the guidelines are not based on literature, but some have been defined as result of the evaluation of the methodological approach. In addition, some guidelines that are supported by other works were defined before the works were published or reviewed. Nonetheless, this fact is not considered very important.

What is important is that most of the guidelines coincide with other works, what also increases their relevance and validity. For those guidelines that are not supported by other works (G1, G4-6, G8), their definition is important to address some of the aspects of a business process model. More specifically, all those guidelines are necessary for correctness, relevance, comparability and systematic design, and may also influence clarity (positively).

Table 4.4 Works that support definition of the guidelines

Work	Guidelines
(Allweyer, 2009)	G21-22, G24, G34
(Becker, Kugeler, Rosemann, 2003)	G7, G11, G13-14, G31-32,
(Bridgeland, Zahavi, 2009)	G9, G11, G16, G20, G31-32, G34-35
(Grosskopf, Decker, Weske, 2009)	G3, G11, G13-16, G20, G22-23
(Gruhn, Laue, 2009)	G27
(Koehler, Vanhatalo, 2007)	G21-22, G25, G29
(Lankhorst, 2009)	G11, G13-G14, G16, G31-32, G34-35
(Mendling, Strembeck, 2008)	G30
(Mendling, Reijers, van der Aalst, 2010)	G11, G13-14, G16-22, G24, G27, G31
(Ould, 1995)	G13, G16, G20
(Recker, 2011)	G2-3, G9-14, G27, G29
(Rolón, et al., 2009)	G13, G16, G24
(Silver, 2009)	G3, G7, G15-20, G28-29, G31-35
(Smith, Miers, 2008)	G17, G22, G26, G30
(zur Muehlen, Indulska, 2010)	G9-12
(zur Muehlen, Wisnosky, Kindrick, 2010)	G13, G20, G29

Table 4.5 shows the aspects of a business process model at which the guidelines presented are targeted. It must be noted that, in general, guidelines for systematic design are influenced by the methodology followed (e.g., the process of the organizational modelling stage). Therefore, they are usually targeted at a specific methodology, although they can also affect other aspects of a business process model.

Table 4.5 Aspects at which the guidelines are targeted

	Aspect					
Guidelines	**CR**	**RL**	**EE**	**CL**	**CM**	**SD**
G1	X				X	X
G2-3	X	X	X		X	X
G4-14	X	X		X	X	X
G15, G26	X	X		X	X	
G16, G20, G30-32, G34-35				X		
G17-19, G21-22, G24-25, G28	X			X	X	
G23	X		X	X	X	
G27	X				X	
G29		X		X	X	
G33				X	X	

Aspects of a business process model: correctness (CR), relevance (RL), economic efficiency (EE), clarity (CL), comparability (CM), systematic design (SD).

The guidelines presented are that, guidelines, not rules. Therefore, they do not impose a completely deterministic and automatable process for modelling of BPDs or oblige system analysts to use them. They just aim to suggest how a BPD should be modelled so that it better addresses the aspects of a business process model, to obtain uniform BPDs and thus to obtain BPDs with a higher quality than if guidelines were not followed (Becker, Kugeler, Rosemann, 2003). It must also be indicated that some guidelines are 100% automatable (G1-3, G12, G18, G23, G25, G27-28, G30), what positively affects economic efficiency.

It must be noted that application of some guidelines can be positive for an aspect of a business process model but negative for another. For example, the guideline G20 (explicit modelling of gateways) is positive for correctness but negative for economic efficiency. Therefore system analysts are responsible for deciding what aspect is more important in

cases like that. Some guidelines may also affect definition of the elements of the previous artefacts. For example, the guideline G29 may cause renaming of some element.

The guidelines do not cover modelling of all the existing BPMN elements, but just of those that can be derived from the information of the rest of artefacts of the organizational modelling stage and of those whose modelling may be problematic (e.g., may lead to obtain an incorrect business process model). Fore example, there are not specific guidelines for modelling of transactions or ad-hoc sub-processes, or for sequencing (most of the) flow objects. In this sense, BPMN elements that are domain-dependant (i.e., related to specific characteristics of an organization) are not considered in the guidelines. For example, modelling of a transaction will depend on how an organization executes or wants to execute its business processes.

Finally, the set of guidelines presented could either be followed or not, but one rule exists that should always be followed: customer stakeholders must be "happy" with the BPDs (Grosskopf, Decker, Weske, 2009; Ould, 1995). Consequently, a guideline may not be followed if customer stakeholders do not like their application, provided that a BPD is still correct. Furthermore, validation by customer stakeholders is necessary for guaranteeing correctness, relevance and clarity.

The 35 guidelines adopted in the organizational modelling stage for business process modelling (i.e, for modelling of BPDs) are the following ones.

G1) Mapping of business processes

For each business process of the process map, a BPD is created.

G2) Mapping of organizations

The organization for which an IS is going to be developed is modelled in the BPDs as a pool. If several organizations were considered (e.g., an organization and its customers), then a pool would be modelled for each organization.

G3) Mapping of participants

Each participant of the process map (and thus each role of the roles model) is modelled as a lane in the BPDs of the business processes in which he participates.

G4) Mapping of business events

Each business event is modelled as an event with a trigger in the BPDs of the business processes that are affected by the event.

G5) Mapping of activities into events with triggers

An activity of the roles model is modelled in the BPD of the business process in which it is executed as an event with a timer trigger when the activity represents that its associated role has to wait, and as an event with a message trigger when the activity represents that the role has to send or receive some information. In both cases, triggering of the events has to be a consequence of an interaction with an external participant (people from other organizations, e.g., a customer or a supplier) or with a role of the same organization that is not modelled in the BPD.

G6) Mapping of activities into decisions

An activity of the roles model is modelled as a decision (i.e., as an exclusive gateway) in the BPDs of the business processes in which it is executed when the activity represents that its associate role has to check some information.

G7) Mapping of activities into no element

An activity of the roles model is not modelled in the BPDs of the business processes in which it is executed if the activity could be mapped into an event with triggers (guideline G5) except for the condition related to interaction (e.g., the activity may represent an interaction with a person of the same organization that is modelled in the same BPD by means of a lane).

G8) Mapping of activities into tasks

If an activity of the roles model is not modelled from the guidelines G5, G6 or G7, then it is modelled as a task in the BPDs of the business processes in which it is executed.

G9) Mapping of obligations, prohibitions or permissions into decisions

An obligation, a prohibition or a permission is modelled as a decision in the BPDs of the business processes that the business rule constrains if it corresponds to an “if” statement and its fulfilment or the moment at which it is checked can be predicted.

G10) Mapping of obligations, prohibitions and permissions into decisions or events with triggers

An obligation, a prohibition or a permission is modelled as an event with a trigger in the BPDs of the business processes that the business rule constrains if it corresponds to an "if" statement and its fulfilment or the moment at which it is checked cannot be predicted. Depending on the nature of the business rule, the trigger can be error, conditional, cancel, compensation, cancel or multiple. The event can also be attached to an activity (task or sub-process) whose execution interrupts.

G11) Mapping of obligations, prohibitions or permissions into documentation

If an obligation, a prohibition or a permission is not modelled from the guidelines G9 or G10, then it is included in the documentation of the BPDs of the business process that the business rule constrain.

For this guideline, other possibility would be to model annotations. Although it may be considered positive for correctness and relevance, it also may be negative for clarity. Clarity is considered more important in the organizational modelling stage than correctness and relevance (in relation to whether modelling annotations or not), thus modelling of annotations is not adopted (a priori) in the methodological approach.

G12) Mapping of responses to events

A response to an event is modelled in the BPDs of the business process that the business rule constrains as a sequence flow that connects the event with the element that represents the activity triggered.

G13) Mapping of domain entities

Input and output domain entities of the activities of the roles model are specified in a table that is part of the documentation of a BPD. For each domain entity, its state must be specified. If a domain entity can have any state, then an asterisk ('*') is used. If a domain entity just has a state for which a name is not defined, then a dash ('-') is used

An example for the software development company is shown in Table 4.6 (business process "management of customer request"). For this guideline, other possibility is to model data objects. However, this way of modelling is not adopted (a priori) in the methodological approach (see discussion in guideline G11 for justification).

Table 4.6 Example of table to specify task inputs and outputs

	Input		Output	
Task	**Entity**	**State**	**Entity**	**State**
Define work unit	Customer	-	Work unit	New
Assign activities to developers	Work unit	New	Activity	Assigned
	Developer	-		
Estimate activity	Activity	Assigned	Activity	Estimated

It must be indicated that this guideline was not followed for modelling of the BPD of Figure 3.5 because in that figure it was considered important to show the inputs and outputs of the tasks in the business process model.

G14) Mapping of definitions

A definition of the glossary that is considered necessary for understanding of a BPD is included in its documentation.

For this guideline, other possibility would be to model annotations in the BPDs. However, this way of modelling is not adopted (a priori) in the methodological approach (see discussion in guideline G11 for justification).

G15) Interactions between organizations

An interaction between two organizations that are modelled in a same BPD (by following the guideline G2) is modelled by means of messages.

G16) Size of BPDs

In general, a BPD should not have more than 9 activities (regardless they are tasks or sub-processes), fit in a page and be described in less than ten minutes.

G17) Start and end events

All BPDs must have a start event and an end event.

G18) Number of start events

In general, a BPD should only have one start event. This guideline should not be followed if different events with triggers can start the process and the first task of the BPD can be different depending on the event that happens.

G19) Number of end events

In general, a BPD should only have one end event. This guideline should not be followed if successful and failure executions of the business process exist (an end event for each situation should be modelled).

G20) Decomposition of a BPD

If modelling of a BPD does not fulfil the guideline G16, then it should be decomposed by means of sub-processes or an event with a link trigger. As a result, new BPDs can be created.

G21) Decisions, forks, mergings and joins

Decisions, forks, merging and joins should be explicitly modelled in BPDs by means of gateways.

G22) Gateways in pairs

Exclusive and parallel gateways should be modelled in pairs. For example, for each decision modelled by means of a gateway, another gateway should be modelled for its merging.

G23) Gateways not in pairs

Guideline G22 should not be followed in the case of looping sequences (the branch or branches of a gateway join or merge in the same gateway) and of a task that follows a decision with two branches and merges into the first task of the other branch.

G24) Outputs of mergings and joins

A merging or a join should just have one outgoing flow.

G25) No combination of decisions and mergings and of forks and joins

Decisions and mergings should not be combined in a same exclusive gateway, and forks and joins should not be combined in a same parallel gateway.

G26) Default flow

A default flow should be modelled in all the decisions. For the methodological approach, they represent the normal (i.e., the most frequent) branch that is followed.

G27) Inclusive gateways

In general, inclusive (OR) gateways should not be used. They can be replaced by using other patterns of gateways.

G28) Loops and multiple instances

Loops should be used in activities of a BPD when they can be executed consecutively several times but the number of executions is unknown at modelling time. Multiple instances should be used when the number is known.

G29) Non-use of other BPMN elements

In general, use of conditional flows (to follow guideline G20), signal triggers for events (they are mainly system oriented, not business oriented and as BPDs are used in this thesis) and annotations (they can and should be specified as documentation as discussed in guideline G11) should be avoided.

It must be noted that just manual tasks are modelled (if the guidelines are followed) in the organizational modelling stage, then use of the rest of types (service, receive, send, script, user, and none) should be avoided too. Modelling of reference tasks is implicitly allowed by modelling of a same activity in different BPDs (those that correspond to the business processes in which the activity is executed).

G30) Length of names

In general, the name of an element of a BPD should not have more than 5 five words. If this condition cannot be fulfilled (e.g., for a complex gateway), then the complete name of the element should be indicated in the documentation of the BPD.

G31) Name of activities

An activity should be named in an action way: a verb followed by an object.

G32) Name of events

An event should be named in a happening way: an adjective or a condition followed by an object or a verb, or an object followed by a past participle.

G33) Common names for events

A catching event for a throwing event modelled in other BPD should have the same name as the latter event.

G34) Name of decisions

A decision should be named in a question form: a condition followed by a question mark.

G35) Name of branches of a decision

A branch of a decision should be named as a response to the question of the decision.

Once presentation of the guidelines is finished, Figure 4.7 shows the As-Is BPDs for the software development company. They are described as follows.

The product manager defines the set of activities that has to be carried out to develop a product through product workflow. When a customer requests a new improvement, a developer defines the work item that is necessary to provide the customer with the request. Next, developers are assigned the activities that are necessary to develop the work item, and developers have to estimate how long the activities will take. The product manager is also responsible for the periodical creation of product versions, which have a strict deadline, and must decide the version in which a work item will be developed.

The developers carry out the activities in order to finish the work items and deliver the improvement requested, and the product manager checks that version development is adequate. However, problems may arise while developing versions. Developers may not be able to finish the activities they are responsible for due to time constraints. If a problem arises, then the product manager has to try to solve it (by changing the version of a work item).

Lastly, it must be noted that the degree of detail of the As-Is BPDs modelled in the organizational modelling stage is high. This is in line with recognised need of detailed business process models when they are used for software development (e.g., (Becker, Kugeler, Rosemann, 2003)), and would correspond to low levels of structured approaches for business process modelling (e.g., (Davis, Brabänder, 2007)).

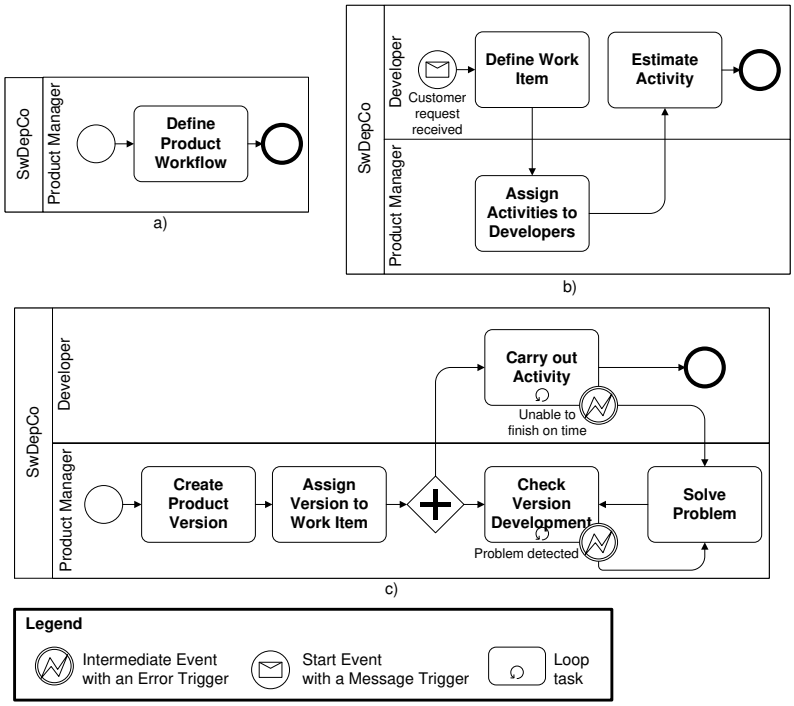

Figure 4.7 Examples of A-Is BPDs: a) definition of workflow; b) request management; c) version development

4.11 Summary

This chapter has presented the organizational modelling stage of the methodological approach of the thesis. For this purpose, its artefacts and their creation processes have been described.

First, information about the organizational activity and environment is gathered through interviewing customer stakeholders and checking organizational documentation. A set of initial artefacts are created from these activities, and they are later mapped into the As-Is BPDs of the organization.

As-Is BPDs are created by following a set of guidelines that aim to guarantee (or increase) the correctness, relevance, clarity and comparability of the models, as well as the economic efficiency and a systematic modelling when creating them. The set of guidelines is

probably the largest and most detailed existing one for modelling with BPMN.

Development of the organizational modelling stage is strongly grounded on existing works. For example, it is influenced by EKD, principles for data modelling, business rules specification and works on guidelines for business process modelling and on BPMN weaknesses.

In conclusion, this chapter has provided a systematic approach for modelling an organization for which an IS is going to be developed. The approach is based on several mechanisms, artefacts and guidance, and is targeted at modelling the As-Is BPDs of the organization. Thanks to the BPDs, an initial picture of the application domain and of the characteristics and needs of an organization is obtained.

Chapter 5
Purpose Analysis

"Without goals, and plans to reach them, you are like a ship that has set sail with no destination"

Fitzhugh Dodson

This chapter presents the second stage of the methodological approach of the thesis (Figure 5.1): purpose analysis. This stage aims to model and analyse the goals of an IS and determine how achievement of such goals will affect the business processes of an organization.

As mentioned in previous chapters, development of this stage may not be necessary. It will depend on the complexity of the purpose of a system. Systematic analysis may not be important if the effect of an IS on business process is clear and straightforward. For example, many times just task automation is the expected effect.

The chapter is organized as follows. First, an overview of the purpose analysis stage and a running example for its explanation are presented. Next, background work on goal discovery in RE and on business process reengineering is presented. The artefacts created as a result of the development of the stage (Figure 5.1) are then described and their creation processes are explained. Finally, a summary of the chapter is presented.

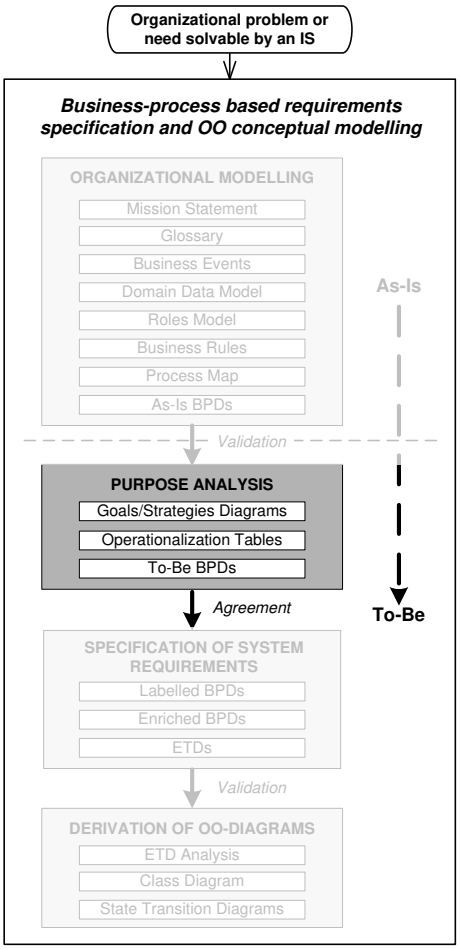

Figure 5.1 Stage and artefacts presented in Chapter 5

5.1 Overview of the Stage

As a summary of the facts and decisions explained in the previous chapters, the main characteristic of the purpose analysis stages is the use of the Map approach (Rolland, 2007) for its development. The stage is targeted at analysing the goals an IS and determining how they can influence the execution of the business processes of an organization.

Another objective is to facilitate communication with customer stakeholder and thus their involvement. As a result, and as justified in Chapter 2, the Map approach was selected for analysis of the purpose of

a system because its characteristics and strong points can facilitate communication among all the stakeholders.

Figure 5.2 shows the activities of the purpose analysis stage, in which the problem (or need) that an organization expects to solve by introducing an IS is analysed. The aim is to determine the IS goals and to find system strategies (features) that can fulfil the goals and thus solve the organizational problem. It must be determined how to operationalize the strategies related to the system purpose, and operationalization tables are used for this purpose. Agreement with customer stakeholders on the effect that the development of the IS may have on the business processes must also be reached. As a result, changes in the business processes can occur on the basis of the effect that the operationalization of the strategies will have on them. To-Be BPDs are modelled according to the effect determined.

Once agreement upon the effect of an IS on business processes has been reached, the purpose analysis stage is finished. It must be noted that several iterations may be necessary for reaching an agreement. This is mainly caused by the possibility of defining alternatives and different ways to achieve IS goals and operationalize the strategies.

It must also be indicated that Map diagrams are called goals/strategies diagrams in the methodological approach of this thesis. Maps diagram are usually called map, but the use of this term in the methodological approach may be confusing because of the definition or an artefact called process map in the organizational modelling stage.

Although all the stages of the methodological approach are correlated, the organizational modelling and purpose analysis stages have a special and very strong interrelationship. On the one hand, the purpose analysis stage affects organizational situation (it changes from As-Is to To-Be) and thus organization modelling. On the other hand, the purpose analysis stage is performed on the basis of information gathered during organizational modelling.

Reference to artefacts of the organizational modelling stage is necessary when performing (and explaining) the purpose analysis stage, as well as reference to artefacts of the purpose analysis stage would have been necessary if a "To-Be" organizational modelling stage had been defined explicitly in the methodological approach.

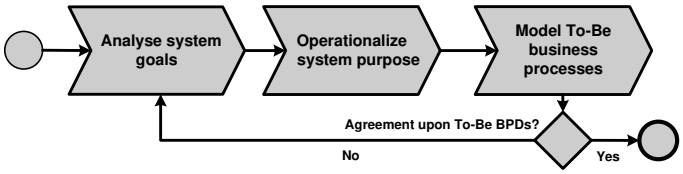

Figure 5.2 Activities of the purpose analysis stage

Although the whole methodological approach is related to business/IT alignment and positively affects it, the relationship is probably stronger and thus clearer in this stage. This alignment is considered to be reached when business goals, activities and processes of an organization are in harmony with the technology that supports them (McKeen, Smith, 2003), so business and IT work together to reach common goals (Campbell, 2005).

When reviewing literature, works and proposals that are in line with the purpose analysis stage can be found. For example, the first four steps of the organization-driven planning process presented in (Olle, et al., 1991) are analyse business problem, identify needs for change, identify IS, and propose change alternatives and new ISs.

Map-based approaches related to business process modelling and reengineering have also been presented in several works. In (Salineis, Presso, 2002), analysis and identification of gaps between As-Is and To-Be situations of a business process are based on Map diagrams, and the operationalization of the sections is addressed too. In (Thevenet, Salinesi, 2007), the Map approach is used to analyse the alignment of ISs and business strategy and to link the strategic and operational levels of an organization.

In relation to EKD, Map diagrams have been proposed to guide the modelling process (Rolland, Nurcan, Grosz, 2000). A Map-based extension of EKD for documentation and management of change in an organization was presented in (Nurcan, Rolland, 2003).

Finally, in the context of B-SCP (a RE approach for alignment of requirements and business strategy; (Bleistein, et al., 2006)), three strategies have been proposed for integration of the Map approach into B-SCP (Babar, et al., 2008a).

One of the strategies is presented in deeper detail in (Babar, et al., 2008b). The strategy for integration aims to provide mechanisms to

validate and verify Map requirements against B-SCP requirements, and vice versa. Map is used as basis for analysis and comparison of the As-Is and To-Be situations of an organization, and for representation of strategic goals in a (non-deterministic) business process-like form. Nonetheless, this way of representing business strategy is not new, but it was previously proposed in the context of the Balanced Scorecard (Kaplan, Norton, 1996). The representation is called strategy maps.

5.2 Running Example: The Software Development Company

The running example for the purpose analysis stage is the same as the one for the organizational modelling stage: the software development company (SwDepCo). The main reason is that introduction of a new example may be counterproductive.

Since reference to and analysis of the As-Is situation of an organization are performed during purpose analysis, it is easier for a reader to understand development of the stage on the basis of an already known case. In addition, introduction of a new example would require to present information in detail about it to understand it. This would imply a larger extension of the chapter, which can be avoided by using the software development company as a running example.

The company is experiencing problems with response time to customer requests since a few months ago. Some customers have started to complain about the time that passes since they request an improvement until it is included in a product, thus their satisfaction with the company has decreased.

The main reason for the slow response to customer requests is that product development is now hardly ever performed as planned and thus improvements are not delivered when expected. The number of customers of the company has increased during the last year considerably, and so the number of requests.

Employees' workload has increased and is now very heavy, or at least heavier than in past. This situation has caused that developers have not been able to finish their job as estimated many times and that product managers have not been able to detect some problems with sufficient anticipation. As a result, delays on response to requests have happened.

The company has analysed possible ways to tackle this situation. The most immediate one will be to hire more employees, so that employees' workload decreases. In addition, the company has decided to introduce a new IS to support product development so that this organizational activity is improved and thus customer satisfaction increases. By using a new IS, a solution to these problems and needs for both present and future is expected to be found.

5.3 Background: Goal Discovery in RE

This section reviews background work on goal discovery in RE in general an in goal-oriented RE approaches in particular. The review is useful for the purpose analysis stage in order to determine the ways in which IS goals should be addressed and discovered.

Clear goals when developing a software system are essential to understand what the real requirements are and thus to elicit them (Alexander, Beus-Dukic, 2009). They can also reduce work of the RE process. However, goal discovery (or elicitation) is not an easy task, and it is recognised as one of the main weaknesses of goal-oriented RE approaches (Rolland, Salinesi, 2005).

As a solution, many works have focused on provision of mechanisms and artefacts for goal discovery. For example, scenarios, organizational documents and goal refinements have been proposed for goal discovery (Rolland, Salinesi, 2005; van Lamsweerde, 2001). Nonetheless, these approaches do not fit the needs of the methodological approach of the thesis, in which IS goals are defined from the organizational problems or needs and as a complement to business process models for understanding of the application domain.

Other works have focused on study of the nature of goals and on proposal of guidelines for goal discovery. Among them, three works can be considered the most relevant ones.

GBRAM (Antón, 1997) can be regarded as the first specific approach for goal discovery. It is based on the assumption that customer stakeholders usually have a better understanding of their general goals than of system requirements. The approach aims to facilitate transition from goals to system requirements, and proposes heuristics, guidelines and a process for goal exploration, identification, organization,

refinement, elaboration and operationalization. Goal discovery is related to exploration, whose inputs can be interviews, policies, requirements, transcripts, workflow diagrams, corporate goals and a mission statement of an organization. Goals are classified as achievement (desired state) or maintenance (condition held true) goals.

In (Regev, Wegmann, 2005), the authors study the principles of goal-oriented RE and the nature of goals. They aim to provide more precise definitions of different types of goals on the basis of mechanisms from general systems thinking and cybernetics, and as part of (or as a complement to) the Lightswitch approach for specification of early (i.e., business) requirements (Regev, Wegmann, 2004). As a result of the study, the relationships between the types of goals are explained in detail. For example, achievement goals are defined (or aim) to fulfil maintenance goals, and they may also be targeted at other achievement goals.

The last and most recent relevant work is (Singh, Woo, 2008), which focuses on goal discovery at the operational, tactical and strategic levels of an organization. The authors provide three approaches for goal discovery (one for each level), which are then integrated into a single approach. Guidelines are also provided for goal discovery at each level. In a later work (Singh, Woo, 2009), the authors use part of the approach to study business/IT alignment.

With regard to the Map approach, its creators have provided some general guidelines for goal discovery (Roland, Salinesi, 2005; Rolland, 2007). For example, system analysts should first focus on business goals and then refine them until defining lower levels goals on the basis of the strategies (i.e., by creating new Map diagrams on the basis of the sections of another diagram). However, the guidelines can be considered too abstract and thus difficult to apply. Some more concrete guidelines have been presented recently for Map-based service engineering (Rolland, Kirsch-Pinheiro, Souveyet, 2010).

Finally, templates for goal documentation can be found in literature (e.g., (Antón, 1997; Pohl, 2010)). Use of templates is expected to facilitate goal discovery and specification. However, goal documentation in detail is not considered very important in this thesis, thus templates are not used and goals are not described in detail.

Goal discovery for the purpose analysis stage is further presented and discussed in Section 5.5.

5.4 Background: Business Process Reengineering

This sections reviews background work on business process reengineering, focusing on its relationship with development of ISs. On the basis of the review, the way to address business process reengineering in the purpose analysis stage is determined.

Business process reengineering can be defined as the analysis and design of business processes for their change and possible improvement in order to achieve some business goal (Grover, Malhorta, 1997). It gained fame quickly in the 90's as a result of the success of several works on the subject (Davenport, 1993; Hammer, Champy, 2001), which proposed novel solutions to important problems of organizations such as loss of competitiveness and low performance.

An organization may need business process reengineering because of three reasons (Hammer, Champy, 2001). First, an organization may need it because it is in deep trouble. The organization would have no choice, and it may not survive if it did not change. Second, an organization may need it because, although not in trouble, its managers see trouble coming. Therefore, the organization would anticipate the change before it was necessary urgently. Third, a leader organization in a peak condition may find a new opportunity. In this case, the change would aim to widen the gap with competitors. Such reasons coincide with the reasons for IS development assumed in this thesis: existence of a problem or need in an organization, which may arise in the three situations described.

Despite its success, discrepancies about business process reengineering exist (Melão, Pidd, 2000). For example, different opinions exist about its originality, its radical or incremental perspective as the most suitable one, the suitability of defining completely new business processes, or the need of methodological approaches to perform it. Adoption of a perspective depends on the needs of business process reengineering and on the criteria of the stakeholders.

An issue on which consensus can be considered to exist is on the importance of IT in business process reengineering (Broadbent, Weill, St.Clair, 1999). Organizations must be aware of the new ways of operation that IT makes possible, which can be useful in all the situations presented above. Business process reengineering initiatives are also usually linked to development of new IT. In the (software-oriented) IS

development research community, this activity is regarded as a reengineering initiative (Grau, Franch, 2007).

The role that IT plays in business process reengineering varies depending on the moment (in relation to business process design) at which it is used (Attaran, 2004). The role can be: 1) enabler, when IT is considered before business processes are designed; 2) facilitator, when IT is considered during business process design, or; 3) implementor, when IT is considered during business process implementation.

There no exists an overall agreement upon the most suitable role. Some authors think that IT must be an enabler of business process reengineering and thus trigger it (e.g., (Carr, Johansson, 1995)), but others think that excessive focus on IT (instead of on business needs) is a risk and IT should never lead reengineering (e.g., (Whitman, 1996)).

The most suitable role of IT will depend on the purpose of a concrete business process reengineering initiative. This issue is also related to radical vs. incremental perspective on business process reengineering, i.e., to the use of technology in a completely new way or in a more conservative manner. Anyway, what is really important about IT is that it meets and is in line with business needs (Davenport, 1993).

On the basis of this discussion and the best practices presented in the following subsection, business process reengineering in the purpose analysis stage is further presented and discussed in the Section 5.6.

5.4.1 Best Practices in Business Process Reengineering

Among the existing works on business process reengineering, many have focused on provision of advice and guidelines for its development and for avoidance of problems on the basis of practical experience (e.g., (Carr, Johansson, 1995; Smith, Fingar, 2002)).

In (Reijers, Mansar, 2005), a set of 29 patterns (called best practices in that work) for business process reengineering can be found. The patterns are based on literature review, and are especially targeted at reengineering of existing business processes that are taken as basis for change. Table 5.1 shows a summary of the patterns, which are divided into eight types of orientations:

- Customer, which focuses on improving contacts with customers; there exist three patterns related to this type of orientation:

1) Control relocation: move controls towards the customer.
2) Contact reduction: reduce the number of contacts with the customers and third parties.
3) Integration: consider the integration with a business process of the customer or supplier.

- Business process operation, which focuses on how to implement a new business process; there exist five patterns related to this type of orientation:
  4) Order types: determine whether tasks are related to the same type of order and, if necessary, distinguish new business processes.
  5) Task elimination: eliminate unnecessary tasks.
  6) Order-based work: consider removing batch-processing and periodic activities from a business process.
  7) Triage: consider the division of a general task into two or more alternative tasks, or the integration of two or more alternative tasks into one general task.
  8) Task composition: combine small tasks into composite tasks and divide large tasks into workable smaller tasks.
- Business process behaviour, which focuses on execution of a business process; there exist four patterns related to this type of orientation:
  9) Resequencing: move tasks to appropriate places.
  10) Knock-out: order knock-outs in a decreasing order of effort and in an increasing order of termination probability.
  11) Parallelism: consider whether tasks may be executed in parallel.
  12) Exception: design business process for typical orders and isolate exceptional orders from normal flow.
- Organizational structure, which focuses on the allocation of resources of an organization in a business process; there exist six patterns related to this type of orientation:

13) Order assignment: let workers perform as many steps as possible for single orders.

14) Flexible assignment: assign resources in such a way that maximal flexibility is preserved for the near future.

15) Centralization: treat geographically dispersed resources as if they are centralized.

16) Split responsibilities: avoid assignment of task responsibilities to people from different functional (i.e., organizational) units.

17) Customer teams: consider assigning teams out of different departmental workers that will take care of the complete handling of specific sorts of orders.

18) Numerical involvement: minimize the number of departments, groups and people involved in a business process.

19) Case manager: appoint one person as responsible for the handling of each type of order.

- Organizational population, which focuses on the types and number of resources of an organization used in a business process; there exist three patterns related to this type of orientation:

  20) Extra resources: if capacity is not sufficient, consider increasing.

  21) Specialist-generalist: consider to make resources more specialized or more generalized.

  22) Empower: give workers most of the decision-making and reduce middle management.

- Information, which focuses on the information that is or may be used and created in a business process; there exist two patterns related to this type of orientation:

  23) Control addition: check the completeness and correctness of incoming materials and check the output before it is sent to customers.

24) Buffering: instead of requesting information from an external source, buffer it by subscribing to updates.

- Technology, which focuses on the technology that may be used for execution of a business process; there exist two patterns related to this type of orientation:

  25) Task automation: consider automating tasks.

  26) IT: try to elevate physical constraints in a business process by applying new technology

- External environment, which focuses on improvement upon the collaboration and communication with third parties; there exist three patterns related to this type of orientation:

  27) Trusted party: instead of determining information oneself, use results of a trusted party.

  28) Outsourcing: consider outsourcing a business process in whole or parts of it.

  29) Interfacing: consider a standardized interface with customers and partners

The patterns have different impacts (positive or negative) on cost, flexibility, time and quality of execution of a business process. It is important that system analysts and customer stakeholders are aware of this impact when deciding to make some change in a business process on the basis of a given pattern. Need of improvement on cost, flexibility, time or quality can make an organization use some patterns or others. The use of a pattern may either be advisable or not depending on the purpose of business process reengineering.

Table 5.1 summarises the impact of the patterns for business process reengineering on cost, flexibility time and quality, which is presented and discussed in detail in (Reijers, Mansar, 2005). The symbols '+' and '-' in the cells represent positive and negative impact of a pattern, respectively, whereas no symbol represents neutral impact. It must be noted that positive impact does not mean increase in all the aspects.

Finally, many guidelines and best practices for business process reengineering can be found in literature, but just one rule exists: it must be linked to and based on business strategy (Carr, Johansson, 1995).

Table 5.1 List of patterns for business process reengineering and their impact on cost, flexibility, time and quality of a business process

		Aspect			
Type	**Pattern**	**Cost**	**Flex.**	**Time**	**Qual.**
Customer	Control relocation	+			+
	Contact reduction	+		-	+
	Integration	-	-	-	
Business process operation	Order types	-	-	-	-
	Task elimination	-	-	-	
	Order-based work	+		-	
	Triage	-	-	-	+
	Task composition	-	-	-	+
Business process behaviour	Resequencing	-		-	
	Knock-out	-		+	
	Parallelism	+	-	-	-
	Exception		-	-	+
Organizational structure	Order assignment		-	-	+
	Flexible assignment		-	-	+
	Centralization	+	+	-	
	Split responsibilities		-	+	+
	Customer teams	-	-	-	-
	Numerical involvement	-		+	-
	Case manager	+			+
Organizational population	Extra resources	+	+	-	
	Specialist-generalist		+	-	
	Empower	-		-	-
Information	Control addition	-		+	+
	Buffering	+		-	
Technology	Task automation		+	-	+
	IT	-		-	+
External environment	Trusted party	-		-	
	Outsourcing	-			-
	Interfacing	-		-	+

5.5 Goals/Strategies Diagrams

Once background work on goal discovery in RE and on business process reengineering has been presented, this section explains how to create the first artefact of the purpose analysis stage: goals/strategies diagrams.

Goals/strategies diagrams are used for modelling and analysis of the organizational problem or need at which an IS is targeted and of the solutions that the system can provide. The diagrams are modelled in a collaborative manner between system analysts and customer stakeholders (usually managers, who are more aware of the strategic and business needs) so that they agree on the solution.

In the methodological approach of the thesis, an IS is a facilitator of the necessary changes in the business processes of an organization because it is considered while they are designed. When adopting the facilitator role, IT can have several positive effects on business processes (Attaran, 2004). For example, IT can enhance employees' ability to make more informed decisions, facilitate identification of enablers for process redesign, capture the nature of the proposed change and link it to strategy, capture and disseminate knowledge and expertise to improve business processes, and reduce and replace work on business processes.

Before modelling of goal/strategies diagrams, system analysts must be aware of the problem or need that an IS must solve and how it is related to business strategy. This is what some authors call system vision (e.g., (Pohl, 2010)). For the software development company, the main problem is the existence of customer complains because of delays in response to requests. As a consequence, the strategic goal "keep customer satisfaction" is not fulfilled.

In comparison to the works presented in Section 5.3, the methodological approach of the thesis focuses on strategic and business goals, whose fulfilment is not or may not be possible because of the existence of problems or needs in an organization. Therefore, some achievement (e.g., reduce customer complaints) or maintenance goal (e.g., keep customer satisfaction) cannot be fulfilled, and some achievement goal(s) (e.g., reduce time of response to customer requests) arises and aims to fulfil the first goal.

The methodological approach focuses on the (business) goals that an IS must fulfil (or whose fulfilment must make possible), not in goals in

general or at any abstraction level. In addition, the above works practically just focus on goal modelling, whereas the thesis focuses both on goal and on business process modelling and analysis.

The goals of the goals/strategies diagrams correspond to achievement goals. They represent need of change and thus desired states in an organization. Otherwise, an IS would not be necessary, or its purpose would be very simple and thus a purpose analysis stage would not be necessary. Consequently, achievement is the type of goals that are mainly addressed in the stage. It must be noted that the goals correspond to the strategic requirements that are addressed in the methodological approach. The goals are named with change-related verbs (improve, minimize, maximize, reduce, increase, facilitate…), referring to some business aspect.

Once goals have been defined, system features (i.e., strategies) to fulfil them must be determined. System features play an important role in the development of many software systems, especially in the contexts of product line-based development (e.g., (Heidenreich, et al., 2010)) and of market-driven development (e.g., for system planning and decision making (Wnuk, Regnell, Karlsson, 2009)). Although features are usually considered to be related to architectural aspects of a software system (Pohl, 2010), they can be used in the RE process and feature-oriented RE approaches exist (e.g., (Shaker, 2010)).

The most common style for modelling of system features is FODA diagrams (Kang, et al., 1990). Nonetheless, this style is not adopted in the thesis because its purposes and perspective for modelling of system features do not coincide. FODA usually focuses on the system domain, whereas features of the purpose analysis stage focuses on the business domain. In addition, FODA features do not aim to represent ways to achieve the goals of an IS. It is also common that features are specified in lists (Lauesen, 2002).

Many definitions exist for system feature (Apel, Kästner, 2009), and its notion has been considered confusing (Classen, Heymans, Schobbens, 2008). On the basis of previous definitions and their purpose in the analysis stage, a system feature is defined in this thesis as a system requirement that:

a) represents an abstraction of the functionality (i.e., a unit of behaviour) of the system;

b) consists of a set of related domain assumptions and requirements;

c) corresponds to distinctive characteristics of the system that is valuable for customer stakeholders;

d) cannot be verified (i.e., tested) unless it is refined and specific criteria are defined (Berenbach, et al., 2009);

e) represents a customer stakeholders' expectation about the system (Lauesen, 2002), and;

f) makes fulfilment of system goals possible.

For example, "the system shall show the available time" is not a system feature because it represents a single requirement, can be tested, and is not an expectation but a fact. In contrast, "the system shall support workflow execution" is a feature.

Features represent domain requirements in the methodological approach and are the first requirements of the system domain that are addressed for support of business requirements. Two types of features are considered when modelling goals/strategies diagrams:

- Collaborative features

  This type corresponds to features in which an IS and its end-users collaborate, thus their participation is necessary. They can be specified in the form "the system shall support…" and refer to support of end-users activities, i.e., system behaviour shall support organizational activity performed by end-users.

- Autonomous features

  This type corresponds to features in which an IS performs some behaviour on its own, without collaboration of end-users. They can be specified in the form "the system shall…" and refer to an autonomous action. Although (as collaboration features) they are targeted at support of organizational activity, they do not directly support end-users activity, but indirectly.

The difference between a collaborative and an autonomous feature is very slight in many cases, and thus selection of a type must be agreed upon with customer stakeholders. For example, a feature could be "the system shall support indication of spent time" (collaborative) or "the

system shall record the spent time" (autonomous). There exits a nuance between both specifications, which in practice are not very different. Specification of system feature of a given type will also depend on the aspect that stakeholders want to emphasise, i.e., if they are more concerned about collaboration between an IS and its users or about autonomous actions of an IS.

By linking system features to goals and business processes (in the next activity), the limitations related to the difficulty to ensure that user tasks are supported and business goals covered and to the representation of unreal customer stakeholders' expectations (Lauesen, 2002) are mitigated.

When naming edges of goals/strategies diagrams, the name of the corresponding feature is adapted. Strategies are named as "By…", and express behaviour of an IS (collaborative or autonomous) that represents a manner to fulfil a goal. In the case of collaborative features, reference to the support is not necessary. For example, the edge of a feature such as "the system shall support workflow execution" is not named as "By supporting workflow execution" but as "By executing workflow".

The reason for this suggestion is that, when discovering and specifying system features, the focus is not only on what an IS will do, but on how it can support organizational activity (regardless it is in a collaborative or autonomous way) and make fulfilment of its goal possible. Anyway, naming of edges will depend on the preferences and criteria of the stakeholders.

In summary, the steps to analyse system purpose are the following ones:

1. A goals/strategies diagram is created.
2. The achievement goals that customer stakeholders want to fulfil by means of a new IS are modelled as nodes.
3. Edges are modelled to represent the system features that can make fulfilment of the goals possible; autonomous features are indicated by means of an asterisk.
4. For those features that are considered to correspond to other achievement goals and for which sub-goals can be defined, the above steps are repeated for the corresponding section.

These steps represent a refinement of those proposed in (Rolland, Salinesi, 2005; Rolland, 2007). Provision of a detail explanation and justification of the types of goals and of the types of system features to address is considered to mitigate the weaknesses in the guidelines of the Map approach outlined in Chapter 2.

It must be noted that analysis of system purpose is a very subjective and extremely creative activity, in which systems analysts' expertise and customer stakeholders' wishes play a major role. The goals are usually stable, but definition of the strategies can vary depending on the people that participate in modelling of goal/strategies diagram.

It is very difficult that diagrams created by different people coincide. Therefore, it is essential that those customer stakeholders that are considered more relevant for IS development (because of their knowledge and awareness of the system goals or of the impact of their opinion on system success) participate in analysis of system purpose and give feedback about the solutions proposed.

Figure 5.3 shows a goals/strategies diagram for the software development company. It is considered that the company has been experiencing problems with delivery requests because of lack of knowledge about version development, and consequently the strategic goal "keep customer satisfaction" is not fulfilled. The main reason for the delay is that activity development is not always performed as planned because of the great amount of work that developers have to do.

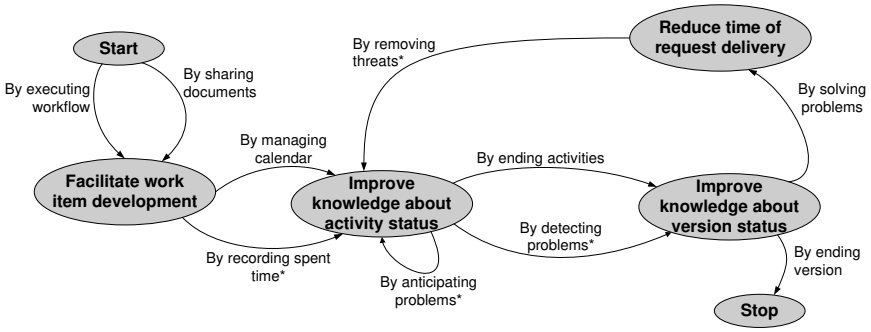

Figure 5.3 Example of goals/strategies diagram

The product manager needs to be able to better project, for example, if an employee will miss working days, or if an employee has spent more time than planned on an activity. The product manager also needs to foresee problems and find solutions quickly. In addition, developers need to be able to determine more accurately the time they have at their disposal to finish their activities and how long these activities will take.

To solve these problems, developers want the IS to facilitate work item development and to improve the knowledge they have about the status of the activities that they have to perform. The product managers want the IS to improve their knowledge about the status of the versions. Finally, company managers want the IS to reduce the time that takes a customer request to be delivered.

The system analyst proposed system features that could fulfil these intentions and modelled them in the goals/strategies diagram after agreement with customer stakeholders was reached. Both collaboration features and autonomous features have been defined.

Figure 5.4 shows an example of an alternative initial goal/strategy diagrams that may have been modelled for the software development company (e.g., because just company managers participated in its creation). Just one goal would have been defined, but analysis of the strategy "by controlling product development" may have resulted in identification of several new goals (the other three goals of the diagram in Figure 5.3), thus it would have been further modelled and analysed in a new goals/strategies diagram.

Finally, it must be indicated that Map elements have parameters (Rolland, 2007) but they are not addressed and thus specified in the methodological approach of the thesis. For example, the beneficiary and the location of an intention (i.e., goals) are some of its parameters.

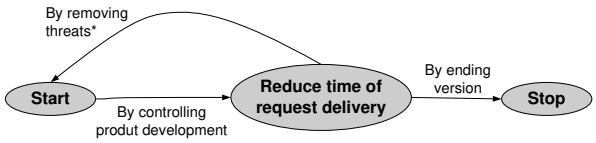

Figure 5.4 Example of alternative goals/strategies diagram

5.6 Operationalization Tables

After analysis of system purpose on the basis of goals/strategies diagrams, the effect that system purpose will have on the business processes of an organization is analysed and determined by means of operationalization tables (one per each diagram obtained).

Table 5.2 shows an example for the goals/strategies diagram shown in Figure 5.3. The table consists of five columns:

1. The strategy whose operationalization is under analysis.
2. The pattern of business process reengineering (Section 5.4) on which operationalization is based; different patterns can affect operationalization of a strategy (in a given business element), but just one pattern is considered to be the most influential one and thus only that is documented;
3. The business element(s) affected by operationalization on the basis of the pattern; such business elements can correspond to information that is part of the "initial" artefacts of the organizational modelling stage (business event, activities, business rules, etc.), or to new elements that are necessary for execution of business processes in the To-Be situation of an organization and did not exist when the As-Is situation was modelled; if an element is new, then the name of the element is followed by "(N)".
4. The type of element for the new business elements.
5. The organizational role that is affected by the operationalization or is in charge of the business element.

Use of patterns makes stakeholders aware of the existing possibilities (based on best practices) that they have for business process reengineering. It must be indicated that the IT pattern is not considered during operationalization of system purpose. This pattern is mainly a technology-based pattern, thus it is much related to specific technical solutions and focuses on the system domain. Operationalization of system purpose is mainly targeted at the business domain, and aims to determine how business elements can be affected by a new IS from a business perspective, independently of the technology (e.g., a workflow management system) that will be used for IS development.

Table 5.2 Example of operationalization table

Strategy	Pattern	Business element	Type	Participant
By executing workflow	TAU	Define product workflow	ACT	Product M.
	TAU	Assign activities to developers	ACT	Product M.
	TCO	Start activity (N)	ACT	Developer
		Carry out activity	ACT	Developer
		Finish activity (N)	ACT	Developer
By sharing documents	BUF	Start activity (N)	ACT	Developer
	BUF	Finish activity (N)	ACT	Developer
By managing calendar	OAS	Manage calendar (N)	ACT	Developer
	BUF	Time slot (Available and Unavailable) (N)	DOE	Developer
By recording spent time	TAU	Carry out activity	ACT	Developer
		Finish activity (N)	ACT	Developer
		Need to start activity (N)	EVE	Developer
By anticipating problems	TAU	Estimate activity	ACT	Developer
	TAU	Need to start activity (N)	EVE	Developer
	OBW	Check version development	ACT	Product M.
		Problem detected	EVE	Product M.
By ending activities	TAU	Finish activity (N)	ACT	Product M.
By detecting problems	TAU	Check version development	ACT	Product M.
		Carry out activity	ACT	Developer
		Unable to finish on time	EVE	Developer
By solving problems	TRI	Solve problem	ACT	Product M.
		Change activity assignment (N)	ACT	Product M.
		Change work item version (N)	ACT	Product M.
		Notify changes (N)	ACT	Product M.
By ending version	CAD	Version deadline (N)	EVE	Product M.
		Release version (N)	ACT	Product M.
	BUF	If the event "version deadline" happens, then the product manager has to release the version under development	BR	Product M.
By removing threats	TAU	Carry out activity	ACT	Developer
	TAU	Notify changes	ACT	Product M.

Patterns for business process reengineering: order-based work (OBW), triage (TRI), task composition (TCO), order assignment (OAS), control addition (CAD), buffering (BUF), task automation (TAU).

Type of element: bus. event (EVE), domain entity (DOE), activity (ACT), bus. rule (BRU).

The new business elements of Table 5.2 are:

- "Start activity", which is an activity that developers perform when they begin the development of an activity and have to receive the necessary documents to carry out the activity;
- "Finish activity", which is the activity that developers perform when they finish an activity and have to share the documents related to development of the activity;
- "Manage calendar", which is an activity that developers perform in order to divide and indicate the time that they can spent in a working day;
- "Time slot", which is a domain entity that developers need to perform the activity "Manage calendar";
- "Need to start activity", which is a business event that happens when a developer must start an activity so that a work item is finished before version deadline;
- "Change activity assignment", which is an activity that product managers performed in order to change the developer that is responsible for an activity so that a work item is finished before version deadline;
- "Change work item version", which is an activity that product manager perform in order to change the version of a work item due to some problem;
- "Notify changes", which is an activity that product managers perform in order to let developers know the possible changes in activity assignment or the version of a work item;
- "Version deadline", which is a business event that happens when the date of version release is reached;
- "Release Version", which is an activity that product managers perform in order to check and later release a finished version of a product, and;
- "If the event "version deadline" happens, then the product manager has to release a version", which is a business rule (response to event).

With regard to how business elements are affected by operationalization of the strategies, the strategy "by executing workflow" is used as an example. The business elements that operationalize the strategy are:

- "Define product workflow", because it is the activity in which the activities and documents of the product workflow are defined, and;
- "Assign activity to developers", "Start activity", "Carry out activity" and "Finish activity", because they refer to activities of the product workflow.

Therefore, the workflow is executed because of the development of these activities once an IS is introduced in the software development company. The first two activities will be mainly affected by the IS as a result of automation, whereas the last three activities are mainly a consequence of task decomposition of the initial activity "Carry out activity".

As happens with modelling of goals/strategies diagrams, both systems analysts and customer stakeholders must participate and agree upon the effect that an IS will have on the business processes of an organization. Again, it is a quite subjective and creative activity, in which expertise and wishes play a major role.

Although the use of patterns for business process reengineering can help stakeholders to perform the activity in a more systematic manner, different people may obtain different operationalization tables. For example, a customer stakeholder may think that removal of an activity on the basis of the order-based work will be very positive, but other may disagree on this decision. Satisfaction of all customer stakeholders is hardly ever possible, so operationalization is also an exercise of trade-off analysis and determination.

When creating operationalization tables, system strategies may be refined (i.e., their associated section in a goals/strategies diagram will be further analysed in another diagram). It would mean that goals associated to the strategy have been discovered, and that they were not discovered during analysis of system purpose.

As a rule of thumb, candidate strategies for refinement are those whose operationalization affects more than five business elements. For

example, on the basis of the goals/strategies diagram shown in Figure 5.4, and assuming that no other diagram had been modelled, operationalization of the strategy "by controlling product development" would have affected much more than five elements, thus it would be a candidate for refinement.

Decision about whether refining or not a strategy is partially subjective, but attempt to find low granularity strategies (i.e., they do no affect many business elements) is considered positive. On the one hand, reduction of the scope of business elements affected by a strategy can facilitate its analysis. When too many elements are affected, effect analysis can be more difficult for stakeholders. They may have problems for identification of all the business elements affected by a strategy, what may also imply loss of focus on the actual effect.

On the other hand, specification of strategies with too high granularity implies not only that sub-goals are not analysed, but that more strategies are not specified. Such a specification is considered advantageous because it facilitates discovery of new ways to achieve system goals and thus of running a business, and such ways can turn to be more adequate or satisfactory for customer stakeholders than if a strategy had not been refined. Nonetheless, refinement and proposal of a strategy (i.e., a system feature) must comply with the definition proposed (e.g., it is not testable unless further details are provided).

Operationalization of a strategy on the basis of a given pattern for business process reengineering can affect more than one element. The clearer example is the triage pattern, on the basis of which a business element (e.g., an activity) is decomposed into several elements.

In addition, those strategies that correspond to autonomous features have automation-related effects on business process. As a result, existing business elements can be automated or removed from a business process. In the latter case, introduction of a new IS would entail that the business element is no longer necessary. For example, execution of an activity by employees would not be necessary or relevant in the To-Be situation of an organization thanks to IS support.

In relation to the review of business process reengineering presented above, the strategy adopted when operationalizing system purpose is incremental, based on existing business processes and methodological (although partially).

5.7 To-Be BPDs

The last activity of the purpose analysis stage is to model the new (To-Be) BPDs of an organization. This activity is based on the effect that the operationalization of the system purpose will have on the business process of an organization. Therefore, and in comparison to As-Is BPDs, new elements may be added, existing elements may be removed, or the structure (e.g., activity sequence) of a business process may be changed.

In summary, changes can occur in As-Is BPDs of an organization as a consequence of the development of an IS and on the basis of the operationalization of its features. Otherwise, the organization may not be able to achieve the goals that it pursues by introducing a new IS. Achievement of such goals implies that the problems of the organization are expected to be solved thanks to the IS, or that its needs are expected to be met.

For modelling of To-Be BPDs, the guidelines presented in Chapter 4 for modelling of BPDs should be followed. System analyst must also agree with customer stakeholders upon the design of the To-Be BPDs, which represents the way in which the organization wants to operate in future. If customer stakeholder do not agree upon the design, looping in the purpose analysis stage may be necessary.

During the validation of the To-Be BPDs, issues related to errors or incompleteness in the previous activities of the stage and in their artefacts may appear, even in artefacts of the organizational modelling stage. Consequently, the artefacts should be revised, the activities may be performed again and changes in the artefacts may be necessary. For example, it is possible that a new IS goal is discovered in validation of To-Be BPDs. Therefore, it would not have been modelled in the goals/strategies diagrams. The goal should be analysed, strategies should be defined to achieve the goal, they would have to be operationalized, and finally new changes in the To-Be BPDs may occur.

Conflicts in operationalization of strategies and thus for modelling of To-Be BPDs may be detected. For example, operationalization of a strategy may involved deletion of a business element that is also affected by other strategy in a non-deletion related way. This would mean that operationalization of a strategy indicates that the element should not be part of To-Be BPDs, whereas another indicates that the element should be

maintained in the To-Be BPDs. In case of conflict detection, system analysts must agree with customer stakeholders what operationalization should be followed.

For the software development company, the activity "check version development" should be maintained according to the strategy "By detecting problems" and on the basis of the pattern "Task automation". However, it may be removed according to the strategy "By anticipating problems" and on the basis of the pattern "Order-based work" (and also influenced by "Empower"). In this case, the final decision is to remove the activity.

As a consequence of the operationalization table-based analysis of the effect of an IS on business processes and of the operationalization of the purpose of the system, changes in other artefacts for organizational modelling may occur too. For example, new business processes may be discovered and thus defined and they would have to be included in the process map, or new domain entities may be defined to support the information needs of the business process.

Figure 5.5 shows a change in the domain data model of the software development company. Time slot and its specializations (available and unavailable time slot) are new domain entities. The relationship between developer and activity has been removed, and a relationship between activity and available time slot has been defined.

With regard to the patterns related to technology, their effect on the business processes of an organization is further analysed in the specification of system requirements stage. For the purpose analysis stage, they are used as basis for indication of activities that are subject to be automated, but the degree of automation (partial or complete) is not considered.

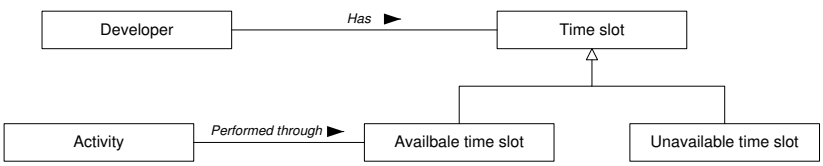

Figure 5.5 Example of change in a domain data model

Figure 5.6 shows the To-Be BPDs for the software development company. The As-Is BPDs have changed, and a new business process (calendar management) and new elements (e.g., the activity "change activity assignment") have been introduced. The BPD for problem resolution is part of the business process "version development", as a result of the decomposition of the BPD via the sub-process "solve problems".

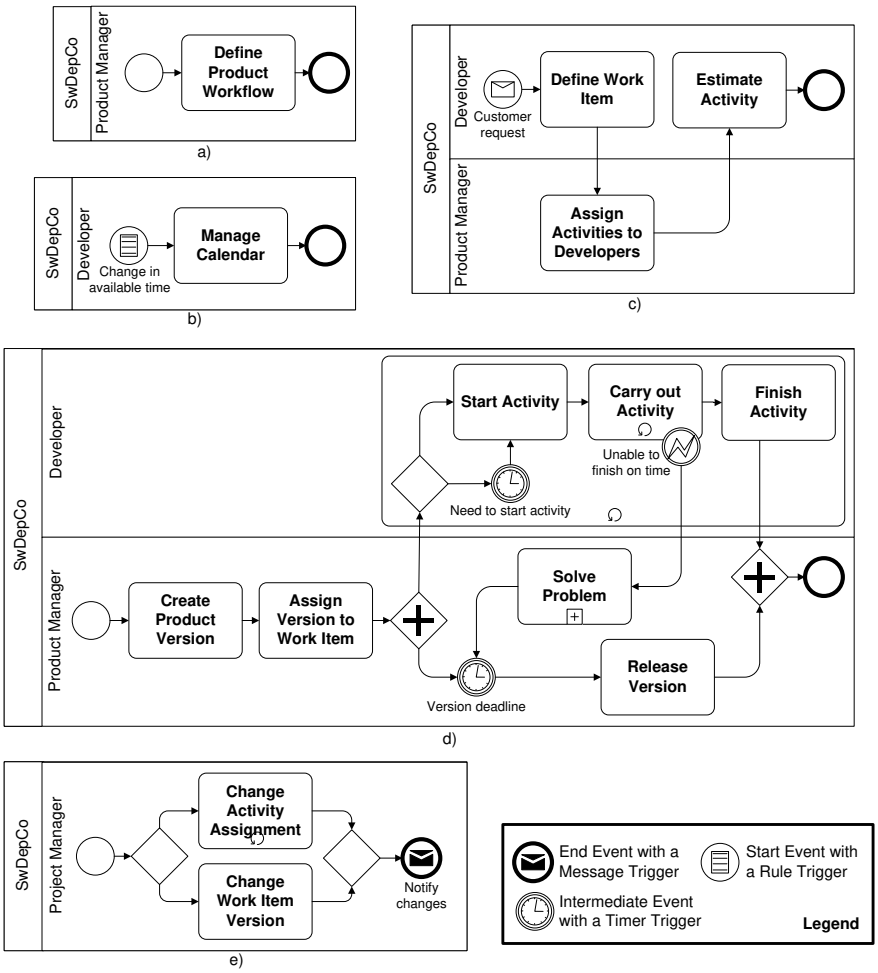

Figure 5.6 Examples of To-Be BPDs: a) definition of product workflow; b) calendar management; c) request management; d) version development; e) solve problem.

5.8 Summary

This chapter has presented purpose analysis, the second stage of the methodological approach of the thesis. By performing this stage, the goals of an IS and their effect of the business processes of an organization can be systematically determined by means of several mechanisms and guidance provided.

The goals of an IS are analysed on the basis of a new Map-based way to create and analyse goals/strategies diagrams. The stakeholders must discover the achievement goals whose fulfilment is expected to be possible thanks to a new IS, and the system features that represent strategies for fulfilment of the goals must be determined.

With regard to operationalization of system purpose, a new way for goal operationalization has been presented. Operationalization tables are used to determine the effect of the strategies of goals/strategies diagrams on the business process of an organization and by taking advantage of well-known and widely used patterns for business process reengineering.

For both cases, the mechanisms and guidance provided are deeply grounded on existing works on goal discovery in RE and on business process reengineering. On the basis of the facilitator role of IT for business process reengineering, the strategy defined and followed is based on discovery of achievement business goals and on existing business processes, and it is incremental, methodological and narrow.

Finally, once the purpose of an IS has been analysed and operationalized, To-Be BPDs of an organization can be modelled. They represent the way that the organization wants to behave when the new IS is running and thus supporting the organizational activity.

Chapter 6
Specification of System Requirements

"Our project plan will follow the usual arc… Requirements will drift until the project is both undesirable and impossible"

Dilbert

This chapter presents specification of system requirements, the third stage of the methodological approach of the thesis (Figure 6.1). It aims to specify the system requirements of an IS to properly support the business processes of an organization.

The stage is performed from the To-Be BPDs of an organization, which are analysed on the basis of several mechanisms. Afterwards, a set of guidelines determine how to derive ETDs from them to specify system requirements. As mentioned in previous chapters, the To-Be BPDs can correspond to the output of the purpose analysis stage or of the organizational modelling stage (in case IS purpose is not analysed).

The chapter is organized as follows. First, an overview of the stage and a running example are presented. Next, the artefacts that are created for specification of system requirements (Figure 6.1) and their creation processes are explained. Finally, a summary of the chapter is presented.

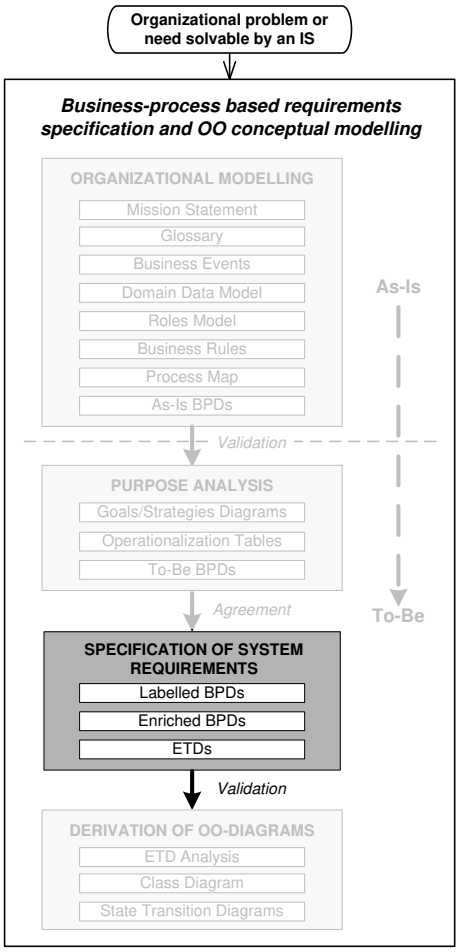

Figure 6.1 Stage and artefacts presented in Chapter 6

6.1 Overview of the Stage

The purpose of the specification of system requirements stage is to help system analysts precisely specify the system requirements of an IS from the business processes of an organization so that its activity is properly supported. System requirements are specified by means of ETDs, which, as explained in Chapter 3, are based on Lauesen's task & support descriptions (Lauesen, 2002), essential use cases (Constantine, Lockwood, 1999), information flows of the Info Cases approach (Fortuna, Werner, Borges, 2008) and the ISO 9126-1 standard (ISO, 2001).

The stage consists of three activities (Figure 6.2). First, the To-Be BPDs of an organization are labelled according to the system control on them and the labelling is agreed upon with customer stakeholders. Next, the flow objects that are always executed consecutively are identified and customer stakeholders validate the identification. These activities aim to help system analysts to properly elicit system requirements and bridge the gap between business and system domains.

Textual templates for ETD specification (hereafter referred to as textual templates) are then filled to specify the system requirements (functional, data and quality requirements) of an IS. This activity is performed from the enriched BPDs by following a set of guidelines that help system analysts to determine: 1) the correspondence between BPD elements and the domain requirements, and; 2) the product requirements of an ETD. Part of the content of the textual templates must be agreed upon with customer stakeholders, and they must check the templates to validate and agree upon the system requirements.

In relation to the requirements taxonomy presented in Chapter 3, all the types of system requirements of an IS are specified in this stage, except design constraints. This type is not addressed in this stage (and no specific stage or activity have been defined for them in the methodological approach of the thesis) because no systematic way to derive them can be defined.

All the guidance that can be provided about specification of design constraints is that they correspond to restrictions that affect an IS as a whole (e.g., the system shall have a web-based user interface), must be specified in a SyRS (e.g., in a specific section for them), and must be elicited from customer stakeholders (e.g., employees that are part of the IT department of an organization and know the possible technical restrictions of a new IS for the organization).

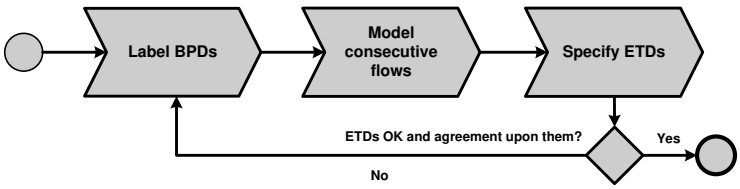

Figure 6.2 Activities and steps of the specification of system requirements stage

6.2 Running Example: The Rent-a-Car Company

As a running example to explain the specification of system requirements stage, a new running example is introduced: a rent-a-car company. Nonetheless, the actual and complete running of the organization is not explained, but just some information of the company is used.

The company is located in a tourist area, and its fleet of cars and workload greatly vary between the summer and the winter seasons. The number of cars in the summer season is around 250, whereas in the winter season is around 50. As a result, cars are usually bought at the beginning of a season and sold at the end. Its main activity is car rental, but it involves other activities (e.g., car maintenance).

Figure 6.3 shows a domain data model for the company and Figure 6.4 shows the To-Be BPD of the business process "car rental" of the company, which is executed by office employees. When a customer wants to rent a car, he has to choose one, what implies that the customer is requesting a rental contract. Rental contracts can also include extras (e.g., a GPS or a baby chair), and the price of the contract is determined on the basis of the rate of the car selected. If a customer is new, then the office employee records his data. Under certain circumstances, customers have to pay a deposit of money. The business process finishes when the office employee prints a copy of the rental contract and gives it to the customer, as well as the keys of the car. Cars need a valid insurance (policy) that covers them in case of accident so that they can be rented.

For the rent-a-car company, it will be assumed that it just aims to automate its business processes. Therefore, the purpose analysis would not have been performed, the artefacts shown correspond to the output of the organizational modelling stage, and thus the As-Is BPDs of the company coincide with its To-Be BPDs.

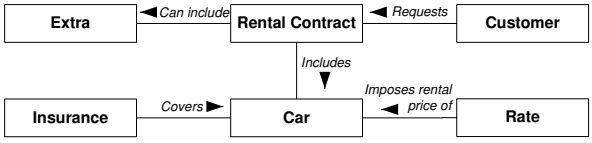

Figure 6.3 Example of domain data model

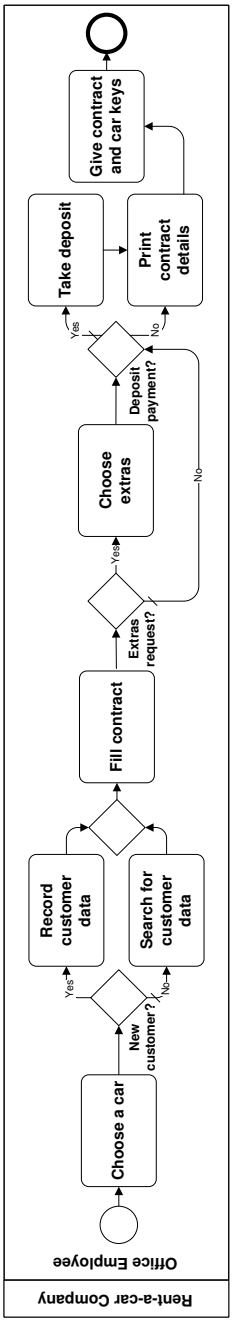

Figure 6.4 Example of To-Be BPD

6.3 Labelled BPDs

The set of To-Be BPDs (which include documentation about business rules and input and output domain entities) depict the operational requirements from which domain requirements are elicited in the specification of system requirements stage. For this purpose, To-Be BPDs are analysed and enriched graphically in order to allow system analysts to properly elicit ETDs from them.

System analysts have to precisely determine the system support for the business processes of an organization and the execution over time of its flow objects, and they do it in collaboration with customer stakeholders. The first activity (BPD labelling) is explained in this section.

In that activity, system analysts and customer stakeholders agree upon the degree of automation of the business processes of an organization. BPMN tasks, events with triggers and gateways that depict decisions of the To-Be BPDs are labelled according to the system support for them. The labels (Figure 6.5) are:

- "O" (out of the system), if the execution of the flow object will not be supported by a software system;
- "L" (controlled by a legacy system), if the execution of the flow object will be supported by an already existing system;
- "U" (controlled by a user), if the flow object will be executed by a person that interacts with the IS, or;
- "IS" (controlled by the system), if the IS will be in charge of the control and execution of the flow object with no human participation.

On the basis of practical experience, the semantics of the flow objects that will be out of the system or controlled by a legacy system is clear, but the semantics of the flow objects that will be controlled by the system or by a user may be confusing. Depending on their label:

- an event happening will be thrown or caught by the IS or by a user (who will use and interact with the system for throwing or catching the event happening);
- the fulfilment of a gateway condition will be checked by the IS or by a user, and;

- a task will be executed by the IS or by a user; in the latter case, the system will also take part in the execution of the task (a user will interact with the system), but it will be executed because of the user's initiative.

In addition, system analysts and customer stakeholders must agree upon the business rules and domain entities that were not modelled graphically and will be part of the IS, i.e., the business rules that will be controlled by the system and the domain entities whose information will be stored and managed in the system.

It must be indicated that the most recent version of BPMN (2.0) includes labels for tasks. As a consequence, the correspondence between labels of labelled BPDs and BPMN labels has been studied in order to be "more compliant" with next versions of the standard. Although the BPMN version 1.2 has been used for development of methodological approach, it is very likely that BPMN labels will be adopted and that their graphical representation will be used when possible.

In fact, BPMN labels have already been considered for development of tool support for the methodological approach (Appendix B). Nonetheless, labels of labelled BPDs are used during presentation of the specification of system requirements stage so that it is consistent with existing publications of the thesis (see Chapter 9). In addition, the labels of labelled BPDs were defined and have been used before BPMN provided labels.

Finally, and as mentioned in Chapter 5, labelling of BPDs is related to business process reengineering (task automation). In this sense, candidate business elements for automation can be discovered in the purpose analysis stage when operationalizing system purpose, but the degree of automation is determined in this stage. Determination of such a degree is also common in workflow modelling (e.g., (Sharp, McDermott, 2009)).

A business element (in general, an activity) is considered to be (in relation to a new IS):

- non-automated, if it will be out of the system or controlled by a legacy system;
- partially automated, if it will be controlled by a user, or;
- completely automated, if it will be controlled by the system.

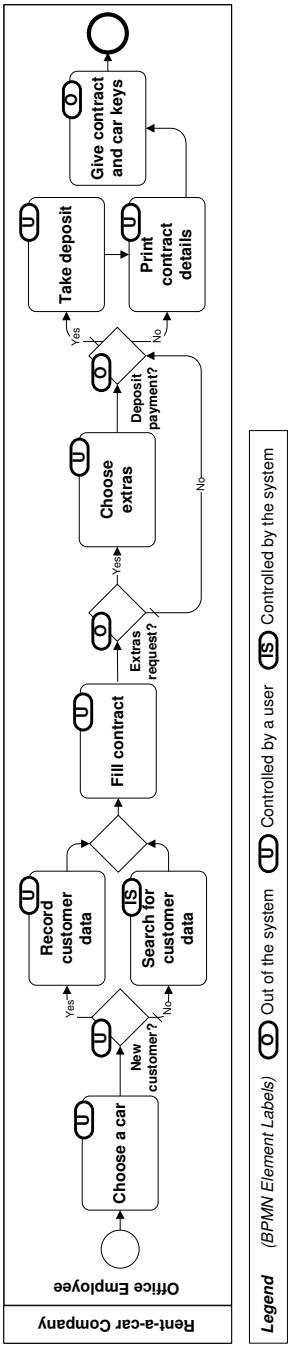

Figure 6.5 Example of labelled BPD

6.4 Enriched BPDs

In the second activity of the stage, labelled BPDs are enriched by specifying those sequence flows that are consecutive flows, i.e., those sequence flows that link two flow objects that are always executed one after another without an interruption. The graphical representation of a consecutive flow is an arrow with two arrowheads. The output of the activity is the set of enriched BPDs of an organization.

This type of connecting object does not exist in BPMN, but, as explained below, it is necessary to properly elicit the ETDs of an IS and so that their granularity is homogeneous. The purpose of the definition of this new type of connecting object is to be able to represent graphically the fact that two flow objects (or a sequence of flow objects) are always executed consecutively. If there are two flow objects that are executed consecutively sometimes, but not consecutively other times, a consecutive flow is not modelled.

The identification of consecutive flows is performed as follows. For each sequence flow of a labelled BPD that links two flow objects of a same lane, system analysts have to determine if the target flow object is always executed immediately after the source flow object when a token is in the sequence flow. If so, both flow objects are linked by means of a consecutive flow.

The existence of a consecutive flow between two flow objects implies that both objects represent a business transaction, i.e., the effect of the first object will be cancelled unless the second one is successfully executed. A sequence of flow objects linked by consecutive flows means that if the role responsible for the execution of the flow objects stopped in executing the business process at the last object of the sequence, then the effect of the flow objects would be recorded in the IS and they would not need to be performed them again.

A sequence of flow objects linked by consecutive flows is similar to the concept of step in workflow (e.g., (Davis, Brabänder, 2007; Dijkman, Joosten, 2002 Sharp, McDermott, 2009)). Nonetheless, the notion and implications of the sequence is more precise and impose more constraints. It is not enough to determine that a participant can execute a set of flow objects in a row, but it is also necessary to determine if it must be done in that way or if it is not necessary.

Customer stakeholders' participation is essential to model consecutive flows this activity. Customer stakeholders are the source of information from which the precise execution order of the flow objects is modelled, and they must validate that the consecutive flows have been properly modelled according to how the organization executes or wants to execute its business processes.

Figure 6.6 shows the enriched BPD for the business process "car rental". Although all sequence flows of Figure 6.5 have turned into consecutive flows in Figure 6.6, this is not always the case. In general, some sequence flows may turn into consecutive flows and others may not as a result of modelling of consecutive flows (Figure 6.7). Even it is possible that no sequence flow turns into a consecutive flow.

6.5 ETDs

In the methodological approach of the thesis, system requirements are elicited from the enriched BPDs and customer stakeholders and specified by means of ETDs in a textual template. The purpose of an ETD is to specify complete, adequate and precise IS support for the business tasks (i.e., activity) of an organization, and thus for its business processes.

The specification of domain requirements in an ETD is an adaptation and extension of task & support descriptions (Lauesen, 2002), the specification of user-system interaction (hereafter referred to as interaction) is based on essential use cases (Constantine, Lockwood, 1999), the specification of data requirements is a modification an extension of the information flows of the Info Cases approach (Fortuna, Werner, Borges, 2008), and quality attributes are specified on the basis of the ISO 9126-1 standard (ISO, 2001). Task & support descriptions, essential use cases, information flows and the ISO 9126-1 standard were chosen as basis for the definition of the textual template because they were considered to allow the purpose of an ETD to be achieved in a straightforward way.

As explained in Chapter 2, task & support descriptions aim to specify adequate support for business tasks (differently from other styles for SyRS; e.g., use cases aim to specify interactions with a system). They are a way to express what the system actors want to perform (user tasks), including domain-level information and how a system could support an activity or solve a problem.

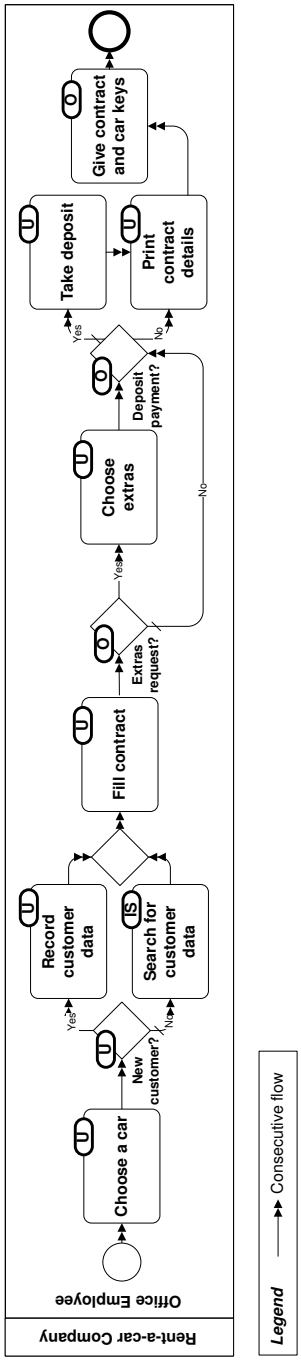

Figure 6.6 Example of enriched BPD

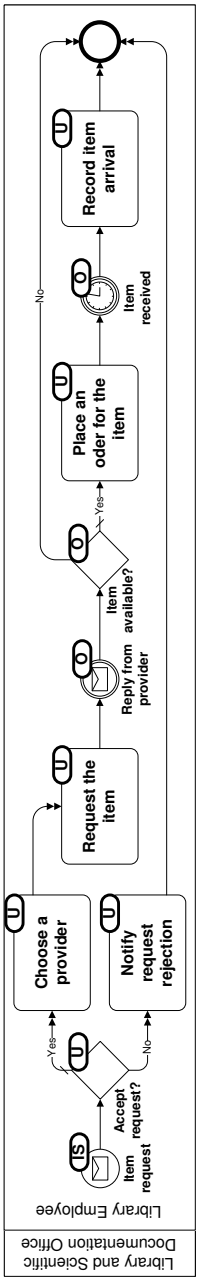

Figure 6.7 Example of enriched BPD in which all sequence flows do not turn into consecutive flows

An essential use case is a simplified (shorter and simpler) form of use case that depicts an abstract scenario for a complete and intrinsically useful interaction with a software system. Originally, essential use cases were a complement to task modelling to provide further details about it. They are specified from a user perspective by means of user intention (e.g., identify self) and system responsibility (e.g., show choices).

Information flows, which were presented in Chapter 2, are an abstract representation of the communication between an IS and its actors.

Task & support descriptions, essential use cases and information flows are intended to contain the fewest presuppositions about the technology with which an IS will be developed.

With regard to the ISO 9126-1 standard, it provides a quality model that can be used for specification of external quality attributes of a software system. The model defines six characteristics, and each one of them is refined in different subcharacteristics (Figure 6.8).

According to some authors (e.g., (Pohl, 2010; Wiegers, 2003)), there exits types of quality requirements (and thus characteristics and subcharacteristics of the ISO 9126-1 standard) that are more important for users and others that are more important for developers. Since ETDs are mainly specified to support users' activity, then it is few likely that types of quality requirements that are more important for developers are specified in them. It is more likely that they are specified as design constraints, or simply in another section of a SyRS. Nonetheless, this does not mean that their specification in ETDs is impossible or not sensible. Their specification will depend on stakeholders' needs and preferences.

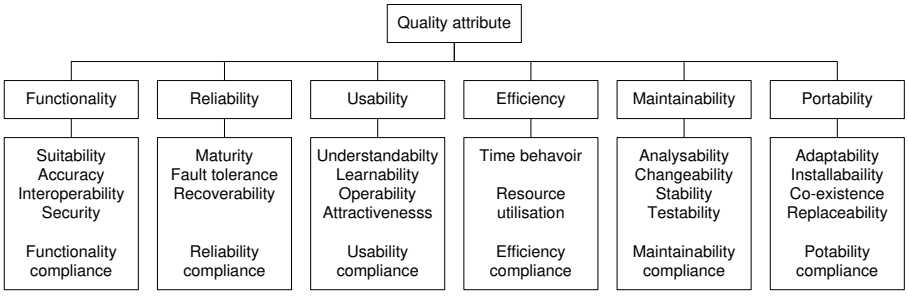

Figure 6.8 Characteristics and subcharacteristics of the ISO 9126-1 standard

The granularity of business tasks modelled in BPDs and of ETDs can be different. An ETD can specify support for several business tasks (activities of the organizational modelling stage) modelled in BPDs. Such tasks correspond to subtasks of the ETD and their determination is based on the consecutive flows of the enriched BPDs. An ETD specifies IS support for the execution of a set of consecutive flow objects of an enriched BPD, and the subtasks of the ETD are part of the set.

In addition, a significance criterion for ETDs has explicitly been defined so that their granularity is homogeneous and, therefore, their specification is consistent and adequate. The criterion is as follows: an ETD is significant if no other ETD is always executed immediately before or after the first one is executed. If there were two ETDs that hindered the fulfilment of this criterion, then both ETDs would represent the same unit of system requirements in conjunction and they should be specified in the same textual template.

This criterion is a result of using analysis of consecutive flows for elicitation of ETDs. As an example of application of this criterion, the system requirements for the enriched BPD of Figure 6.6 are specified in a single ETD. It is considered that specification of IS support for the tasks is only significant as a whole (an only ETD) because the tasks are always executed together and consecutively and their effect would be cancelled unless all the tasks are executed.

When reviewing literature, criteria for homogeneous granularity of system requirements that are in line with or related to the significance criterion of ETDs can be found:

- A step is a sequence of tasks of a business process that can be performed without interruption by the same role. A step is mapped into a use case (Dijkman, Joosten, 2002) .
- A communicative event (or a use case) must be triggered by an external interaction, provide meaningful information and consists of synchronous activities. Communicative events are asynchronous to each other (España, González, Pastor, 2008).
- Execution of a use cases implies a change of state in the system and in its environment, and such a state must be steady (Fortuna, Werner, Borges, 2008).

- A user performs a task description (or task & support description) and either achieves its goal or cancels the whole activity. When the task is finished, it means that the user would deserve a cup of coffee (Lauesen, 2002).
- A use case should embody at least one transaction in a business process and should support at least one activity leading a change of state (Odeh, Kamm, 2003)

However, these criteria are considered to be less precise or more complex than the significance criterion of ETDs (in conjunction to modelling of consecutive flows). As a result, they may be misinterpreted, their application may be more difficult for system analysts and customer stakeholders, and non-homogeneous system requirements may be obtained. Furthermore, just the criteria in (Dijkman, Joosten, 2002; Odeh, Kamm, 2003) address elicitation of system requirements from task-oriented business processes. The criterion proposed in (España, González, Pastor, 2008) analyses business processes from a communicative perspective, which is different from the perspective for business process modelling adopted in this thesis.

The following subsections present the sections of the textual template and the guidelines to fill it. Although some parts of the explanation of the subsections might be intuitive or well-known for some readers, understanding of the semantics of the sections is not always as straightforward as expected, what can lead to misinterpretation of ETD specification. Therefore, it has been considered important to explain the sections of the templates in detail.

6.5.1 Sections of the Textual Template

Table 6.1 lists the sections of the textual template and shows what types of requirements are specified in each one of them. When a cell contains a cross, it means that the section of its row always specifies the type of requirement of its column. For example, user intention represents a product requirement. When a cell contains an asterisk, it means that the section of its row may specify the type of requirement of its column, but it may also not do it. For example, business rules may represent functional or data requirements.

Table 6.1 Sections and types of requirements of the textual template

	Type of system Requirements			
Section	**Domain**	**Product**	**Functional**	**Data**
Name	X			
Business process	X			
Role	X		X	
Subtasks	X		*	
Triggers	X		X	
Preconditions	X		X	
Postconditions	X		X	
Frequency	X			
Critical	X			
Input	X		X	X
Output	X		X	X
Business rules	X		*	*
User intention		X		
System responsibility		X	X	
Information flows		X		X
Quality attributes		X	*	*

The sections of the textual template can be categorized as domain requirements or product requirements, and can also correspond to functional or data requirements. Figure 6.9 shows an example of textual template that has been specified from the enriched BPD of Figure 6.6. As shown in Figure 6.9, textual templates are divided into three parts (thick-lined parts): the name of the ETD, the rest of domain requirements, and the product requirements.

With regard to the combination of existing approaches and types of system requirements in the textual template, its sections for specification of domain requirements represent an extension of Lauesen's template for specification of task (and task & support) descriptions. The sections for specification of product requirements represent a combination and extension of essential use cases, information flows of the Info Cases approach and the ISO 9126-1 standard, as well of task descriptions.

Extended Task Description: **CAR RENTAL**

Business process: Car rental	**Role:** Office employee
Subtasks: Choose a car, Check whether a customer is new or not, Record customer data, Search for customer data, Fill contract, Choose extras, Take deposit, Print contract details	
Triggers: - **Preconditions:** - **Postconditions:** -	
Frequency: 10 times per day during winter season; 40 times per day during summer season **Critical:** Days on which a holiday period begins in summer season	

Input		**Output**	
Domain Entity	**State**	**Domain Entity**	**State**
Car	Ready	Rental contract	Open
Customer (1)	-	Car	Rented
Extra	Ready	Customer (2)	-
		Extra	Rented

Business Rules

- The insurance of a car must be valid during the rental period
- A car cannot be rented if it has more than 300000 km
- The total cost of a rental contract is calculated from the rate of a car and the price of the extras requested, multiplied by the number of rental days and VAT

User Intention	**System Responsibility**	**Information Flows**
	Normal interaction	
	1. Show cars	← $_1\{$ Car / make + model / $\}_n$
2. Select a car		→ Car
	3. Show customers	← $_1\{$ Customer / name + surname + ID number/ $\}_n$
4. Select a customer		→ Customer (1)
5. Introduce rental contract information		→ Rental contract / contract number + current date + current time + office + return date + return office /
	6. Show rental contract details	← Rental contract / contract number + current date + current time + office + return date + return office + rental cost + extras cost + VAT + deposit + total cost / + Car / make + model + plate number / + (Customer (1) / name + surname + ID number / | Customer (2) / name + surname + ID number /) + [$_1\{$ Extra / name / $\}_n$]
	7. Print rental contract details	
	Alternatives	
(New customer) 4.a.1. Introduce customer data [5]		→ Customer (2) / number + name + surname + ID number + address + city + telephone number + credit card type + credit card number + credit card expiration date /
	Extensions	
	(Extras request) 5.a.1. Show extras	← $_1\{$ Extra / name / $\}_n$
5.a.2. Select extras		→ $_1\{$ Extra $\}_n$
(Deposit payment) 5.b.1. Introduce deposit amount		→ Rental contract / deposit /

Quality attributes

- When an office employee selects a car, no other office employee will be able to select the same car (*Functionality/Suitability*)

Figure 6.9 Example of ETD

Each section of the textual template is described in the following subsections. For this purpose, the sections are divided into domain requirements, product requirements and information flows. Although information flows are product requirements, they have a specific subsection because of the length of their description compared to the rest of product requirements.

6.5.1.1 Domain Requirements

The domain requirements of the textual template are those system requirements that are elicited from operational ones. As mentioned above, in the methodological approach of the thesis, operational requirements correspond to the enriched BPDs of an organization and the textual specification of their business rules and input and output domain entities. The sections of the textual template that represent domain requirements are the following ones:

- Name: it is a sentence that identifies an ETD.
- Business process: it corresponds to the enriched BPD (i.e., business process) that is supported by an ETD.
- Role: it represents the role that the user responsible for the execution of an ETD will play, or the system.
- Subtasks: they are the consecutive flow objects of the business process of an ETD that depict business tasks that will be supported by the ETD.
- Triggers: they are the flow objects of the business process of an ETD that precede its first subtask, depict conditions that cause the need to execute the ETD when they are fulfilled and will be controlled by the system.
- Preconditions: they are the flow objects of the business process of an ETD that precede its first subtask, depict conditions that must be fulfilled before the ETD can be executed and will be controlled by the system.
- Postconditions: they are the flow objects of the business process of an ETD that follow its last subtask, depict conditions that must be fulfilled after the ETD is executed and will be controlled by the system.

- Frequency: it is the expected number of times that an ETD will be executed within a time period.
- Critical: it represents a time period or condition in which an ETD will be executed under abnormal or extreme conditions.
- Input: it is the set of domain entities that are used or consumed by the subtasks of an ETD and whose information will be stored in the IS.
- Output: it is the set of domain entities that are modified or generated after the execution of the subtasks of an ETD and whose information will be stored in the IS.
- Business rules: they are the business rules of the business process of an ETD that were specified textually, affect the ETD and will be controlled by the system; as explained below, new business rules can be specified.

6.5.1.2 Product Requirements

The product requirements of the textual template are those system requirements that are not elicited from operational ones and are necessary to support domain requirements. In the methodological approach of the thesis, product requirements correspond to specific characteristics of an IS that are not derived from enriched BPDs. The sections of the textual template that represent product requirements are the following ones:

- User intention: it corresponds to the set of actions that a user may perform during the execution of an ETD to interact with the system.
- System responsibility: it corresponds to the set of actions that the system may perform during the execution of an ETD.

  The actions of user intention and system responsibility can be part of three different interactions:

  a) Normal interaction: actions that are performed when executing the set of subtasks of the ETD that are always executed in its business process, or that are part of the default flow of a gateway and the branches that follow

the gateway are not always executed in the business process of the ETD.

b) Alternatives: actions that may be performed when executing the ETD and are an alternative to the actions of normal interaction (i.e., actions that imply that some action of normal interaction is not executed and substitute it).

c) Extensions: actions that may be performed when executing the ETD and are an extension to the actions of normal interaction (i.e., additional actions that may be executed but do not imply that some action of normal interaction is not executed).

- Quality attributes: they are the set of product requirements that are not specified in any other section of the textual template and represent quality requirements of an ETD.

6.5.1.3 Information Flows

Detailed specification of data requirements is performed in ETDs by means of information flows. They are mainly an abstract representation of the communication between an IS and its users, and correspond to data requirements for interaction (i.e., for user intention and system responsibility).

Information flows depict all the pieces of information that are necessary in an ETD so that user intention and system responsibility are properly supported. These pieces will be those that the IS and its users will exchange, thus information flows are specified for each action of user intention and system responsibility. Completeness of data requirements will be reached if all the pieces of information that are necessary for interaction are specified, and consistency between data and functional requirements will exist if all the domain entities of the pieces of information that are specified are used as input or output of the ETD.

Information flows can be considered a specialization of data expressions (Lauesen, 2002), i.e., data expressions that are exchanged between an IS and its users during the execution of an ETD. The main advantages of data expressions and, therefore, of information flows are that they are very compact, precise and easy for system analysts and

customer stakeholders to use and understand. In addition, the problem of complexity (size) of data expressions when trying to model an entire system is overcome by using interaction in ETDs as scope for specification of information flows.

In the Info Cases approach, information flows specify only the communication between a system and its users and in a unique flow. This is a problem because all the possible functions and modes of an IS (Olivé, 2007) are not (properly) addressed. Consequently, information flows are modified in this thesis for ETD specification.

The possible functions of an IS are:

- Memory: its purpose is to maintain an internal representation of the state of a domain, which is necessary for the other functions.
- Informative: its purpose is to provide users with information about the state of a domain.
- Active: its purpose is to perform actions that change the representation of the state of a domain.

The possible modes of an IS are:

- On request: it is followed when an IS performs a function as a response to a user request.
- Autonomous: it is followed when an IS performs a function on his own, without a user request.

It must also be indicated that external systems with which an IS interacts are considered users for ETD specification, i.e., users of an IS can corresponds to both people and software systems that interact (exchange information) with the IS (Pohl, 2010).

Specification of information flows is performed on the basis of the BNF grammar shown in Figure 6.10, the domain entities that are used as input and output in an ETD, and from its interactions (normal interaction, alternatives and extensions).

An input flow represents the pieces of information that a user has to communicate to an IS when executing an action of user intention. An output flow represents the pieces of information that an IS has to communicate to a user when executing an action of system responsibility. An autonomous flow represents the pieces of information that an IS has

to store on his own to keep a correct representation of the domain. Such information will be later available for users (so that the external view criterion for system requirements is fulfilled). An input or autonomous flows corresponds to information that must be part of the memory of (i.e., must be stored in) an IS, whereas an output flow corresponds to information obtained (directly or derived) from the memory of an IS.

```
<Information flow> ::= <Input flow> | <Output flow> | <Autonomous flow>
<Input flow> ::=  → <Input data expression>
<Output flow> ::=  ← <Output data expression>
<Autonomous flow> ::=  ~ <Autonomous data expression>
<Input data expression> ::= <Domain entity> |
   <Domain entity> / <Attribute> / |
   <Input data expression> + <Input data expression> |
   <Lower limit> { <Input data expression> } <Upper limit>
<Output data expression> ::= <Domain entity> / <Attribute> / |
   <Output data expression> + <Output data expression> |
   ( <Output data expression> '|' <Output data expression> ) |
   <Lower limit> { <Output data expression> } <Upper limit> |
   [ <Output data expression> ]
<Autonomous data expression> ::= <Domain entity> / <Attribute> / |
   <Autonomous data expression> + <Autonomous data expression> |
   <Lower limit> { <Autonomous data expression> } <Upper limit>
<Attribute> ::= <Attribute name> | <Attribute> + <Attribute> |
   ( <Attribute> '|' <Attribute> ) | [ <Attribute> ]
<Domain entity> ::= <String>
<Attribute name> ::= <String>
<String> ::= <Character> | <Character><String>
<Character> ::= <Letter> | <Number> | '(' | ')' |
<Lower limit> ::= <Number>
<Upper limit> ::= <Number> | n
<Letter> ::= A | a | B | b | C | c | D | d …
<Number> ::= <Digit> | <Digit><Number>
<Digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Figure 6.10 BNF grammar for specification of information flows

The symbols that can appear in an information flow and their semantics are:

- '→', for input pieces of information to an IS;
- '←', for output pieces of information from an IS;
- '~', for autonomous memory and active actions of an IS;
- '/ /', for membership;
- '+', for aggregation;

- ‘(|)’, for alternative;
- ‘{ }’, for repetition;
- ‘[ ]’, for option, and;
- ‘n’, for indeterminate number of repetitions.

Table 6.2 shows the correspondence between the types of information flows of an ETD and the functions and modes of an IS. It must be indicated that not only information flows (and their associated user intention or system responsibility) are affected or participate in the development of the functions and modes of an IS, but also other sections do. For example, a business rule (such as the one shown for total cost in Figure 6.9) can correspond to the active function of an IS. Details about the functions and modes also affect the derivation of OO diagrams stage (Chapter 7).

Finally, it must be noted that the semantics of the input and output flows and the semantics of the input and output of an ETD are different. The input of an ETD is the set of domain entities that exist and are used or consumed by its subtasks, and the output is the set domain entities that are generated or changed after the execution of its subtasks.

Table 6.2 Types of information flows to represent the possible functions and modes of an IS

	Mode	
Function	**On request**	**Autonomous**
Memory	→	~
Informative	←	←
Active	→	~

6.5.2 Filling of the Textual Template

The sections of the textual template either are derived from an enriched BPD or must be agreed upon with customer stakeholders, and guidelines have been defined to fill them. The guidelines specify what BPD elements correspond to domain requirements of the textual template and how to specify product requirements, and there are 31 guidelines to fill the sections.

Apart from the guidelines that are presented in the next subsection, all the sections of the textual template that depict product requirements of an ETD have an implicit guideline: they must be agreed upon with customer stakeholders. This guideline is also applicable to frequency, critical, input, output and business rules sections of the textual template. In the case of product requirements and of frequency and critical sections, customer stakeholders are the source of information from which they are elicited and specified in an ETD, thus they must indicate them and validate them.

With regard to business rules, it must be remembered that just some action assertions are specified in the organizational modelling stage. However, IS behaviour may be constrained by other types of business rules that have not been addressed in the methodological approach yet. These types correspond to structural assertions and derivations. More specifically, they correspond to those business rules related to restrictions on the information (data) of an organization and that must be controlled by the IS. For example, in Figure 6.9, a business rules that has been defined for ETD specification is that "the total cost of a rental contract is calculated from the rate of a car and the price of the extras requested, multiplied by the number of rental days and VAT".

In summary, all the business rules that represent constraints on the information of an organization and of its IS and are not specified in any other artefact (e.g., information flows of the ETDs) must be specified as business rules in the textual template. Nonetheless, it must be indicated that some types of business rules are addressed later in the methodological approach. They are those related to the data types of the attributes of the domain entities and the cardinality of the relationships between domain entities. They are addressed in the derivation of OO diagrams stage.

Since some guidelines might be difficult to understand without an example, Figure 6.11 shows examples of labelled flow objects and their specification in the sections of the textual template (application of the guidelines).

Finally, the guidelines allow system analysts to bridge the gap between business and system domains for specification of system requirements, as well as BPD labelling and modelling of consecutive flows.

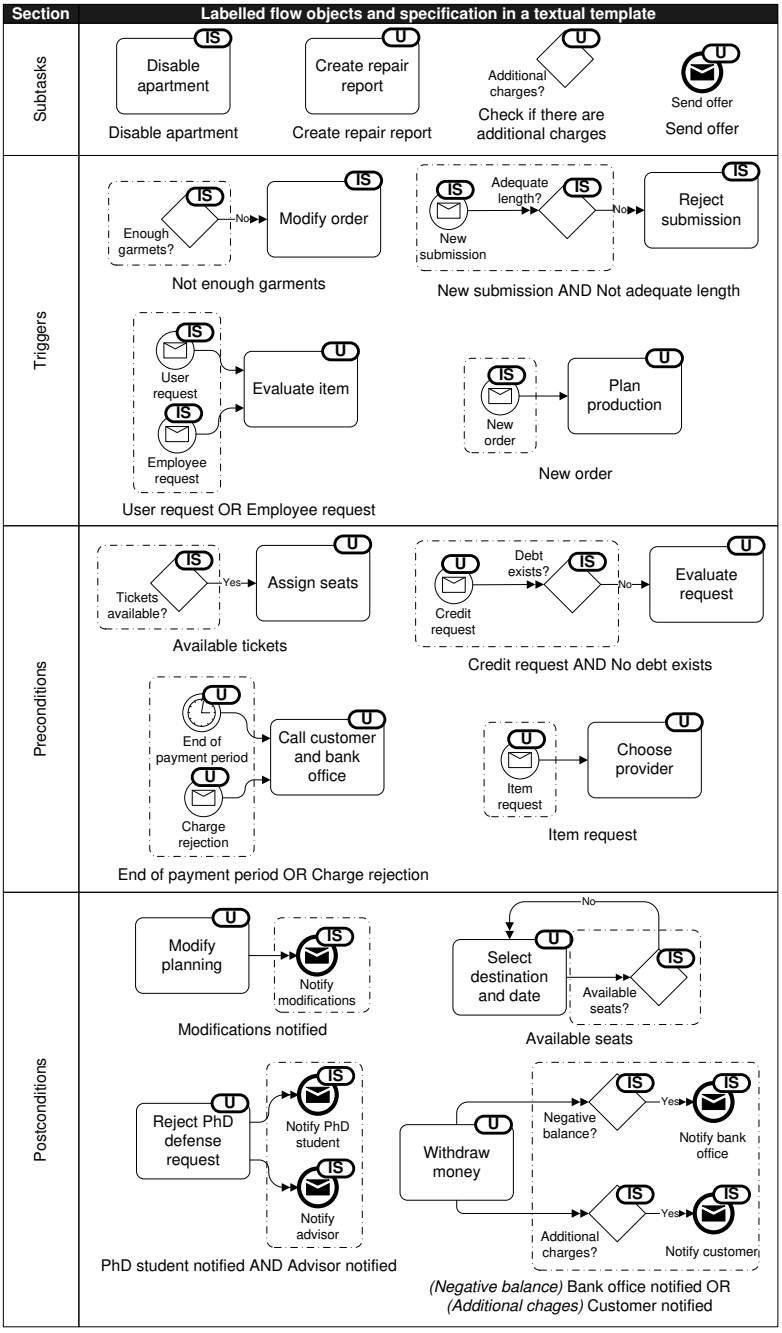

Figure 6.11 Examples of labelled flow objects and of their specification as sections of the textual template

6.5.2.1 Guidelines for Filling of the Textual Template

Name

G1) If an ETD just supports a subtask, then its name is the same as the name of the subtask that is supported.

G2) The name of an ETD must be agreed upon with customer stakeholders if it supports several subtasks.

Business Process

G3) The business process of an ETD corresponds to the enriched BPD (name) from which the ETD has been elicited. If the BPD has been modelled for a sub-process, then the business process corresponds to the root BPD.

Role

G4) The role of an ETD is the system if all its subtasks are controlled by the system.

G5) The role of an ETD is the participant in the business process of the ETD whose lane contains the flow objects from which the subtasks of the ETD that will be controlled by a user are specified.

Subtasks

G6) The subtasks of an ETD are the BPMN tasks that will be controlled by the system or by a user and the gateways and "throwing" events that will be controlled by a user.

Triggers

G7) The triggers of an ETD are the "catching" events that will be controlled by the system. If the role of an ETD is the system, then the gateways that will be controlled by the system are triggers too.

G8) The triggers of an ETD can be combined conjunctively (the fulfilment of all of them causes the need of execution) and disjunctively (the fulfilment of any of them causes the need of execution).

Preconditions

G9) The preconditions of an ETD are the "catching" events that will be controlled by a user. If the role of an ETD is not the system, then the gateways that will be controlled by the system are preconditions too.

G10) The preconditions of an ETD can be combined conjunctively (the fulfilment of all of them is necessary so that the ETD can be executed) and disjunctively (the fulfilment of some of them is necessary so that the ETD can be executed).

Postconditions

G11) The postconditions of an ETD are the "throwing" events that will be controlled by the system and the gateways that will be controlled by the system and can make the subtasks of the ETD iterate.

G12) The postconditions of an ETD can be combined conjunctively (all of them must be fulfilled after the ETD is executed) and disjunctively (some of them must be fulfilled after the ETD is executes).

G13) If the postconditions of an ETD are combined disjunctively, then it must be specified when they must be fulfilled (i.e., a condition precedes the postcondition).

Input and Output

G14) If more than an instance of a domain entity is part of the input or output of an ETD, then the instances must be differentiated by means of numbers in parentheses.

User Intention and System Responsibility

G15) For each subtask of an ETD that is controlled by a user, at least one action in user intention or in system responsibility must be specified.

G16) For each alternative and extension of an ETD, a name that identifies the alternative or extension and refers to the condition under which it is executed must be specified.

G17) The actions of normal interaction of an ETD are jointly ordered according to their expected execution[1].

G18) The actions of an alternative are ordered by means of three components: the same number as the first action of normal interaction that they substitute; a letter to distinguish among alternatives; and another number to order the actions of the alternative.

[1] Although such an order represents the expected sequence of the steps of user intention and system responsibility, it may finally not represent the actual sequence to perform an ETD, i.e., it is just a possible sequence. In this sense, some authors have discussed the convenience of step ordering (e.g., (Lauesen, 2002)), as well as its possible problems.

G19) The action of normal interaction that follows the last action of an alternative is put in square brackets at the end of the action of the alternative.

G20) The actions of an extension are ordered like the actions of an alternative, but their first number corresponds to the action of normal interaction that precedes the actions of the extension.

G21) In general, the actions of user intention correspond to the following verbs (Table 6.3): select (existing information), introduce (new information, which may affect existing information) and check (existing information).

G22) In general, the actions of system responsibility correspond to the following verbs (Table 6.4): show (existing information), store (new information) and change (existing information).

Information flows

G23) An input flow must be specified for each action of user intention in which the user has to select pieces of information of the system or to introduce new pieces of information in the system.

G24) An output flow has to be specified for each action of system responsibility in which the system has to show pieces of information to users.

G25) An autonomous flow has to be specified for each action of user intention or system responsibility in which the system has to store or change some information without input from users.

G26) For each information flow, the input or output data expression is specified on the basis of the BNF grammar for specification of information flows.

Table 6.3 Verbs for user intention on the basis of the possible functions and modes of an IS

	Mode	
Function	**On request**	**Autonomous**
Memory	Introduce, Select	-
Informative	Check	-
Active	Introduce, Select	-

G27) For each information flow of an ETD, the domain entities that are used must be part of the input or output of the ETD.

G28) The domain entities and the attributes of an autonomous flow of an ETD must be part of an output flow of some ETD (the same ETD or another).

Quality attributes

G29) For each quality attribute of an ETD, a type of characteristic and subcharacteristic of the ISO 9126-1 standard must be specified.

G30) If an external system participates in an ETD (as a user), then a quality attribute of "Functionality/Interoperability" must be specified and must refer to the system.

G31) If both a human user (which will correspond to the role) and an external system participate in an ETD, then the steps of interaction performed by the system must be indicated by means of a quality attribute of "Functionality/Interoperability".

Table 6.4 Verbs for system responsibility on the basis of the possible functions and modes of an IS

	Mode	
Function	**On request**	**Autonomous**
Memory	-	Store
Informative	Show	Show
Active	Show	Change

6.6 Summary

This chapter has presented specification of system requirements, the third stage of the methodological approach of the thesis. The stage aims to specify system requirements by means of ETDs and is based on the analysis of the To-Be BPDs of an organization. The stage is performed collaboratively by system analysts and customer stakeholders.

Mechanisms and detailed guidance have been presented in order to properly elicit and specify the system requirements of an IS from the business processes of an organization. As a result, the gap between BPMN and ETDs has been bridged, BPMN has been extended

graphically by specifying the automation of its elements with labels and by defining the concept of consecutive flow, and thus BPMN expressiveness and usefulness for the RE process have been improved.

ETDs are specified in a standard textual template that integrates functional, data and quality requirements and that is filled by following a set of guidelines that determines the correspondence between the business processes of an organization and the system support that they need. Data requirements are specified in detail by means of a new and improved style for specification of information flows and on the basis of a BNF grammar. Guidelines have also been presented to precisely specify the information flows and to facilitate completeness and consistency of data requirements. Furthermore, ETDs are homogeneously specified on the basis of a significance criterion.

Chapter 7
Derivation of OO Diagrams

"Essentially, all models are wrong, but some are useful"

George E. P. Box

This chapter presents the fourth and last stage of the methodological approach of the thesis (Figure 7.1): derivation of OO diagrams. Such diagrams correspond to an OO conceptual schema of an IS, and their derivation allows business process-based system requirements to be integrated into OO conceptual modelling-based IS development.

As a result of the integration, system requirements can be useful for an OO perspective for IS modelling and development. A SyRS in the form of ETDs (as well as the first three stages of the methodological approach) would be useful for IS development on the basis of OO-Method (Pastor, Molina, 2007) or other OO conceptual modelling-based approach. The diagrams derived would meet the system requirements of an IS and support the business processes of an organization.

The chapter is organized as follows. First, an overview of the stage and a running example are presented. Next, the artefacts of the stage (Figure 7.1) and the guidance and rules for their creation are described. Further link of ETDs with OO-Method is then discussed, and finally a summary of the chapter is presented.

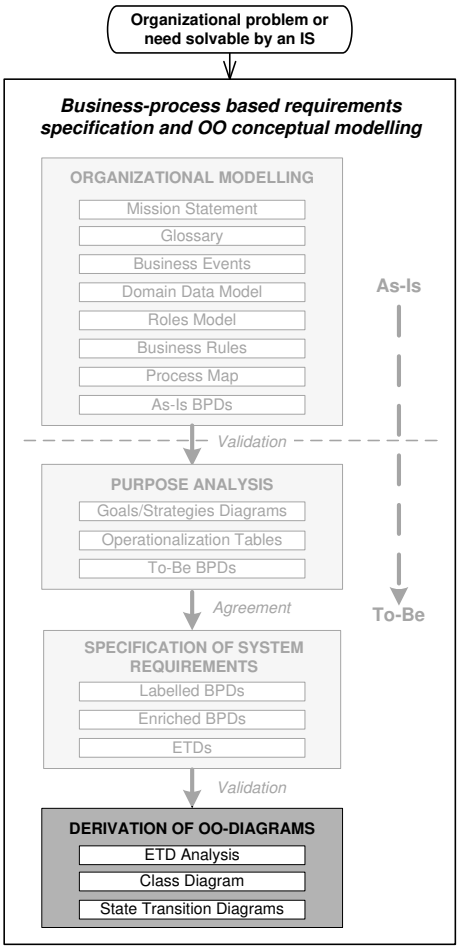

Figure 7.1 Stage and artefacts presented in Chapter 7

7.1 Overview of the Stage

The main purpose of the derivation of OO diagrams stage is to obtain an OO conceptual schema of an IS in the form of a class diagram and of the state transition diagrams of the classes. The conceptual schema must meet the system requirements of the IS (Insfrán, Pastor, Wieringa, 2002), be consistent with them (Glinz, 2000), and be complete from a requirements perspectives (i.e., the conceptual schema must contain all the necessary information to support the system requirements of an IS; (Olivé, 2007)).

Since system requirements are defined from the business process models of an organization, then the OO conceptual schema derived from the system requirements will support the business processes. Therefore, business process-based requirements specification and OO conceptual modelling of an IS are linked, and the system requirements can be useful for any OO conceptual modelling-based approach for IS development.

Derivation of OO diagrams is performed through three activities (Figure 7.2). First, ETDs are analysed to specify the necessary information for derivation of an OO conceptual schema that has not been documented in the previous stages of the methodological approach. Such information corresponds to the ETDs in which relationships between domain entities are created or deleted.

Next, a class diagram and state transition diagrams of the classes are derived by following a set of rules that determine the properties of the diagrams that can be determined from existing artefacts (the domain data model, the ETDs an the ETD analysis). Part of the information can be automatically derived, but other cannot. The latter case corresponds to decisions that system analysts must make on the basis of ETDs and customer stakeholders' knowledge and needs.

It must be indicated that both a class diagram and the state transition diagrams cannot be completely derived until part of the information of the other diagram is specified. Some rules for derivation of a diagram are based on information of the other.

Finally, customer stakeholders must validate the OO conceptual schema derived. The class diagram must contain all the classes and attributes that represent the information that the customer stakeholders need to execute the business processes of their organization, and the state transition diagrams must contain all the necessary states of the classes.

The rules and guidance provided guarantee that the conceptual schema is correct and complete on the basis of the ETDs of an IS, but incompleteness in the schema may be discovered because of incompleteness in the SyRS. Although the ETDs had been previously validated, incompleteness in them may be found in this stage. For example, an attribute may not be present in the class diagram because it was not specified in any information flow. If the attribute was necessary and thus should be part of the conceptual schema, then it should be part of an input or autonomous flow of some ETD.

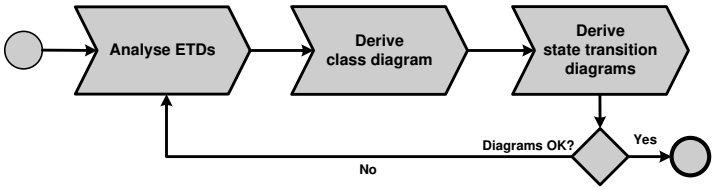

Figure 7.2 Activities of the derivation of OO diagrams stage

Definition of the rules for derivation of the class diagram and of the state transition diagrams of the classes is based on the assumption that a SyRS (i.e., the set of ETDs of an IS) is complete. Therefore, it contains the specification of all the system requirements of the IS, such as all the information flows that the IS users need. Otherwise, the diagrams derived may need to be completed by system analysts with that (requirements) information that is missing in the ETDs but will be necessary in the IS to support the business processes of an organization.

For example, it may be decided that ETDs for management of the information of the domain entities (such as modification of the value of its attributes) are not explicitly specified in the specification of system requirements stage. Consequently, elements to support this activity could not be derived to the OO diagrams (e.g., methods) because no explicit information would exist in the ETDs to derive the elements from them. Therefore, the elements would have to be modelled by the system analysts according to customer stakeholders' needs.

Finally, and in the context of this thesis, the derivation of OO diagrams stage makes integration of ETDs into IS development with OO-Method possible. Therefore, the whole RE process proposed in the first three stages of the methodological approach could be used as a RE approach for OO-Method. Nonetheless, it is again emphasised that the derivation of OO diagrams stage has been defined to provide a standard integration with OO conceptual modelling-based approaches for IS development that use class diagrams and state transitions diagrams.

In addition to creation of a class diagram and of state transition diagrams of the classes, further link of ETDs with other models of OO-Method is possible by deriving part of the information that has to be specified in them. However, many specific details of the link are out of the scope of this thesis.

7.2 Running Example: The Rent-A-Car Company

As a running example for the derivation of OO diagrams stage, the rent-a-car company presented for the specification of system requirements stage (Chapter 6) is used. In addition to the information and details about the company presented in the previous chapter, some new information is used to perform the derivation of OO diagrams stage. More concretely, more ETDs are used in this chapter and the domain data model is extended.

The part of the rent-a-car company that is used to show the fourth stage of the methodological approach is that related to car management. In Chapter 6, the business process "car rental" and its corresponding ETD (Figure 6.9) were presented, and the rest of ETDs related to car lifecycle are used in this chapter to derive the class diagram and the state transition diagrams of an IS. Such ETDs are:

- car purchase;
- car return;
- car maintenance;
- operation end, and;
- car sale.

Part of the ETDs (input, output and information flows) is shown in Figure 7.4. The domain data model for the rent-a-car company presented in Chapter 6 is extended with new domain entities and relationships in Figure 7.3.

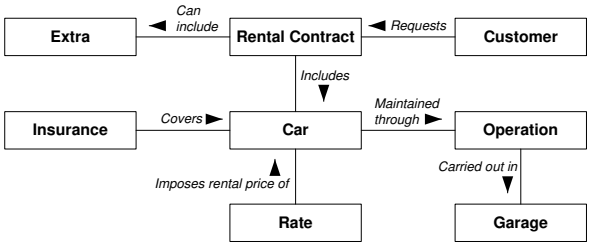

Figure 7.3 Extended domain data model of the rent-a-car company

Extended Task Description: **CAR PURCHASE**			
Input		**Output**	
Domain Entity	**State**	**Domain Entity**	**State**
Insurance Rate	- -	Car	Ready
Information flows			
(Normal) ← $_1${ Insurance / company + expiration date / }$_n$ → Insurance ← $_1${ Rate / name + price / }$_n$ → Rate → Car / plate number + make + model + engine + colour + seats + purchase date /			

Extended Task Description: **CAR RETURN**			
Input		**Output**	
Domain Entity	**State**	**Domain Entity**	**State**
Rental Contract Car Extra	Open Rented Rented	Rental Contract Car Extra	Closed Ready | Disabled Ready
Information flows			
(Normal) ← $_1${ Car / plate number / }$_n$ → Car / km + fuel / ~ Rental Contract / return date / ← Rental Contract / amount to pay / (Extension) → Car / disability date /			

Extended Task Description: **CAR MAINTENANCE**			
Input		**Output**	
Domain Entity	**State**	**Domain Entity**	**State**
Car Garage	Disabled -	Operation	To-Do
Information flows			
(Normal) ← $_1${ Car / plate number / }$_n$ → Car ← $_1${ Garage / name + address + phone number / }$_n$ → Garage → Operation / number + date + description /			

Extended Task Description: **OPERATION END**			
Input		**Output**	
Domain Entity	**State**	**Domain Entity**	**State**
Operation Car	To-Do Disable	Operation Car	Finished Ready
Information flows			
(Normal) ← $_1${ Car / plate number / }$_n$ → Car → Operation / end date + price /			

Extended Task Description: **CAR SALE**			
Input		**Output**	
Domain Entity	**State**	**Domain Entity**	**State**
Car	Ready	Car	Sold
Information flows			
(Normal) ← $_1${ Car / plate number / }$_n$ → Car / sale date /			

Figure 7.4 Input, output and information flows of the ETDs related to car lifecycle of the rent-a-car company

7.3 ETD Analysis

The first activity of the derivation of OO diagrams stage aims to specify information about the ETDs and the domain entities that has not been documented in the previous stages of the methodological approach but it is necessary for derivation of the class diagram of an IS and of the state transition diagrams of the classes.

System analysts have to determine:

a) the ETDs in which the relationships between the domain entities of the domain data model are created, and;

b) the ETDs in which the relationships between the domain entities of the domain data model are deleted.

The activity is performed by creating a table as the one shown in Table 7.1. The relationships between the domain entities of the domain data model of the rent-a-car company that are created or deleted in each ETD are specified in the table. These relationships represent information that will be managed by the IS of an organization.

All the types of relationships are analysed (association, aggregation and inheritance relationships). In the case of inheritance relationships, creation and deletion of a relationship means that a domain entity is specialised (i.e., an object becomes an instance of the child domain entity) and generalised (i.e., the object becomes an instance of the parent domain entity), respectively. For example, a person may be considered a child when he is born, but the person may become a regular person once he is 18. Therefore, he would not be considered a child anymore.

Table 7.1 Example of table for relationship analysis

ETD	Relationship	
	Creation	Deletion
Car Purchase	Covers, Imposes rental price of	-
Car Rental	Requests, Includes, Can include	-
Car Return	-	-
Car Maintenance	Maintained through, Carried out in	
Operation End	-	-
Car Sale	-	-

Nonetheless, all inheritance relationships may not be created or deleted this way. An instance of a parent domain entity may never become an instance of a child entity, and instances of the child entity may be created directly and without the need of previously creating an instance of the parent domain entity.

Inconsistencies may be detected in the ETDs (or in the analysis performed):

- The ETD in which a relationship between two domain entities is created or deleted requires that both entities are part of the input or output of the ETD. Otherwise, no relationship can exist or some domain entity is missing in the ETD.
- All relationships that are deleted must be created.

ETD analysis may be extended or changed by specifying more information about the ETDs that it is specified in subsequent activities of the derivation of OO diagrams stage. For example, and as currently defined in the methodological approach of the thesis, the data types of the attributes of the domain entities are specified during derivation of the class diagram. However, the data types may also be specified in another artefact when analysing ETDs.

What has been considered important is not when to specify certain information (if it could be specified at different "moments" of a stage), but that all the necessary information is specified at some moment. It is considered that specification of information at a given moment will depend on the criteria and preferences of the systems analysts. Provision of mechanisms and guidance to facilitate their decision and make it possible, as well as recognition of the fact that some steps may me performed at different moments, is regarded as the relevant point.

Finally, it must be indicated that the more information was pre-specified before application of the rules for derivation of the OO conceptual schema, the more automatic the derivation would be. As said above, it is a decision of system analysts when to perform some steps and what degree of automation they need or want at a given moment of a stage, and trade-offs may be necessary. It must be noted that more automation does not imply that a step has not to be performed, but that it can be performed previously to gain some degree of automation later.

7.4 Class Diagram

The class diagram of an IS is derived from (some sections of) its ETDs, ETD analysis, its state transition diagrams, and the domain data model of the corresponding organization. This activity is performed by following 16 rules. The graphical representation of the class diagram is completed by documenting the information (e.g., integrity constraints) that affects the classes and is not represented graphically. Although plain text is used in this thesis, no specific language is assumed for documentation and any could be used (e.g., OCL (OMG, 2006)).

The rules allow system analysts to model classes (Rule C1) and their attributes (Rule C2) and methods (Rules C3, C4, C5, C6, C8, C9 and C10), as well as the relationships between the classes (rule C7) and their multiplicities (Rules C13 and C14) for associations and aggregations. In addition, four rules have been defined for completing a class diagram with integrity constraints (Rule C15) derivation rules (Rule C16) and details about the methods (Rules C11 and C12) in its documentation.

Some rules can be automatically applied from existing artefacts, but others cannot (Rules C11, C12, C14, C15 and C16). Such rules need that system analysts make decisions or indicate some extra information that cannot be automatically (and deterministically) derived from the available artefacts. System analysts also have to name the methods.

Figure 7.5 shows a class diagram for the rent-a-car-company. It has been derived from the ETDs shown in Figure 6.9 and Figure 7.4 and the ETD analysis shown in Table 7.1. It must be noted that these artefacts correspond to just a part of the whole SyRS and ETD analysis of the rent-a-car company, thus the class diagram is incomplete. In addition, parameters of the methods and data types have not been modelled to keep Figure 7.5 as small as possible.

The rules are defined as follows.

Rule C1 (classes)

A class is modelled in a class diagram for each domain entity of an information flow.

For the running example, the classes are “Insurance”, “Rate”, “Car”, “Customer”, “Rental Contract”, “Extra”, “Garage” and “Operation”.

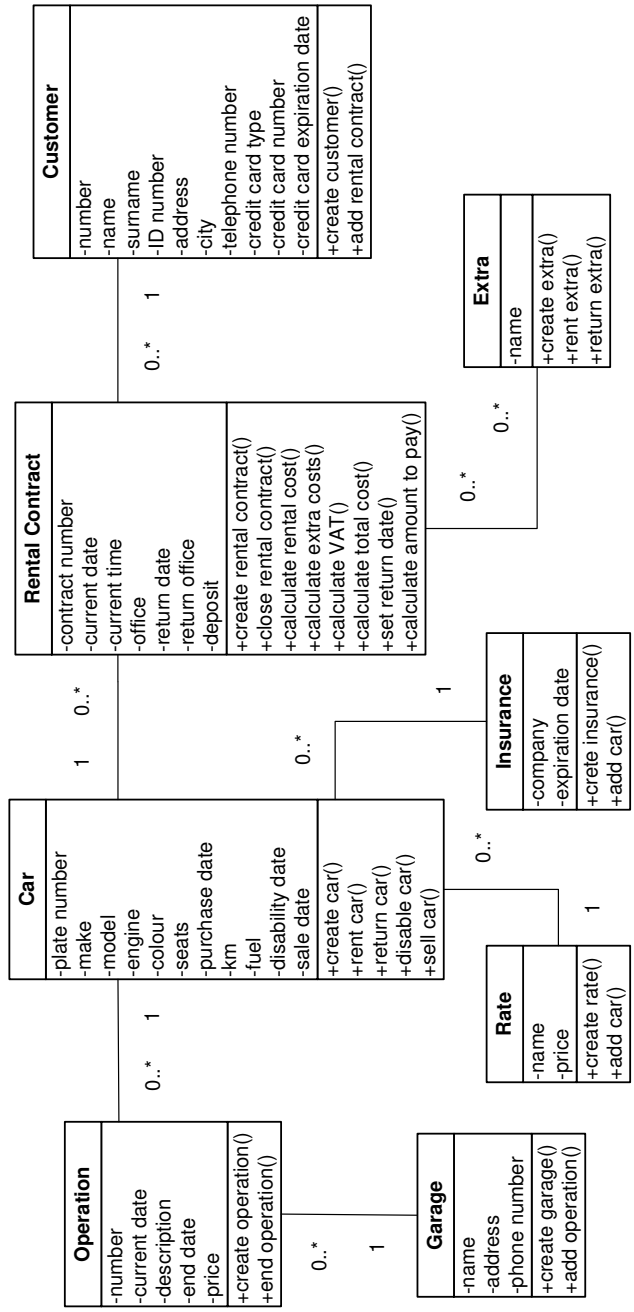

Figure 7.5 Example of class diagrams

Rule C2 (attributes)

An attribute is modelled in a class for each attribute that belongs to the domain entity from which the class was modelled and that is in an input or autonomous flow of an ETD. A data type must be specified for each attribute.

For the running example, the attributes of the class "Operation" are "number", "date", "description", "end date", and "price".

Rule C3 (creation method)

A creation method is modelled for each class. Its parameters are the attributes of the domain entity from which the class was modelled in the ETD(s) where the domain entity appears in an input or autonomous flow and it is part of the output of the ETD but not of the input. A data type must be specified for each parameter.

It must be noted that a class may have several creation methods, and it must have at least one. For the running example, the creation method of the class "Operation" is "create operation (number, date, description)", which is modelled from the ETD "Car Maintenance".

Rule C4 (deletion method)

A deletion method is modelled in a class if the domain entity from which the class was modelled can reach a state in the output of an ETD that is not used in the input of other ETD.

For the running example, a deletion method ("sell car") is modelled for the class "Car." Once it reaches the state "Sold" (in the ETD "Car Sale"), the state is not used in of other ETD.

Rule C5 (modification method)

A modification method is modelled in a class for each ETD in which the domain entity from which the class was modelled has attributes in an input or autonomous flow and a creation method of the class was not modelled from the ETD. Its parameters are the attributes of the domain entity in the input or autonomous flow of the ETD where the domain entity appears. A data type must be specified for each parameter.

For the running example, a modification method of the class "Operation" is "end operation (end date, price)", which is modelled from the ETD "Operation End".

Rule C6 (calculation method)

A calculation method is modelled in a class for each attribute that: 1) belongs to the domain entity from which the class was modelled; 2) is in an output flow; and 3) does not correspond to an attribute of the class. A return data type must be specified for each calculation method.

For the running example, a calculation method of the class "Rental Contract" is "calculate rental cost", which is modelled from the ETD "Car Rental" (Figure 6.9).

Rule C7 (relationships)

A relationship between two classes is modelled if there exists a relationship between the domain entities from which the classes were modelled that is created in some ETD. For inheritance relationships in a domain data model between domain entities from which classes are modelled, all the relationships are modelled.

For the running example, a relationship between the classes "Car" and "Rental Contract" is modelled from the ETD "Car rental" (the relationship "Includes").

Rule C8 (relationship creation method)

A relationship creation method is modelled in a class if no method has been defined in the class from the ETD in which a relationship of the domain entity from which the class was modelled is created.

For the running example, a relationship creation method of the class "Garage" is "add operation", which is modelled form the ETD "Car Maintenance".

Rule C9 (relationship deletion method)

A relationship deletion method is modelled in a class if no method has been defined in the class from the ETD in which a relationship of the domain entity from which the class was modelled is deleted.

For the running example, this rule is not applied. Nonetheless, it would have been if the relationship "Carried out in" between the classes "Operation" and "Garage" had been considered to be deleted after execution of the ETD "Operation End". This would imply that the rent-a-car company would not need to know the garage where an operation is carried out once it is finished.

Rule C10 (state change method)

A state change method is modelled in a class for each ETD in which the domain entity from which the class was modelled has different states in the input and output of the ETD and no other method has been modelled in the class from the ETD.

For the running example, the method "return extra" of the class "Extra" is a state change method. It is defined from the ETD "Car return".

Rule C11 (methods details)

The type of each method of a class must be indicated in the documentation of the class diagram.

For example, the method "return car" of the class "Car" is a modification method.

Rule C12 (relationship methods details)

If the methods related to creation and deletion of a relationship have not been modelled from the Rules C8 and C9, then the methods of a class from which a relationship with other classes is created and deleted must be indicated.

For the running example, the relationship "Carried out in" is created from the method "create operation" of the class "Operation" and the method "add operation" of the class Garage. The first one is defined from Rule C3, and the second one is from the Rule C8. This means that the method "create operation" is both a creation method and a relationship creation method.

Rule C13 (minimum multiplicity)

The minimum multiplicity of a class in an association or aggregation is 0 if the corresponding relationship creation method does not correspond to a creation method of the class. Otherwise, the minimum multiplicity is the minimum number of occurrences of the domain entity from which the class was modelled in the input flows of the ETD from which the relationship creation method was modelled (1, 0 if optional, or lower limit of repetitions).

For the running example, the minimum multiplicities of the association between the classes "Car" and "Rental Contract" are 0 for "Car" and 1 for "Rental Contract".

It must be noted that the minimum multiplicity of the compound class of an aggregation cannot be 0. Existence of the component class is only possible if the compound class exists. Otherwise, the relationship between the classes would be an association, not an aggregation.

Rule C14 (maximum multiplicity)

The maximum multiplicity of a class in an association or aggregation is the maximum number of times that the corresponding relationship creation method can be executed in the lifecycle of the class without executing the corresponding relationship deletion method. Such a number is calculated from the state transition diagram of the class.

For the running example, the maximum multiplicities of the association between the classes "Car" and "Rental Contract" are indeterminate ('*') for "Car" and 1 for "Rental Contract".

It must be indicated that the maximum multiplicity cannot be lower than the number of occurrences of the domain entity from which the class was modelled in the input flows of the ETD in which the relationship creation method (1 or upper limit of repetitions).

Rule C15 (integrity constraints)

The triggers, preconditions, postconditions and business rules of the ETDs must be checked to determine if some of them should be included in the documentation of the class diagram as integrity constraints. In addition, other integrity constraint may be discovered.

For the running example, an integrity constraint is that the insurance of a car must be valid during the rental period.

Rule C16 (derivation rules)

The business rules of the ETDs must be checked to determine if some of them should be included in the documentation of the class diagram as derivation rules. In addition, other derivation rules may be discovered.

For the running example, a derivation rule is that the total cost of a rental contract is calculated from the rate of a car and the price of the extras requested multiplied by the number of rental days and VAT.

7.5 State Transition Diagrams

A state transition diagram is derived for each class of the class diagram of an IS from (some sections of) its ETDs and its class diagram by following a set of rules. There are 11 rules, and they allow system analysts to model states (Rules S1, S2 and S3), transitions (Rules S4, S5 and S8) and preconditions (Rule S9) and postconditions (Rule S10) of the transitions. System analysts also have to indicate information when ambiguity exists (Rules S6 and S7) and to check that a diagram is correct (Rule S11).

As for derivation of the class diagram, some of the rules are not completely automatable (Rules S6, S8, S9 and S11).

Figure 7.6 shows the state transition diagram of the class "Car" for the running example, which is used as an example to explain the rules. It has been derived from the ETDs shown in Figures 6.9 and 7.4 and the class diagram shown in Figure 7.5. It must be noted that the diagram has not been modelled completely (e.g., preconditions are not shown) to keep Figure 7.6 as small as possible.

The rules are defined as follows.

Rule S1 (initial state)

An initial state is modelled in each state transition diagram.

This rule is always applied, thus it has been in Figure 7.6.

Rule S2 (final state)

A final state is modelled in a state transition diagram if the class has a deletion method in the class diagram.

A final state is part of Figure 7.6 because the class "Car" has a deletion method ("sell car").

Rule S3 (intermediate states)

An intermediate state is modelled in a state transition diagram for each state that the domain entity from which the class was modelled can reach in the ETDs and does not correspond to the state that the domain entity reaches in the ETD from which a deletion method was modelled. The names of the states are those of the domain entity in the in the ETDs.

The intermediate states of the class "Car" are "Ready", "Rented", and "Disabled".

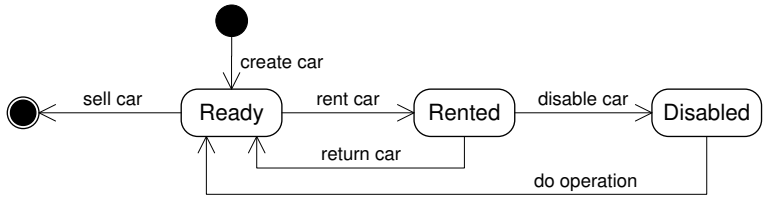

Figure 7.6 Example of state transition diagram

Rule S4 (first transition)

A transition is modelled from the initial state of a state transition diagram for each creation method of the corresponding class. The event of the transition is the creation method of the class and the target state is the state of the domain entity from which the class was modelled in the output of the ETD from which the creation method was modelled.

The first transition of the state transition diagram of the class "Car" is targeted at the state "Ready". Its event is "create car".

Rule S5 (last transition)

If a state transition diagram has a final state, then a transition is modelled to it for each deletion method of the corresponding class. The event of the transition is the deletion method of the class and the source state is the state of the domain entity from which the class was modelled in the input of the ETD from which the deletion method was modelled.

In Figure 7.6, the source state of the last transition of the class "Car" is "Ready". The event of the transition is "sell car".

Rule S6 (multiple possible target states)

If the domain entity from which the class was modelled can reach several (different) states in the output of an ETD, then the states that can be reached from each method modelled in the class diagram from that ETD must be indicated.

For the lifecycle of the class "Car", the states "Ready" and "Disabled" can be reached in the ETD "Car Return", from which the methods "return car" and "disable car" were modelled in the class diagram shown in Figure 7.5. The method "return car" allows the class "Car" to reach the state "Ready", whereas the method "disable car" allows the class "Car" to reach the state "Disabled".

Rule S7 (multiple possible transitions)

If several methods have been modelled in a class from a same ETD, then the states that can be reached from each method modelled in the class diagram from that ETD must be indicated.

This rule is not applied in Figure 7.6, but it is for the state transition diagram of the class "Rental Contract". The methods "set return date" and "calculate amount to pay" are modelled from the ETD "Car Return". It is considered that the first one is the event of a transition between the states "Open" and "Closed", whereas the second method represents the event of a transition whose source and target state is "Closed".

Rule S8 (intermediate transitions)

For each method of a class not analysed through Rules S4, S5, S6 and S7, a transition is modelled in its state transition diagram. The event of the transition is the method. If the state of the domain entity in the input of the ETD cannot be anyone ('*'), then the source state is the state of the domain entity in the input of the ETD. The target state is the state of the domain entity in the output of the ETD. If the state of the domain entity can be anyone (*'), then a cyclic transition is modelled in all the intermediates states.

In Figure 7.6, an intermediate transition for the class "Car" corresponds to the event "rent car". The source state is "Ready" and the target state is "Rented".

Rule S9 (preconditions)

The integrity constraints of the class diagram must be checked to determine if some of them are preconditions of some event.

For the transitions of Figure 7.6, a precondition for execution of the event "rent car" is that a car cannot be rented if it has more than 300000 kilometres.

Rule S10 (postconditions)

The integrity constraints of the class diagram must be checked to determine if some of them are postconditions of some event.

This rule is not applied in Figure 7.6, but it would have been if, for instance, it had been decided that a car could not be returned unless the

customer had paid the whole rental contract (i.e., the amount to pay was 0).

Rule S11 (diagram check)

The integrity constraints of the class diagram must be checked to determine if some of them impose restrictions on the lifecycle of a class that have not been modelled.

This rule is not applied in Figure 7.6. An example of application would be that an ETD had been specified in which it had determined that the domain entity "Car" was part of its input and could have any state ('*'). For example, an ETD for modification of the colour of a car (after it had been painted) may exist. Nonetheless, a business rule may also specify that a car cannot be used as input (i.e., its colour cannot be modified) if its state is "Sold", and this business rules would later turned into an integrity constraint of the class diagram. Furthermore, this rule would also cause the need to make changes in the class diagram.

If all the other rules (both the ones for derivation of the class diagram and the ones for derivation of the state transition diagrams) were applied, and for the example used in the previous paragraph, then the class "Car": 1) would not have a deletion method; 2) would not have a final state; 3) would not have a final transition; 4) would have an intermediate state "Sold", and; 5) would have a transition whose source and target states were "Sold" and whose event was the method modelled from the ETD for modification.

Consequently, the state transition and class diagrams derived would not be correct, but system analyst would have to modify them so that they meet system requirements (according to the integrity constraint).

The point on this discussion (as well as on many other aspects of the methodological approach) is that the rules determine mappings between artefacts when possible. They also aim that these mapping are deterministic and as automatable as possible. However, some details and specific information and needs that may exist are specified in a way in the methodological approach that makes the previous purpose unreachable. The purpose may be reachable, for instance, by imposing more restrictions on how ETDs should be specified, but it would also imply other trade-offs.

In summary, the rules could be more automatable and allow derivation of (100%) complete diagrams, but this would imply further work and restrictions on previous stages and steps that, in general, practitioners do not like. Furthermore, in most of the cases, the situations discussed do not occur, thus the necessary effort for more automatable and complete rules may be regarded as not worthy.

7.6 Further Link with OO-Method

This section presents and discusses how the methodological approach of the thesis (more concretely, a SyRS in the form of ETDs) could be further linked to OO-Method (Pastor, Molina, 2007).

OO-Method has more details and models that those that can be derived by following the rules presented in the two previous sections. Therefore, the link must be determined or at least suggestions and ideas about it must be provided so that the methodological approach could be more useful for those practitioners that use OO-Method. Nonetheless, and as mentioned below, a deeper analysis of the further link of ETDs with OO-Method is out of the scope of this thesis.

The following subsections outline OO-Method by presenting its conceptual models and how software can be generated by using it and discuss the link of the methodological approach with OO-Method.

7.6.1 Conceptual Modelling and Software Generation with OO-Method

As mentioned in Chapter 1, OO-Method is an approach for automatic software generation on the basis of OO conceptual modelling. It is supported by the OlivaNova tool and can decrease development time and increase productivity. Conceptual modelling with OO-Method is independent from the target technological platform (e.g., Java or .Net).

OO-Method consists of the following conceptual models (Figure 7.7):

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

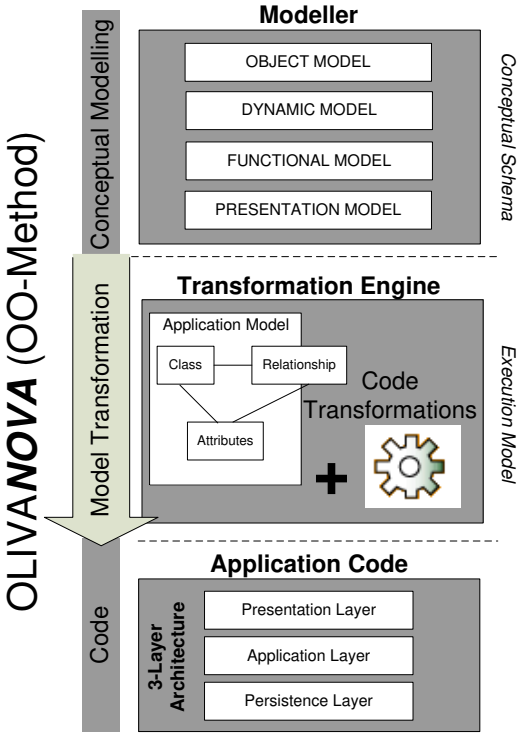

Figure 7.7 General view of OO-Method

- **Dynamic model:** this model specifies the dynamic and behavioural side of the classes of the object model by means of graphical diagrams that can be considered equivalent to UML state transitions diagrams; the valid lifecycles of the classes are represented in this model, as well as the possible interactions between the objects (i.e., instances of the classes).
- **Functional model:** this model specifies the semantics of the change of an object state as a result of method execution (e.g., a change in the number of kilometres of a car) by means of a declarative textual specification.
- **Presentation model:** this model specifies the characteristics of the user interface of a software system and how the users will interact with the system; the model is created by means of a pattern-based graphical model through 3 levels of detail, from more general to more specific characteristics.

OO-Method corresponds to a graphical representation for OASIS, which is an OO language for specification of ISs. OASIS is based on dynamic logic and process algebra, and allows OO-Method conceptual schemas to be formally defined. An OASIS specification also represents a high-level repository or data dictionary of a software system.

Applications generated from conceptual modelling with OO-Method have a three-layer architecture (Figure 7.7). The presentation layer contains the software components responsible for presenting users the application interface to interact with a software system. The application layer provides services that implement the functionality of an application. The persistence layer provides services to store and obtain the pieces of data necessary for execution of an application.

The software process of OO-Method consists of two stages. First, system analysts (modellers) create a conceptual schema, which corresponds to a representation of the problem space (i.e., the application domain). A UML-based notation and textual specifications are used. Second, the code of an application is generated on the basis of the Execution Model of OlivaNova, which corresponds to a representation of the solution space and can be targeted at different technologies.

When comparing OO-Method with other approaches for software modelling and development, it deals with the static (data-oriented) and the dynamic (behaviour-oriented) views of an IS. Both views are necessary for complete IS modelling and development. In addition, it relies on an underlying formal model, integrates formal and semi-formal techniques, and (in conjunction with OlivaNova) allows generation of complete and ready-for-running applications by precisely specifying an IS.

In relation to MDA (Model Driven Architecture; (OMG, 2003)), a detailed description of its correspondence with OO-Method can be found in (Pastor, Molina, 2007). The main points are that: 1) an OO-Method conceptual schema corresponds to a Platform-Independent-Model; 2) the execution model corresponds to a Platform-Specific-Model, and; 3) the code generated corresponds to an Implementation Model.

OO-Method was and is targeted at system modelling, thus it has weaknesses related to previous stages of software development (e.g., the RE process). As a solution, several approaches have been defined for the last decade to extend OO-Method and provide its users with means to

deal with requirements specification and modelling and with their link with OO conceptual modelling. In this sense, the methodological approach of this thesis arose to meet the need of OO-Method of having a business process-based RE approach.

Other RE approaches for OO-Method are based on use cases (Insfrán, Pastor, Wieringa, 2002), linguistic patters (Díaz, Sánchez, Matteo, 2005), the i* framework (Estrada, et al., 2006) and Communication Analysis (González, et al., 2011). Existence of such a set of approaches is in line with the recommendation of having and using different approaches and tools for different problems (Dieste, Juristo, Shull, 2008).

More details about OO-Method can be consulted in (Pastor, Molina, 2007). Some of them are presented in the next section to discuss further link of ETDs with OO-Method. Finally, some potential problems associated to IS modelling and development with OO-Method and that are related to requirements (e.g., difficulty of customer stakeholders to understand OO conceptual schemas) are discussed in Chapter 8.

7.6.2 Details and Discussion about the Link

Sections 7.4 and 7.5 have presented a "standard" way to derive the class diagram of an IS and the state transition diagrams of its classes. It could be used for any OO conceptual modelling-based approach for IS development that uses those diagrams. However, OO-Method has specific characteristics that differentiate it from other approaches. They must be considered for further link of ETDs with OO-Method. Anyway, other OO diagrams (e.g., UML diagrams) can be derived from and transformed into OO-Method diagrams (Giachetti, Marím, Pastor, 2009).

The distinctive models and characteristics of OO-Method are a consequence of an important characteristic and need on it. For generation of a (100%) complete application, a conceptual schema must be precise and complete. Otherwise, a model compiler could not generate the application.

Most of the specific details for further link with OO-Method are out of the scope of this thesis. For example, no specific way to link ETDs to the presentation model of OO-Method is defined. Nonetheless, the following subsections outline the details of the different models and characteristics of OO-Method that can or may be specified from ETDs. When not

considered possible (without much further study), the way in which the models and characteristics should be specified is explained. Specification would be based on information from or on agreement with customer stakeholders.

It must also be indicated that system analysts that use OO-Method can model some products requirements that may not have been specified in the ETDs of an IS (e.g., a help message in user interface) and which should be elicited from customer stakeholders too.

In addition, it is common that system analysts include design-level details in the OO-Methods diagrams. Such details correspond to design decisions that cannot be directly derived from requirements, unless correspondence patterns were defined. For example, an association between classes could be modelled as an inheritance relationship because of a design decision related to the final implementation of the system and data storage into a database management system.

Finally, it must be indicated that not only a single way to use OO-Method exists, but its models can be used and combined in different ways to generate an application. A concrete use depends on the preferences of a system analyst. For example, some analysts prefer not to model state transitions diagrams but to restrict service execution (on the basis of the states of a class) in the object model.

Consequently, some details about further link with OO-Method that may seem redundant because they are indicated in other model actually are not. The point is that some ways for further link would be necessary or not depending on the preferences of the system analysts. Some of the indications provided may not be necessary because they correspond to information specified in other part of an OO-Method conceptual schema.

7.6.2.1 Object Model

The characteristics and details of the object model of OO-Method that are not defined from the rules presented in Sections 7.4 and 7.5 but that could (and in general must) be specified in an OO-Method conceptual schema are the following ones.

- Attributes

  For each attribute of a class, system analysts must indicate if it is an identifier of the class, and may also indicate its default value.

Constant attributes would be those that correspond to the parameters of the creation method(s) of a class and are not later modified by other method, and variable attributes would be the rest. A calculation method (i.e., the result of its execution) could be transformed into a derived attribute. Required on creation attributes would be those that correspond to the parameters of the creation method of a class. An attribute could have a null value if it does not correspond to one of the parameters of the creation method.

- Services

  In general, OO-Method services of classes correspond to the methods of a class diagram. Nonetheless, OO-Method defines some special types of services. Shared events correspond to pairs of methods of two different classes that must be executed in conjunction so that their effect is relevant. These services would correspond to the relationship methods of a given relationship, which affect more than one class. Transactions represent the joint execution of a set of services. These services would correspond to a set of methods that must be executed in an ETD for its completion. In addition, the creation and destruction (deletion) methods are graphically specified (shown) by means of stereotypes.

- Arguments

  OO-Method arguments correspond to the parameters of the methods of a class diagram. As attributes, they can have a default value and be null. In the case of output arguments, their value expression would correspond to an integrity constraint or derivation rule.

- Agents

  OO-Method agents are a special type of classes that depict that a class can execute services of other classes. IS users would be represented in the OO-Method object model as agents, even though they did not correspond to classes already modelled in the class diagram. The services that they could execute could be determined from the information flows of the ETDs. For example, a user (i.e., an agent) could execute a method of a class that has

been defined from an input flow of him in an ETD. Attribute visibility must also be specified for agents. This could be performed from the output flows of the ETDs.

- Associations

  In OO-Method, the roles of the classes of an association must be specified. In addition, associations can be dynamic or static. Dynamic associations mean that the object of an association can change during the lifecycle of the other object. Static associations mean that the relationship is constant. Determination of a dynamic association would be based on the number of times that a relationship creation method can be executed without executing its corresponding relationship deletion method, in a similar way to how maximum multiplicity of a relationship is determined. For dynamic associations, both relationship methods should be executed in the same ETD. First the relationship deletion method should be executed, and then the relationship creation method.

- Identification dependency

  In an aggregation, a component class may need its corresponding composed class to be identified. Existence of such a dependency must be specified.

- Other characteristics of associations

  In OO-Method, associations can be: 1) flexible or strict; 2) disjoint or non-disjoint; 3) null or not null, and; 4) mono-valued or multi-valued. The cases (3) and (4) can be determined from the multiplicities of the relationships of a class diagram. A strict association corresponds to the aggregation defined in the methodological approach. Disjoint and not disjoint associations must be determined.

- Specialization and generalization

  OO-Method specialization and generalization corresponds to the inheritance relationships of the methodological approach. A specialization is temporary if an object can switch between the parent class and the child class, and it is universal otherwise. A temporary specialization would be detected because of the existence of creation or deletion methods associated to an

inheritance relationship of a class diagram. Generalizations can also be disjoint or non-disjoint, and this fact must be indicated.

- Derivations, preconditions and integrity constraints

  OO-Method has its own syntax for specification of derivations, preconditions and integrity constraints. Therefore, those defined in the class diagram or the state transition diagrams must be translated.

7.6.2.2 Dynamic Model

The characteristics and details of the dynamic model of OO-Method that are not defined from the rules presented in Sections 7.5 and 7.6 but that could (and in general must) be specified in an OO-Method conceptual schema are the following ones.

- Elements of the state transition diagram

  In OO-Method, the initial state of a class is called pre-creation state, the final state is called destruction state and the intermediate states are called simple states. Events are called actions and preconditions are called control conditions. Agents responsible of action execution can also be defined, and they could be derived from the ETDs as explained above. OO-Method does not directly support specification of postconditions for the events of the transition, but they can be specified with other mechanisms (e.g., by including a "fictitious" service to check the postcondition and defining a transaction that includes the event and the new service).

- Statecharts

  Complexity of state transition diagrams can be managed in OO-Method by defining statecharts from them, which could be modelled from the "standard" state transition diagrams derived with the methodological approach too.

- Elements of the object interaction diagram

  The object interaction diagram complements the state transition diagrams of the classes of a software system by specifying interaction and communication between objects. These objects can be instances of a same class or of different classes. In relation to

the link of ETDs with an object interaction diagram, triggers could be derived from those methods of a class defined from autonomous flows as well as from the triggers of the ETDs. Candidates for global transactions and interactions are those sets of methods that are modelled in a class diagram from a same ETD and have been defined from autonomous flows.

It must be indicated that many details of the dynamic model of OO-Method do not correspond to requirements information but to design decisions. This is due to the fact that they specify internal characteristics of a software system, which are out of the scope of this thesis. For example, and in general, the way in which objects (instances of the classes) will communicate each other is considered a design decision that system analysts must make.

7.6.2.3 Functional Model

The characteristics and details of the functional model of OO-Method that are not defined from the rules presented in Sections 7.4 and 7.5 but that could (and in general must) be specified in an OO-Method conceptual schema are the following ones.

- Evaluations

  Evaluations of the functional model of OO-Method allow specification of service effect. For example, assignation of a new value to an attribute of a class as a result of service execution is specified by means of an evaluation. In general, all the methods defined in the class diagram that have parameters imply determination of the initial values or changes in the values of the attributes of a class. For more complex evaluations (that do not represent design decisions), their formulae should be defined on the basis of the integrity constraints and derivation rules of the class diagram. Nonetheless, it may be possible that no business rules had been defined (e.g., because it was not considered necessary before). In this case, system analysts should discover service effect.

- Evaluation conditions

  Execution of an evaluation may be constrained by a condition, i.e., a precondition. Such a precondition should correspond to an

integrity constraint of the class diagram, a precondition of some transition of the sate transition diagram or a business rule of an ETD.

- Default values in evaluations

  When specifying evaluations, system analysts can specify default values by using functions. For example, "getSystemDate()" is a function that could be used to indicate that the system should obtain and store the current date. These functions may be used for specification of evaluations for methods modelled from autonomous flows. For input flows, users are responsible for introducing the values that they want the IS to store, thus no automatic generation is necessary.

7.6.2.4 Presentation Model

Modelling and specification of the user interface of and of the interaction with a software system can be considered a very creative activity. It highly depends on system analysts' expertise and is strongly constrained by end-users' preferences. Therefore, a deterministic mapping between ETDs and the presentation model of OO-Model is hard to define. It would require further research, which is out of the scope of this thesis.

Nonetheless, some characteristics and details of the presentation model of OO-Method that are not defined from the rules presented in Sections 7.4 and 7.5 but that could (and in general must) be specified in an OO-Method conceptual schema are the following ones.

- Level 1: Action hierarchy tree

  This level represents the way in which users and a system will interact, e.g., by using a menu in which users will select options. In relation to ETDs, it is considered that the action hierarchy tree of an application whose conceptual schema has been created after performing the methodological approach should be structured according to business processes and ETDs. This would imply that the user interface would be task-oriented, and that end-users would indicate the business process and the ETD (of all the associated with that business process) that they need or want to execute.

- Level 2: Interaction units

  This level represents the specific units for interaction (e.g., a user interface for execution of an ETD) and the possible navigations between them. On the basis of the information flows of the ETDs, initial interaction units may be defined by analysing the information that a (human) user and a system exchange. For example, if a system shows a set of objects of a class ("← $_1${ Car / make + model / }$_n$"), then the population pattern (which can be used to show all the instances of a class) should be used. Navigation may be based on the order in which the states of the classes can be reached, or even execution order of the ETDs could be determined, for instance, as a part of ETD analysis.

- Level 3: Basic elements

  This level represents the specific components of the interaction units, e.g., the set of buttons that will be displayed. Again, it is considered that an initial set of basic elements could be derived from the ETDs of an IS. For example, if an input flow from a (human) user includes attributes, then something (e.g., a textfield) is necessary in the user interface to introduce such information.

Finally, it must be indicated that no single user interface for and interaction with a software system exists, but different and alternatives ones can be defined. Even though an initial presentation model was derived from ETDs, the model may require many modifications. As indicated many times during presentation of the methodological approach of the thesis for different aspects of the RE process, just one rule exist for modelling of the user interface and of the interaction: that customer stakeholders like and agree upon the proposed alternative.

7.6.2.5 Conceptual Modelling of Legacy Systems

A characteristic of conceptual modelling with OO-Method that has not been mentioned above is the possibility of modelling elements of a legacy system with which a new software system should interoperate.

In relation to the methodological approach, such systems have not been considered much. The only mechanisms provided have been: 1) the definition of a label to indicate that a flow object of a To-Be BPD would be controlled by a legacy system, and; 2) the acknowledgement of the

possibility of the fact that an IS may interoperate with other systems, that these system should be regarded as users and that interoperability details can be specified in the quality attributes sections of an ETD. Both mechanisms were presented in Chapter 6.

Modelling of legacy elements has not been considered when defining the methodological approach, but it is not conflicting with it. Furthermore, if such elements existed, then it could be assumed that its system requirements and its conceptual schema would be already known. Therefore, the main objective of the derivation of OO diagrams stage (to obtain a complete and consistent OO conceptual schema that meets system requirements) would have already been achieved.

The main difficulty of including legacy elements in an OO-Method conceptual schema would be to exactly determine how they would interoperate with a new IS. Such information should be obtained from customer stakeholders (those who know how the legacy system works) and may also imply the need of making design decisions related to internal management of the interoperability in the new IS, which are out of the scope of this thesis.

7.7 Summary

This chapter has presented derivation of OO diagrams, the fourth stage of the methodological approach of the thesis. This stage allows system analysts to obtain an OO conceptual schema of an IS that meets its system requirements and is complete and consistent from a requirements-perspective. Consequently, a SyRS in the form of ETDs is linked to OO conceptual modelling-based IS development.

For performing the stage, first ETDs are analysed to specify several details that are necessary for derivation of the OO conceptual schema of an IS. Next, a class diagram and state transition diagrams are modelled by following two sets of rules that determine the correspondence between the system requirements of an IS and its OO conceptual schema. Most of the rules are fully automatable, but system analysts must always make some decisions when deriving an OO conceptual schema.

Finally, further link of the methodological approach with OO-Method has been discussed. The initial OO conceptual schema derived can be useful for any OO conceptual modelling-based approach for IS development, but OO-Method has specific and distinctive characteristics

that require a deeper analysis of the correspondence between ETDs and an OO-Method conceptual schema.

For those elements of an OO-Method conceptual schema that are not derived by following the rules proposed for derivation of a class diagram and of the state transitions diagrams of the classes, the (possible) way to model or specify them has been presented. As a consequence, more information in the object, dynamic, functional and presentation models of OO-Method could be determined from ETDs. Nonetheless, many specific details about the link require further research that is out of the scope of this thesis.

Chapter 8
Evaluation

"In theory, there is no difference between theory and practice. But, in practice, there is"

Jan L.A. van de Snepscheut

Once presentation of the stages of the methodological approach of the thesis is finished, this chapter describes the evaluation that has been performed to validate it. Evaluation has three main characteristics. First, it has been targeted at gaining insights into the practical usefulness of the methodological approach. Second, it corresponds to a qualitative (flexible) research approach. And third, it is mainly based on a survey with industry stakeholders.

Interviews have been conducted after explaining and applying the methodological approach by means of different methods for validation of RE approaches. As result of the survey, several lessons about the methodological approach have been learned.

The chapter is organized as follows. First, evaluation of RE approaches in industry and qualitative research are discussed. Next, the methods, the industrial contexts and the process for evaluation are described. Validity and lessons learned are then presented and discussed. Finally, a summary of the chapter is provided.

8.1 Background: Evaluation of RE Approaches in Industry

Gaining insights into RE in industry is recognised as one of the main needs for RE research because of the gap that exists between academia and practice (Paech, et al., 2005). This is related to the fact that RE approaches are not usually empirically evaluated. For example, a recent study (Condori, et al., 2009) showed that very few publications on techniques for requirements specification had an empirical evaluation.

Furthermore, most of the evaluations are not performed in industrial contexts (see the references of the previous paragraph), but in laboratory environments. If RE approaches are not evaluated in industry, then their adoption in practice can be hindered (Ivarsson, Gorschek, 2009).

Although the validity and implications of laboratory evaluations have been discussed and justified in literature (e.g., the adequacy of using students as subjects (Runeson, 2003)), generalizability is always affected. For example, and in general, conclusions from students would just be useful to characterize the situation of novice practitioners.

As a solution, several authors have acknowledged the need and convenience of evaluating RE approaches in actual industrial contexts (e.g., (Davis, Hickey, 2002; Potts, 1993; Wieringa, 2005)). For example, a RE approach should be evaluated with practitioners that use it or may use it. This does not mean that evaluation in laboratory is inappropriate but that evaluation in industry is considered more adequate.

Users can indicate how useful they considered a RE approach is or can be on the basis of their expertise, as well as possible weaknesses and improvements in relation to actual practice. This is the line that has been followed for evaluation of the methodological approach.

In summary, evaluation of the methodological approach of the thesis aims to gain insights into its practical usefulness, thus it has been (mainly) validated with industry people. Such people can play the different roles of the stakeholders taxonomy (Chapter 2): customer stakeholder (customer manager, employee and end-user) and supplier stakeholder (supplier manager, system analyst and programmer). These roles are considered to correspond to the users (and stakeholders) of the methodological approach, thus their feedback about it is very valuable.

8.2 Background: Qualitative Research

When addressing empirical evaluation (or research) in any research field in general (Robson, 2002) and in software engineering in particular (Wohlin, et al, 2000), two overall approaches exist: quantitative (aka fixed) research and qualitative (aka flexible) research.

Quantitative research is mainly targeted at quantifying a relationship between or comparing two or more groups as a result of the discovery of a cause-effect relationship. Qualitative research is mainly targeted at studying objects in their natural setting and at interpreting phenomena on the basis of explanations from people of the setting (Wohlin, et al., 2000). Nonetheless, a mixed approach could be adopted too (Robson, 2002). For example, a research approach may be used to facilitate design of the other, or triangulation may be used to check results obtained from a quantitative approach with those obtained from a qualitative approach (or vice versa).

For evaluation of the methodological approach of the thesis, a qualitative research approach has been adopted. Qualitative research is usually characterised by the production of qualitative data (though quantitative data can be produced and analysed too), the lack of a complete pre-specification of the analysis to perform on the pieces of data, and the possibility of design evolution as evaluation proceeds. Qualitative research also aims to investigate and understand phenomena within their real context and to seek new insights, ideas and possible hypotheses for future research (Robson, 2002).

Other characteristics of qualitative research are that it is targeted at learning from others, its data typically come from fieldwork by interacting with people about their experiences and perceptions, the pieces of data produced are more detailed than those produced in quantitative research but comes from a smaller number of subjects, the researcher is usually considered the instrument, and the data do not include judgements but simply describe phenomena (Patton, 2001).

The above aspects about qualitative research allow the evaluation of the methodological approach to be considered to correspond to a qualitative research approach. This fact is further explained and shown in the next sections.

8.3 Methods for Evaluation

As mentioned above, evaluation of the methodological has been mainly based on a survey with industry people. Details about them and the industrial context in which evaluation has been performed are presented in the next sections.

A survey is not just the instrument (e.g., the questionnaire or checklist) for gathering information. It is a comprehensive research method for collecting information to describe, compare or explain knowledge, attitudes and behaviour (Fink, 1995). It is usually performed in retrospect, for instance, after an approach has been presented or used, and usually aims to produce descriptive conclusions (Wohlin, et al., 2000).

A set of methods for validation of RE approaches is presented in (Wieringa, 2008). Among them, methods in which an approach is applied by its designers have been used in evaluation of the methodological approach. Application of these methods allowed the survey to be performed in retrospect, after the participants had been explained about the methodological approach, had been involved in its application or had observed it.

The methods that were used are the following ones.

- Lab demo

  This method consists in using a RE approach on a realistic example in an artificial environment in order to show that the approach could work in practice. For evaluation of the methodological approach, lab demos allowed analysis of the mechanisms and guidelines defined, identification of potential weaknesses that stakeholders may find and, consequently, proposal of improvements. Lab demos were conducted prior to evaluation with industry people.

- Illustration

  This method consists in using a RE approach in a small example in order to explain it and allow someone to understand it. For evaluation of the methodological approach, illustrations allowed presentation of the approach to industry people before getting feedback from them.

- Field trial

  This method consists in using a RE approach in the field to gain knowledge and show that the approach can be used in practice. For evaluation of the methodological approach, field trials allowed application of the approach in real settings in order to identify actual weaknesses and improvements (i.e., based on industrial practice) related to application. Customer stakeholders were directly involved in application of the methodological approach, whereas supplier stakeholders observed application. Feedback was obtained during and after the development of the field trials.

In summary, lab demos allowed analysis of application of the methodological approach in existing, known cases to check if it would be effective in past projects. Documentation about the projects and the proposed solutions were available. Illustrations allowed development of initial in-lab evaluations with stakeholders in order to know their opinion about the methodological approach and their experience with the challenges and objectives that it addresses.

With regard to field trials, they were performed as follows. First, meetings with customer stakeholders to understand and model their organizations were held. They described the activity of the organization, organizational documentation was compiled and organizational modelling was performed. As-Is BPDs were then modelled and customer stakeholders validated them.

Second, purpose analysis was performed to discover means to meet systems goals and to determine and agree upon their effect on business processes. In most of the cases the purpose was straightforward (mainly automation), thus the stage was not completely performed and To-Be BPDs and As-Is BPDs were very similar.

Next, system requirements were specified and validated by customer stakeholders, and OO diagrams were derived. Finally, development of the field trials was discussed with the stakeholders that participated.

Although iterations were performed in all the stages (i.e., modifications were performed after initial validation by customer stakeholders), the stage in which more iterations were necessary to obtain the final versions of its output artefacts was organizational modelling.

It must be noted that data collection was mainly based on feedback obtained from stakeholders by means of interviews. Nonetheless, it is recognised that data collection can also be performed, for instance, from available documents (Yin, 2009), as done in the lab demos. In this case, existing proposed solutions and solutions obtained by applying the methodological approach were compared.

Although evaluation is mainly based on survey design, characteristics of other methods for evaluation can be found. This is a result of the very thin line that exists between different research methods (Wohlin, Höst, Henningsson, 2003). Research methods share characteristics, thus their application can be similar in several aspects.

Aspects of case study research (Runeson, Höst, 2009; Yin, 2009) can be found in evaluation. For example, evaluation studied contemporary phenomena in its natural context, was (mainly) exploratory, qualitative and flexible, addressed ways (how questions) in which industry works and the reasons (why questions) behind past problems and current opinions, and there were many more variables than data points.

Post-mortem analysis (Wohlin, Höst, Henningsson, 2003) is conducted on the basis of past events. For example, past experiences in IS development were discussed with participants in evaluation. This method can be performed by looking at project documentation (as done for lab demos) or by interviewing people, and focuses on typical situations that have occurred. The basic idea behind post-mortem analysis is to capture the knowledge and experience from a specific case or activity after it has been finished.

Some authors have called interview study to similar or related evaluations (e.g., (Berntsson-Svensson, Gorschek, Regnell, 2009)). However, in general it is considered that interviews are more a method for data collection than a research method itself (e.g., (Kitchenham, Pfleeger, 2008; Robson, 2002)).

Finally, it must be indicated that the methodological approach has been developed progressively and changes have been made as a result of evaluation. Weaknesses have been discovered and mitigated, and advantage from possible improvements has been taken. For example, relationship methods (Chapter 7) were not initially addressed. Modelling of a relationship was considered enough, but practitioners' opinion about the convenience of the methods pointed that they should be considered.

8.4 Industrial Contexts

This section presents the different industrial contexts where the methodological approach has been evaluated. They correspond to the situations in which illustrations and field trials were performed, and are divided into two types of contexts: evaluation in a collaborative project with a company (hereafter referred to as project partner) and evaluation with other industry partners.

The next subsections present more details about each context.

8.4.1 Collaborative Project

This thesis arose from a collaborative project between Unniversidad Politécnica de Valencia and a company that uses OO-Method. The purpose of the project was to link business and system domains in order to solve problems that the project partner experienced and were related to requirements. These problems mainly arose when system analysts were inexperienced, they modelled large or complex systems, or the organization for which an IS was going to be developed was part of an application domain with which the system analysts had not previously dealt.

As explained previously in the thesis, OO-Method can decrease development time and increase productivity. However, these advantages might disappear if requirements-related problems arise. System analysts might have difficulty in modelling systems, and systems may be deployed later than planned.

After analyzing its requirements practices, it was argued that the project partner did not properly address business understanding, communication with customer stakeholders, and, therefore, requirements elicitation, specification and validation.

When OO-Method was applied in the project partner, the object model was the main system model, and system analysts usually just provided some unstructured textual descriptions about the requirements and validated them on the object model or on the final application. Senior system analysts felt comfortable with this approach, but it was considered that it should be improved:

- Class diagrams (e.g., an object model) alone might not be appropriate for communicating and validating requirements, there are few studies addressing the ability of customer stakeholders to understand class models, and they can be complex for people that have not been trained in object-oriented modelling (Dobing, Parsons, 2000; Lubars, Potts, Richter, 1992). In addition, objects might not be a good way of thinking about an application domain (Vessey, Coner, 1994).
- Requirements validation should be carried out with reference to organizational concerns instead of with reference to system functionality (Rolland, Prakash, 2000). Furthermore, requirements validation on the generated applications can cause late detection of errors, thus their correction might be much more expensive than if errors had been detected in earlier development stages (Davis, 1993).
- Maintenance might be complex for systems whose analysts do not longer work for the project partner. System and requirements documentation is usually scarce, thus system modification can be very difficult for new analysts of a system when they just have an OO-Method conceptual schema and the application to understand the system and its requirements.

As traditional approaches for conceptual modelling (e.g., (Olivé, 2007)), system analysts focused on obtaining the conceptual schema of an IS instead of on analysing the application domain. However, understanding of the application domain is essential for project and system success (as explained below and justified in works such as (Jackson, 1995)), and focus on obtaining a conceptual schema and disregard of the process for creation may lead the RE process and a software system to failure (Insfrán, Pastor, Wieringa, 2002; Rolland, Prakash, 2000). System analysts needed detailed guidance for developing the RE process so that a conceptual schemas meet system requirements.

As a solution, it was decided to develop a business process-based RE approach for OO-Method. Its current state corresponds to the methodological approach of the thesis. It must be indicated that none of the RE approaches defined for OO-Method (Chapter 7) had been widely adopted by the project partner and they were not used when the collaborative project started.

A very positive point of collaborating with the project partner was that it belonged to a holding company. As a result, other organizations of this company participated in evaluation of the methodological approach. The organizations were small/medium-size and operated in different application domains (water supply, insurance, apartment rental…). Since the project partner had developed ISs for the organizations previously, comparison among the approach that the system analysts usually used and the methodological approach was also possible.

Employees of the project partner played the supplier stakeholder role in evaluation, whereas employees of the other organizations of the holding company played the customer stakeholder role.

8.4.2 Other Industry Partners

In addition to the project partner, the methodological approach was evaluated with other industry partners. These industry partners can be divided into two categories: customer stakeholders and supplier stakeholders.

Customer stakeholders of the other industry partners that participated in evaluation corresponded to people that had played or were playing the customer role in IS development. They worked as managers or employees for their organizations and were IS users. The organizations were part of different application domains (automobile manufacture, construction, food and goods manufacture, food and goods distribution, public transport, media, telecommunications, finance, public administration, education, agriculture and hotel business).

In addition, the methodological approach was evaluated with supplier stakeholders. More specifically, managers (including company managers and project leaders), system analysts and programmers from several software development companies participated in evaluation. The application domain at which the software systems of the companies were targeted was variable in most of the companies (i.e., they companies did not only develop ISs for a domain). Nonetheless, some of companies were specialised in specific domains (e.g., finance, public administration, education and telecommunications).

In both cases, participants' backgrounds were heterogeneous (education, years of experience, number of IS development projects…).

When field trials were performed with other industry partners, just a part of an organization was used. Therefore, they probably should be regarded as illustrations. Although they may be regarded as field trials because actual cases were used, they mainly aimed to show application of the methodological approach by means of an example.

8.5 Evaluation Process

This section describes the evaluation process performed to conduct the survey. Information about the methods for validation of RE approaches that were applied before data collection has been presented in Section 8.3. The evaluation process is presented as proposed in (Kitchenham, Pfleeger, 2008).

Presentation of the evaluation process mainly focuses on the part of the survey related to the interviews (after use of illustrations and field trials). Nonetheless, details about lab demos are provided when considered necessary.

8.5.1 Objectives Definition

As already mentioned, the main goal of the survey was to gain insights into the practical usefulness of the methodological approach (in IS development). The research questions analysed in relation to application of the approach were:

RQ1: What weaknesses may stakeholders find?

RQ2: What advantages do stakeholders find?

RQ3: What limitations do stakeholders find?

Lab demos were used to answer the first questions, whereas illustrations and field trials were used to answer the rest of questions. In relation to the advantages and limitations, analysis of both current and past situations and experiences was aimed, trying to discover and understand the reasons behind stakeholders' perspectives.

8.5.2 Survey Design

In addition to qualitative, the survey is both cross sectional and longitudinal (Kitchenham, Pfleeger, 2008), and explorative (Wohlin, et al., 2000). Interviews were chosen for data collection.

The survey was cross sectional because most of the participants were interviewed once and at a fix point of time. Nonetheless, others were interviewed several times, thus the survey was longitudinal too. For example, some employees of the project partner provided feedback several times, as development of and modifications in the methodological approach were presented to them.

The survey was explorative because it mainly aimed to get data and analyse results to improve the methodological approach from a practical perspective. The survey was not targeted at determining the distribution of some perspective (as a descriptive survey) neither at explaining a given aspect of a population (as an explanatory survey). It also aimed to discover what is happening in little-known situations, what is almost exclusive of qualitative research (Robson, 2002).

With regard to the interviews, they are one of the most common methods for data collection in qualitative studies (Patton, 2001) and in surveys (Wohlin, et al., 2000). Among their types, unstructured interviews (Robson, 2002) were conducted.

In this type of interviews, the interviewer has a general area of interest and concern but lets the conversation develop within this area. The interview can be completely informal. For example, a meeting in which people discuss about a specific subject (i.e., the area of interest) can be considered an unstructured interview.

The areas of interest of the interviews were related to the research questions of the survey. They focused on the challenges and objectives addressed in the methodological approach (Chapter 1) and on its principles (Chapter 3), as well as on the artefacts, mechanisms and guidance proposed (Chapters 4, 5, 6 and 7). As mentioned above, feedback from both past and current experiences was aimed.

8.5.3 Development of a Survey Instrument and Instrument Evaluation

Given that interviews were unstructured, a specific instrument was not developed (consequently neither evaluated). The researchers (i.e., the designers of the methodological approach) were regarded as the instruments, what it is usual in qualitative research as explained above.

How validity was affected by the lack of a specific survey instrument is discussed in Section 8.6. Nonetheless, it must be noted that all the interviews focused on aspects (challenges, objectives and principles) of the methodological approach. Such aspects have been defined both from literature and from problems found in industry. This fact can be considered to address the step related to search the relevant literature when developing a survey instrument. Furthermore, references and details about works whose results and conclusions are in line with the lessons learned from evaluation are provided in Section 8.7.

8.5.4 Data Collection

The designers of the methodological approach were the interviewers for data collection. Participants in the survey were selected through a combination of convenience, snowball and maximum variation sampling (Patton, 2001). Known (industry) people were contacted in order to ask them about the possibility of providing feedback about the methodological approach. All of them worked in Spain, for regional, national or international organizations.

In the case of the project partner, meetings were agreed and held periodically in order to explain and discuss the progress of the methodological approach. For application of the methodological approach with people from other organizations of the holding company of the project partner, the project partner was in charge of agreeing with the other organizations when they would participate and who would participate.

The people contacted and those who work for the project partner and for other organizations of the holding company corresponded to people that played the roles of the stakeholders taxonomy (proposed) in their organizations. It was planned to try to obtain feedback from at least 5 participants of each role.

In relation to the number of subjects that participated in the interviews, the interviews with participants from other industry partners were individual. The interviews with participants from the project partner and from the other organizations of the holding company were group interviews, in which several people participated in the discussion and provided feedback, in a similar way to workshops.

For data collection, the duration of the interviews and meetings were limited to two hours, although in some case less time was enough. All interviews lasted at least 1 hour. Data were collected by writing and taking notes about facts and opinions that the participants provided in relation to the methodological approach and past and current experiences.

Although it is recognised that taping and transcription are better suited when conducting interviews (Patton, 2001; Robson, 2002), problems were found when people were requested to consent to record the interviews. Most of them did not agree upon (they did not feel comfortable), thus taking notes was finally adopted for data collection in all the interviews.

Since the methodological approach has been defined progressively and modifications have been made as weaknesses and improvements were found, feedback about the whole approach was not obtained throughout all the evaluation. For example, feedback from the project partner and people that work for other organizations of the holding company was not obtained for the current sate of the derivation of OO diagrams stage. The collaborative project finished before the current state of the stage was reached.

In many interviews only feedback from specific aspects of the methodological approach was obtained, i.e., feedback from those aspects in which the interviewees were more interested or that were more familiar to them. In addition, feedback for the whole methodological approach could not be obtained in other cases because enough interviews could not be conducted. As a consequence of the little available time that many participants in evaluation had, they could not be interviewed as many times as necessary and desired.

With regard to the lab demos, designers of the methodological approach used information about past projects (realistic examples) that they had or that they were provided with. Since the methodological approach has been defined progressively and modifications have been made, it was been applied several times in some examples. This means that different versions of the approach were applied in the same examples.

8.5.5 Analysis

Table 8.1 shows the number of interviewees for each role of the stakeholders taxonomy that participated in the survey and whose results from interview analysis are reported. The number of organizations involved is shown too. With regard to lab demos, 6 examples were used.

Further details (i.e., background information) about the participants cannot be provided for confidentiality reasons. Nonetheless, some agreed on presentation and explanation of specific situations related to their experience when presenting the lessons learned, provided that the information was presented in an anonymous way. Therefore, more details about the practice and experience of the participants and their organizations, and that could identify them, cannot be provided.

Analysis was performed for 49 people from 38 organizations. The organizations of employees and end-users coincided in 5 cases, of supplier managers and system analysts coincided in 1 case, and of system analysts and programmers coincided in 1 case. In addition, 4 system analysts worked for the same company. As explained below, results from some interviews (people different to these 49) are not reported.

Although some participants played different roles in their organizations (e.g., a person could be both an employee and an end-user), just one of the roles was considered for the survey. In summary, each interviewee of Table 8.1 corresponds to a different person.

The main activity of analysis was to partition the responses. It consisted in identifying how similar/distinct the feedback from different participants was in order to obtain a set of lessons learned (Section 8.7) from evaluation and try to improve the methodological approach.

Table 8.1 Summary of subjects

Stakeholder role	Number of interviewees	Number of organizations
Customer manager	5	5
Employee	10	10
End-user	9	9
Supplier manager	8	7
System analyst	10	7
Programmer	7	7

Both individual interviewees, sets of interviewees and organizations were analyzed. It was determined that lessons learned may correspond to perceptions, experiences or practices (e.g., importance of a principle of the methodological approach in industry) from specific types of stakeholders (first column of Table 8.1) or sets of specific types of stakeholders (e.g., customer stakeholders), or to organization-wide issues (e.g., use of a given approach for requirements specification).

Three types of lessons learned were sought on the basis of their source according to three criteria:

- Lessons learned that corresponded to direct feedback from interviewees: they corresponded to facts and opinions that the participants provided during evaluation of the methodological approach; their acronym is DFI (Direct Feedback from Interviewees).
- Lessons learned derived by the designers of the methodological approach on the basis of feedback from interviewees: they corresponded to assumptions that the designers made about the methodological approach after evaluating it; their acronym is AFF (Assumption From Feedback).
- Lessons learned derived by the designers of the methodological approach on the basis of application of the approach: they corresponded to assumptions that the designers made about the methodological approach after applying it in the lab demos, illustrations and field trials; their acronym is AFA (Assumption From Application).

For determination of those lessons learned considered relevant, and their degree of importance/relevance, other two types of lessons learned were defined on the basis of two criteria:

- Primary lesson learned (PLL): this type corresponded to those lessons learned that 1) held for half or more the subjects under analysis, and 2) the number of subjects for which it held was bigger than 2 (for individual subjects, they had to work for different organization); depending on the issue, the subjects under analysis could be organizations, specific types of stakeholders, union of specific types of stakeholders or the examples used in the lab demos.

- Secondary lesson learned (SLL): this type corresponded to those lessons learned that 1) did not fulfil the criterion of PLLs and 2) held for at least a 10% of the subjects under analysis or for two subjects (when 10% of the sample for a given part of the sample was smaller than two, such as for system analysts).

In short, the first types and criteria allowed determination of the source of a lesson learned, whereas the latter allowed determination of the degree of importance/relevance of a lesson learned. Both categories were not disjoint, but they were combined for characterization of a lesson learned. For example, a lesson learned could be a DFI-PLL if it was defined from the opinion of the majority of the subjects. A lesson learned may even be a combination of DFI, AFF and AFA (e.g., AFF/AFA-PL).

As a result of the definition of the types of lessons learned and of their associated criteria, report on lessons learned that may be very specific to a specific subject (and thus probably may not be generalizable) is avoided and feedback from some interviewees is not reported. More specifically, feedback from five interviewees (and their corresponding five organizations) is not considered when reporting on the lessons learned.

The inclusion/exclusion of the feedback from these interviewees did not have any impact on the determination of the source and relevance of the lessons learned. For example, for PLLs their criterion still held despite exclusion of the feedback, and for SLLs inclusion of the feedback did not cause determination of more lessons learned.

Nonetheless, feedback from the interviewees excluded is considered as relevant or important for development, evaluation and evolution of the methodological as the rest of facts or opinions. Indeed, some modifications in the methodological approach were made on the basis of feedback from just one subject (e.g., the project partner). For most of the cases, relevance of such feedback was confirmed by coinciding later with feedback from other subjects.

Awareness and solution of a single problem or of problems of a single organization can be very important. In fact, their study is possible and very common, for instance, in case study research (Runeson, Höst, 2009) and the value of such studies cannot be denied despite their limitations. However, for determination and report of the lessons learned, individual cases were considered to be less valuable than common cases (between subjects) and negatively affect generalization of the evaluation.

8.6 Validity

A fundamental question concerning the results and conclusions of an evaluation is how valid they are (Wohlin, et al., 2000). Validity of qualitative research highly depends on the quality of the researchers that perform it (Patton, 2001). For example, the methodological skill, sensitivity and integrity of a researcher (regarded as quality aspects) affect validity.

Researcher's quality can be considered related to researcher's expertise. It can be presumed that the validity of a qualitative study conducted by a junior researcher (e.g., a PhD student) is more threatened that the validity of a study conducted by a senior researcher (e.g., a Professor that has advised and supervised several PhD students in conduction of qualitative studies).

Even though a junior researcher strives for properly performing qualitative research, expertise is usually gained as a result of practice by participating in different studies. It is likely that many details and aspects will be better known and thus addressed after having performed or supervised several studies.

Nonetheless, review of and reliance on relevant, existing and widely-accepted literature from well-known and experienced authors (e.g., (Kitchenham, Pfleeger, 2008; Patton, 2001; Robson, 2002, Wohlin, et al., 2000)) can help a junior researcher to better address and thus perform qualitative research.

It must also be indicated that the methodological was first evaluated with the project partner and later with the other industry partners. Therefore, and as a result of the expertise gained from evaluation with the project partner, it can be considered that validity of evaluation with other industry partners is less threatened. In short, the later an interview was performed, the more valid it can be considered.

Other aspects of evaluation validity are discussed in the rest of this section on the basis of the perspectives proposed in (Wohlin, et al., 2000). In addition, some limitations associated to specific lessons learned are presented in the next section.

8.6.1 Construct Validity

Construct validity is concerned with the relation between a theory and its observation. Threats to construct validity refer to the extent to which the setting of an empirical study actually corresponds to the construct under study. In evaluation of the methodological approach of the thesis, this validity is related to the information sources used (participants and examples of the lab demos) and the characterization of the theory under analysis (business process-based requirements specification and object-oriented conceptual modelling of ISs).

Inadequate preoperational explication of construct can be considered mitigated by extensive reference to literature and actual problems in industry to characterise the theory. Mono-operation bias was addressed by interviewing people that had the different roles of interest and by using several examples in the lab demos, and mono-method bias by aiming to get feedback both from current and from past experiences.

In relation to interaction of different treatments, there exists a threat in the fact that, for instance, participants from the project partner were interviewed several times as development of the methodological approach progressed. Their previous perspectives on the approach may have influenced their opinion at a given later moment. In a similar way, interaction of testing and treatment may be affected because participants may have changed their behaviour as a result of previous knowledge about the methodological approach.

Hypothesis guessing was addressed by emphasising the need of obtaining actual opinions, information and facts from participants on the basis of their experience. Evaluation apprehension was mitigated by guaranteeing anonymity and confidentiality when publishing the results. Nonetheless, a remaining threat to evaluation apprehension is that participants may not feel completely free to express their opinions in group interviews.

Finally, experimenter expectancies are considered to having been properly addressed because both positive aspects and negative aspects of the methodological approach were discovered and thus are reported as a result of evaluation. Existence and report of only positive results may suggest that experimenter expectancies may have affected the observation.

8.6.2 Conclusion Validity

Conclusion validity is concerned with the relationship between a treatment and the conclusions drawn from it. Threats to conclusion validity refer to the ability to draw correct conclusions about relationships between the treatments and the results of an empirical study. In evaluation of the methodological approach of the thesis, this is related to the correctness of the lessons learned about the roles and organizations of the participants and the examples of the lab demos.

A way to address conclusion validity and mitigate possible threats was to confirm with interviewees those facts and opinions whose understanding (on the basis of the notes taken) was not completely clear, both during and after the interviews. For lab demos, the two designers of the methodological approach had to agree on the lessons learned.

Since obtaining statistical significance was not a goal of the survey (otherwise, probabilistic sampling methods should have been used (Kitchenham, Pfleeger, 2008)), the threats related to statistical issues were not addressed.

Fishing is considered to not have affected the survey because of the same reasons discussed for experimenter expectancies. Both negative results and positive results about the practical usefulness of the methodological approach were obtained. This means that some lessons learned indicate positive aspects, whereas others indicate negative aspects. Furthermore, evaluation was (partially) targeted at identification of possible weaknesses to mitigate them, not just at determination of positive aspects of the methodological approach.

Threats to reliability of measures and reliability of treatment implementation were that unstructured interviews were conducted, no specific instrument was developed for the survey, all the participants were not interviewed about the same, exact issues and the details presented to interviewees about the methodological approach varied according to their background (e.g., knowledge about a subject and experience on it). Therefore, it is difficult (or even impossible) to guarantee that, for instance, the outcome would be same if measurement of a phenomenon (from conduction of an interview with a participant) was performed twice.

Random irrelevancies in experimental setting were mitigated by according meetings for conducting the interviews at moments that were convenient both for the interviewer and the interviewee. In addition, the meetings were held in places that were chosen so that participants did not get distracted.

8.6.3 Internal Validity

Internal validity is concerned with the causal relationship between a treatment and its results. Threats to internal validity refer to discovery of a causal relationship in an empirical study that does not exist. In evaluation of the methodological approach of the thesis, this validity is related to the truth of the lessons learned for the roles and organizations of the participants and for the lab demos.

History was mitigated by considering just one role for those participants that may correspond to several, and maturation by limiting the duration of the interviews to two hours. Furthermore, many interviews lasted less time. Nonetheless, participation in several interviews may have affected maturation.

The risk of instrumentation is evident because of the lack of a specific instrument for the survey. Interaction with selection may have occurred in the people that participated in several interviews. Compensatory rivalry may also have happened if participants tried to unjustifiably show that their practice is better than the methodological approach. Nonetheless, emphasising the need of actual opinions, information and facts should have mitigated it.

In summary and in general, threats to internal validity are not considered very important for the survey because discovery of (statistically) significant cause-effect relationships was not aimed. It is also considered that criteria defined for identification of relevant lessons learned increased internal validity.

8.6.4 External Validity

External validity is concerned with the generalization of the conclusions of an empirical study. Threats to external validity refer to the ability to generalize the results and conclusions beyond the setting of the study. In evaluation of the methodological approach, this validity is related to how

general the lessons learned are, i.e., if they are related to general practice and could be considered that they could affect more cases that those associated to the participants in the survey.

In general, threats to external validity are not considered very important for the survey. The survey is qualitative and exploratory, thus it does not aim to generalize conclusions beyond its actual setting but to explain and understand the phenomena under analysis. Nonetheless, understanding the phenomena under study may help in understanding other cases. In addition, many results and conclusions coincide with other works, as presented and discussed in the next section, and the subjects' background was heterogeneous.

Use of realistic environments (practitioners that corresponded to supplier stakeholders and people that corresponded to customer stakeholder) and of realistic examples (in the lab demos) is considered very positive for external validity despite other threats existed.

Interaction of selection and treatment was mitigated by interviewing people that played those roles considered to correspond to the most relevant stakeholders for IS development (stakeholders taxonomy). Interaction of setting and treatment was addressed by searching subjects that had participated or were participating in development projects.

A remaining threat to external validity is that many details about the participants and their interviews cannot be presented for confidentiality reasons. This also affects the value of the evaluation (Ivarsson, Gorschek, 2011).

8.7 Lessons Learned

Presentation of lessons learned is a common way to report on results and interpretation of qualitative research (Ivarsson, Gorschek, 2011). Furthermore, lessons learned can be very compelling and valuable for practitioners (if they correspond to industrial issues) because they provide evidence of actual situations that they face or may face.

It must be indicated that lessons learned may even be regarded as a research method itself, as case study or survey research. Nonetheless, for evaluation of the methodological approach, lessons learned are just considered to correspond to the results of the evaluation and their interpretation.

For this thesis, several lessons have been learned after evaluating the methodological approach on the basis of the feedback obtained from participants in evaluation and from the observations made by the designers of the approach. Some of them have driven definition of the mechanisms and guidance of the approach and have been implicitly addressed in previous chapters (e.g., problems related to specification of information flows as a unique flow were mentioned in Chapter 6). Indeed, many decisions and choices on the methodological approach have been based on lessons learned.

This section presents the lessons learned distinct from issues already discussed in the thesis. It is considered that those issues do not need to be discussed (again) in this chapter because the lessons learned presented are related to concerns that affect and are related to application of the methodological as currently defined.

Application of and discussion about previous versions of the methodological approach is considered unnecessary because the corresponding issues do not affect the approach anymore. For example, it is considered unnecessary to discuss about the fact that BPDs can get tangled if domain entities are (always) modelled with data objects.

Nonetheless, issues mentioned in previous chapters are presented in lessons learned when it is considered that they have not been extensively enough discussed or not enough evidence has been provided. For example, the need of detailed guidance for application of a RE approach has been indicated throughout the thesis, but details about concrete, actual cases are shown in this section.

Lessons learned are presented from more general ones (they can be considered to affect the RE process of IS development projects in general, i.e, they are related to RE practice in general) to more concrete ones (they mainly affect and are related to the methodological approach of the thesis). How the lessons learned have been and are addressed in the methodological approach is also discussed.

Although it may be argued that some of the lessons learned (especially for those for RE practice in general) correspond to issues that have been widely acknowledged in literature and thus are not very novel, it is considered relevant to present and discuss them. They are concerns that still affects RE practice, and thus research. Therefore, they are important and should be reported.

The main lessons learned are presented in the next subsections, firstly the general ones (10 lessons learned) and secondly the specific ones (7 lessons learned). For each lesson learned, its type (source and relevance) is specified. It must be noted that the names (and thus the explanation) of some lessons learned can enclose lessons learned of several types. In addition, some lessons learned (not considered main ones) are presented within explanation of others.

Table 8.2 shows the correspondence between the lessons learned and the research questions of evaluation for which the lessons learned provided answers. As shown in the table, it was considered that a lesson learned could provide answers for more than one research question. For example, a lesson learned may be related to a positive aspect of the methodological approach identified from (some) interviewees' perspectives (RQ2), but some details of the approach may not be completely satisfactory for other interviewees (RQ3).

Table 8.2 Research questions of evaluation addressed in each lesson learned

Lesson Learned	Research Question		
	RQ1	RQ2	RQ3
LL1		X	
LL2		X	
LL3		X	X
LL4			X
LL5		X	
LL6		X	
LL7		X	
LL8		X	
LL9		X	X
LL10		X	X
LL11	X	X	
LL12		X	
LL13		X	
LL14	X		
LL15	X	X	X
LL16	X	X	X
LL17		X	

When reviewing Table 8.2, it can be observed that there exist more lessons learned related to positive aspects of the methodological approach (RQ2) than to negative aspects (RQ1 and RQ3). This is a consequence of having already addressed most of the negative aspects of the methodological approach discovered during its definition. When it was considered that a weakness could be mitigated, means to achieve it were studied and introduced.

8.7.1 Lessons Learned Related to RE Practice in General

LL1) Understanding and knowledge of the application domain is a key for success of software development projects and of ISs (AFF-PLL)

The first part of the interviews with supplier stakeholders was aimed to know and understand the mission of their organizations, i.e., what types of projects and systems developed and at what applications domain they were targeted. It also involved discussion about the success of the organizations, of their projects and of their products.

In relation to and on the basis of these aspects, feedback about the importance of understanding of and knowledge about the application domain was obtained. These aspects were considered essential when defining the methodological approach, but gaining insights into to what extent they were important in practice was aimed.

The feedback obtained was in line with the initial believes of the interviewers. Many cases and much evidence were found about the fact that a project or system can be deemed to failure if supplier stakeholders do not know the application domain. In the same way, likelihood of success is higher when the application domain is familiar.

As an example, a software development company was found that not only was specialised in software system for a particular application domain (finance), but also for specific needs of the domain. In addition, that part of the domain was not well-known for most of industry.

The company had found a market niche in which it stood out over competitors and had obtained projects when competing with major IT companies. Even the company had found cases in which potential clients had not believed in the possibility of obtaining the products that it offered. Past projects had failed in the potential clients because of the

difficulty of understanding and the complexity of the application domain and thus of developing an adequate product for it.

Understanding of the application domain is one of the principles of the thesis (Chapter 3). Works that point its importance when performing the RE process (and developing an IS) can be found easily (e.g., (Jackson, 1995)). Some works are based on empirical studies or practical experience (e.g., (Berntsson-Sevensson, Aurum, 2006; Firesmith, 2007; Gulla, 2004; Lauesen, Vium, 2005; Sadraei, et al., 2007)).

LL2) IS support to business processes is essential (DFI-PLL)

An issue on which consensus was reached with all the interviewees was the fact that ISs must support the business processes of an organization. Even though business processes are not always modelled during the RE process of and IS, the activity of an organization is always considered when an IS is going to be developed.

In the customer stakeholder side, interviewees indicated the dissatisfaction that had existed in their organizations when a new IS had been introduced but it had not allowed end-users to perform their work as expected or wished. For example, cases in which ERP systems had not been adequately customized for their use in organizations resulted in problems when using them. Instead of facilitating and improving work, they negatively affected end-users' performance. End-users even stopped using the "new" ERP and resumed using the previous IS until the ERP was modified, despite the limitations of the IS.

In the supplier stakeholder side, interviewees indicated how important was that an IS supported end-users' (organizational) activity so that a project was successful, and some examples of problems were obtained. What was partially surprising was that, even though supplier stakeholder acknowledged problems related to inadequate knowledge about and thus to support for business processes, most of the organizations neither used nor planned to use business process modelling as part of their RE process. Nonetheless, some interviewees stated that they had to study and try its adoption.

The problem of inadequate support to business processes in the supplier stakeholder side was also clearly related to the need of knowledge and understanding of an application domain. For example, a case was found in which an IS had been developed for a factory on the

basis of the activity in a different factory of the same company. It had been assumed that all the factories worked in the same way, but it was not so The application domain was not known enough, and consequently the business processes of the first factory were not properly supported.

Support to business processes is one of the principles of the thesis, and its importance is widely acknowledged in literature (e.g., (Becker, Kugeler, Rosemann, 2003; Dumas, van der Aalst, ter Hofstede, 2005)). Some works that report on empirical studies or practical experience and have also indicated the importance of this issue are (Cardoso, Almeida, Guizzardi, 2009; Gulla, 2004; Indulska, et al., 2009; Lauesen, Vium, 2005; Sadraei, et al., 2007).

LL3) Awareness of IS goals is important (DFI-PLL), but systematic modelling and analysis is not a wide-spread practice (AFF-PLL)

Another issue for success of an IS and its development project that was widely acknowledged among supplier stakeholders was the need of knowing the (ultimate) goal of an IS.

In most of the projects it is not enough to know that a customer wants an ERP system, but also it is necessary to know why the system is necessary to specify the (actual) requirements of the system and provide an adequate solution. For example, an ERP system in a factory may be necessary so that employees follow the quality policies of the company and the managers of the company can know in real time how such policies are being addressed.

Despite the acknowledged 1) importance of awareness of the purpose of a system, 2) importance of analysis and determination of its effect on the business processes of a customer (DFI-SLL), 3) interest that interviewees showed in the purpose analysis stage (specially in the patterns for business process reengineering; DFI-SLL), no supplier stakeholder neither software development company was found that used some approach for modelling and analysis of the purpose of an IS, i.e., use of some goal-oriented RE approach was not found.

The importance of knowing and addressing the goals of a software system for project success has been acknowledged in the principles of the thesis and in works such as (Borg, et al., 2003; Charette, 2005; Gulla, 2004; Lausen, Vium, 2005; Liu, et al., 2009). In relation to the narrow adoption of goal-oriented RE approaches in industry, it is acknowledged in

(Matulevicius, Heymans, 2007). Recent surveys about techniques and approaches for requirements elicitation and specification also indicates that either goal-oriented RE approaches are not used (e.g., (Liu, et al., 2009; Neill, Laplante, 2003)) or that they are used by very few practitioners (e.g., (Rouibah, Al-Rafee, 2009)).

A limitation of this lesson learned is that the interviewees worked for companies that focused on provision of IT solutions and not on business consulting. It is considered that business consulting companies may be more concerned about reengineering the business process of an organization from the analysis of its business goals, for instance, on the basis of strategy maps of the Balanced Scorecard (Kaplan, Norton, 1996). In other words, it is more likely that business consulting practitioners analyse business goals and how they are related to business processes.

LL4) Practitioners may be reluctant to use and combine several modelling techniques (AFF-PLL)

One of the main characteristics of the methodological approach is that it combines several modelling techniques (and approaches) in order to address (part of) the RE process of an IS. Although it was decided so because of the belief in its need, the actual practice is that such a combination of techniques is not liked by practitioners in general.

Combination has been found in some cases (DFI-SLL; e.g., use of different UML diagrams), but most of the system analysts interviewed do not like having to use more than one technique. They prefer to just use one even though it may present limitations and problems may arise later.

Use of several techniques is usually based on company recommendations and policies, not on the belief of practitioners in their suitability (DFI-SLL). Furthermore, system analysts rely more on their expertise than on the possible advantages of using a technique (AFF-PLL). For example, practitioners that use OO-Method usually do not create a dynamic model, but "adapt" the object model so that it includes the details that are supposed to be in the dynamic model.

In relation to the methodological approach, the combination of BPMN and the Map approach first and the later derivation of OO diagrams was not "very" welcome in general among system analysts (AFF-PLL). This issue is further explained, analysed and discussed in LL10.

Some authors have provided evidence of the reluctance and dislike of practitioners to use and combine several modelling techniques (e.g., (Becker, et al., 2010) for business process modelling). Nonetheless, other works have acknowledged the need of combining and using different techniques in the RE process depending on the problem to solve (e.g., (Dieste, Juristo, Shull, 2008)). It has also been acknowledged that system analysts relay mainly on heuristics acquired through experience when working or deciding how to work (Cox, Phalp, 2007), and programmers may have problems when using several RE techniques/models and may not like having to use new models (Lauesen, Vinter, 2001).

Other authors have recognised that problems when using and interpreting different models of a same system can arise (Kim, Hahn, Hahn, 2000). A way to mitigate such problems is to determine clear references between the models. In this sense, traceability and relationships among the models of the methodological approach exist. This issue is clearer and explicitly shown in Appendix A (conceptual framework) of the thesis.

Furthermore, and as acknowledged by the same authors of the previous work, the use of several models is usually the norm, not the exception, in IS modelling in real projects. This has been further confirmed by studies that have surveyed the use and combination of diagrams and techniques for IS modelling in industry (e.g., (Dobing, Parsons, 2006)).

It is considered that this lesson learned is limited in the same way as LL3. It is believed that use and combination of different modelling techniques is more usual in business consulting than in IT consulting. For example, ARIS (which, as mentioned in Chapter 2, is widely used in industry in several European countries) can be considered to be more focused on business issues and requires the combination of several modelling techniques, if followed as defined.

Another limitation is related to the size of the systems that the interviewees developed. It is widely recognised that the more complex a system is, the more models to analyse it and design it may be necessary to better understand it and find solutions (e.g., (Kim, Hahn, Hahn, 2000)). In addition, system analysts' feedback may have been influenced by the fact that most of them had previously worked as programmers.

LL5) Homogeneous granularity of system requirements can be important (AFF/AFA-SLL)

During the definition of the methodological approach, the need of homogeneous granularity (abstraction level) of ETDs so that their specification can be considered consistent and significant (Chapter 6) was determined. As a result, the significant criterion for ETDs was defined.

When discussing homogeneous granularity with supplier stakeholders, some interesting insights were gained. On the one hand, it was not regarded as a very important concern for most of the supplier stakeholders (DFI-PLL) and use of explicit criteria to guarantee it was uncommon (DPI-PLL). Granularity of system requirements (e.g., use cases) usually depended on system analysts' experience and own criteria and was not wide-spread knowledge/practice in companies.

Nonetheless, explicit criteria were found in some organizations (AFF-SLL). For example, a company defined the granularity of a use case for activation of a service provided to its clients as the sequence of steps necessary since a client requests the activation (via SMS or the Internet) until he receives the notification of activation (via SMS). For each use case, scenarios were specified and the necessary workflow was designed.

Furthermore, cases in which non-homogeneous granularity was a source of problems were found (AFF-SLL). For example, they arose when programmers received a SyRS from system analysts in the form of use cases whose granularity was considerably heterogeneous. Some of them represented very concrete interactions whereas others represented practically business goals or business activity at a high abstraction level.

As a consequence, programmers had to "re-analysed" the use cases so that they were relevant for implementation (implementation of a given use case in isolation was nonsense) or had to consult system analysts about the purpose of some use cases to order their implementation. These problems involved delay in implementation of use cases and extra work for programmers.

The importance of homogeneous granularity of system requirements has been acknowledged, for instance, for improving the quality of a SyRS (España, et al., 2009), comparing and prioritising requirements (Gorschek, Wohlin, 2006) and for properly applying a RE approach (Dutoit, Paech, 2002).

LL6) Imprecise specification of system requirements can have negative consequences (DFI/AFA-SLL)

Obtaining a precise SyRS is considered so important that it is explicitly mentioned in the objectives of the thesis (Chapter 1). However, it was not considered very important by some supplier stakeholders (AFF-SLL), who considered that modelling of a use case diagram and provision of some unstructured information was enough.

However, negative consequences of imprecise SyRSs were found in some cases. An example (with no very negative consequences) was that programmers had to ask system analysts about details of, for instance, a SyRS in the form of use cases. It entailed loss of time for both programmers and system analysts, but not much time.

A much more serious situation was found in the company (mentioned in LL5) that used service request-notification for homogeneous granularity of use case. This company usually outsourced software development, and a system analyst assumed that the necessary software for a new service would be developed by the same company that had previously developed software for another service. As a result, he omitted some details about the scenarios of the corresponding (new) use case because they were similar to the scenarios of the previous service.

What the system analyst did not know was that the outsourced company also outsourced development sometimes, as for the new service. The final consequence of detail omission was that the company of the system analyst had provided clients with a service that was not billed, what turned to be regarded as a loss of money in the company.

This lesson learned is also related to LL1. The more knowledge about an application domain system analysts had, the less precise SyRSs usually were (AFF-SLL). Nonetheless, system analysts were told about the important risk in assuming a wide or complete knowledge about an application domain, which may lead a system or project to failure (Berenbach, et al., 2009). They agreed upon this issue (DFI/AFA-SLL).

The importance of creating a clear, precise SyRS and problems that may arise if not fulfilled (e.g., project failure) have been indicated, for instance, in (Charette, 2005; Firesmith, 2007; Hofman, Lehner, 2001; Kamsties, Hörmann, Schlich, 1998; Lauesen, Vium, 2005; Procaccino, Verner, Lorenzet, 2006).

LL7) The notion of non-functional requirement is ambiguous (AFF-PLL)

"What do you mean by non-functional requirement?" This was the most frequent answer that stakeholders provided when they were asked about aspects of management of non-functional requirements and when discussing about them.

Even in cases in which this answer was not obtained, or when it had been explained that non-functional requirements referred to characteristics related to aspects such as usability, performance and interoperability of an IS, most of the stakeholders did not make a clear distinction between functional and non-functional requirements or did not interpret a non-functional requirement as such (AFF-PLL). For example, some supplier stakeholders regarded some security concerns (e.g., authentication) as related to functional requirements.

Therefore, it was assumed that the notion of non-functional requirements was ambiguous, at least for the interviewees. As a solution, the term quality requirement was adopted and the ISO 9126-1 standard was chosen as a reference to specify quality requirements of ETDs.

It is considered that these decisions improved communication with stakeholders and understanding for them because less confusion arose in later interviews (AFA-PLL). When talking or discussing about a given system requirement, it was easier for stakeholders to identify it as a security aspect (quality attribute and requirement) than as a non-functional requirement.

The ambiguity and the inconvenience of the term non-functional requirements has been discussed and justified in literature (e.g., (Glinz, 2007; Pohl, 2010)). With regard to empirical studies, misinterpretation of non-functional requirements, communication problems among different organizations related to their notion and language problems related to them in documents have been reported as important risks in software development in (Borg, et al., 2003).

Finally, it is considered (in this thesis) that use of the term quality requirement has increased in recent years (e.g., (Bersntsson, Gorschek, Regnell, 2009; Lauesen, 2002)). Nonetheless, the term non-functional requirement is still used (e.g., (Ameller, Franch, Cabot, 2010; Chung, Leite, 2009)).

LL8) Detailed and straightforward methodological guidance can be important for a RE approach (DFI/AFA-SLL)

Provision of detailed methodological guidance for application of a RE approach is one of the principles of the thesis. It is considered that it can facilitate adoption because practitioners would find fewer problems when using an approach. For example, they would exactly know from where and how system requirements should be elicited.

Although the interviewees indicated that guidance is important (DFI-PLL), they did not consider that detailed guidance is essential (DFI-PLL). For example, system analysts and programmers attended several courses and talks about agile development and studied different techniques and books for adoption of an agile software process in a company.

The company adopted user stories because they were believed to be very useful (and indeed they were), but they were used in a very lightweight and non-systematic way if compared to reference works on how to apply them (e.g.,(Cohn, 2004)). In short, the company considered necessary to follow general guidelines, but it did not considered important to follow most of the specific, detailed aspects.

In contrast to this situation, cases in which adoption and application of a RE approach has ended in abandonment because of insufficient or inadequate guidance were found too (DFI-SLL). Supplier stakeholders indicated that not only (detailed) guidance was necessary, but that it was also necessary that the guidance was straightforward for those who had to apply a RE approach (DFI-SLL). For example, application of criteria for identification of use cases on the basis of their goals had resulted in confusion among system analysts.

The need of providing detailed guidance in a RE approach probably is not widely acknowledged in literature. Nonetheless, there exist works that have recognised the advantages of detailed guidance in a RE approach (e.g., higher quality in SyRS (España, et al., 2009)) and the problems that may arise, such as, different results if applied by different people (Dutoit, Paech, 2002) or ambiguity in a SyRS (Hsia, Davis, Kung, 1993). Even there are authors who have recognised detailed guidance as important and necessary to apply a RE approach on the basis of feedback from practitioners (e.g., (Karahanna, Straub, Chervany, 1999; Regnell, Berntsson-Svensson, Olsson, 2008)). Such guidance must be targeted at ease of use of an approach.

LL9) Creation of OO diagrams that meet system requirements is important (DFI-PLL) but it is not a main concern in practice (DFI-PLL)

An objective of the thesis (and maybe the ultimate one, given that it arose from the need of OO-Method of having a business process-based RE approach) is to derive the OO conceptual schema of an IS from its system requirements in the form of a class diagram and state transition diagrams.

Such diagrams must be consistent with and complete in relation to the system requirements of an IS. In essence, it is considered in this thesis that the OO diagrams of an IS must properly meet its system requirements. This line of thought coincides with industrial perspectives. Most of the interviewees (including system analysts) regarded this characteristic as a very positive aspect of the methodological approach and as important for practice.

However, the survey also indicated that, although stakeholders considered it important, most of system analysts were not much concerned about obtaining OO diagrams that met system requirements. They simply cared about obtaining OO diagrams. Determination of whether a diagram met system requirements or not was later analysed and determined.

Furthermore, most of the system analysts did not considered that creation of the OO diagrams of an IS and that fulfilment of system requirements were main (i.e., highly important) activities in IS development (DFI-PLL). In relation to the methodological approach, they considered, for instance, that requirements elicitation and interaction with customers are much more important concerns (DFI-SLL).

Nonetheless, problems in creation of OO conceptual schemas and in their creation to meet system requirements were found (DFI-SLL), but on a minor scale. For example, a system analysts had been requested to create the data model (in the form of an ER diagram, which can be considered equivalent in many aspects to a class diagram) for a new IS.

This IS was not modelled and developed from scratch, but it was the extension and improvement of an existing system. The data model had to reflect the data requirements of the "old" system, and also incorporate the new data needs of the new system.

Initially, it did not seem a difficult duty. The problems arose when the system analyst was told that no documentation about the requirements of the old system existed. Therefore, he just had the actual old system (e.g., its forms) and the corresponding database to create the data model.

Creation of the data model was tough, but not the toughest part. Although not easy, the system analyst finally managed to obtain it. However, he was not able to guarantee that the model met data requirements, thus directly using it as a reference for the new system would have been risky. Such a risk is clearer if it is indicated that parts of the new (and thus of the old) IS corresponded to a safety-critical system.

At the end, and as result of the lack of a SyRS and thus of being unable to guarantee the correctness of the model, the system analysts had to request help from some colleagues that better knew the system. This turned into a delay in development of the project, which it was later cancelled because of breach of contract for several milestones.

It is not said that the project was cancelled because of the problems related to creation of the data model, and it may be considered that the ultimate "guilty" was the person that requested the system analysts to create a data model or that the project was cancelled because of other problems. Indeed, projects usually fail because of several types of problems (Charette, 2005). What this example showed was that fulfilment of system requirements in OO diagrams (and guaranteeing it) could be a problem, especially if the RE process was not carefully performed. Such a (supposedly) unimportant problem turned into a serious situation.

There is a contradictory point in this lesson learned (as well as in others). On the one hand, system analysts considered that obtaining adequate OO diagrams is important. However, and on the hand, system analysts were not much worried about it. This contradiction may be regarded as a threat to internal validity of the lesson. It is considered that somehow industry perspectives on the issue are different from industry practice.

When reviewing literature, the need of OO conceptual schemas of meeting system requirements is acknowledged because, for instance, it is related to its correctness (Olivé, 2007). Furthermore, works that report on problems when creating OO diagrams that meet system requirements if this activity is not properly performed exist (e.g., (Fortuna, Werner, Borges, 2008; Svetinovic, Berry, Godfrey, 2005)). The need of detailed and

more guidance to properly model OO conceptual schemas from SyRSs is recognised in (Cox, Phalp, 2007), which also indicates that, in general, industrial acceptance of guidelines is still limited.

A possible limitation of this lesson is that the number of senior system analysts (4 or more years of experience) interviewed was higher than the number of junior ones. The lessons learned is also related to the next.

LL10) The perceived usefulness of a RE approach varies among stakeholders and projects (AFF-PLL)

This lesson learned can be considered to be the main and most important one of the thesis. Although it may seem obvious, it was surprising how perception about the value of the methodological approach greatly varied among stakeholders. Furthermore, somehow this lessons learned is related to and summarises most of (or even all) the others.

When asking customer stakeholders about the usefulness of the methodological approach, the general perception was that it could be effective for the RE process of an IS (DFI-PLL). They considered that its use could lead to ISs that adequately fitted their needs, and several positive points lessons learned (presented below) were derived and assumed from customer stakeholders' feedback.

System analysts stated that use of the methodological approach would allow them to adequately understand and specify system requirements and to create an OO conceptual schema (AFF-PLL). However, there were some senior system analysts who claimed that they may just use it eventually because they did not think that the approach could improve their job significantly in most of their projects (AFF-SLL).

These system analysts corresponded to two types. The first type corresponded to systems analysts that were very skilled in using other approaches and interacting with customers. For example, senior system analysts of the project partner were experts at using OO-Method. They usually modelled a system while the customer described what the system should do, thus they could quickly generate it, validate it and fix it (if necessary).

The second type corresponded to system analysts that had worked for many years and thus were specialised in a given application domain. As a result, they did not need to gather and model much information (i.e., activity of an organization) of the business (application) domain because

they were already aware of most of the behaviour and possible needs of an organization.

In contrast to the opinion of senior system analysts, most of the junior system analysts considered that the methodological approach could really help them (AFF-PLL). They had less experience in dealing with customers and, therefore, in understanding what they need, and also had less knowledge about many applications domains.

Even though supplier managers agreed upon the principles of the methodological approach and acknowledged that (parts of) it could be useful for and improve practice in their companies (AFF-PLL), their main concerns about application of the methodological were the necessary effort and time to learn to apply the approach and to apply it in a project (AFF-PLL). Many of them were not sure that the expected improvement was worth the necessary investment (AFF-SLL).

Last but not least, programmers claimed that the degree of detail and precision of ETDs was good enough for implementation (DFI-PLL) and that (semi-) automatic derivation of an OO conceptual could be very useful (DFI-PLL). Other positive aspects for programmers have been reported above.

In summary, although it is considered that all supplier stakeholders can benefit from the methodological approach, it is also considered that its use in a project will mainly depend on system analysts' experience and knowledge about the application domain and on system complexity (AFF-PLL).

For IS development for application domains with which a system analyst has not previously deal or with junior system analysts, it is considered that the approach could really help them to better address elicitation and specification of system requirements and subsequently obtain an OO conceptual schema that meets system requirements (AFF-PLL). Nevertheless, there can also be projects in which application of the methodological approach (or of some of its parts) may not be necessary (AFF-PLL). The problem is how to exactly determine whether a RE approach should be used or not in a given project.

It is also assumed (AFF-PLL) that a gap exist in the perceived useful between those who would apply or would control application of the methodological approach (system analysts and supplier stakeholders)

and those who would received the outcome of the approach in the form of a SyRS and an OO conceptual schema (programmers) or, once development is finished, an IS (customer stakeholders).

The first ones are more concerned about performance issues (related to how much the approach may "hinder" an activity or make it slower), whereas the latter are more concern about other quality issues related to how much the approach would facilitate their activity (AFF-PLL).

Some works that have reported on differences among opinion on the usefulness of a RE approach are (Cox, Phalp, 2007; Davis, Hickey, 2002; Kaindl, et al., 2002; Mead, 2000). Other authors have indicated that industry is more concerned about RE and thus more clearly perceive its usefulness in large projects (Juristo, Moreno, Silva, 2002) and that models for IS development in general (Davies, et al., 2007; Karahanna, Straub, Chervany, 1999) and for the RE process in particular (Kamsties, Hörmann, Schlich, 1998) are only used when they are believed to be useful.

Compatibility with current practices may also influence adoption and perceived usefulness of a RE approach (Davies, et al., 2007; Karahanna, Straub, Chervany, 1999), as well as organizational culture (Cox, Phalp, 2007). It has been widely acknowledged the difficulty of transferring RE research to practice (e.g., (Kaindl, et al., 2002; Lauesen, Vinter, 2001; Mead, 2000). For example, practitioners may not want to use new RE approaches even though they may improve their work because of the necessary effort.

Finally, it is recognised that a good RE approach either reduces the cost of a development project or increases the quality of the resulting product when used in specific situations (Davis, Zowghi, 2006). In this sense, the methodological approach fulfils the second conditions (AFF-PLL). The same authors also indicated that a RE approach should not be discarded because failure in a project, that a good RE approach does not guarantee project success and that use of a "non-good" RE approach does not imply project failure.

The main and final point of this lesson learned has been indicated in literature: quality is in the eye of the beholder (Davis, 1995). The perceived quality of a RE approach (if not empirically demonstrated, or even in that case) can be considered to be closely related to the expected benefits and problems from its application.

8.7.2 Specific Lessons Learned Related to the Methodological Approach

LL11) There exists a gap between BPMN and system requirements (AFF/AFA-PLL)

One of the problems of the RE process and of IS development that is addressed in the methodological approach is the gap between business and systems domains. The solution to this problem is essential for the approach, and its importance is acknowledged in its second principle (Chapter 3): business processes must be properly analysed so that the gap is bridged when specifying system requirements from them.

BPMN is used in the methodological approach from a business perspective (it can also be used from a system perspective), thus BPMN (BPDs) is used as model for the business domain and ETDs are used as model for the system domain.

A gap exists between BPDs and ETDs because of the differences in characteristics such as terminology, semantics, abstraction levels and granularity. The gap is bridged by providing mechanisms and guidance for proper analysis of BPDs (labels and consecutive flows) and for correct elicitation and specification of ETDs (significance criterion and guidelines for ETD specification).

This gap is clear if the guidelines for filling of the textual template are analysed. For example, BPMN triggers do not always map to ETD triggers and granularity of ETDs and BPMN tasks can be different. In addition, the textual template contains information that is not part of the business environment, i.e., it corresponds to specific characteristics of the system that determine support for business requirements.

LL12) Graphical extension of BPMN can facilitate understanding (AFF-PLL)

When it was decided to adopt BPMN graphically, probably the main reason was that it was expected to facilitate communication between system analysts and customer stakeholder. As explained in the next lesson learned, this condition was shown in the evaluation, but it was also discovered that graphical extension of BPMN could further improve the notation.

System analysts and supplier managers did not initially like that BPMN had been graphically extended (DFI-PLL) with labels for flow objects and with consecutive flows. They wondered if more graphical elements were really necessary given that BPMN was quite extensive.

Nonetheless, it was argued that labelling and modelling of consecutive flows were useful to bridge the gap between business and system domains. As a result, system analysts and supplier managers finally agreed upon their use (DFI-PLL). Furthermore, after using these elements, they acknowledged that these graphical elements helped them to better understand BPDs (DFI-PLL), and thus BPMN expressiveness was improved (AFF-PLL).

Customer stakeholders and programmers also liked the graphical extensions (DFI-PLL). Understanding of the proposed solution (system support) for a business process was easy, and better than if new graphical elements had not been used (AFF-PLL).

The convenience and usefulness of extending BPMN graphically is evident if two facts are considered. First, BPMN specification has always provided the possibility of extending the notation if considered necessary by its users. Second, and more important, the last version of BPMN (version 2.0) has defined labels for the tasks so that their type is graphically shown and their semantics is more easily understood.

Existing works related to this lesson learned are those that have found benefits from graphical extension of BPDs. They have studied graphical extension of BPMN to improve its expressiveness and facilitate understanding of BPDs in actual projects (e.g., (zur Muehlen, Ho., 2008)) and graphical extension of business process models by labelling their elements (e.g., (Mendling, Reijers, Recker, 2010)).

LL13) The RE models of the methodological approach are easy to understand for stakeholders and can facilitate customer stakeholders' involvement (AFF-PLL)

Since its initial conception, one of the needs that the methodological approach had to fulfil was that it had to facilitate communication with customer stakeholders and their involvement in its application. As a result, and as justified in previous chapters of the thesis, when a RE

technique or approach (i.e., a model)[1] was selected, one of the criteria was how useful it would be for fulfilment of the above need. Although theoretically all the RE models were suitable, this had to be shown in practice. Furthermore, some of them had been modified.

When evaluating the methodological approach with customer stakeholders, they stated that they could understand and validate the artefacts of the RE models (e.g., BPDs, goals/strategies diagrams and ETDs) easily (DFI-PLL), more easily than other models such as class diagrams or dataflow diagrams (DFI-SLL). The only negative aspect found was that ETDs sometimes became very large (DFI/AFA-PLL), what may hinder their understanding and validation. Nonetheless, this was a consequence of the inherent complexity and the amount of needs of the subtasks of the ETDs.

In addition, customer stakeholders of the field trials get used to BPMN even quicker than expected. In very little time, they were able to completely understand BPDs and they even fixed modelling errors (AFA-PLL). It was considered that communication and interaction with customer stakeholders was facilitated (AFA-PLL), and they also claimed that had felt more involved in system development and had had a more participative attitude than in past projects (DFI-PLL).

With regard to supplier stakeholders, they were asked to interpret BPDs of organizations that they did not know in depth. The result was that they understood them easily most of times (DFI/AFF-PLL), and they just had problems when documentation of the organizational activity (business rules, data entities…) was insufficient (AFF-PLL).

A limitation of this lesson learned is that all the elements of BPMN (the complete notation) were not used in any field trial or illustration.

With regard to related work, (expected) benefits in communication between customer stakeholders and system analysts when using the RE models of the methodological approach have been indicated in previous chapters of the thesis. In addition, some works have reported on empirical studies or practical experiences that have provided evidence of the suitability of BPMN to interact with customer stakeholders (e.g., (zur Muehlen, Ho, 2008)).

[1] OO diagrams are not considered to be part of the RE models of the methodological approach, but to be part of a system model (e.g., an OO-Method conceptual model)

LL14) Filling of the textual template of an ETD is variable (AFA-PLL)

Domain requirements of an ETD depends on (are elicited from) BPDs, and their correspondence is determined by the guidelines for filling of the textual template. However, BPD modelling is variable, and specification of domain requirements also is.

For example, a business rule might be specified in a BPD: 1) as a gateway, or; 2) textually in its documentation. Therefore, the business rule could correspond: 1) to a trigger, precondition, postcondition or the source of an alternative or of an extension, or; 2) to a business rule of an ETD.

It was not considered that variability of BPD modelling and, therefore, of specification of domain requirements was a weakness or problem of the methodological approach. What was considered important was that all the organizational concerns that would affect and would be controlled by an IS (e.g., the business rules of an organization) and corresponded to domain requirements were specified in the ETDs, regardless of the section of the textual template in which it was done.

In summary, some domain requirements could be specified in different styles (sections of the textual template) (AFA-PLL).

LL15) The OO diagrams derived can be regarded as complete from a requirements perspective (AFA-PLL)

Completeness of the OO conceptual schema of an IS created by applying the methodological was evaluated by comparing it with existing OO diagrams of the IS under study. This means that the actual OO conceptual schema of some ISs (e.g., an OO-Method conceptual schema) or a schema that corresponded to possible solutions (proposed by people different to the designers of the methodological approach) were available.

The main result was that, in general, the OO diagrams derived by following the methodological approach were smaller and differ in some characteristics with the other diagrams (AFA-PLL). The reason was that internal (design) characteristics of an IS had been modelled in the available diagrams, whereas the diagrams derived by applying the methodological approach only contained characteristics related to (derived from) the corresponding SyRS.

For example, and as explained in Chapter 7, it is usual that system analyst that use OO-Method in industry make some design decisions when creating a conceptual schema and (given that the OO-Method allows it) include them, for instance, in the object model. The example shown in Chapter 7 was modelling of an association between two classes as an inheritance relationship.

Nonetheless, all the external characteristics that had been modelled in the available OO diagrams coincided with the details of the OO conceptual schemas derived by applying the methodological approach (AFA-PLL).

It must also be indicated that the previous statement is partially "false". Some external characteristics of the available diagrams were missing, but because incompleteness (in relation to the OO conceptual schema that should have been derived) existed in artefacts created in stages of the methodological approach previous to derivation of OO diagrams. For example, if an attribute of a domain entity had not been specified in an input or autonomous flow of some ETDs (but it should have been), then it was impossible that the attribute was present in the corresponding class of the class diagrams.

In summary, it was determined that the OO conceptual schemas derived by applying the methodological approach could be considered complete from a requirements perspective, provided that the SyRS from which the schema was derived was complete and correct. It must also be noted that all the characteristics of an OO-Method conceptual schema whose derivation has not been addressed (i.e., studied in depth) in this thesis (e.g., the presentation model) were out of the scope of evaluation of the completeness of the OO conceptual schemas derived by applying the methodological approach.

LL16) The OO diagrams derived represent a refinement of artefacts of previous stages (DFI/AFA-PLL)

An aspect of the methodological about which discussion with interviewees arose and of which designers of the methodological approach realised while applying it was the fact that there exist similarities and relationships between the OO conceptual schema created in the derivation of OO diagrams stage and artefacts created in previous stages of the methodological approach.

It was evident that a class diagram and a domain data model were very similar (DFI/AFA-PLL). The classes and associations of the class diagrams were a subset of the domain entities and relationships of the domain data model of the organizations under study. Nonetheless, this fact was not considered to be a weakness or problem of the methodological, but to be a reflection of common practice in IS development.

The pieces of information (data) that are stored and managed in an IS correspond to a part of the application domain that will be represented, stored and controlled by the system (Olivé, 2007). In the methodological approach, a domain data model is a part of the application domain that can be considered to be later refined to model a class diagram that depicts the part of the domain data that will be controlled by an IS. The methodological approach aims to provide system analysts with mechanisms and guidance to model a class diagram of an IS.

Furthermore, the purposes of a domain data model and of a class diagram are different in the methodological approach. The purpose of a domain data model is to understand the application domain of an IS, whereas the purpose of a class diagram is to model the static properties of the OO conceptual schema of the IS and that it meets its system requirements. Furthermore, it includes system details, such as methods.

With regard to state transition diagrams, they were related to ETDs (DFI/AFA-PLL). Apart from other information, ETDs specified the state of its input and output domain entities, which implicitly specified the sequence (execution order) of the ETDs. These characteristics were clearly related and were similar to specification of states in a state transition diagram and to their sequence.

ETDs could be regarded as a representation of the states and transitions of an IS (AFF/AFA-PLL), i.e., the sequence of ETDs that the IS would have to follow to support the business processes of an organization. ETD execution implied state changes of domain entities, and thus in (the instances of) the state transition diagrams of the classes that had been modelled from them.

Therefore, ETDs could also be regarded as transactions in an IS in which events (transitions) of different state transition diagrams were executed. ETD sequence could be regarded as a macro-state transition diagram, i.e., a state transition diagram that affected the states of the state

transition diagrams of several classes (AFA-PLL). Therefore, the state transition diagrams represented a refinement of ETDs.

In summary, the relation between ETDs and state transition diagrams was that ETD execution affected lifecycles of classes (state change, event happening…). Consequently, they could be considered to be semantically similar.

Nonetheless, this fact did not mean that state transition diagrams were considered unnecessary in the methodological approach. ETDs were part of a requirements model, whereas state transition diagrams were part of a system model (e.g., an OO-Method conceptual schema). Both models represented and shared common phenomena, in different ways and in different development stages. They also had different purposes (requirements vs. system modelling), as indicated for a domain data model and a class diagram.

LL17) The principles, artefacts, mechanisms and guidance of the methodological approach can be useful for and applied in other RE approaches (AFF/AFA-SLL)

The last lesson learned resulted from evaluation and reported is related to an aspect about which the designers of the methodological approach had thought but that also was later indicated by supplier stakeholders. Specific ideas (principles, artefacts, mechanisms and guidance) of the methodological may be applied in other RE processes or approaches, both in academic research and in industrial practice. The reason is that they could benefit from the ideas.

Theoretically, the benefit was evident, for instance, when extensions and improvements had been proposed for the existing RE approaches adopted in the methodological approach. For example, Lauesen's task descriptions may include a new section for specification of the business rules of the work area under analysis so that they are more complete and precise.

Nonetheless, usefulness of the ideas of the methodological approach was considered to go further, and they could be adopted in other RE processes and in other RE approaches. For example, a system analyst acknowledged that an explicit criterion for homogeneous granularity of the use cases specified in his company would be useful, and a supplier manager indicated he would like to adopt BPMN for business process

modelling in his organization[2]. The system analyst also asked if the significance criterion for ETDs could be adapted for or adopted for use case modelling, and the answer was "yes".

In summary, although the methodological approach of the thesis was defined from the collaborative project and thus was initially targeted at the project partner, similar problems and needs (e.g., lack of business understanding and inadequate specification of system requirements) were observed in other industry partners. Therefore, it was considered that the ideas of the methodological approach could be useful for them.

Furthermore, the methodological approach should not be regarded as a monolithic RE approach, but as the conjunction of a set of ideas, principles and good practices that can improve the RE process of any company and that can be adopted separately.

For those ideas of the methodological approach that have been or may be adopted in other RE processes and approaches, it would be interesting to study their effectiveness and their practical usefulness. The results would positively affect, for instance, external validity of the evaluation of the methodological, provided that the results (e.g., lessons learned) coincided.

8.8 Summary

This chapter has presented the evaluation of the methodological approach of the thesis. It has been performed from an industry perspective by following a qualitative research approach and on the basis of a survey.

Data from 49 (relevant) participants were collected through unstructured interviews after application of the methodological approach in illustrations and field trial. The participants provided feedback (facts and opinions) about the usefulness of the approach and how its challenges, objectives and principles affected them. In addition, the designers of the methodological approach performed lab demos to gain insights into potential limitations of the approach in practice.

[2] Although BPMN had not been defined as notation for business process modelling in this thesis, it was considered that its adoption by the supplier manager would be a result of its adoption and use in the methodological approach, after having decided that it would the best-suited notation for business process modelling (in general and for the RE process in particular) and after having explained its strong points to the supplier manager.

As a result of the survey, several lessons about the methodological approach have been learned. Some are useful for the RE process of an IS in general, and some are useful for application of the methodological approach in particular. Anyway, it is considered that all the lessons learned are important both in academia and in industry for awareness of the state of practice and of possible situations that may occur in business process-based requirements specification and OO conceptual modelling of ISs.

The evaluation performed is considered to have been very positive. On the one hand, it has allowed the designers of the methodological approach to modify it and improve it from lessons learned in real settings. On the other hand, it has allowed further understanding of industrial practice to be gained.

Nonetheless, readers must be aware of the limitations and threats to validity indicated when using and interpreting the results (i.e., the lessons learned) of the evaluation. Probably the two main threats are: 1) the use of unstructured interviews for data collection (and thus the lack of a survey instrument), and; 2) the impossibility to provide more details about the subjects and the survey for confidentiality reasons.

Chapter 9
Conclusions

"You rise. You fall. You're down, then you rise again. What don't kill ya make ya more strong"

Metallica

Development of any activity allows people to draw conclusions from it. They can be useful for those who perform them and for others to gain insights into and better understand the activity under analysis in the form of a summary of facts and opinions. This chapter presents the main conclusions that can be drawn after development of this thesis.

Such conclusions correspond to different aspects of the methodological approach of the thesis related to what it provides and to what could be performed from it. Nonetheless, and as explained in the chapter, more aspects of the thesis may be presented and discussed as part of its conclusions.

The rest of the chapter is organised as follows. First, the contributions that have been made in the thesis are presented. Next, the impact of the thesis is discussed and possible future woks are described. Lastly, a final reflection is presented.

9.1 Contributions

This thesis has presented a methodological approach for business process-based requirements specification and OO conceptual modelling of ISs. Its general purpose is to facilitate system analysts' work when addressing the RE process of an IS and creating an OO conceptual schema of the IS from its systems requirements. Mechanisms and detailed guidance have been provided to achieve such a purpose, and other stakeholders can benefit from the approach too.

As a result of the development of the thesis, ways to address current, actual problems acknowledged in academia and in industry have been provided. Therefore, several contributions have been made. The main contributions are the following ones.

Provision of a set of fundamentals for business process-based requirements specification and OO conceptual modelling of ISs

In Chapter 3, the fundamentals of the proposed solution of the thesis have been presented. Such fundamentals can be useful not only for the methodological approach, but also for other RE approaches for IS development.

A precise definition of business process has been provided, and the types of stakeholders and of requirements that must be considered in the RE process of an IS have been discussed. It is considered that the top ten principles should be addressed by any RE approach for IS development because they represent essential issues in the RE process. Finally, the correspondence between business process models and goal models and its implications have been presented and discussed.

Development of an integrated, systematic approach for modelling of the business processes of an organization and analysis of the purpose of an IS

In Chapters 4 and 5, an integrated way to model the current state of an organization by focusing on its business processes and to analyse how a new IS may affect them has been presented. It is based on the combination of BPMN and the Map approach.

Guidance and guidelines for BPMN-based modelling of an organization have been provided, as well as for creation of goals/strategies diagrams. For determination of the effect of an IS on

business processes, operationalization tables and their analysis have been proposed. Furthermore, the mechanisms and guidance presented allow several weaknesses of BPMN and of the Map approach to be mitigated.

Development of a systematic approach and a style for specification of system requirements from business process models

In Chapter 6, a systematic way to specify the system requirements of an IS from the business process of the organization in which the system will be introduced (modelled in the form of BPDs) has been presented.

It provides a new style for SyRS, ETDs, which are elicited by analysing and enriching BPDs and specified by following a set of guidelines. ETDs represent the union of several techniques, which are improved by providing new mechanisms and guidance to address some of their weaknesses. Consequently, system requirements for support of business processes are adequately specified.

Provision of guidance for systematic derivation of an OO conceptual schema from business-process based system requirements

In Chapter 7, a systematic way to obtain the OO conceptual schema of an IS (in the form of a class diagram and state transition diagrams) from its system requirements (in the form of ETDs) has been presented.

This allows system analysts to adequately link system requirements to subsequent development stages by following a set of rules. As a result, ETDs can be integrated into OO conceptual modelling-based IS development. In addition, further details about how ETDs should be linked to OO-Method models have been provided and discussed.

Presentation of and discussion about the lessons learned by evaluating the above contributions

Finally, Chapter 8 has presented the evaluation that has been performed on the methodological approach. As a result of a survey, several lessons have been learned, which have been presented and discussed.

Lessons learned are useful for awareness of the current state of practice related to the thesis. Existing problems and needs in industry have been pointed, as well as aspects about which practitioners are not very concerned. Lessons learned are also important to analyse the practical usefulness of the methodological approach.

In summary, the thesis has dealt with challenges and objectives that, in conjunction, are not properly addressed by other RE approach. Therefore, its contributions can be considered evident and important. Furthermore, the contributions can be useful both for academia and for industry.

For academia, the thesis has indicated important issues of business process-based requirements specification and OO conceptual modelling of ISs that may be addressed in future research or should be considered when developing a RE approach. For industry, the thesis has presented and addressed actual problems in practice, thus practitioners can be aware of their existence and of possible ways to address them.

9.2 Thesis Impact

The impact of this thesis is explained and discussed in this section on the basis of these criteria: 1) the publications that are related to the thesis and have been accepted at research forums; 2) the quality of the forums of the publications; 3) the number of citations of the publications; 4) the collaborations that have arisen with researchers of other institutions during the development of the thesis, and; 5) the research stays that have been performed during the development of the thesis.

Another criterion to asses the impact of a thesis is the research projects on which it has influenced. The list of such projects for this thesis has been presented in Chapter 1. In addition, other criteria may have been defined and thus used. For example, the teaching duties performed and related to the development of the thesis, the talks given or the collaborations and contacts with industry could be criteria for assessment of the impact of a thesis.

Probably the main criterion to assess the impact of a thesis should be its use and adoption in industry and by people different to its designers. For this thesis, it is hard to predict whether the methodological approach as a whole will be adopted by practitioners (at this moment it is not), but it can be guaranteed that some of its ideas and principles (e.g., business process modelling with BPMN) have been and are being adopted in industry after their presentation to practitioners (see Chapter 8).

The following subsections present the impact of the thesis for each criterion defined in the first paragraph of this section.

9.2.1 Publications

The set of publications that are related to this thesis consists of 17 papers that have been accepted at twelve international conferences, four international workshops and one national workshop, and of one book chapter. The publications (in chronological order of publication) are the following ones:

- **de la Vara, J.L.**, Anes, D., Sánchez, J. (2007) Construcción de modelos de requisitos a partir de modelos de procesos y de metas. In: X Workshop de Ingeniería de Requisitos y Ambientes de Software (IDEAS 2007), pp 33-46.
- **de la Vara, J.L.**, Sánchez, J., Pastor, O. (2007) Integración de un Entorno de Producción Automática de Software en un Marco de Alineamiento Estratégico. In: X Workshop on Requirements Engineering (WER 2007), pp 68-79.
- **de la Vara, J.L.**, Anes, D., Sánchez, J. (2007) Descomposicón de Árboles de Metas a partir de Modelos de Procesos. In: X Workshop on Requirements Engineering (WER 2007), pp 35-46.
- **de la Vara, J.L.**, Sánchez, J. (2007) Business process-driven requirements engineering: a goal-based approach. In: VIII International Workshop on Business Process Modeling, Development and Support (BPMDS'07), pp 299-307.
- **de la Vara, J.L.**, Sánchez, J. (2007) Derivación de modelos de tareas a partir de modelos BPMN. In: I Taller sobre Procesos de Negocio e Ingeniería del Software (PNIS 2007).
- **de la Vara, J.L.**, Sánchez, J. (2008) Improving Requirements Analysis through Business Process Modelling: A Participative Approach. In: Abramowicz, W., Fensel, D. (eds.) BIS 2008, LNBIP 7. Springer, Heidelberg, pp 165-176.
- **de la Vara, J.L.**, Sánchez, J. (2008) Facilitating and Benefiting from End-User Involvement during Requirements Analysis. In: 10th International Conference on Enterprise Information Systems (ICEIS 2008), pp 316-319.
- **de la Vara, J.L.**, Sánchez, J., Pastor, O. (2008) Business Process Modelling and Purpose Analysis for Requirements Analysis of

Information Systems. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008, LNCS 5074. Springer, Heidelberg, pp 213-227.

- **de la Vara, J.L.**, Sánchez, J. (2009) Business Process-Driven Requirements Engineering: A Goal-Based Approach. In: P.K. Banerjea (ed.) Technology & Business Strategy: A Global Perspective. Icfai, Hyderabad, pp 81-92.
- **de la Vara, J.L.**, Fortuna, M.H., Sánchez, J., Werner, C.M.L., Borges, M.R.S. (2009) Modelado de Requisitos de Datos para Sistemas de Información basados en Procesos de Negocio. In: XII Conferencia Iberoamericana de Ingeniería de Requisitos y Ambientes de Software (CIbSE 2009), pp 43-57.
- **de la Vara, J.L.**, Fortuna, M.H., Sánchez, J., Werner, C.M.L., Borges, M.R.S. (2009) A Requirements Engineering Approach for Data Modelling of Process-Aware Information Systems. In: Abramowicz, W. (ed.) BIS 2009, LNBIP 21. Springer, Heidelberg, pp 133-144.
- **de la Vara, J.L.**, Sánchez, J. (2009) BPMN-Based Specification of Task Descriptions: Approach and Lessons Learnt. In: Glinz, M., Heymans, P. (eds.) REFSQ 2009, LNCS 5512. Springer, Heidelberg, pp 124-138.
- **de la Vara, J.L.**, Sánchez, J. (2009) Specification of Data Requirements from Task Descriptions. In: 21st International Conference on Software Engineering and Knowledge Engineering (SEKE 2009), pp 55-60.
- **de la Vara, J.L.**, Sánchez, J. (2010) System Modeling from Extended Task Descriptions. In: 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010), pp 425-429.
- Koschmider, A., **de la Vara, J.L.**, Sánchez, J. (2010) Measuring the Progress of Reference Model-Based Business Process Modeling. In: 3rd International Conference on Business Process and Services Computing (BPSC 2010), pp 218-229.
- **de la Vara, J.L.**, Ali, R., Dalpiaz, F., Sánchez, J., Giorgini, P. (2010) Business Process Contextualisation via Context Analysis. In:

Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010, LNCS 6412. Springer, Heidelberg, pp 471-476.

- **de la Vara, J.L.**, Ali, R., Dalpiaz, F., Sánchez, J., Giorgini, P. (2010) COMPRO: A Methodological Approach for Business Process Contextualisation. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2010, Part I, LNCS 6426. Springer, Heidelberg, pp. 132-149 (CoopIS 2010)
- **de la Vara, J.L.**, Wnuk, K., Berntsson-Svensson, R., Sánchez, J., Regnell, B. (2011) An Empirical Study on the Importance of Quality Requirements in Industry. In: 23rd International Conference on Software Engineering and Knowledge Engineering (SEKE 2011) (accepted paper)

9.2.2 Forums Quality

Table 9.1 shows the rating of the forums at which the publications of the thesis have been accepted. The rating is based on international conference rankings that are used by the Spanish Government and by Spanish universities to asses the quality of a publication. In this sense, quality of a publication is determined from the the ranking of the forum at which it was accepted.The rankings used are:

- Computing Research and Education Association of Australasia (CORE)[1];
- Conference Ranking in Computer Science (CSCR)[2], and;
- Citeseer[3].

Of the 15 forums at which some publication of thesis has been accepted, eight of them are included in some of the international conference rankings considered. Four of the conferences are considered top forums (CAiSE, ER, CoopIS and SEKE), thus six publications of the thesis can be regarded as vey high quality publications. Among these conferences, probably the toughest one is CAiSE. For the 2008 edition of the conference, just a 13% of the papers submitted were accepted.

[1] http://core.edu.au/index.php/categories/conference%20rankings. Accessed July 13, 2011
[2] http://www.grc.upv.es/localdocs/Conference_category_CS.pdf. Accessed July 13 2011
[3] http://citeseerx.ist.psu.edu/stats/venues. Accessed July 13, 2011

Table 9.1 Rating of the forums of the publications of the thesis

Conference	CORE	CSCR	Citeseer
CAiSE	A	-	209/581
ER	A	0,91	117/581
CoopIS	A	0,87	475/581
SEKE	B	0,88	-
REFSQ	B	-	-
BIS	B	-	-
ICEIS	C	-	400/581
BPMDS	C	-	-

9.2.3 Citations

Other perspective to asses the impact of a thesis is the number of times that its publications have been referenced in other works, i.e., their citations. This perspective determines how the publications and thus the work of a thesis have influenced on later works. Among all of them, the works considered relevant are those developed by authors different to the author of a thesis. This means that these authors have taken the work of the thesis as a reference for their own work and have realised of the relationship between the works.

For determination of the citations, exploratory searches in Google[4] and in Google Scholar[5] have been performed. Table 9.2 shows the number of citations of the publications of the thesis that have been referenced in some work[6]. For counting of the citations, just publications of other authors have been considered, i.e., publications of which Jose Luis de la Vara is not one of the authors.

The result of citation counting is that the publications of the thesis have been referenced in 54 works and that the h-index of the PhD candidate is four (if self-references are not considered). Nonetheless, these results may be lower than in reality given the exploratory and non-systematic nature of the searches for determination of citations.

[4] http://www.google.es. Accessed July 13, 2011
[5] http://scholar.google.es. Accessed July 13, 2011
[6] The detailed list of publications can be found in:
http://hci.dsic.upv.es/jdelavara/Citations.pdf. Accessed July 13, 2011

Table 9.2 Number of citations of the publications of the thesis

Publication	Citations
Business process-driven RE: a goal-based approach (BPMDS'07)	17
Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems (CAiSE 2008)	16
Descomposición de Árboles de Metas a partir de Modelos de Procesos (WER 2007)	4
A Requirements Engineering Approach for Data Modelling of Process-Aware Information Systems (BIS 2009)	4
Improving Requirements Analysis through Business Process Modelling: A Participative Approach (BIS 2008)	3
Derivación de modelos de tareas a partir de modelos BPMN (PNIS 2007)	2
Modelado de Requisitos de Datos para Sistemas de Información basados en Procesos de Negocio (IDEAS 2009)	2
BPMN-Based Specification of Task Descriptions: Approach and Lessons Learnt (REFSQ 2009)	2
System Modeling from Extended Task Descriptions (SEKE 2010)	2
Construcción de modelos de requisitos a partir de modelos de procesos y de metas (IDEAS 2007)	1
Measuring the Progress of Reference Model-Based Business Process Modeling (BPSC 2010)	1

9.2.4 Collaborations

Collaboration with researchers from other institutions can be considered a criterion to asses the impact of a thesis too. It is related to the interests that other researchers have shown in the work of the thesis and to their interest in researching on the same or on similar subjects.

During the development of this thesis, collaborations with researchers from the following institutions have arisen:

- Universidade Federal de Juiz de Fora, Universidade Federal do Rio de Janeiro and Universidade Federal de Pernambuco (Brazil)
- Karlsruhe Institut für Technologie (Germany)
- Università degli Studi di Trento (Italy)
- University of Pretoria (South Africa)
- Universitat Politècnica de Catalunya (Spain)
- Lunds Universitet (Sweden)

9.2.5 Research Stays

The set of research stays performed during the development of a thesis can be a criterion for assessment of its impact. The justification for the use of this criterion is almost the same as the justification for the use of collaborations: other researchers have shown interests in the work of the thesis.

Nonetheless, research stays could be considered a further step, in which researchers from the host institution and the guest researcher commit to closely work and collaborate during a given period of time. In addition, the importance of the host institution can be considered related to the quality of the work performed and thus of the thesis.

During the development of this thesis, two research stays have been performed:

- February 2010, Università degli Studi di Trento

  This research stay was performed at the Software Engineering and Formal Methods Research Group of the Department of Information and Computer Engineering, and under the supervision of Prof. Paolo Giorgini. Researchers from this Research Group have a strong background and influence on RE and conceptual modelling research and participate in major international research projects[7].

- May-August 2010, Lunds Universitet

  This research stays was performed at the Software Engineering Research Group of the Department of Computer Science, and under the supervision of Prof. Björn Regnell. Lunds Universitet and the members of the Software Engineering Research Group are considered a top institution and top researchers worldwide, respectively, on system and software engineering (Sjoberg, et al., 2005; Wong, et al., 2011)). In addition, Lunds Universitet is well-known for its strong and successful relationships with industry and for performing industry-driven, empirical research[8].

[7] http://disi.unitn.it/research/research_programs/sweng. Accessed July 13, 2011
[8] http://ease.cs.lth.se. Accessed July 13, 2011

9.3 Future Work

No PhD thesis is perfect or complete. The reason is very simple: it is impossible that a single thesis solves all the problems that may arise in real world within its research area. Problems appear and disappear as new technologies and approaches are proposed and the wishes of stakeholders evolve over time. Therefore, more work related to a thesis can always be performed to reduce it possible weak points or to address new challenges of its research area.

More work can be performed on the basis of this thesis for further advance of and improvement on the RE process of IS development. Therefore, many future works could be performed as a continuation of this thesis. A set of future works related to this thesis that have already been started or that are expected to be started in a short period of time is the following one.

Further evaluation

Current evaluation of the methodological approach presents some limitations that may be addressed to improve it and to gain further insights into its practical usefulness.

Experiments would allow specific aspects of the approach to be assessed in more detail. For example, and as performed in other works (e.g., (Marín, et al., 2010)), the effectiveness of its guidelines and rules can be evaluated. Case studies in which the approach is applied by people different to its designers would also be very valuable. Finally, surveys targeted at conclusion generalization could be performed too.

In contrast to the above empirical-based further evaluation, another stream for further evaluation (and possible improvement) would be based on theoretical models and recommended practices. For example, the methodological approach could be evaluated on the basis of its support to the fitness relationship (Salinesi, Rolland, 2003). Other evaluation could be performed on the basis of the support to CMMI (SEI, 2010). Improvement opportunities detected may increase the industrial acceptance of the approach, for instance, in companies that follow CMMI.

Extensions and improvements on business process modelling

Although the (business process-based) organizational modelling stage is considered to have been adequately defined, improvements on it can

be suggested. This a result of the evolving needs and of the opportunities that frequently arise in the research field of business process modelling. Two specific opportunities are the following ones.

Business process modelling could be improved by creating the models from a repository or a recommendation system (e.g., (Koschmider, Hornung, Oberweis, 2011)). These systems facilitate business process modelling by providing users with models similar to the ones that they want to create, thus the process of modelling a business process can be accelerated. In addition, a new idea that could be studied is how use of these systems could even be more positive by including progress measurement techniques. They would allow modellers to know the completion degree of their business process models under construction.

Another opportunity is the incorporation of techniques for context analysis (e.g., (Ali, Dalpiaz, Giorgini, 2010)) so that fitness between a business process and its context is better analysed and determined. Context can strongly influence execution of a business process. If this fact is disregarded when designing a business process, then the business process may not properly responds to the events of its context and would not be adequately executed.

Further link with OO-Method

Although link of business process-based SyRS with OO conceptual modelling has been addressed in Chapter 7, many specific details of OO-Method have not been addressed in depth in this thesis. These details have been indicated in Chapter 7, but most of the times in a very abstract way that requires further research.

Further link with OO-Method would mean that the methodological approach could be even more useful for those practitioners that use the approach for automatic software generation. It would also mean that a whole approach for automatic IS development (methodological approach in conjunction with OO-Method) would be defined and obtained. This would provide many benefits for both approaches and would represent a significant contribution for practice.

In addition, link of the methodological approach with advanced features of model-driven development with OO-Method that have been proposed recently to extend it (e.g., (Aquino, Vanderdonckt, Pastor, 2010; Panach, et al., 2008; Valverde, Pastor, 2009)) could be studied.

Extension and application of the methodological approach for new contexts

The methodological approach defines (part of) a RE process for IS development on the basis of OO conceptual modelling in general and of OO-Method in particular. Nonetheless, many other approaches, techniques, paradigms and contexts exist for IS development nowadays. Use of the methodological approach in and with them would require further study.

For example, it may necessary that some aspects of the methodological were tailored to meet specific needs of service-oriented systems (as discussed in (Graham, 2008)), of ERP systems or of agile development. Consequently, extensions or modifications in the methodological approach may be necessary for its application in these types of IS development projects.

Furthermore, once extensions have been proposed and included, the (new) methodological approach should be applied in the new contexts to evaluate its actual, practical usefulness.

Improvement on tool support

Tool support for the methodological approach is presented in Appendix B. As mentioned in Chapter 1, it corresponds to a set of prototypes whose purpose is to show feasibility of automation of the approach. Although the tool support developed is considered important and useful, it could be improved.

Improvement should be targeted mainly at two aspects. First, it should aim to further automate application of the methodological approach. Since current tool support corresponds to prototypes, just partial automation has been addressed. Second, usability of tool support should be increased. The prototypes are highly based on software generated automatically, whose usability could be improved by hand coding specific parts of tool support.

The goal of the improvements is clear: easier and faster application of the methodological approach. Consequently, application of the approach by practitioners would be facilitated and its (possible) adoption in industry may increase.

Finally, and as stated above for a PhD thesis, future work is neither perfect nor complete. Many other future works can be defined, but their relevance will depend on the needs and interest of industry and academia (i.e., researchers). In addition, a future work that has not been listed is further publication of the different parts of the thesis. This future work is implicit in any thesis, provided that all its results have not been published yet in journals, conferences or workshops.

9.4 Final Reflection

After having stayed for ten years in academia (five years as a student of a 5-years degree in computer engineering and five years as a MSc and a PhD student on computer science, and six of the ten years being involved in research), and having interacted many times with researchers and practitioners, many things seem more limpid.

In relation to this thesis, the perception about the role of RE (and thus of RE approaches) on software processes has evolved. Things that seemed inexplicable or at least hard to understand have now a justification. A picture that was blurred is now clear.

Belief in the importance of RE is evident. Otherwise, this thesis should have never been developed. However, there exist people (both practitioners and researchers) who do not agree on this opinion and undervalue RE. They claim, for instance, that they do not care about RE, that they do not develop neither need a RE process or that the effort it requires is not worth.

What these people do not realize is that RE is not only modelling business processes or specifying use cases. They do not even realize that, in fact, they deal with RE, although in a different way to what they think RE is. For sure, any software development company tries to discover customer needs and implement software systems that meet them. The use of a given RE approach or other is not the relevant point. The relevant point is to try to successfully develop systems that fulfil their purpose and satisfy stakeholders.

No existing RE approach is well-suited for all companies, types of project or types of system. Furthermore, most of (or even all) the RE approaches that are targeted at specific contexts will not always be useful

and weak points will be found. For example, further (future) work can be performed for the RE approach presented in this thesis.

The actual value of a RE approach should never be denied because of its weak points. The actual value of a RE approach should be assessed on the basis of the problems whose solution it may facilitate. Therefore, it is responsibility of both researchers and practitioners to determine on what situations a RE approach will be useful. If weaknesses are found, it does not mean that an approach is useless, but that improvement opportunities exist or that other approach would be better-suited.

This line of thought has been adopted and followed in this thesis. The methodological approach developed is based on many ideas and mechanisms of existing RE approaches. Therefore, the methodological approach has tried to take advantage of existing solutions to achieve the objectives of the thesis.

It cannot be said that the proposed solution is better than, for instance, all the approaches reviewed in Chapter 2. What can be said is that the proposed solution aims to help system analysts face the challenges described in Chapter 1, and that it is expected that the methodological approach will do it better than the existing RE approaches.

Furthermore, usefulness of a RE approach not only depends on how well and sensible it was conceived and developed. It highly depends on the skills of those that use the approach and on their goals. A paradigmatic example for me is specification and modelling of use cases.

For some time, I thought that what I had been taught about the importance and usefulness for software development of use cases was partially false. I had heard many practitioners and researchers doubting their value and saying that they would never use them (in some cases again).

Some time later, I found cases in which the success of software development highly depended on adequate and precise elicitation and specification of use cases, and practitioners completely believed in their usefulness. The explanation of the first perception became then clear. Use cases simply did not fit the needs and wishes of those practitioners (or researchers), or they simply did not know how to use or take advantage of them.

In conclusion, researchers on software development methods in general and on RE approaches in particular must listen to all stakeholders, both in academia and in industry, and look for solutions to existing or potential problems and for improvements on existing practice when necessary. What researchers should never do is to consider a personal opinion as a universal truth, or a situation as a permanent state. For any opinion or situation that is taken for granted, cases in which their bases do not hold can be found.

Although the above claims may seem obvious, they have not always been so, at least for me. Nevertheless, they are personal opinions.

References

Aagesen, G., Krogstie, J. (2010) Analysis and Design of Business Processes Using BPMN. In: vom Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management 1, International Handbooks on Information Systems. Springer, Heidelberg, pp 213-235.

Aguilar-Savén, R.S. (2004) Business process modelling: Review and Framework. International Journal of Production Economics 90(2): 129-149.

Alexander, I.F., Stevens, R. (2002) Writing Better Requirements. Pearson Education, London.

Alexander, I., Bider, I., Regev, G. (2003) REBPS 2003: Motivations, Objectives and Overview. Message from the Workshop Organizers. In: CAiSE 2003 Workshops.

Alexander, I.F., Maiden, N. (eds.) (2004) Scenarios, Stories, Use Cases. John Wiley and Sons, Chichester.

Alexander, I., Beus-Dukic, L. (2009) Discovering Requirements: How to Specify Products and Services. Wiley, Chichester.

Ali, R., Dalpiaz, F., Giorgini, P. (2010) A goal-based framework for contextual requirements modeling and analysis. Requirements Engineering 15(4): 439-458.

Allweyer, T. (2010) BPMN 2.0: Introduction to the Standard for Business Process Modeling. Books on Demand, Noderstedt.

Ameller, D., Franch, X., Cabot, J. (2010) Dealing with Non-Functional Requirements in Model-Driven Development. In: 18th IEEE International Requirements Engineering Conference (RE 2010), pp 189-198.

Antón, A.I. (1997) Goal Identification and Refinement in the Specification of Software-Based Information Systems. PhD Thesis, Georgia Institute of Technology.

Apel, S., Kästner, C. (2009) An Overview of Feature-Oriented Software Development. Journal of Object Technology 8(5): 49-84.

Aquino, N., Vanderdonckt, J., Pastor, O. (2010) Transformation templates: adding flexibility to model-driven engineering of user interfaces. In: 25th ACM Symposium On Applied Computing (SAC 2010), pp 1195-1202.

Arsanjani, A. (2005) Empowering the business analysts for on demand computing. IBM Systems Journal 44(1): 67-80.

Attaran, M. (2004) Exploring the relationship between information technology and business process reengineering. Information & Management 41(5): 585-596.

Aurum, A., Wohlin, C. (eds.) (2005a) Engineering and Managing Software Requirements. Springer, Heidelberg.

Aurum, A., Wohlin, C. (2005b) Requirements Engineering: Setting the Context. In: (Aurum, Wohlin, 2005a), pp 1-15.

Babar, A., Zowghi, D., Cox, K., Tosic, V., (2008a) Three Integration Approaches for Map and B-SCP Requirements Engineering Techniques. In: 2008 ACM symposium on Applied computing (SAC'08), pp 650-655.

Babar, A., Cox, K., Tosic, V., Bleistein, S., Verner, J. (2008b) Integrating B-SCP and MAP to manage the evolution of strategic IT requirements. Information and Software Technology 50(7-8): 815-831.

Bandara, W., Gable, G.G., Rosemann, M. (2005) Factors and measures of business process modeling: model building through a multiple case study. European Journal of Information Systems 14: 347-360.

Batra, D. (2007) Cognitive complexity in data modelling: causes and recommendations. Requirements Engineering 12(4): 231-244.

Becker, J., Rosemann, M., von Uthmann, C. (2000) Guidelines of Business Process Modeling. In: van der Aalst, W., Desel, J., Oberweis, A. (eds.) Business Process Management, LNCS 1806. Springer, Heidelberg, pp 30-49.

Becker, J., Kugeler, M., Rosemann, M. (eds.) (2003) Process Management: A Guide for the Design of Business Processes. Springer, Berlin.

Becker, J., Breuker, D., Weiß, B., Winkelman, A. (2010) Exploring the status quo of business process modelling languages in the banking sector – An empirical insight into the usage of methods in banks. In: 21st Australasian Conference on Information Systems (ACIS 2010)

Berenbach, B., Paulish, D.J., Kazmeier, J., Rudorfer, A. (2009) Software and Systems Requirements Engineering: in Practice. McGraw-Hill, New York.

Berntsson-Svensson, R., Aurum, A. (2006) Successful software project and products: An empirical investigation. In: 2006 International Symposium on Empirical Software Engineering (ISESE 2006), pp 144-153.

Berntsson-Svensson, R. (2009) Managing Quality Requirements in Software Product Development. Licentiate Thesis, Lund University.

Berntsson-Svensson, R., Gorschek, T., Regnell, B (2009) Quality Requirements in Practice: An Interview Study in Requirements Engineering for Embedded Systems. In: Glinz, M., Heymans, P. (eds) REFSQ 2009, LNCS 5512. Springer, Heidelberg, pp 218-232.

Bhattacharya, K., Gerede, C., Hull, R., Liu, R., Su, J. (2007) Towards Formal Analysis of Artifact-Centric Business Process Models. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007, LNCS 4714. Springer, Heidelberg, pp 288-304.

Bider, I. (2005) Choosing Approach to Business Process Modeling– Practical Perspective. Journal of Conceptual Modeling 34.

Bidgeland, D.M., Zahavi, R. (2009) Business Modeling: A Practical Guide to Realizing Business Value. The MK/OMG Press, San Francisco.

Birkmeier, D., Klöchner, S., Overhagen, S. (2010) An empiricial comparison of the usability of BPMN and UML activity diagrams for business users. In: 18th European Conference on Information Systems (ECIS 2010)

Bleistein, S., Cox, K., Verner, J., Phalp, K.T. (2006) B-SCP: A requirements analysis framework for validating strategic alignment of organizational IT based on strategy, context, and process. Information and Software Technology 48(9): 846-868.

Borg, A., Yong, A., Carlshamre, P., Sandahl, K. (2003) The bad conscience of requirements engineering: an investigation in real-world treatment of non-functional requirements. In: Third Conference of Software Engineering Research and Practise in Sweden (SERPS'03), pp 1-8.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J. (2004) Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems 8(3): 203-236.

Broadbent, M., Weill P., St.Clair, D. (1999) The implications of information technology infrastructure for business process redesign. MIS Quarterly 23(2): 159-182.

Bubenko, J., Persson, A., Stirna, J. (2001) EKD User Guide (online) http://people.dsv.su.se/~js/ekd_user_guide.html. Accessed July 13, 2011.

Campbell, B. (2005). Alignment: Resolving ambiguity within bounded choices. In: Ninth Pacific Asia Conference on Information Systems (PACIS 2005)

Cardoso, E.C.S., Almeida, J.P.A., Guizzardi, G. (2009) Requirements Engineering Based on Business Process Models: A Case Study. In: 13th Enterprise Distributed Object Computing Conference Workshops (EDOCW 2009), pp 320-327.

Carr, D.K., Johansson, H.J. (1995) Best Practices in Reengineering: What Works and What Doesn't in the Reengineering Process. McGraw Hill, New York.

Castro, J., Alencar, F.M.R., Filhol, G.A.C., Mylopoulos, J. (2001) Integrating organizational requirements and object oriented modelling. In: 5th IEEE International Symposium on Requirements Engineering Conference (RE'01), pp 146-153.

Castro, J., Kolp, M., Mylopoulos, J. (2002) Towards requirements-driven information systems engineering: the Tropos Project. Information Systems 27(6): 365-389.

Chalin, P., Sinnig, D., Torkzadeh, K. (2008) Capturing business transaction requirements in use case models. In: 23rd Annual ACM Symposium on Applied Computing (SAC'08), pp 602-606.

Charette, R.N. (2005) Why Software Fails. IEEE Spectrum 42(9): 42-49.

Cheng, B.H.C., Atlee, J.M. (2007) Research Directions in Requirements Engineering. In: 29th International Conference on Software Engineering (ICSE 2007), Workshop on the Future of Software Engineering (FOSE 2007), pp 285-303.

Chung, L., Nixon, B., Yu, E., Mylopoulos, J. (2000) Non-Functional Requirements in Software Engineering. Kluwer Academic Press, Boston.

Chung, L., Leite, J.C.S.P. (2009) On Non-Functional Requirements in Software Engineering. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos, LNCS 5600. Springer, Heidelberg, pp 363-379.

Classen, A., Heymans, P., Schobbens, P.Y. (2008) What's in a Feature: A Requirements Engineering Perspective. In: Fiadeiro, J., Inverardi, P. (eds.) FASE 2008, LNCS 4691. Springer, Heidelberg, 16-30.

Cockburn, A. (2001) Writing Effective Use Cases. Addison-Wesley, Boston.

Cohn, M. (2004) User Stories Applied: For Agile Software Development. Addison-Wesley, Redwood City.

Condori, N., Daneva, M., Sikkel, K., Wieringa, R., Dieste, O., Pastor, O. (2009: A systematic mapping study on empirical evaluation of software requirements specifications techniques. In: 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009), pp 502-505

Constantine, L., Lockwood, L. (1999) Software for use: a practical guide to the models and methods of usage-centered design. Addison-Wesley, New York.

Correa, A.L., Werner, C.M.L. (2004) Precise specification and validation of transactional business software. In: 12th IEEE International Requirements Engineering Conference (RE'04), pp 16-25.

Coskuncay, A., Aysolmaz, B., Demirors, O., Bilen, O., Dogani, I. (2010) Bridging the gap between business process modelling and software requirements analysis: a case study. In: Fifth Mediterranean Conference on Information Systems (MCIS 2010)

Cox, K., Phalp, K.T. (2007) Practical experience of eliciting classes from use case descriptions. Journal of Systems and Software 80(8): 1286-1304.

Dardenne, A., van Lamsweerde, A., Fickas, S. (1993) Goal-directed Requirements Acquisition. Science of Computer Programming 20: 3-50.

Davenport, T.H. (1993) Process innovation: reengineering work through information technology. Harvard Business School Press, Boston.

Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S. (2007) How do practitioners use conceptual modeling in practice? Data & Knowledge Engineering 58(3): 358-380.

Davis, A.M. (1993) Software requirements: objects, functions and states. Prentice-Hall, Upper Saddle River.

Davis, A.M. (1995) 201 principles of software development. McGraw-Hill, New York.

Davis, A.M., Hickey, A.M. (2002) Requirements Researchers: Do We Practice What We Preach. Requirements Engineering 7(2): 107-111.

Davis, A.M. (2003) System phenotypes. IEEE Software 20(4): 54-56.

Davis, A.M., Zowghi, D. (2006) Good requirements practices are neither necessary nor sufficient. Requirements Engineering 11(1): 1-3.

Davis, R., Brabänder, E. (2007) ARIS Design Platform: Getting Started with BPM. Springer-Verlag, London.

de Castro, V., Marcos, E., Vara, J.M. (2010) Applying CIM-to-PIM model transformations for the service-oriented development of information systems. Information and Software Technology 53(1): 87-105.

Decker, G., Dijkman, R., Dumas, M., García-Bañuelos, L. (2010) The Business Process Modeling Notation. In: (ter Hofstede, et al., 2010), pp 347-368.

Díaz, I., Sánchez, J., Matteo, A. (2005) Conceptual Modelling Based on Transformation Linguistic Patterns. In: Delcambre, L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, O (eds.) ER 2005, LNCS 3716. Springer, Heidelberg, pp 192-208.

Dieste, O., Juristo, N., Shull, F. (2008) Understanding the Customer: What Do We Know about Requirements Elicitation? IEEE Software 25(2): 11-13.

Dijkman, R.M., Joosten, S.M.M. (2002) Deriving Use Case Diagrams from Business Process Models. CTIT Technical Report 02-08, University of Twente.

Dijkman, R.M., Dumas, M., Ouyang, C. (2008) Semantics and analysis of business process models in BPMN. Information and Software Technology 50(12): 1281-1294.

Dobing, B., Parsons, J. (2000) Understanding the role of use cases in UML: a review and research agenda. Journal of Database Management 11(4): 28-36.

Dobing, B., Parsons, J. (2006) How UML is used. Communications of the ACM 49(5): 109-113.

Dumas, M., van der Aalst, W., ter Hofstede, A.H.M. (eds.) (2005) Process-Aware Information Systems. John Wiley & Sons, Chichester.

Dutoit, A.H., Paech, B. (2002) Rationale-Based Use Case Specification. Requirements Engineering 7(1): 3-19.

Effinger, P., Jogsch, N., Seiz, S. (2010) On a Study of Layout Aesthetics for Business Process Models Using BPMN. In: Mendling, J., Weidlich, M., Weske, M. (eds.) BPMN 2010, LNBIP 67. Springer, Heidelberg, pp 31-45.

Eriksson, H., Penker, M. (2000). Business Modeling with UML: Busines Patterns at Work. John Wiley & Sons, New York.

Eshuis, R. (2006) Symbolic Model Checking of UML Activity Diagrams. ACM Transactions on Software Engineering and Methodology 15(1): 1-38.

España, S., González, A., Pastor, O. (2009) Communication Analysis: A Requirements Engineering Method for Information Systems. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009, LNCS 5565. Springer, Heidelberg, pp 530-545.

España, S., Condori-Fernández, N., González, A., Pastor, O. (2009) Evaluating the Completeness and Granularity of Functional Requirements Specifications: A Controlled Experiment. In: 17th IEEE International Requirements Engineering Conference (RE 2009), pp 161-170.

España, S., Ruiz, M., Pastor, Ó., González, A. (2011) Systematic derivation of state machines from communication-oriented business process models. In: IEEE 5th International Conference on Research Challenges in Information Science (RCIS 2011) (accepted paper)

Estrada, H., Martínez, A., Pastor, O., Mylopoulos, J. (2006) An Empirical Evaluation if the i* Framework in a Model-Based Software Generation Environment. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006, LNCS 4001. Springer, Heidelberg, pp 513-527.

Figl, K., Mendling, J., Strembeck, M., Recker, J. (2010) On the Cognitive Effectiveness of Routing Symbols in Process Modeling Languages. In: Abramowicz, W., Tolksdorf, R. (eds.) BIS 2010, LNBIP 47. Springer, Heidelberg, pp 230-241.

Fink, A. (1995) The Survey Handbook. Sage Publications, Thousand Oaks.

Finkelstein, A. (1994) Requirements Engineering: a review and research agenda. In: 1st Asia-Pacific Software Engineering Conference (APSEC'94), pp 10-19.

Firesmith, D. (2005) Are Your Requirements Complete? Journal of Object Technology 4(1): 27-43.

Firesmith, D. (2007) Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them. Journal of Object Technology 6(1): 17-33.

Franch, X. (2010) Fostering the Adoption of i∗ by Practitioners: Some Challenges and Research Directions. In: Nurcan, S., Salinesi, C., Souveyet, C., Ralité, J. (eds) Intentional Perspectives on Information Systems Engineering. Springer, Heidelberg, pp 177-193.

Fortuna, M.H., Werner, C.M.L., Borges, M.R.S. (2008) Info Cases: Integrating Use Cases and Domain Models. In: 16th International Requirements Engineering Conference (RE'08), pp 81-84.

García Molina, J., Ortín, M.J., Moros, B., Nicolás, J. (2002) Transforming the OOram Three-Model Architecture into a UML-based Process. Journal of Object Technology 1(4): 119-135.

Genon, N., Heymans, P., Amyot, D. (2011) Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation. In: Malloy, B., Staab, S., van der Brand, M. (eds.) SLE 2010, LNCS 6563. Springer, Heidelberg, pp 377-396.

Giachetti, G., Marín, B., Pastor, O (2009) Using UML as a Domain-Specific Modeling Language: A Proposal for Automatic Generation of UML Profiles. In: In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009, LNCS 5565. Springer, Heidelberg, pp 110-124.

Glinz, M. (2000) A Lightweight Approach to Consistency of Scenarios and Class Models. In: 4th International Conference on Requirements Engineering (ICRE'00), pp 49-58.

Glinz, M., Berner, S., Joos, S. (2002) Object-oriented modeling with ADORA. Information Systems 27(6): 425-444.

Glinz, M. (2007) On Non-Functional Requirements. In: 15th International Requirements Engineering Conference (RE'07), pp 21-26.

González, A., España, S., Ruiz, M., Pastor, O. (2011) Systematic derivation of class diagrams from communication-oriented business process models. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPDMS 2011 and EMMSAD 2011, LNBIP 81. Springer, Heidelberg, pp 246-260.

Gordjin, J., Akkermans, J.M. (2003) Value-based requirements engineering: exploring innovative e-commerce idea. Requirements engineering 8(2): 114-134.

Gorschek, T., Wohlin, C. (2006) Requirements Abstraction Model. Requirements Engineering 11(1): 79-101.

Graham, I. (2008) Requirements modelling and specification for service oriented architecture. Wiley, Chichester.

Grau, G., Franch, X. (2007) ReeF: Defining a Customizable Reengineering Framework. In: In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007, LNCS 4495. Springer, Heidelberg, pp 485-500.

Grau, G., Franch, X., Maiden, N.A.M. (2008) PRiM: An i*-based process reengineering method for information systems specification. Information and Software Technology 50(1-2): 76-100.

Grosskopf, A., Decker, G., Weske, M. (2009) The Process: Business Process Modeling Using BPMN. Megahn-Kiffer Press, Tampa.

Grover, V., Malhorta, M. (1997) Business process reengineering: A tutorial on the concept, evolution, method, technology and application. Journal of Operations Management 15(3): 193-213.

Gruhn, V., Laue, R. (2009) Reducing the Cognitive Complexity of Business Process Models. In: 8th IEEE International Conference on Cognitive Informatics (ICCI'09), pp 339-345.

Gulla, J.A. (2004) Understanding Requirements in Enterprise Systems Projects. In: 12th IEEE International Requirements Engineering Conference (RE'04), pp 176-185.

Hammer, M., Champy, J. (2001) Reengineering the Corporation: A Manifesto for Business Revolution, 2nd ed. Collins, New York.

Harrington, H.J. (1991) Business Process Improvement: The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness. McGraw-Hill, New York.

Heidenreich, F., Sánchez, P., Santos, J., Zschaler, S., Alférez, M., Araújo, J., Fuentes, L., Kulesza, U., Moreira, A., Rashid. A. (2010) Relating Feature Models to Other Models of a Software Product Line - A Comparative Study of FeatureMapper and VML*. In: Katz, S., Mezini, M., Kienzle, J. (eds.) Transactions on AOSD VII, LNCS 6210. Springer, Heidelberg, pp 69–114.

Hofman, H.F., Lehner, F. (2001) Requirements Engineering as a Success Factor in Software Projects. IEEE Software 18(4): 58-66.

Hsia, P., Davis, A., Kung, D. (1993) Status report: requirements engineering. IEEE Software 10(6): 75-79.

IIBA (2009) A Guide to the Business Analysis Body of Knowledge (BABOK Guide), Version 2.0.

Indulska, M., Recker, J., Rosemann, M., Green, P. (2009) Business Process Modeling: Current Issues and Future Challenges. In: van Eck, P., Gordjin, J., Wieringa, R. (eds.) CAiSE 2009, LNCS 5565. Springer, Heidelberg, pp 501-514.

Insfrán, E., Pastor, O., Wieringa, R. (2002) Requirements Engineering-Based Conceptual Modelling. Requirements Engineering 7(2): 61-72.

ISO (2001) International Standard ISO/IEC 9126-1: Software engineering – Product quality – Part 1: Quality Model.

Ivarsson, M., Gorschek, T. (2011) A Method for evaluating rigor and industrial relevance of technology evaluations. Empirical Software Engineering 16(3): 365-395.

Ivarsson, M., Gorschek, T. (2009) Technology transfer decision support in requirements engineering research: a systematic review of REj. Requirements Engineering 14(3): 155-175.

Jackson, M. (1995) Software Requirements & Specifications: a lexicon of practice, principles and prejudices. ACM Press/Addison-Wesley, New York.

Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G. (1992) Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, Reading.

Jarke, M., Loucopoulos, P., Lyytinen, K., Mylopoulos, J., Robinson, W. (2010) The Brave New World of Design Requirements: Four Key Principles. In: Pernici, B. (ed.) CAiSE 2010, LNCS 6051. Springer, Heidelberg, pp 470-482.

Jones, S., Maiden, N. (2004) RESCUE: An Integrated Method for Specifying Requirements for Complex Socio-Technical Systems. In: Mate, J.L., Silva, A. (eds.) Requirements Engineering for Sociotechnical Systems. Information Resources Press, Arlington, pp 245-265.

Jones, S., Maiden, N.A.M., Manning, S., Greenwood, J. (2007) Informing the Specification of a Large-Scale Socio-technical System with Models of Human Activity. In: Sawyer, P., Paech, B., Heymans, P. (eds.) REFSQ 2007, LNCS 4542. Springer, Heidelberg, pp 175-189.

Juristo, N., Moreno, A.M., Silva, A. (2002) Is the European Industry Moving toward Solving Requirements Engineering Problems? IEEE Software 19(6): 70-77.

Kaindl, H., Brinkkemper, S., Bubenko, J.A., Farbey, B., Greenspan, S.L., Heitmeyer, C.L., Leite, J.C.S.P., Mead, N., Mylopoulos, J., Siddiqi, J. (2002) Requirements Engineering and Technology Transfer: Obstacles, Incentives and Improvement Agenda. Requirements Engineering 7(3): 113-123.

Kamsties, E., Hörmann, K., Schlich, M. (1998) Requirements engineering in small and medium enterprises. Requirements Engineering 3(2): 84 -90.

Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S. (1990) Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SRI-90-TR-21. Software Engineering Institute, Carnegie Mellon University.

Kaplan, R., Norton, D. (1996) The Balanced Scorecard: Translating Strategy into Action. Harvard Business School Press, Boston.

Karahanna, E., Straub, D.W., Chervany, N.L. (1999) Information technology adoption across time: a cross-sectional comparison of pre-adoption and post-adoption beliefs. MIS Quarterly 23(2): 183-213.

Kavakli, E. (2002) Goal-Oriented Requirements Engineering: A Unifying Framework. Requirements Engineering 6(4): 237-251.

Kavakli, E., Loucopoulos, P. (2006) Experiences With Goal-Oriented Modeling of Organizational Change. IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews 36(2): 221-235.

Kim, J., Hahn, J., Hahn, H. (2000) How Do We Understand a System with (So) Many Diagrams? Cognitive Integration Processes in Diagrammatic Reasoning. Information Systems Research 11(3): 284-303.

Kirchmer, M. (1999) Business Process Oriented Implementation of Standard Software: How to Achieve Competitive Advantage Efficiently and Effectively, 2nd ed. Springer, Berlin.

Kitchenham, B.A., Pfleeger, S.L. (2008) Personal opinion surveys. In: Shull, F., Singer, J., Sjoberg, D.I.K. (eds.) Guide to Advanced Empirical Software Engineering. Springer, London, pp 63-92.

Ko, R.K.L., Lee, S.S.G., Lee, E.W. (2009) Business process management (BPM) standards: a survey. Business Process Management Journal 15(5): 744-791.

Koehler, J., Vanhatalo, J. (2007) Process anti-patterns: How to avoid the common traps of business process modelling. WebSphere Technical Development Journal (online) http://www.ibm.com/developerworks/websphere/techjournal/0702_koehler/0702_koehler.html. Accessed July 13, 2011.

Koschmider, A., Hornung, T., Oberweis, A. (2011) Recommendation-based editor for business process modelling. Data & Knowledge Engineering 70(6): 483-503.

Kösters, G., Six, H.W., Winter, M. (2001) Coupling Use Case and Class Models as a Means for Validation and Verification of Requirements Specifications. Requirements Engineering 6(1): 3-17.

Kotonya, G., Sommerville, I. (1998) Requirements Engineering: Processes and Techniques. John Wiley & Sons, Chichester.

Kueng, P., Kawalek, P. (1997) Goal-based Business Process models: creation and evaluation. Business Process Management Journal 3(1): 17-38.

Kumaran, S., Liu, R., Wu, F.Y. (2008) On the Duality of Information-Centric and Activity-Centric Models of Business Processes. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008, LNCS 5074. Springer, Heidelberg, pp 32-47.

Küsters, J.M., Ryndina, K., Gall, H. (2007) Generation of Business Process Models for Object Life Cycle Compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007, LNCS 4714. Springer, Heidelberg, pp 165-181.

Lankhorst, M. (2009) Enterprise Architecture at Work: Modelling, Communication and Analysis, 2nd ed. Springer, Heidelberg.

Larman, C. (2005) Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd ed. Prenticw-Hall, Upper Saddle Rive.

Lauesen, S., Vinter, O. (2001) Preventing Requirement Defects: An Experiment in Process Improvement. Requirements Engineering 6(1): 37-50.

Lauesen, S. (2002) Software Requirements: Styles and Techniques. Addison-Wesley, London.

Lauesen, S., Vium, J.P. (2005) Communication gaps in a tender process. Requirements Engineering 10(4): 247-261.

Lauesen, S., Kuhail, M.A. (2011) Use Cases versus Task Descriptions. In: Berry, D., Franch, X. (eds.) REFSQ 2011, LNCS 6606. Springer, Heidelberg, pp 106-120.

Lawrence, B., Wiegers, K., Ebert, C. (2001) The Top Risks of Requirements Engineering. IEEE Software 18(6): 62-63.

Liang, Y. (2003) From use cases to classes: a way of building object model with UML. Information and Software Technology 45(2): 83-93.

Lindsay, A., Downs, A., Lunn, K. (2003) Business processes - attempts to find a definition. Information and Software Technology 45(15): 1015-1019.

List, B., Korherr, B. (2006) An Evaluation of Business Process Modelling Languages. In: 21st Annual ACM Symposium on Applied Computing (SAC'06), pp 1532-539.

Liu, L., Zhang, H., Peng, F., Ma, W., Shan, Y., Xu, J., Burda, T. (2009) Understanding Chinese Characteristics of Requirements Engineering. In: 17th IEEE International Requirements Engineering Conference (RE 2009), pp 261-266.

Loniewski, G., Insfrán, E., Abrahão, S. (2010) A Systematic Review of the Use of Requirements Engineering Techniques in Model-Driven Development. In: Petriu, D.C., Rouquette, N., Haugen, O. (eds.) MODELS 2010, Part II, LNCS 6395. Springer, Heidelber, pp 213-227.

Loucopoulos, P., Karakostas, V. (1995) System requirements engineering. McGraw-Hill, New York.

Lubars, M., Potts, C., Richter, C. (1993) A review of the state of the practice in requirements modelling. In: IEEE International Symposium on Requirements Engineering (ISRE 1993), pp 2-14.

Luftman, J., Ben-Zvi, T. (2010) Key Issues for IT Executives 2010: Judicious IT Investments Continue Post-Recession. MIS Quarterly Executive 9(4)

Maiden, N.A.M., Jones, S., Manning, S., Greenwood, J., Renou, L., (2004) Model-Driven Requirements Engineering: Synchronising Models in an Air Traffic Management Case Study. In: Persson, A., Stirna, J. (eds.) CAiSE 2004, LNCS 3084. Springer, Heidelberg, pp 368-383.

Marín, B., Giachetti, G., Pastor, O., Vos, T.E.J., Abran, A. (2010) Evaluating the usefulness of a functional size measurement procedure to detect defects in MDD models. In: 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2010)

Marshall, C. (2000) Enterprise Modeling with UML. Addison Wesley, Essex.

Matulevicius, R., Heymans, P. (2007) Comparing Goal Modelling Languages: An Experiment. In: Sawyer, P., Paech, B., Heymans, P. (eds.) REFSQ 2007, LNCS 4542. Springer, Heidelberg, pp 18-32.

Matulevicius, R., Heymans, P., Opdahl, A. (2007) Comparing GRL and KAOS using the UEML Approach. In: Gonçalves, R.J., Müller, J.P, Mertins, K., Zelm, M. (eds.) Enterprise Interoperability II, Part I. Springer, London, pp 77-88.

McKeen, J., Smith, H.A. (2003) Making IT Happen: Critical Issues in IT Management. . John Wiley & Sons, Chichester.

Mead, N.R. (2000) Why is it so Difficult to Introduce Requirements Engineering Research Results into Mainstream Requirements Engineering Practice? In: In: 4th International Conference on Requirements Engineering (ICRE'00), pp 75-76.

Melão, N., Pidd, M. (2000) A conceptual framework for understanding business processes and business process modelling. Information Systems Journal 10(2): 105-129.

Mendling, J., Strembeck, M. (2008) Influence Factors of Understanding Business Process Models. In: Abramowicz, W., Fensel, D. (eds.) BIS 2008, LNBIP 7. Springer, Heidelberg, pp 142-153.

Mendling, J., Reijers, H.A., Recker, J. (2010) Activity labeling in process modeling: Empirical insights and recommendations. Information Systems 35(4): 467-482.

Mendling, J., Reijers, H.A., van der Aalst, W.M.P. (2010) Seven process modelling guidelines (7PMG). Information and Software Technology 52(2): 127-136.

Monteiro, R., Araújo, J., Amaral, V., Patrício, P. (2010) MDGore: Towards Model-Driven and Goal-Oriented Requirements Engineering. In: 18th IEEE International Requirements Engineering Conference (RE 2010), pp 405-406.

Moody, D.L., Heymans, P., Matulevicius, R. (2010) Visual syntax does matter: improving the cognitive effectiveness of the i* visual notation. Requirements Engineering 15(2): 141-175.

Neill, C.J., Laplante, P.A. (2003) Requirements Engineering: The State of the Practice. IEEE Software 20(6): 40-45.

Nigam, A., Caswell, N.S. (2003) Business artifacts: An approach to operational specification. IBM Systems Journal 42(3): 428-445.

Nurcan, S., Rolland, C. (2003) A multi-method for defining the organizational change. Information and Software Technology 45(2): 61-82.

Nuseibeh, B., Easterbrook, S. (2000) Requirements engineering: a roadmap. In: Conference on The Future of Software Engineering, 22nd International Conference on Software Engineering (ICSE'00), pp 35-46.

Nysetvold, A.G., Krogstie, J. (2005) Assessing Business Process Modelling Languages Using a Generic Quality Framework. In: 10th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD05)

Odeh, M., Kamm, R. (2003) Bridging the gap between business models and system models. Information and Software Technology 45(15): 1053-1060.

Olivé, A. (2007) Conceptual Modeling of Information Systems. Springer, Heidelberg.

Olle, T.W., Hagelstein, J., Macdonald, I.G., Rolland, C., Sol, H.G., van Assche, F.J.M., Verrijn-Stuart, A.A. (1991) Information Systems Methodologies: A Framework for Understanding, 2nd ed. Addison-Wesley, Workingham.

OMG (2003) MDA Guide Version 1.0.1 (online) www.omg.org/mda. Accessed July 13, 2011.

OMG (2005) Unified Modeling Language: Superstructure Version 2.0 (online) http://www.uml.org. Accessed July 13, 2011.

OMG (2006) Object Constraint Language Version 2.0 (online) http://www.omg.org/spec/OCL/2.0/. Accessed July 13, 2011.

OMG (2009) Business Process Model and Notation (BPMN) Specification v.1.2 (online) http://www.bpmn.org. Accessed July 13, 2011.

O'Neill, P., Sohal, A.S. (1999) Business Process Reengineering: A review of recent literature. Technovation 19(9): 571-581.

Ould, M. (1995) Business processes: modelling and analysis for re-engineering and improvement. Wiley, Chichester.

Overmyer, S.P., Lavoie, B., Rambow, O. (2001) Conceptual modeling through linguistic analysis using LIDA. In: 23rd International Conference on Software Engineering (ICSE'01), pp 401-410.

Paech, B., Koenig, T., Borner, L., Aurum, A. (2005) An analysis of empirical requirements engineering survey data. In: (Aurum, Wohlin, 2005a), pp 427-452.

Panach, J.I., España, S., Moreno, A.M., Pastor, O. (2008) Dealing with Usability in Model Transformation Technologies. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008, LNCS 5231. Springer, Heidelberg, pp 498-511.

Parsons, J., Wand, Y. (1997) Choosing Classes in Conceptual Modeling. Communications of the ACM 40(6): 63-69.

Parsons, J., Wand, Y. (2000) Emancipating Instances from the Tyranny of Classes in Information Modeling. ACM Transactions on Database Systems 25(2): 228-268.

Pastor, O., Molina, J.C. (2007) Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. Springer, Heidelberg.

Patton, M.Q. (2001) Qualitative Research and Evaluation Methods, 3rd ed. Sage Publications, London.

Pinheiro, F.A.C. (2003) Requirements honesty. Requirements Engineering 8(3): 183-192.

Pohl, K. (2010) Requirements Engineering: Fundamentals, Principles, and Techniques. Springer, Heidelberg.

Porter, M.E. (1985) Competitive Advantage: Creating and Sustaining Superior Performance. Free Press, New York.

Potts, C. (1993) Software-Engineering Research Revisited. IEEE Software 10(5): 19-28.

Procaccino, J.D., Verner, J.M., Lorenzet, S.J. (2006) Defining and contributing to software development success. Communications of the ACM 49(8): 79-83.

Ratcliffe, M., Budgen, D. (2005) The application of use cases in systems analysis and design specification. Information and Software Technology 47(9): 623-641.

Recker, J., Indulska, M., Green, P. (2007) Extending Representational Analysis: BPMN User and Developer Perspectives. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007, LNCS 4714. Springer, Heidelberg, pp 384-399.

Recker, J., zur Muehlen, M., Siau, K, Erickson, J., Indulska, M. (2009) Measuring method complexity: UML versus BPMN. In: 15th Americas Conference on Information Systems (AMCIS 2009)

Recker, J. (2010) Opportunities and constraints: the current struggle with BPMN. Business Process Management Journal 16(1): 181-201.

Recker, J. (2011) Evaluations of Process Modeling Grammars: Ontological Qualitative and Quantitative Analyses Using the Example of BPMN, LNBIP 71. Springer, Heidelberg.

Regev, G., Wegmann, A. (2004) Defining early IT system requirements with regulation principles: the Lightswitch approach. In: 12th IEEE International Requirements Engineering Conference (RE'04), pp 144-153.

Regev, G., Wegmann, A. (2005) Where do Goals Come from: the Uncerlying Principles of Goal-Oriented Requirements Engineering. In: 13th IEEE International Conference on Requirements Engineering (RE'05), pp 353-362.

Regnell, B., Berntsson-Svensson, R., Olsson, T. (2008) Supporting Roadmapping of Quality Requirements. IEEE Software 25(2): 42-47.

Reich, B.H., Benbasat, I. (200) Factors that influence the social dimension of alignment between business and information technology objectives. MIS Quarterly 24(1): 81-113.

Reijers, H.A., Liman, S., van der Aalst, W.M.P. (2003) Product-Based Workflow Design. Journal of Management Information Systems 20(1): 229-262.

Reijers, H.A., Mansar, S.L. (2005) Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. Omega 33(4): 283-306.

Robertson, S., Roberson, J. (2006) Mastering the Requirements Process, 2nd ed. Addison-Wesley, Boston.

Robson, C. (2002) Real World Research, 2nd ed. Blackwell, Oxford.

Rodriguez, A., de Guzmán, I.G.R., Fernández-Medina, E., Piattini, M. (2009) Semi-formal transformation of secure business processes into analysis class and use case models: An MDA approach. Information and Software Technology 52(9): 945-971.

Rolland, C., Nurcan, S., Grosz, G. (2000) A decision-making pattern for guiding the enterprise knowledge development process. Information and Software Technology 42(5): 313-331.

Rolland, C., Prakash, N. (2000) From conceptual modelling to requirements engineering, Annals of Software Engineering 10(1-4): 151-176.

Rolland, C., Salinesi, C. (2005) Modeling Goals and Reasoning with Them. In: (Aurum, Wohlin, 2005a), pp 189-217.

Rolland, C. (2007) Capturing System Intentionality with Maps. In: Krogstie, J., Opdhal, A.L., Brinkkemper, S. (eds.) Conceptual Modelling in Information Systems Engineering. Springer, Heidelberg, pp 141-158.

Rolland, C., Kirsch-Pinheiro, M., Souveyet, C. (2010) An Intentional Approach to Service Engineering. IEEE Transactions on Service Computing 3(4): 292-205.

Rolón, E., Sánchez, L., García, F., Ruiz, F., Piattini, M., Caivano, D., Visaggio, G. (2009) Prediction Models for BPMN Usability and Maintainability. In: 2009 IEEE Conference on Commerce and Enterprise Computing (CEC'09), pp 383-390.

Rosemann, M., Recker, J., Indulska, M., Green, P. (2006) A Study of the Evolution of the Representational Capabilities of Process Modeling Grammar. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006, LNCS 4001. Springer, Heidelberg, pp 447-461.

Rouibah, K., Al-Rafee, S. (2009) Requirement engineering elicitation methods: A Kuwaiti empirical study about familiarity, usage and perceived value. Information Management & Computer Security 17(3): 192-217.

Rubens, J. (2007) Business analysis and requirements engineering: the same, only different? Requirements Engineering 12(2): 121-123.

Runeson, P (2003) Using Students as Experiment Subjects - An Analysis on Graduate and Freshmen Student Data. In: 7th International Conference on Empirical Assessment in Software Engineering (EASE 2003), pp 95-102.

Runeson, P., Höst, M. (2009) Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14(2): 131-164.

Sadraei, E., Aurum, A., Beydoun, G., Paech, B. (2007) A field study of the requirements engineering practice in Australian software industry. Requirements Engineering 12(3): 145-162.

Salinesi, C., Presso, M.J. (2002) A Method to Analyse Changes in the Realisation of Business Intentions and Strategies for Information System Adaptation. In: Sixth International Enterprise Distribute Object Computing (EDOC'02), pp 84-95.

Salinesi, C., Rolland, C. (2003) Fitting Business Models to System Functionality Exploring the Fitness Relationship. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003, LNCS 2681. Springer, Heidelberg, pp 647-664.

Sánchez L., García, F., Ruiz, F., Piattini, M. (2010) Measurement in business processes: a systematic review. Business Process Management Journal 16(1): 114-134.

Scheer, A.W. (2000) ARIS - Business Process Modeling, 3rd ed. Springer, Heidelberg.

Schrepfer, M., Wolf, J., Mendling, J., Reijers, H.A. (2009) The Impact of Secondary Notation on Process Model Understanding. In: Persson, A., Stirna, J. (eds.) PoEM 2009, LNBIP 39. Springer, Heidelberg, pp 161-175.

SEI (2010) CMMI for Development, Version 1.3, CMU/SEI-2010-TR-033 (online) http://www.sei.cmu.edu. Accessed July 13, 2011.

Shaker, P. (2010) Feature-Oriented Requirements Modeling. In: 32nd ACM/IEEE International Conference on Software Engineering (ICSE 2010), vol. 2, pp 365-368.

Sharp, A., McDermott, P. (2009) Workflow modelling: tools for process improvement and application development. Artech, Norwood.

Shun, S.X., Zhao, J.L., Numaker, J.F. (2006) Formulating the Data-Flow Perspective for Business Process Management. Information Systems Research 17(4): 374-391.

Siau, K., Cao, Q. (2001) Unified modeling language - a complexity analysis. Journal of Database Management 12(1): 26-34.

Siau, K., Lee, L. (2004) Are use case and class diagram complementary in requirements analysis? Requirements Engineering 9(4): 229-237.

Silver, B. (2009) BPMN Method and Style. Cody-Cassidy Press, Aptos.

Singh, S.N., Woo, C. (2008) A Methodology for Discovering Goals at Different Organizational Levels. In: 3rd International Workshop on Business/IT Alignment (BUSITAL'08)

Singh, S.N., Woo, C. (2009) Investigating business-IT alignment through multi-disciplinary goal concepts. Requirements Engineering 14(3): 177-207.

Sjøberg, D.I.K, Hannay, J.E., Hansen, O., Kampenes, V.B., Karahasanovic, A., Liborg, N.K., Rekdal, A.C. (2005) A Survey of Controlled Experiments in Software Engineering. IEEE Transactions on Software Engineering 31(9): 733-753.

Smith, H., Fingar, P. (2002) Business Process Management: The Third Wave. Meghan-Kiffer Press, Tampa.

Sommerville, I., Sawyer, P. (1997) Requirements Engineering: A Good Practice Guide. Wiley, Chichester.

Sommerville, I. (2005) Integrated Requirements Engineering: A Tutorial. IEEE Software 22(1): 16-23.

Stirna, J., Persson, A., Sandkuhl, K. (2007) Participative Enterprise Modeling: Experiences and Recommendations. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007, LNCS 4495. Springer, Heidelberg, pp 546-560.

Stirna, J., Persson, A. (2009) Anti-patterns as a Means of Focusing on Critical Quality Aspects in Enterprise Modeling. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) BPMDS 2009 and EMMSAD 2009, LNBIP 29. Springer, Heidelberg, pp 407-418.

Svetinovic, D., Berry, D.M., Godfrey, M.W. (2005) Concept Identification in Object-Oriented Domain Analysis: Why Some Students Just Don't Get It. In: 13th IEEE International Requirements Engineering Conference (RE'05), pp 189-198.

Taylor-Cummings, A. (1998) Bridging the user-IS gap: a study of major information systems projects. Journal of Information Technology 13: 29-54.

ter Hofstede, A.H.M., ven der Aalst, W.M.P., Adams, M., Russell, N. (eds.) (2010) Modern Business Process Automation: YAWL and its Support Environment. Springer, Heidelberg.

The Business Rules Group (2000) Defining Business Rules ~ What Are They Really? Final Report, revision 1.3 (online) http://www.businessrulesgroup.org/first_paper/BRGwhatisBR_3ed.pdf. Accessed July 13, 2011.

Thevent, L.H., Salinesi, c. (2007) Aligning IS to Organization's Strategy: The INSTAL Method. In: Krogstie, J., Opdahl, A.I., Sindre, G. (eds.) CAiSE 2007, LNCS 4495. Springer, Heidelberg, pp 203-217.

Vaishnavi, V.K., Kuechler, W. (2008) Design Science Research Methods and Patterns: Innovating Information and Communication. Auerbach Publications, Boca Raton.

Valverde, F., Pastor, O. (2009) Facing the Technological Challenges of Web 2.0: A RIA Model-Driven Engineering Approach. In: Vossen, G., Long, D.D.E., Yu, J.X. (eds.) WISE 2009, LNCS 5820. Springer, Heidelberg, pp 131-144.

van Lamsweerde, A. (2001) Goal-oriented requirements engineering: a guided tour. In: 5th IEEE International Symposium on Requirements Engineering (RE'01), pp 249-262.

Vernadat, F. (1996) Enterprise Modelling and Integration: Principles and Applications. Chapman & Hall, London.

Verner, J., Cox, K., Bleistein, S., Cerpa, N. (2005) Requirements engineering and software project success: an industrial survey in Australia and the U.S. Australasian Journal of Information Systems 13(1): 225-238.

Vessey,I., Coner, S. (1994) Requirements Specification: Learning Objects, Process and Data Methodologies. Communications of the ACM 37(5): 102-113.

Wahl, T., Sindre, G. (2005) An Analytical Evaluation of BPMN Using a Semiotic Quality Framework. In: 10th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD05)

Wang, J., Kumar, A. (2005) A Framework for Document-Driven Workflow Systems. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005, LNCS 3649. Springer, Heidelberg, pp 285-301.

Wan-Kadir, W.M.N., Loucopoulos, P. (2004) Relating evolving business rules to software design. Journal of Systems Architecture 50(7): 367-382.

Weidenhaupt K., Pohl, K., Jarke, M., Haumer, P. (1998) Scenarios in System Development: Current Practice. IEEE Software 15(2): 34-45.

Weske, M. (2007) Business Process Management: Concepts, Languages, Architectures. Springer, Heidelberg.

WfMC (1999) Workflow Management Coalition: Terminology & Glossary (online) http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf. Accessed July 13, 2011.

White, S.A., Miers, D. (2008) BPMN Modeling and Reference Guide. Future Strategies Inc., Lighthouse Point.

Whitman, M. (1996) IT divergence in reengineering support: Performance expectations vs. perceptions. Information & Management 30(5): 239-250.

Whittle, J., Schumann, J. (2000) Generating Statechart Designs From Scenarios. In: 22nd International Conference on Software Engineering (ICSE'00), pp 314-323.

Wiegers, K.E. (2003) Software Requirements, 2nd ed. Microsoft Press, Redmond.

Wieringa, R. (2005) Requirements researchers: are we really doing research? Requirements Engineering 10(4): 304-306.

Wieringa, R. (2008) Requirements Engineering Research Methodology: Principles and Practice. University of Twente.

Wnuk, K., Regnell, B., Karlsson, L. (2009) What Happened to Our Features? Visualization and Understanding of Scope Change Dynamics in a Large-Scale Industrial Setting. In: 17th IEEE International Requirements Engineering Conference (RE 2009), pp 89-98.

Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hostede, A.H.M., Russell, N. (2006) On the suitability of BPMN for Business Process Modelling. In: Dustdar, S., Fiadeiro, J.L., Sheth, A. (eds.) BPM 2006, LNCS 4102. Springer, Heidelberg, pp 161-176.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A. (2000) Experimentation in Software Engineering: An Introduction. Kluwer Academic Press, Boston.

Wohlin, C., Höst, M, Henningsson, K. (2003) Empirical Research Methods in Software Engineering. In: Conradi, R., Wang, A.I. (eds.) ESERNET 2001-2003, LNCS 2765. Springer, Heidelberg, pp 7–23.

Wong, W.A, Tse, T.H., Glass, R.L., Basili, V.R., Chen, T.Y. (2011) An assessment of systems and software engineering scholars and institutions (2003–2007 and 2004–2008) Journal of Systems and Software 84(1): 162-168.

Yin, R.K. (2009) Case Study Research: Design and Methods, 4th ed. Sage Publications, Los Angeles.

Yourdon, E., Nevermann, P., Oppel, K., Thomann, J., Whitehead, K. (1995) Mainstream objects: an analysis and design approach for business. Yourdon Press, Upper Saddle River.

Yu, E. (1995) Modelling Strategic Relationships for Process Reengineering. PhD Thesis, University of Toronto.

Yue, T., Briand, L.C., Labiche, Y. (2009) A Use Case Modeling Approach to Facilitate the Transition towards Analysis Models: Concepts and Empirical Evaluation. In: Schürr, A., Selic, B. (eds.) MODELS 2009, LNCS 5795. Spinger, Heidelberg, pp 484-498.

Yue. T., Briand, L.C., Labiche, Y. (2010) A systematic review of transformation approaches between user requirements and analysis models. Requirements Engineering 16(2): 75-99.

Zave, P. (1997) Classification of research efforts in requirements engineering. ACM Computing Surveys 29(4): 315-321.

Zave, P., Jackson, M. (1997) Four Dark Corners of Requirements Engineering. ACM Transactions on Software Engineering and Methodology 6(1): 1-30.

zur Muehlen, M., Ho, D.T. (2008) Service Process Innovation: A Case Study of BPMN in Practice. In: 41st Hawaii International Conference on Systems Science (HICSS-41 2008)

zur Muehlen, M., Recker, J. (2008) How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008, LNCS 5074. Springer, Heidelberg, pp 465-479.

zur Muehlen, M., Indulska, M. (2010) Modeling languages for business processes and business rules: A representational analysis. Information Systems 35(4): 378-390.

zur Muehlen, M., Wisnosky, D., Kindrick, J. (2010) Primitives: Design Guidelines and Architecture for BPMN Models. In: 21st Australasian Conference on Information Systems (ACIS 2010)

Appendix A
Conceptual Framework

This appendix presents a conceptual framework for the methodological approach of the thesis. Such a framework corresponds to the metamodel or conceptual schema of the approach, and includes the main, basic concepts proposed and used in the stages of approach and the relationships that exist between them.

The conceptual framework is divided into five different sub-conceptual frameworks for:

- Organizational modelling (Figure A.1)
- Purpose analysis (Figure A.3)
- Specification of system requirements (Figure A.2)
- Class diagrams (Figure A.4)
- State transition diagrams (Figure A.5)

They have been modelled in a domain data model-like way.

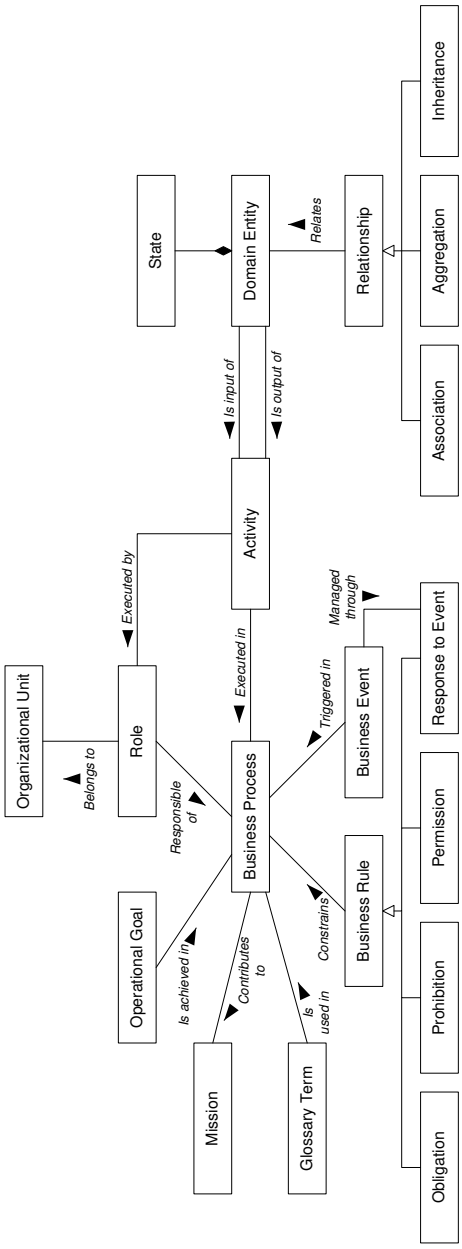

Figure A.1 Conceptual framework for organizational modelling

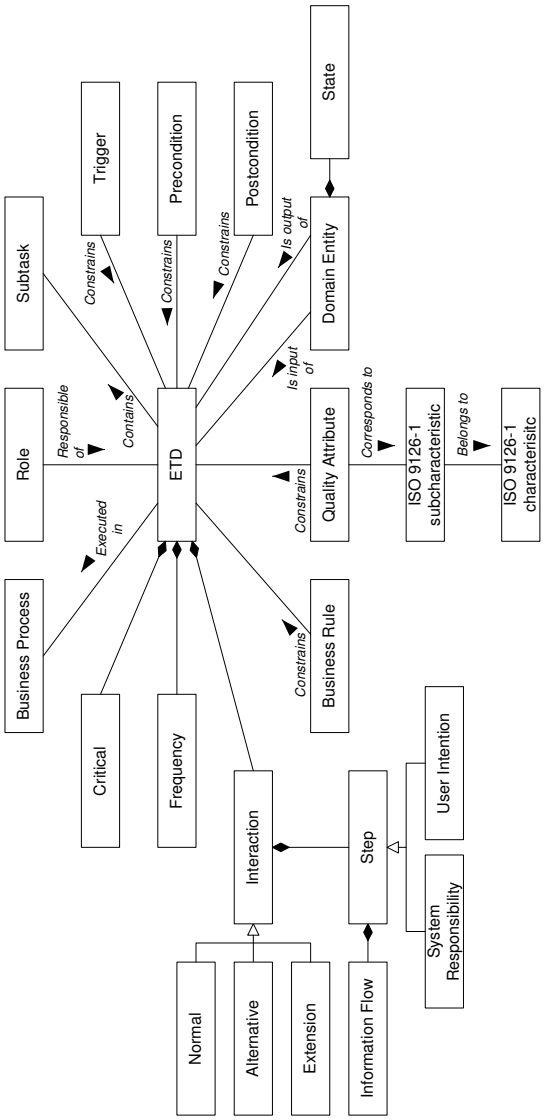

Figure A.2 Conceptual framework for specification of system requirements

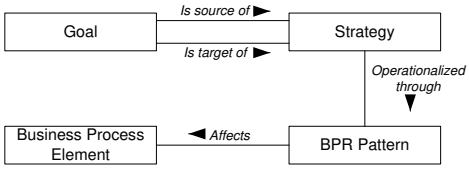

Figure A.3 Conceptual framework for purpose analysis

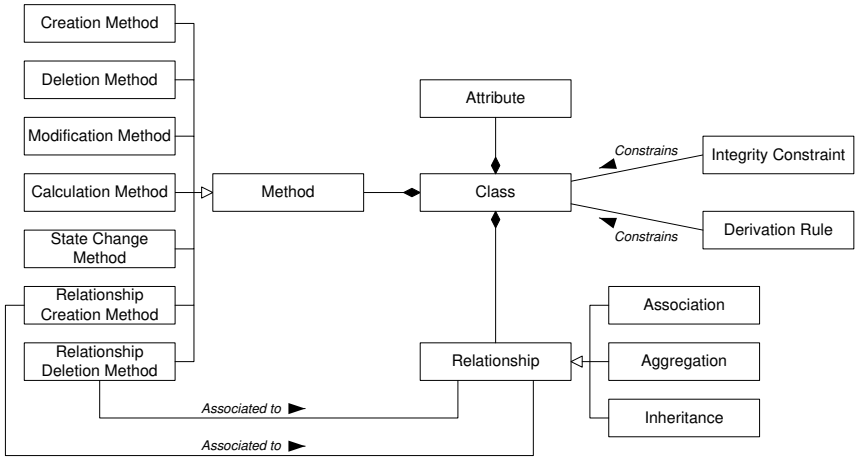

Figure A.4 Conceptual framework for class diagrams

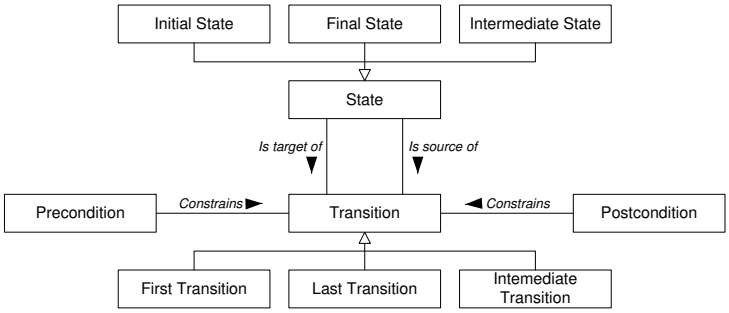

Figure A.5 Conceptual framework for state transition diagrams

Appendix B
Tool Support

This appendix outlines the tool support that has been developed for the methodological approach of the thesis. It aimed to show feasibility of automation of the approach.

Tool support consists of six main components (i.e., editors):

- An organizational model editor
- A Map editor
- A BPMN editor
- An ETD editor
- A class diagram editor
- A state transition diagram editor

Although most the details about the editors are out of the scope of the thesis and thus are not presented in this appendix, it must be indicated that development of tool support has been based on the Eclipse environment[1]. As a result, components and characteristics of the environment that facilitate creation of editors have been used, such as Ecore modelling, EMF and GMF. The class diagram editor and the state

[1] http://www.eclipse.org. Accessed July 13, 2011.

transition diagram editor have not been specifically developed for the methodological approach of the thesis, but UML2 tools provided by Eclipse have been used.

The guidelines and rules proposed in the methodological approach have been partially automated. More concretely, part of the textual template of the ETDs can be filled from enriched BPDs and part of an OO conceptual schema can be derived from ETDs. This implies that further automation is yet possible, as well as other improvements on tool support (see Chapter 9). Guidelines and rules have been implemented by hand coding them and by taking advantage of the ATL language for specification of transformations between models.

The following figures show some screenshots and examples of:

- an ecore diagram (Figure B.1);
- a tree-based editor generated from an ecore diagram and on the basis of EMF (Figure B.2);
- a diagram modelled with the Map editor (Figure B.3), and;
- the BPMN editor (Figure B.4), which also integrates a form-based ETD editor; it supports BPMN 2.0 labels for labelling of flow objects, as well as modelling of consecutive flows.

Tool support for RE approaches is considered essential for adoption in industry (e.g., (Kaindl, et al., 2002)). This fact was confirmed during evaluation of the methodological approach of the thesis. Many supplier stakeholders asked about the tool support developed and the automation provided. Even though full automation was not yet available, existence of tool support and possibility of improvement were considered positive.

Nonetheless, cases in which existence of tool support had not implied success in adoption of a RE approach were found. Some organizations had acquired or even developed their own tools as part of the adoption of a RE approach, but adoption failed because of other reasons. The most usual reason was the lack of enough and adequate guidance to apply the approaches. Consequently, tool support did not provide any advantage because of the existence of other inherent difficulties for application.

In relation to the above fact, development of the methodological approach has been in line with those authors that think that techniques can be more important that tools (e.g., (Davis, 1995)).

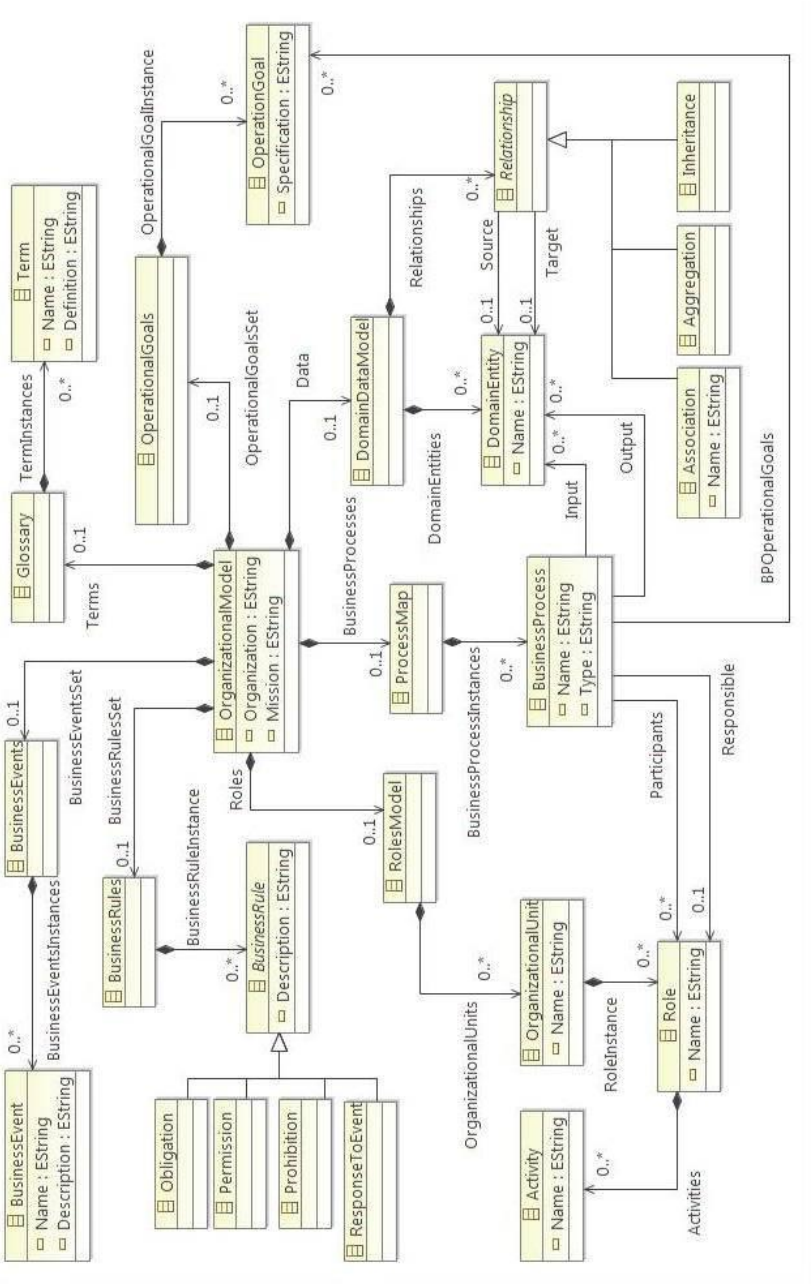

Figure B.1 Ecore diagram for the organizational modelling stage

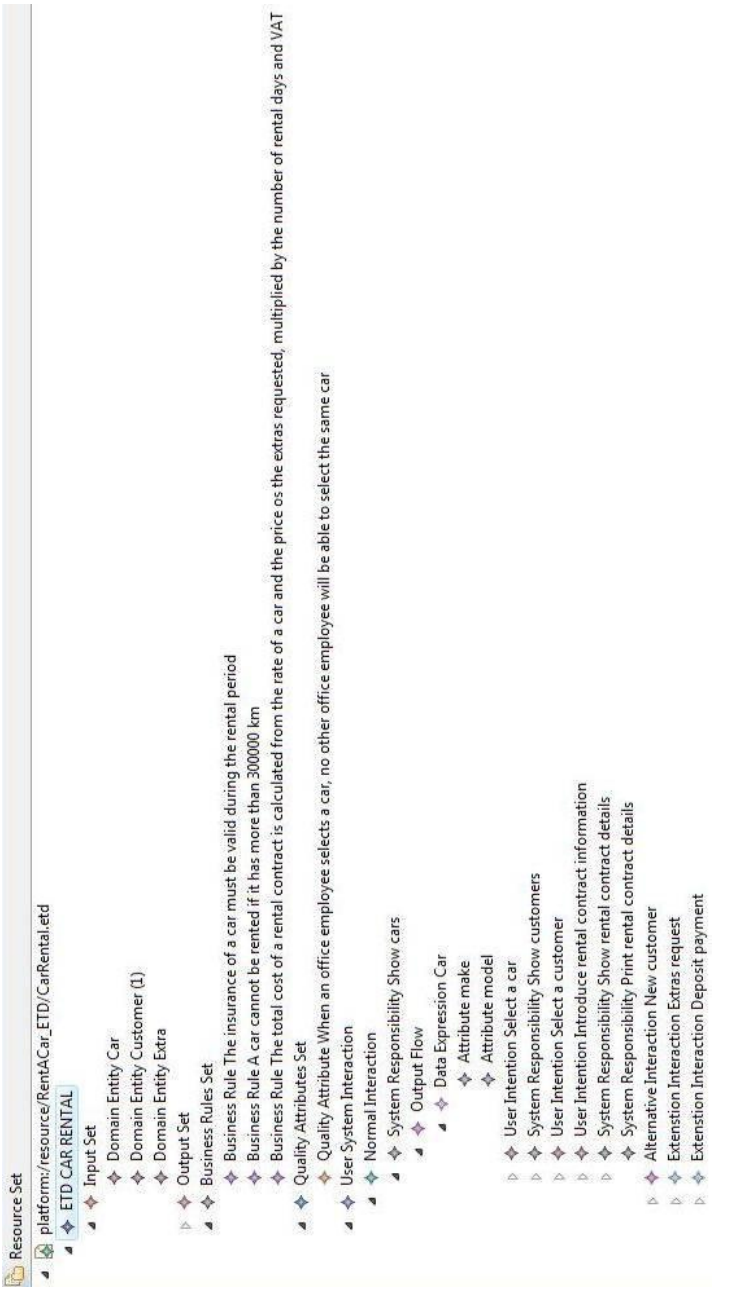

Figure B.2 Tree-based ETD editor

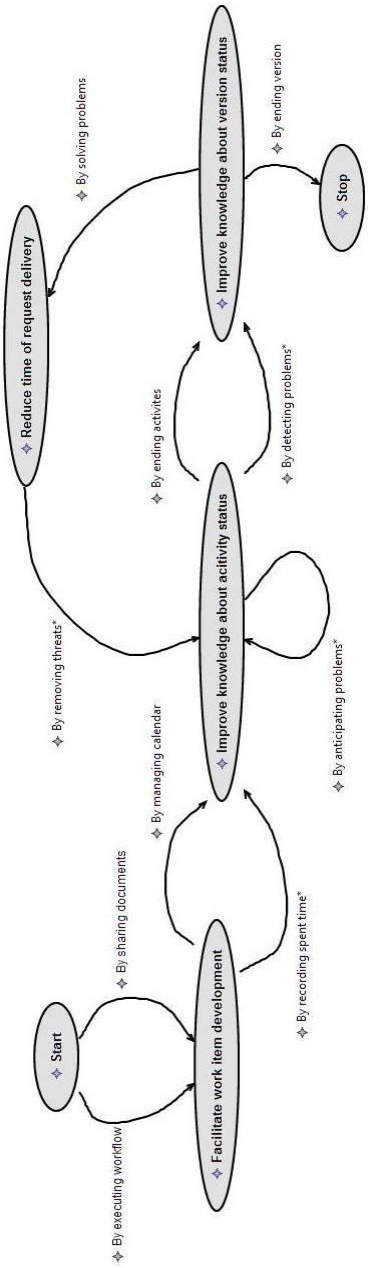

Figure B.3 Example of goals/strategies diagram

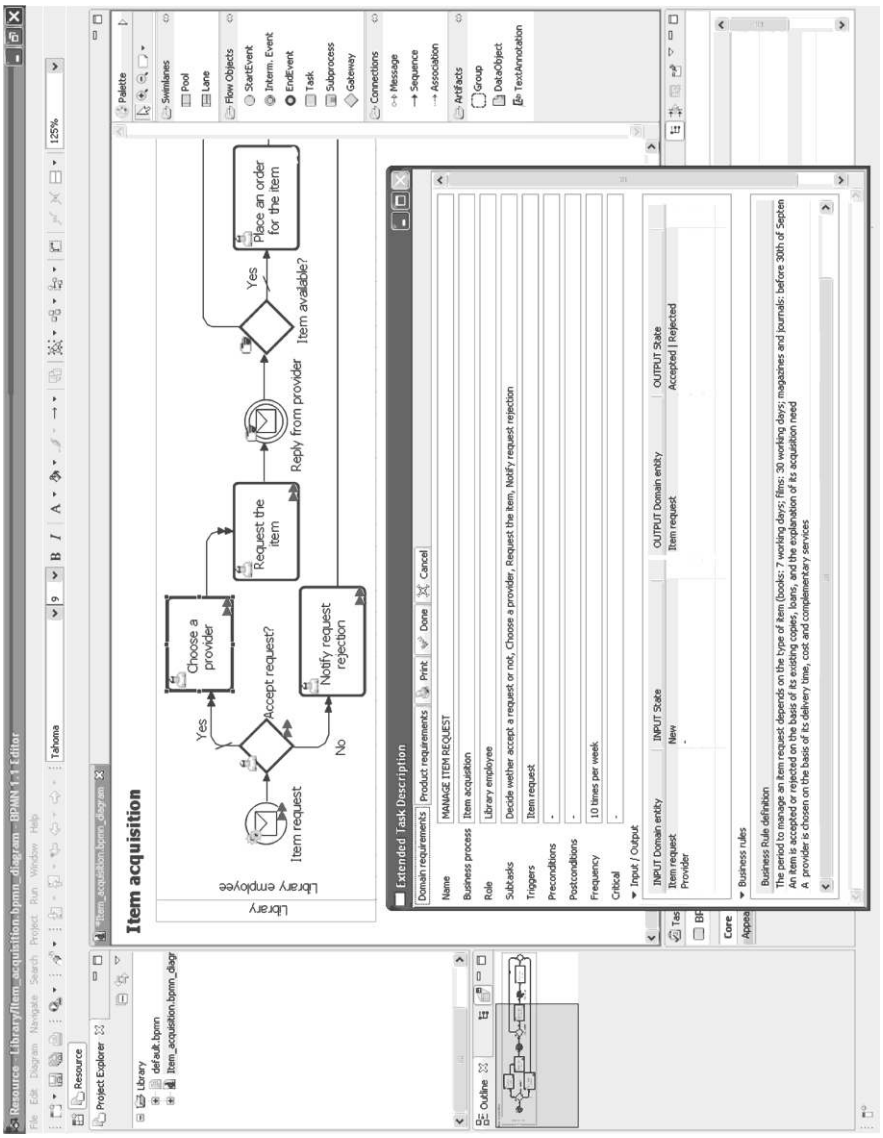

Figure B.4 BPMN editor and form-based ETD editor