PHD THESIS:

MODELADO DE LAS RELACIONES ENTRE ELEMENTOS DEL NEGOCIO Y ENTRE REQUISITOS PARA LA SELECCIÓN DE PATRONES ARQUITECTÓNICOS

JAVIER BERROCAL

DEPARTMENT OF COMPUTER AND TELEMATIC SYSTEMS ENGINEERING

Conformity of the Supervisor:

Signed: D. Juan Manuel Murillo Rodríguez

Associate Professor

Department of Computer and Telematic Systems Engineering

University of Extremadura

2014

To my mother, sister and grandparents, and especially
to the memory of my father, for all the values
that they have instilled in me.

To Éric, the best source of joy and amusement.

To María José, for supporting my dreams
like if they were hers.

Acknowledgement

"There comes a point in your life when you realize who really matters, who never did, and who always will."

The realization of a PhD thesis involves a continuous and constant work for a long time. Even though its development leads to the grant of the doctoral degree to a single person, its successful completion is usually only possible by the advices, helps and the constant encouragement provided by the people close to the PhD student. In my case, there have been a large number of people helping me and pushing me in this journey. I hope that the following paragraphs serve to convey my deepest and sincere gratitude.

Firstly, I would like to thank my supervisor, D. Juan Manuel Murillo, for all the time devoted to this project, for his patience and the wise advices he has given to me, and, above all, for creating an environment of friendship, fellowship and collaboration that has greatly facilitated the achievement of not only my PhD, but also any other projects addressed.

Secondly, I would like to thank all members of Gloin for the help and support they have given me. In particular, I would like to thank José García-Alonso, he has been a tireless companion on this journey that we have shared. Your support has been one of the pillars that allowed me to get this PhD.

Third, I would like to thank all the members of the Quercus Software Engineering group and, in particular, its director Juan Hernández. I will never stop to thank you the comments and feedback provided in the different meetings we have had and for encouraging me to follow the right path. Also, I would like to thank Cristina

Vicente-Chicote, her invaluable advices and comments gave me the final push to successfully complete this thesis.

I would also like to express my gratitude to all the colleagues of the Department of Computer and Telematic Systems Engineering for their support.

I have left for the end the most important people. I would like wholeheartedly thank my family. To my father, who although unfortunately did not accompany me in the final stretch, instilled in me perseverance, effort and perfection in everything I do. I wish you would have shared this moment with me, which is also yours. To my mother and my sister, they taught me that any adversity can be overcome with strength and courage. They are an example to follow. To my grandparents, my uncles and my cousins, for their continued support and concern about my future. To Nandi and José, for taking care of Éric and María José, so that I could continue working.

Finally, I would especially like to thank Éric for getting me out a smile independently of how tired I came home. And to María José, for sharing this journey with me, for helping me in everything that you could, for cheering me up in the bad times and celebrating with me the good ones, and for, even, taking care of our son during so many months, so that I could continue working and do my research stay. I hope that I can give you back all the lost moments. Your support has been essential for the successful conclusion of this thesis.

Abstract

This thesis began to take shape seven years ago in the Teseo software factory, developing software projects for SGAE. To develop them, interviews to business experts were conducted, the requirements were specified (the functional ones were detailed by means of Use Cases and Activity diagrams, and the non-functional ones were documented in natural language), and the requirements were analysed for designing the system architecture. Nevertheless, recurrently, integration problems and misalignment with the business arose in the final development stages. Problems that were costly to solve in such phases.

With the aim of reducing the number of problems, we analysed their causes, noting that they mainly occurred because the used notations and artefacts focused on detailing the requirements, but they did not pay much attention to document the relationships between requirements. Essential information for designing correctly integrated and aligned systems. This situation was also detected by other researchers as an open issue, who proposed specific solutions for documenting some relationships, but did not detail a comprehensive approach for modeling and treat them from the business to the design stages.

This lack of solutions led us to start this thesis, which has the following goals:

- To define a process guiding in the modeling of the interdependencies.
- To design a notation for modeling the business interdependencies.
- To propose profiles for modeling the requirements interdependencies.
- To facilitate the analysis of the interdependencies in the architectural design.

- To simplify the derivation of the requirements and their relationships from the business to the design phase.

All these goals have been achieved in this thesis by defining a methodology focused on modeling and analysing these interdependencies. This methodology is supported, first, by an extension of BPMN and a set of profiles for Use Case and Activity diagrams for modeling the interdependencies at the business and requirements stages. In addition, we have proposed a metamodel to document the architectural impact of each non-functional requirement depending on its interdependencies with the functional ones. Finally, we have also defined a set of patterns and ATL transformations guiding and semi-automating the derivation of the requirements and their relationships from the business to the design stage. All the modeled information is provided to architects, facilitating the acquisition of knowledge on requirements, reducing the likelihood of misinterpretations and favouring the design of systems better integrated and aligned with the business.

The contributions of this thesis have been validated in three industrial projects. This validation has proved that they are feasible and their use produces more complete business and requirements models. Furthermore, it has also demonstrated that, although modeling these interdependencies entails additional effort in the early development stages, this effort is recovered during the architecture design. In the projects in which the proposed methodology has been applied, we got a ROI of 5,5% in the first three development phases. Moreover, a reduction of 12% in the number of integration bugs was detected, with respect to historical data of projects in which the methodology was not applied. Therefore, the validation shows that the application of the proposed methodology makes software development companies more competitive.

Resumen

La presente tesis comenzó a gestarse hace siete años en la factoría de software Teseo, desarrollando proyectos para SGAE. Para llevarlos a cabo, normalmente se realizaban entrevistas a los expertos del negocio, se especificaban los requisitos (los funcionales eran documentados mediante diagramas de Casos de Uso y de Actividad, y los no funcionales eran detallados en lenguaje natural), y se analizaban los requisitos para diseñar la arquitectura del sistema. No obstante, de forma recurrente, en las fases finales del desarrollo se detectaban problemas de alineación del sistema con los objetivos del negocio y de integración entre funcionalidades o con el entorno. Errores que en esas fases eran muy costosos de corregir.

Con el objetivo de reducir el número de problemas, se analizaron para identificar su origen, observando que principalmente se debían a que las notaciones y artefactos utilizados se centraban en detallar los requisitos, pero no prestaban mucha atención a documentar las relaciones entre requisitos. Información esencial para el diseño de sistemas perfectamente integrados y alineados con el negocio. Esta situación también había sido identificada por otros investigadores que, si bien proponían soluciones específicas para tratar determinadas relaciones, no detallaban soluciones integrales que permitieran documentarlas y gestionarlas desde el negocio hasta el diseño del sistema.

Dicha falta de soluciones fue lo que nos motivó a comenzar esta tesis con los siguientes objetivos:

- Definir un proceso que guíe en el modelado de las interdependencias.
- Diseñar una notación para modelar las relaciones entre elementos del negocio.

- Proponer perfiles para el modelado de las interdependencias entre requisitos.
- Facilitar el análisis de las interdependencias durante el diseño arquitectónico.
- Simplificar la derivación de los requisitos y sus relaciones desde el negocio hasta la fase de diseño.

Para satisfacer todos estos objetivos, en esta tesis se ha definido una metodología centrada en el tratamiento y análisis de dichas interdependencias. Dicha metodología es soportada, primero, por una extensión de BPMN y por un conjunto de perfiles para diagramas de Casos de Uso y Actividad que permiten modelar las interdependencias a nivel de negocio y de requisitos. Además, se ha diseñado un metamodelo para documentar el impacto arquitectónico de cada requisito no funcional a partir de sus interdependencias con las funcionalidades del sistema. Finalmente, se ha definido una serie de patrones y transformaciones ATL que guían y semiautomatizan la derivación de los requisitos y sus relaciones desde la fase de negocio hasta la de diseño. Toda esta información es proporcionada al arquitecto, facilitando la adquisición del conocimiento sobre los requisitos, reduciendo las probabilidades de que cometa errores de interpretación y propiciando el diseño de sistemas mejor integrados y alineados con el negocio.

La metodología presentada en esta tesis ha sido evaluada en tres proyectos industriales. Esta evaluación ha demostrado que todas las contribuciones son factibles y permiten generar modelos de negocio y de requisitos más completos. Además, también se ha demostrado que, si bien el modelado de estas interdependencias conlleva un incremento del esfuerzo en las fases iniciales, dicho esfuerzo es recuperado durante el diseño de la arquitectura. En los proyectos en los que se ha aplicado se ha llegado a obtener un ROI en las tres primeras fases del desarrollo de hasta el 5,5%. Además, se detectó una reducción de errores de integración de hasta un 12% con respecto a los obtenidos en proyectos en los que la metodología no fue aplicada. Por lo tanto, la validación corrobora que la aplicación de la metodología propuesta hace que las empresas de desarrollo puedan ser más competitivas.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

"There is only one thing that makes a dream impossible to achieve: the fear of failure."

Paulo Coelho

In this thesis, we detail a set of techniques and tools for designing systems better integrated and aligned with the business. To that end, the interdependencies between business elements and the relationships among requirements are modeled, treated and derived to the architectural design stage. The perfect understanding of these interdependencies is essential for designing a system correctly integrated and meeting all the requirements and business goals. These interdependencies provide to architects information on how the quality attributes affect the system functionalities. Information imperative for making architectural decisions complying with the requirements, and hence with the business goals.

This chapter introduces the research context, the problem faced and the proposed solution. In Section 1.1, first, the reasons that motivated this thesis are presented. In Section 1.2, we describe the research context and the scope of our work. Section 1.3 details the problem faced in the thesis. In Section 1.4, the main goals of the thesis are presented. In Section 1.5, the contributions of the thesis achieving the defined goals are described. Section 1.6 introduces some details about the context in which the thesis has been developed. Finally, in Section 1.7, we detail how the content of this thesis is organised.

1.1 Thesis Origins

The idea of this thesis arose in the course 2006/2007 from the work done in the software factory **Teseo**[1]. This was a factory inside the University of Extremadura that developed projects for the copyright society SGAE[2]. In these projects, the software development process followed was UP (*Unified Process*) (OpenUP, 2013). So, first, an analysis of the business was done, by means of interviews to the business experts, in order to elicit the system requirements. Then, all the requirements identified were modeled with Use Case and Activity diagrams, which subsequently were analysed by architects to acquire an accurate picture of the system and to design the architecture.

Although all the business information, requirements and designs were properly detailed using the correct models, recurrently in the final development stages integrations problems and misalignment of the system with the business were detected. Analysing the causes of these problems, we identified that the methodology and notations used were focused on documenting the characteristics of the requirements, but they did not pay attention in the modeling of the relationships between them. This lack of information required architects to deeply analyse the generated models and diagrams to identify these interdependencies in order to design the architecture. Nevertheless, some relationships were not identified, or were incorrectly identified by misinterpretations. This caused the integration and misalignment problems.

Once the origin of the problems was identified, we sought solutions that could be applied to solve them. We found that these problems were already observed by other researchers, who also proposed some partial solutions for them. However, these solutions were focused on documenting specific relationships at concrete development stages. No comprehensive solutions documenting and managing such relationships from the business to the system design stages were identified. The lack of these complete and comprehensive solutions is what motivated us to start this thesis.

[1] http://www.unex.es/investigacion/grupos/quercus/
[2] http://www.sgae.es/

1.2 Research Context

The development of a software application usually begins doing a detailed analysis of the business and the organization for which it is developed. This analysis allows engineers to identify its business goals and needs, its strategies and tactics and its business processes. Thus, this information on the goals, tactics and processes to be supported can be used to design a system aligned with them (Jackson, 1995; Grau, Franch, & Maiden, 2008; Cardoso, Almeida, & Guizzardi, 2009; Traetteberg & Krogstie, 2009).

Approaches such as BMM (2013) and EKD (Zikra, Stirna, & Zdravkovic, 2011) define different methodologies guiding in the analysis of an organization to get information on the business. These methodologies guide engineers in identifying, refining, and modeling the business goals and processes.

The business goals are usually modeled using *Goals* trees (E. S.-K. Yu, 1995). These trees allow engineers to detail high-level goals and to refine them into more specific and concrete objectives. In addition, they facilitate the documentation of different relationships between objectives; for example, to document whether two objectives are mandatory or alternative.

The business processes are usually detailed with workflow diagrams, such as UML Activity diagrams (UML 2, 2014) or BPMN (2014). These notations model what tasks are performed within a given process, the data exchanged between them, their sequentially, the people responsible for each task, or the relationships with other processes or sub-processes. BPMN is currently one of the most widely used notation because of its ease of use for both business experts and requirements engineers.

The artifacts documenting business goals and processes are subsequently used as the basis to identify the functional and non-functional requirements of the system to develop (Alexander & Stevens, 2002; Lauesen, 2002; van Lamsweerde, 2004). Thus, it is easier to identify and document system requirements aligned with the business. The models detailing the business goals are usually analysed to identify the quality attributes, or the non-functional requirements, of the system. These

attributes detail the restrictions that the system have to meet to provide the desired quality. From the business process models, the system functional requirements or Use Cases are identified. These requirements specify the tasks and the actions that the system have to support to achieve the desired functionality.

As in the business analysis, the identified requirements are detailed and modeled with specific artifacts and notations (Chung & do Prado Leite, 2009). Thus, following for example the *UP (Unified Process)* (OpenUP, 2013) software development process, the functional requirements are detailed in the *Vision Document* and in the *Use Case Model*, using UML Use Case and Sequence or Activity diagrams for detailing the actions that have to be done by each functionality. The non-functional requirements are detailed in the *Supplementary Specification Document*, detailing in natural language the system quality attributes.

Subsequently, in the system design phase, all the information on the business and on the requirements must be consolidated and analysed. (Verner, Cox, Bleistein, & Cerpa, 2005; Capilla, Ali Babar, & Pastor, 2012). Architects need to fully know the system functional and non-functional requirements, and the interdependencies between them in order to make architectural decisions for designing a system achieving the greatest number of requirements, perfectly integrated and aligned with the business.

The interdependencies are the objectives or actions defined in a requirement that interact with another requirement or that indicate how its actions should perform. The requirements should not be treated independently, since they usually are related and affect each other (Regnell et al., 2001). These relationships are highly important, since they can be used to manage the changes in the requirements (Kotonya & Sommerville, 1998), to plan the releases (Carlshamre, Sandahl, Lindvall, Regnell, & Natt och Dag, 2001) or to design the system architecture (Shakil Khan, Greenwood, Garcia, & Rashid, 2008).

Specifically, in the architecture design stage, the requirements and their interdependencies are analysed in order to assess the impact and importance of each requirement and, thus, identify the Architectural Significant Requirements (ASRs)

driving the architecture design. These are the requirements (i.e. functional or non-functional) that are expected to have architectural level consideration and addressing. In addition, it is vital that architects have in-depth understanding of not only the ASRs but also their interdependencies in order to make appropriate design decisions by selecting suitable patterns and tactics. A lack of or missing information about requirements' relationships may lead to a poorly integrated system and to a pattern being selected that can hinder rather than facilitate the satisfaction of the ASRs. Inappropriate selection and application of patterns usually results in a system that is unlikely to meet the business requirements. That is why it is vital that architects have a good understanding of all the requirements and their relationships.

Since both at the business level and at the requirements level each kind of business element or requirement is detailed in specific artifacts and with specific notations, the interdependencies between different kind of elements are usually kept implicit, that is, hidden in the graphical models and textual descriptions (Dahlstedt & Persson, 2003). Therefore, it must be the architects the ones who make explicit the whole network of relationships. The identification and understanding of these interdependencies is a complicated activity (Xu, Ziv, & Richardson, 2005). To correctly identify them, architects need to have a great experience and ability to analyse all the artifacts generated in the business and requirements analysis phases.

The aim of this thesis is to make explicit these interdependencies from the early software development stages in order to facilitate the design of systems perfectly integrated and aligned with the business with less effort. To that end, first, the contributions presented in this dissertation make their modeling in the early development phases possible. Second, they semi-automate their derivation to the successive development phases. Finally, they facilitate their analysis by architects for the architectural decision making.

1.3 Problem Statement

As we stated above, due to the use of specific notations and the documentation of each kind of element in concrete artifacts, there are several difficulties for model-

ing the interdependencies between business elements and the relationships between requirements (Dahlstedt & Persson, 2003). These interdependencies are essential for the proper design of a system. If they are not correctly treated the likelihood of poor integration between functionalities, of requirements not correctly achieved and of misalignment with the business is increased. To overcome this problem, new techniques documenting these interdependencies from the initial development stages and deriving them to the system design are necessary. This is the problem addressed by this thesis.

Below, the most important interdependencies existing at the business and requirements levels, that currently cannot be modeled, are detailed. Also, the approaches addressing the documentation of some of them are also specified.

In the business analysis stage, the business goals are detailed with a goal tree, while the business processes are documented using workflow diagrams (i.e. BPMN) (Ullah & Lai, 2011). Natively, engineers cannot model what business processes, or parts of the processes, are restricted by certain business goals. These interdependencies kept implicitly detailed. Therefore, they have to be identified later by architects for the architectural design.

Currently, there are some approaches making these interdependencies explicit. In (Aburub, Odeh, & Beeson, 2007), the authors link business restrictions with business processes in order to identify areas of the processes that need to be improved to meet the goals. In (Pavlovski & Zou, 2008), the authors define how to model business operational constraints in business processes in order to be able to define controls to manage these constraints. However, in order to model a greater number of interdependencies between business goals and business processes, it would also be desirable to be able to model the relationships with non-operational restrictions.

In the requirements analysis stage, the functional requirements are specified in the *Vision Document* and in the *Use Case Model* using UML, while the non-functional requirements are detailed in the *Supplementary Specification Document* using natural language. These requirements are identified analysing the business information. Thus, it is easier to identify them and in turn are also better aligned with the busi-

ness specification (Kearns & Lederer, 2000; de Leede, Looise, & Alders, 2002). This alignment lead to benefits such as: a greater business performance (Cardoso et al., 2009), technology used strategically (Singh & Woo, 2009) and IT systems that can evolve at the same pace as the business (Chan, Huff, Barclay, & Copeland, 1997).

Approaches, such as the ones detailed by Dijkman and Joosten (2002), Stolfa and Vondrak (2006) o Siqueira and Silva (2011), propose mechanisms facilitating the derivation of functional requirements from business processes. These approaches, although reduce the effort required to identify the functional requirements, only identifies some of the relationships between functional requirements. There are situations (e.g. exception flows in the business processes or the merging or splitting of flows through gateways) and relationships (e.g. *include* or *generalization*) that are not identified by these proposals. This means that engineers still have to carefully analyse the business information in order to detect these situations and the functional requirements derived from them. Also, these approaches do not address the relationship between business goals and business processes, nor derive them to the system requirements. Therefore, to get a requirements specification better aligned with the business, it would be desirable to address more situations and to get these relationships.

In addition, natively with UML, engineers cannot model what Use Cases, or parts of the Use Cases, are restricted by a specific quality attribute or non-functional requirement. Approaches, such as IESE-NFR (Dörr, 2011) and ICRAD (Herrmann, Paech, & Plaza, 2006), improve the requirements specification in order to facilitate the modeling of the interdependencies between functional and non-functional requirements. IESE-NFR, for example, allows requirements engineers to include notes in the Use Case diagrams detailing in natural language the quality attributes affecting each Use Case. However, these notes, first, are oriented to check if the quality attributes are satisfied in a specific architectural design and, second, since they are detailed in natural language, they cannot be reused by tools that assist architects in the subsequent development phases. It would be desirable that the requirements specifications, along with their interdependencies, can be reused in successive development phases, such as during the system architecture design (Verner et al., 2005).

Finally, if these interdependencies are not documented, or are partially documented, they have to be identified in the architecture design phase. Architects need to deeply know the business goals and processes, the functional and non-functional requirements, and the interdependencies between them in order to identify the ASRs and to make architectural decisions meeting the greater possible number of requirements. If these relationships have to be manually identified by architects, there is a higher risk of eliciting incorrect interdependencies that in turn involve making inadequate architectural decisions, jeopardizing the achievement of the business goals and needs.

Currently, there are several approaches facilitating specific activities of the architecture design process (Avgeriou, Grundy, Hall, Lago, & Mistrík, 2011). Works such as (Bachmann, Bass, Klein, & Shelton, 2005) and (Kim, Kim, Lu, & Park, 2009) document the different patterns and how they affect each non-functional in order to guide architects in the selection of the most suitable patterns. For these techniques to be fully effective, architects needs to provide them the ASRs of the system and their interdependences. Likewise, their identification requires architects to perform a deep analysis of the requirements, and their relationships, in order to have a complete picture of the system. However, any misinterpretation made can be transmitted to these tools, which in turn can lead to select inappropriate patterns, hindering the achievement of the requirements.

This thesis aims to overcome the limitations and problems presented above. The following are the main issues addressed:

- There are interdependencies between business goals and business processes that currently cannot be modeled.
- The relationships between functional and non-functional requirements cannot be documented to be reused in the subsequent development phases.
- There are interdependencies between requirements that are not semi-automatically derived from the business information.
- The identification of the relationships in the architecture design stage increase the risk of misinterpretations.

1.4 Thesis Objectives

The main goal of this thesis is to define a process, and the tools supporting it, to improve the alignment of the IT systems with the business by means of, first, modeling the interdependencies between business elements and among requirements and, second, facilitating the reuse of this information for designing the architecture. To achieve this goal, we identified a set of sub-goals specific for each development phase and focused on solving the issues outlined in the previous section.

For the business analysis stage, the sub-goals defined for increasing the number of interdependencies between business elements that can be modeled are the following:

1. Identify what interdependencies between business elements, and useful for the architecture design, currently cannot be modeled.
2. Extend the existing notations for documenting such interdependencies.
3. Develop tools supporting the defined extensions and facilitating the modeling of the interdependencies.

For the requirements analysis phase, the following are the sub-goals defined for augmenting the number of interdependencies between requirements that can be modeled:

4. Identify what interdependencies between requirements, and important for the architecture design, cannot be currently modeled with UML.
5. Define profiles extending the existing notations for documenting such interdependencies.
6. Develop tools, or extend the existing ones, for supporting the defined profiles.

In addition, for the requirements stage, the following are the sub-goals detailed with the aim of increasing the number of relationships between requirements that can be derived from the business:

7. Identify what interdependencies currently are not automatically derived from the business information.

8. Define patterns guiding in the derivation of such relationships.

9. Develop rules and transformations automating the application of the patterns.

Finally, for the architecture design phase, the sub-goals defined with the aim of facilitating the analysis of the interdependencies and reducing the risk of misinterpretations are the following:

10. Identify the shortcomings that existing approaches assisting architects have in the evaluation of requirements and their interdependencies.

11. Design notations, or extend the existing ones, to document the results of the requirements analysis.

12. Define a methodology guiding architects in the reutilization of the interdependencies for the architectural decision making, and integrate it with the existing approaches assisting architects.

1.5 Thesis Contributions

The goals defined for this thesis have been met throughout its implementation, as the following sections show. As result, this thesis provides a number of contributions for modeling and analysing the previously defined interdependencies. Specifically, we have defined a methodology guiding in the documentation of the interdependencies between business elements and between requirements, and facilitating their reuse for the architectural decision making. Figure 1.1 details the complete picture of the contributions provided by this thesis. These contributions are numbered and highlighted in blue.

The initial part of this methodology, and the first contribution of this thesis, is a modification of the software design process (contribution 1 of Figure 1.1) adding three new activities to it focused on the documentation and use of these interdependencies. The first activity, *Model Contex Information*, guides business experts and requirements engineers in the modeling of the interdependencies between business elements (i.e. between business quality goals, business processes and legacy sys-

tems). The second one, *Define Requirements Interdependencies*, drives requirements engineers in the modeling of the relationships between requirements (i.e. between functional and non-functional requirements). Finally, the third activity, *Analyse Requirements Interdependencies*, guides architects in the reutilization of the previously modeled information to facilitate the architectural decision making. This process was published in (Berrocal, García-Alonso, & Murillo, 2013) (Core C) and (Berrocal, Garcia Alonso, & Murillo, 2014).

This process is supported by a suit of extensions, profiles, metamodels and tools allowing engineers to make these interdependencies explicit from the first development stages. In addition, we have defined a set of patterns and rules semi-automating the derivation of the requirements and their interdependencies from the business models, facilitating the definition of requirements models aligned with the business. Below, the specific contributions defined for each development stage are detailed.

In order to document the interdependencies in the business analysis phase, an extension of BPMN has been defined. This extension allows one to model the relationships between business processes and other elements of the organization (such as business goals or legacy systems) or useful information for the subsequent development phases (such as the business tasks that will be supported by each system functionality). In addition, with the aim of facilitating the modeling of the elements defined in this extension, a modification of the Eclipse plugin BPMN2 Modeler was implemented. Both the extension and the plugin supporting it (contributions 2 and 3 of Figure 1.1) were published in (Berrocal, Garcia Alonso, Vicente Chicote, & Murillo, 2014) (JCR).

In order to allow and facilitate the modeling of the interdependencies in the requirements analysis stage, a set of profiles were defined for UML Use Case and Activity diagrams (contributions 5 and 6). This profiles, which have been implemented as an extension of the Eclipse plugin Papyrus (*Eclipse Papyrus*, 2014) (contribution 7) allow requirements engineers to model the interdependencies between functional and non-functional requirements. Both the profiles and their implementations have been published in (Berrocal, Garcia Alonso, Vicente Chicote, & Murillo, 2014) (JCR). In addition, the identification and the modeling of the requirements and their interde-

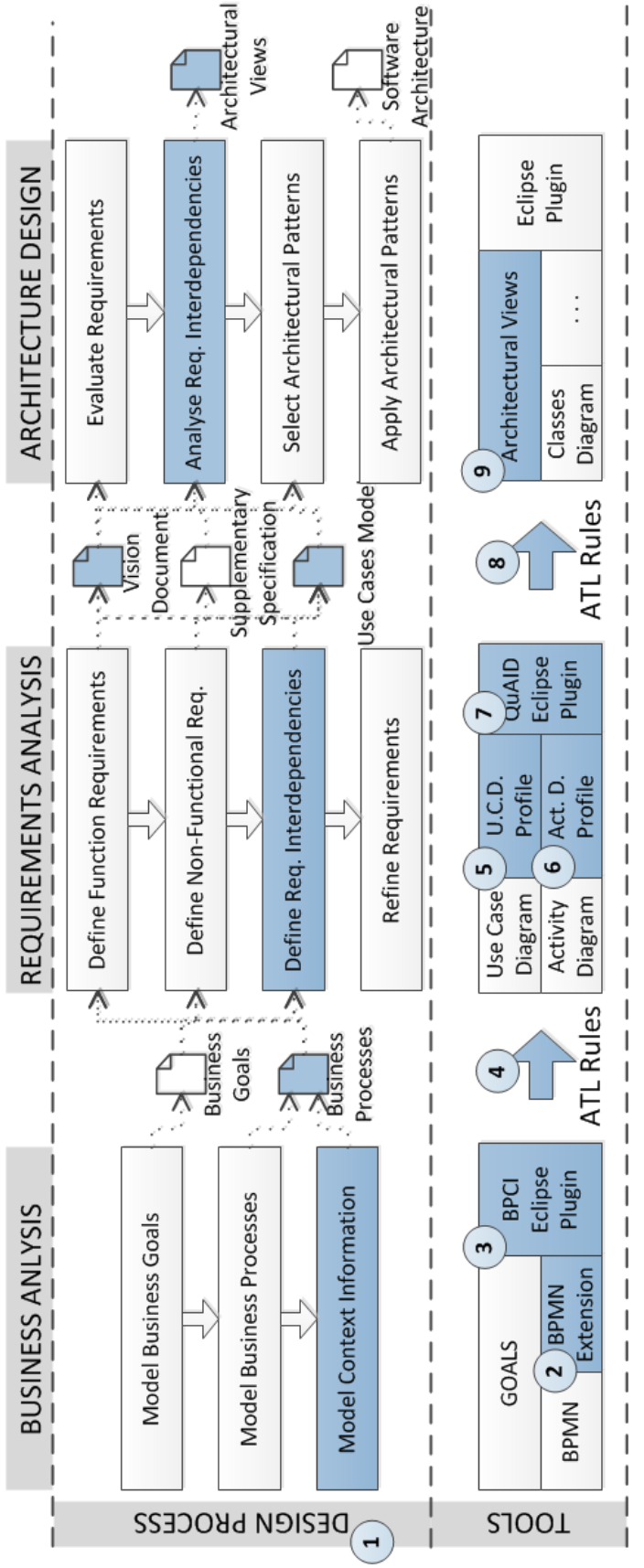


Figure 1.1: Overview of the thesis contributions

pendencies is guided by a set of patterns. These patterns define the most common situation in the annotated business processes usually leading to the definition of Use Cases and relationships between them or with non-functional requirements. The application of these patterns has been automated by means of Model Driven Development techniques (contribution 4 in Figure 1.1). The defined patterns and ATL transformations were published in (Berrocal, García-Alonso, Vicente-Chicote, & Murillo, 2013) (Core A). Thanks to these patterns and rules, requirements models more complete and better aligned with the business are obtained.

To further facilitate the analysis of the interdependencies in the architecture design phase, a metamodel for documenting the different architectural views and how each quality attribute impact on each view was defined (contributions 9). The generation of these models is semi-automated by means of ATL transformations based on the annotated requirements models (contribution 8 in Figure 1.1). Thus, it is easier for architects to get the impact of each quality attribute and identify the ASRs. This contribution was published in (Berrocal, Garcia Alonso, & Murillo, 2014). In addition, through the profiles defined for Use Case and Activity diagrams, architects can easily analyse the interdependencies of the ASRs with other requirements, facilitating the requirements analysis and the architectural decision making. The process reusing these interdependencies for the architectural decision making was published in (Berrocal, García-Alonso, & Murillo, 2010a) (Core A).

The new activities incorporated to the software design process, along with the extensions, profiles and tools supporting them, enable and guide in the modeling of the interdependencies, and in their use during the architectural decision making, facilitating the design of IT systems better integrated and aligned with the business.

1.6 Thesis Impact

This section details the impact of this thesis in relation with the context in which it has been developed, the projects in which the author of this thesis has participated, the papers published in different forum (along with the importance of each forum) and the research stays conducted.

1.6.1 Context

As Section 1.1 details, this thesis is motivated by the integration and misalignment problems identified during the development of projects in the software factory Teseo for the company SGAE.

Once the problems and the lack of solutions were identified, this thesis started and was developed in the environment of the **Quercus Software Engineering** research group[3] of the University of Extremadura[4], where the author hold a position of assistant professor. Quercus is one of the bigger and more important research group of this university. In addition, it participates in a number of project with both private and public funding. In fact, it has been recognized as the research group in the University of Extremadura that more private funds attracted in the years 2011 and 2012.

Finally, in the development of this thesis, the company **Gloin**[5] was co-founded by the author of this thesis, the supervisor and another research partner. The most important goals of Gloin are to apply the research knowledge and advances done in the research group, and to help other companies to apply them. Thus, the techniques and tools detailed in this thesis have been successfully applied by the company to different projects.

1.6.2 Research and Industrial Projects

This section details the different projects in which the author of this thesis has been involved and in which some of the contributions of this thesis have been developed. The following are the national research projects in which the author has participated:

- **Model-driven development of business process in software factories: applications to the Web 2.0 and J2EE multi-tier architectures. (TIN2008-02985).** In this project, the author of this thesis defined a set of patterns for deriving Use Cases and the relationships between them from busi-

[3] http://www.unex.es/investigacion/grupos/quercus
[4] http://www.unex.es
[5] http://www.gloin.es

ness process models. The papers (Berrocal, García-Alonso, & Murillo, 2009, 2010c) details some of the results of this work.

- **Updating the development of framework based on multi-layer applications: Integrating MDD and PL techniques.(TIN2011-24278).** In this project, we delved into the reuse of the interdependencies between requirements for the selection of architectural patterns facilitating their compliance. This selection process was published in (Berrocal et al., 2010a).

- **PATTERN: Product line And Transformation TEcniques in the multi-LayeR architectures desigN. (TIN2012-34945).** In the context of this work we developed some techniques to identify the Architectural Significant Requirements guiding the design of the system architecture. Some of these techniques were published in (Berrocal, García-Alonso, & Murillo, 2013).

The regional research projects in which the author has participated are the following:

- **JACA: Java for the regional government applications. (PDT08A034).** In this project, the author colaborated in the definition of a development environment for the regional goverment in which developer can easily select the development framework to use in a software project depending on the non-functional requirements or quality attributes to be met. This work led to publications such as (Berrocal et al., 2010a).

- **Product lines in the development of multi-layer applications.(ACVII-08).** In this project, the author of this thesis contributed defining a process guiding in the definition of an initial configuration of a product line (detailing the different architectural patterns that are usually applied in a company, and their variations) from the requirements interdependencies information. Some parts of this process were published in (Berrocal et al., 2010a; Berrocal, García-Alonso, & Murillo, 2013).

- **Methodology for the extraction of Use Cases from business processes. (ACVII-09).** In this project, we dug in the definition of patterns for the derivation of Use Cases and their relationships (with other Use Cases or

with non-functional requirements) from business process mdoels. As result of this project, we obtained the following publications (Berrocal, García-Alonso, & Murillo, 2013; Berrocal, García-Alonso, Vicente-Chicote, & Murillo, 2013).

The following are the industrial projects and the agreements of collaboration with companies in which the author of this thesis has participated. The main goal of these projects was to transfer of the results of different researches to companies.

- **GEPRODIST. Distributed projects management.** In this project, we made the analysis and design of a system for Indra[6] that facilitates the management of distributed software projects. To that end, this system takes as input the software process to follow (modeled as a business process) and the business processes that have to be supported by the system to develop. From this information, GEPRODIST guides the different roles in the realization and management of the tasks that have to be done. As result of this project, the following publications were made (Berrocal, García-Alonso, & Murillo, 2007, 2008, 2010b).

- **GLOCO. Global Connector.** In this project we colaborated with Gloin in the development of a system connecting different information sources of Latin-American Digital Copyright Societies. The methodology detailed in this thesis was applied in this project for the definition of the societies' business processes and the derivation of the system Use Cases from them.

- **DCI. Data Clean-up and Integrity.** This project involved the development of an application supporting a set of business processes assesing the data quality of the musical repertoires of the Digital Copyright Societies. To develop this system, the contributions presented in this thesis were used. Section 4 details how they were applied and the results obtained.

- **BeeFun.** In this project, we colaborated with Gloin in the development of a messaging app for mobile devices that can send messages to a group of users based on information captured by their mobile device (i.e. their positions, preferences, etc.). This app was also developed using the methodology presented

[6] http://www.indracompany.com/

in this thesis. Section 4 details how it was applied and the results obtained.

- **NimBees.** NimBees is an API developed by Gloin that enables mobile devices as cloud providers, in which services providing information on the data registered by the sensors of the device can be deployed. To develop this API, the contributions here presented were used. Section 4 details how they were applied and the results obtained.

1.6.3 Summary of publications

Table 1.1 shows a summary of the forums in which papers have been published, the number of publications in each forum and its importance. The complete list of the papers published in the context of the thesis is detailed in Section 5.2. The importance of the congress is obtained from the *Computing Research and Education Association of Australasia (CORE)*[7]. The importance of the journals is obtained from the *Journal Citation Report (JCR)*[8].

As can be seen in Table 1.1, in the development of this thesis, its author has published in total thirteen papers, of which five have national scope and eight have international scope. Ten of these papers were published in conferences/workshops, of which four were accepted in conferences indexed in the CORE ranking. Finally, the other three papers were published in journals, of which two were indexed in JCR, with an impact factor of 0,218 and 1,616.

1.6.4 Collaborations

In the development of this thesis, a five-month research stay was done in the IT University of Copenhagen. This university is considered a top institution on system and software engineering

This research stay was supervised by Prof. Muhammad Ali Babar. The main research areas of Prof. Muhammad Ali Babar is to develop and/or rigorously evaluate approaches and tools for supporting the design, analysis, and evolution of complex

[7] http://core.edu.au/index.php/categories/conference%20rankings
[8] http://thomsonreuters.com/journal-citation-reports/

Table 1.1: Summary of the published papers, together with the foum relevance.

Forum	**Kind**	**Scope**	**Num**	CORE	**JCR**
PNIS	Workshop	Nat	3	-	-
JISBD	Conference	Nat	1	-	-
ECSA	Conference	Int	1	A	-
SERA	Conference	Int	1	C	-
ISD	Conference	Int	1	A	-
REMIDI	Workshop	Int	1	-	-
PROFES	Conference	Int	1	B	-
Agile-Spain	Conference	Nat	1	-	-
IEEE Latin Transactions	Journal	Int	1	-	0,218
IEEE Software	Journal	Int	1	-	1,616
IJSI	Journal	Int	1	-	
Total	**4 Works. 6 Conf. 3 Journal**	**5 Nat. 8 Int.**	**13**	**4**	**2**

and dependable software intensive system and services that meet both the functional and non-functional requirements as derived from the quality goals.

As a result of this stay, and thanks to the experience and contributions of Prof. Muhammad Ali Babar, the use of the interdependencies between requirements for identifying the Architectural Significant Requirements was improved. In addition, the integration of the thesis outcomes with approaches guiding the architectural decision making was refined.

1.7 Thesis Structure

The rest of this thesis is organized as follows:

Chapter 1: Introduction. It comprises this introduction, which contains the thesis origins, the research context of the work developed needed to understand the content of this thesis, a detailed statement of the most common problems in such context and the goals to be achieved with this work. Additionally, it contains a summary of the contributions of the thesis and their impacts.

Chapter 2: State of the Art. This chapter reviews the most important approaches related with this thesis. Specifically, it details the approaches modeling the inter-

dependencies between business elements, the works documenting the relationships among requirements and the proposals basing the architectural decision making in the requirements analysis.

Chapter 3: Modeling Business and Requirements Interdependencies. This chapter is the core of this thesis since it describes the main ideas and contributions. First, it specifies the process for modeling the interdependencies and their further reuse in the architectural making process. Then, it details the contributions for modeling the business interdependencies, for deriving functional requirements models annotated with the interdependencies with the non-functional ones, and for the identification of the Architectural Significant Requirements. Finally, this chapter presents how the outcomes of this thesis integrates and facilitates existing approaches guiding the architectural decision making.

Chapter 4: Validation of the Proposal. This chapter presents the validation of the ideas and contributions of this thesis based on their application to three real industrial projects. The contributions have been validated according to three characteristics: their feasibility, their completeness and the effort required to apply them to the three projects.

Chapter 5: Conclusions and Future Works. This chapter presents the main conclusions drawn after developing this thesis, specifies the papers written during its realization, details some future works, and finalizes with a final remark.

Appendix A:BPCI Extension. This appendix shows the complete extension of BPMN, detailing the new objects defined for modeling the contextual information.

Appendix B: UC-QuAID Profile. This appendix details the complete profile defined for modeling the interdependencies between Use Cases and non-functional requirements.

Appendix C: Patterns for Deriving Functional Requirements and Interdependencies. This appendix presents the defined patterns guiding the derivation of functional requirements, and their interdependencies with non-functional requirements, from the information annotated in the business processes of the organization.

Appendix D: ATL Transformations for the Derivation of Annotated Requirements Models. This appendix shows the ATL transformations automatizing the generation of Use Case diagrams annotated with the interdependencies between Use Cases and non-functional requirements.

Appendix E: ATL Transformations for the Derivation of the Architectural Views. This appendix details the ATL transformations defined for automatically generate, from the annotated Use Case diagrams, the model documenting the impact of the quality attributes on the Use Case View.

Chapter 2

State of the Art

"I hear and I forget. I see and I remember. I do and I understand."

Confucius

Designing a system aligned with the business requires the synchronization of a great number of activities and stages, such us the enterprise modeling, the requirements modeling and the system architecture design. This thesis is focused on improving the business-IT alignment enhancing the specification of interdependencies. Therefore, this chapter review how these relationships are currently addressed at each development stage.

The following sections present the approaches that either are the most relevant ones at each stage or are highly related to the work presented in this thesis. Section 2.1 introduces this chapter and the discipline in which this thesis is framed, which is Business Driven Development. Section 2.2 specifies a running example used to illustrate the approaches reviewed and the contributions of this thesis. Section 2.3 details different techniques for analyzing and modeling the business information. Section 2.4 analyzes the notation for modeling and deriving requirements aligned with the business. Finally, Section 2.5 specifies some approaches focused on reusing the requirements information for design the architecture.

2.1 Introduction

Businesses face constant changes in their environment. Therefore, they have to shift their strategies and adapt their business processes to reflect market trends and competition. In order for this adaption to provide the maximum benefit, the IT systems supporting the business should be able to adapt at the same rate.

Business Driven Development aims at developing IT implementations that directly satisfy business requirements (Koehler et al., 2008). BDD requires that "IT efforts are interlocked with business strategy and requirements"(Mitra, 2005). BDD states the documents and artifacts generated as a result of the business analysis should be actively used for specifying and designing the IT systems that will support the business. If there is a mapping between business requirements and IT systems, these systems will be more responsive and will agilely adapt to the business changes.

BDD is a holistic approach based on the following actions:

1. Identification and modeling of business processes and tasks.
2. Analysis of the business information in order to derivate the IT system requirements.
3. Design an appropriate IT solution aligned with the business and system requirements.
4. Development and deployment of the IT systems.
5. Monitorization and analysis of the IT solutions for identifying improvement both on the business processes and in the IT systems.
6. Adoption of improvements both on the business and the IT level.

Developing IT systems aligned with the business using the BDD methodology ensures that these systems enhance the business performance (Cragg, King, & Hussin, 2002), that technology is used strategically for improving the business (Singh & Woo, 2009) and that the IT systems and the business can evolve at the same rate (Chan et al., 1997).

This thesis gives support to the first, second and third actions of the BDD methodology. Currently, there are different methodologies and notations supporting these actions, or part of them. Figure 2.1 shows an scheme of the most important approaches, related with this thesis, that support them. These approaches have been reviewed in order to identify the benefits they provide and the different areas that could be improved to design IT systems better aligned with the business and with less effort. The following sections detail the results of these reviews.

This thesis focuses on enhancing these areas. Specifically, it details how to model the interdependencies between business elements; how to derive the system requirements and the relationships between them from the business model; and how to reuse these interdependencies to facilitate the design of the system architecture aligned with the business.

2.1.1 BDD and MDD

Model-Driven Development (MDD) is a software development methodology which focuses on creating and exploiting models (that is, abstract representations of the knowledge of a particular application domain) as central artifact along the entire software development process (Stahl, Voelter, & Czarnecki, 2006). MDD is meant to increase productivity by maximizing the reuse of models, simplifying the process of design, and promoting communication between teams working on the system.

Model-Driven Architecture (MDA) is an approach to the MDD methodology proposed by Object Management Group (OMG, 2014). It provides a set of guidelines for structuring the models specified in the software development process. Specifically, this approach defines three different tiers: CIM (Computational Independent Models), PIM (Platform Independent Models) and PSM (Platform Specific Models). Each of these tiers defines models at different abstraction level, so that for example the same models detailed at the PIM level can be used to develop systems with different platforms. Moreover, this approach proposes the use of transformations for the generation and creation of more specific models from the abstract ones.

In this sense, BDD follows the MDA approach (Fazal-Baqaie, 2009). In the CIM

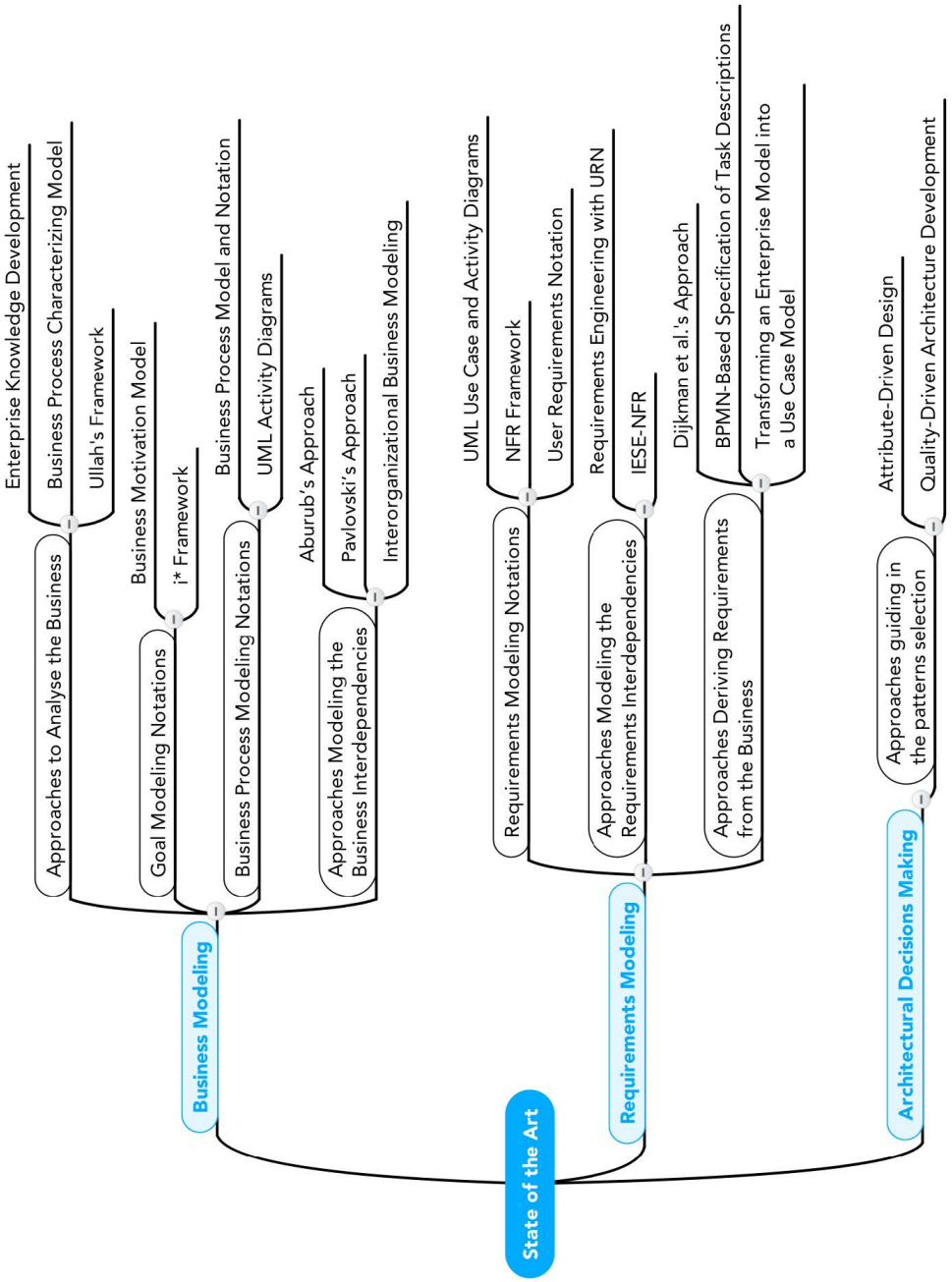


Figure 2.1: Scheme of the approaches reviewed in this thesis.

tier are the models detailing the business information and requirements. And in the PIM and PSM tiers are the models documenting the system architecture and design aligned with the business information.

The work developed in this thesis is framed on the BDD and MDA disciplines. Concretely, it allows engineers to document the interdependencies between the elements modeled in the CIM tier and facilitates their analysis and derivation to the PIM tier in order to design systems better integrated and aligned with the business.

2.2 Running Example

The following sections detail the approaches that partially or totally support the actions of the BDD methodology covered by this thesis. With the aim of better detail them, and their contributions, a running example is used. This running example is also used to illustrate the contributions of this thesis.

This example is a e-commerce solution to sell products via electronics media to customers nation-wide. This business has the following goals:

- *Get Online Orders.* The system should be able to get orders through a web portal and manage its validation and preparation.
- *Easy to Adapt to Changes.* The system should be easy to adapt to modifications in the workflows or in the business rules.
- *Increase the customer satisfaction.*
  - The orders must be validated in less than 0.7 seconds, since different studies done by the company indicate that this is the acceptable waiting time for users.
  - The information on the customers and their orders is private. Therefore, the access to these data must be secure.
- *Integration with external systems.* The management of the warehouse stock is done by means of legacy systems. The new system should integrate with these applications.

The most important business process that must be supported by the new system is *Place Online Order*, that is used to illustrate the thesis contributions. This process handles an order made by a customer, validating it, showing products related with the ones ordered, checking the product availability and managing its preparation and payment.

Once the organization goals and business processes have been identified, analysing them the business analyst can obtain information on the goals motivating each process (for example, the goal *Get Online Orders* motivates the process *Place Online Orders*) or the goals affecting and/or restricting each process or part of the process (for example, the goal concerning the time behaviour for checking an order restrict a part of the process *Place Online Order*). At business analysis level, it is relatively simple to identify these relationships. However, if they are not modeled or documented, as the system development progresses their identification in the subsequent development stages gets complicated, or may even be unidentified, increasing the likelihood of designs with issues that usually are not detected until the integration and deployment stages. These are drawbacks that this thesis intended to prevent.

2.3 Modeling the Business Information

The analysis and modeling of the business information (Enterprise Modeling) is the set of activities aimed at obtaining and documenting the knowledge of an organization (Vernadat, 1996). Normally, during this analysis, the company vision, its strategies, objectives and/or processes are identified and modeled. Enterprise modeling is usually used to capture the purpose of a system, by describing the behavior of the organization in which that system will be deployed (Loucopoulos & Kavakli, 1995).

With the aim of facilitating and guiding in the business analysis, nowadays different techniques propose activities and the use of specific notations for the identification and documentation of the business information. This section focuses on introducing these techniques and the most common notations used for modeling the business goals and processes.

2.3.1 Approaches to Analyse the Business

The business analysis is a vital activity in the early phase of an IT system development process. It is vital to gather information on the organization, such as its vision, mission, strategies, tactics, goals and/or processes.

Correctly identifying and analysing an enterprise requires a number of activities and conversations between business experts and IT engineers. A good understanding of the business domain is key for the IT project success (Curtis, Krasner, & Iscoe, 1988). In this sense, in the last few years some techniques have been defined guiding the analysis of the enterprise. The most significant techniques are EKD (Enterprise Knowledge Development) (Bubenko, Brash, & Stirna, 1998), Business Process Characterizing Model (BPCM) (Gao & Krogstie, 2009) and Ullah's framework (Ullah & Lai, 2011)

Enterprise Knowledge Development (EKD)

EKD is an approach that provides a systematic and controlled way of analyzing, understanding, developing and documenting an enterprise and its main elements, by using Enterprise Modelling (Bubenko et al., 1998).

The purpose of applying EKD is to provide a clear, unambiguous picture of how the enterprise works currently, what are the requirements and the reasons driving the aimed change, what alternatives could be devised to meet these requirements and what are the criteria and arguments for evaluating these alternatives.

The EKD framework is based on two main cornerstones: a set of guidelines for analysing the enterprise information, and a set of techniques for describing the analysed information.

The guidelines define a process for diagnosing and understanding the enterprise with the aim of being able to discuss and define the objectives, tactics and strategies of the company. They define different approaches to gathering domain knowledge. Common approaches are interviews with domain experts, analysis of existing documentation, observation of existing work practices, and facilitated group modeling.

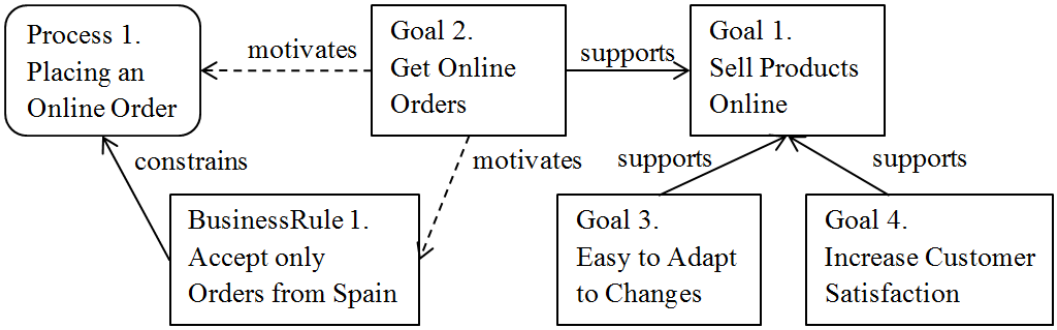


Figure 2.2: An excerpt of the model generated applying the EKD approach to the online-shop running example.

The set of description techniques provides a set of models used for describing the organisation information. Some of the most important models are: Goals Model (focused on describing the goals of the enterprise), Business Rule Model (which is used to formulate the business rules), Business Process Model (used to define the enterprise processes), Actor and Resources Model (which describe how different actors and resources are related to each other), and Technical Components and Requirements Model (defining requirements of the information system).

In addition, they define *inter-model relationships* in order to be able to trace decisions, components and other aspects throughout the enterprise. These relationships, for instance, allow one to indicate if a goal, in the Goals Model, motivates a particular process, in the Business Processes Model. However, one cannot define if a process, or a part of it, is restricted by a specific goal. Consequently, some links are missed between these models that are especially important in the future development phases to identify if a specific functionality is restricted by a quality restriction.

With the aim of using this models and inter-relationships in a MDD process, in (Zikra et al., 2011) the authors propose a metamodel to represent them. Thus, these models can be re-used in the system design. However, this metamodel neither allows one to document restrictions of a goal over a process, it only facilitates to detail if a business rule (which can be motivated by a goal) constrains a process.

Applying this approach to the online-shop running example, one would obtain a set of model and inter-relationships. Figure 2.2 shows an excerpt of the model

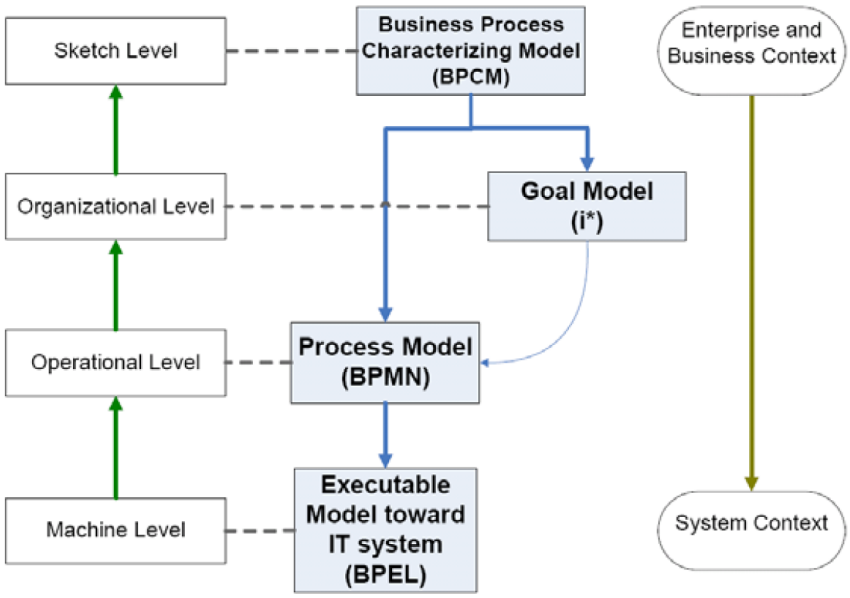


Figure 2.3: BPCM Framework

documenting the inter-relationships between goals, processes and business rules. As can be seen, the achievement of the main goal is supported by the subgoals *Get Online Orders*, *Easy to Adapt to Changes* and *Increase the Customer Satisfaction*. In turn, the goal *Get Online Orders*' motivates both the business process *Placing an Online Order* and the business rule *Accept only Orders from Spain*. However, one can only model that the defined rule constrains the business process, but cannot document if any goals, such as *Easy to Adapt to Changes*, also affects the process. This means that these relationships have to be identified during the architecture design at the risk of misinterpretation, or even they are forgotten, which may lead to design a system that is not easy to adapt or do not provide the desired satisfaction to customers.

Business Process Characterizing Model (BPCM)

BPCM is a technique developed by Gao and Krogstie (2009) to support sense-making and communication between business experts and model developers. It is especially used to document the key attributes and elements involved in a business domain. Based on the detailed characteristics, this approach guides and supports the modeling of goals and business processes.

An image of the proposed framework is shown in Figure 2.3 (extracted from (Gao & Krogstie, 2009)). As can be seen, the BPCM is the cornerstone to identify and anlayse the business information. This model is a textual table detailing per each business process the resources, actors, concepts, and goals related to it. Once defined, BPCM is used for deriving both the goal models, specified with i* (E. S.-K. Yu, 1995), and the process models, detailed with BPMN (BPMN, 2014).

The goals model is defined at the organizational level based on the information detailed in the BPCM. In this context, goals are defined as statements to declare the states to be achieved. In connection to the process model at the operational level, goals are used to state what has to be achieved in the context of process modeling.

Process models are defined at the operational level based on the BPCM and the goal model. These processes reflect the involved participant (who), functional goals (what), coordination and cooperation (when). The model developer is supposed to design and develop the business process models based on models from the upper two levels (sketch level and organizational level) as a blueprint for the IT systems to be implemented.

This approach allows one to document the business information in a specific model (the BPCM) in order to facilitate the further derivation of the goal and process models from it. This model even allows one to detail the business goals that have to be achieved by each process.

Thus, for example, applying this framework to the online-shop running example, the modeler can define a BPCM for the business process *Placing Online Order* as the one detailed in the Table 2.1. As can be seen, the modeler can easily document for a specific process the actors performing it, its resources, context, or domain and the goal (both operational and non-operational) with which it is related. This information facilitates the generation of the goal and process models. However, the derivation of the defined interdependencies between processes and goals to the organizational and operational levels highly depend on the languages used for modeling them, i.e. i* and BPMN (which are detailed in further sections). Thus, the derivation of the interdependencies, between the *Placing Online Order* process and, for

Table 2.1: Summary of the derived BPCMs for the online-shop example

Name / Elements	Online-Shop
Process	Placing online order
Resource	Order, Items, Payment Information, Stock, Customer Information, Bank Information, Payment Status
Actors	Seller, Customer, Store, Marketing, Accountant, Bank
Context	Web Portal
Business Domain	454111 Electronic Shopping
Goal	Soft Goal: Easy to adapt to changes, Increase the customer satisfaction Hard Goal: Get online orders
Process Type	Exchange

example, the *Easy to Adapt to Changes* soft-goal, to the operational level largely depends on whether BPMN permits the documentation of this information. If it cannot be modeled, as is detailed below, these interdependencies will kept implicit and hidden in the models, having to analyse the BPCMs and the generated artifacts to identify them in the subsequent development phases.

Ullah's Framework

This technique aims to better understand the business expectations of an IT system. This approach is structured in two parts as Figure 2.4 shown (Ullah & Lai, 2011). The former part, called *Organization Fit* in Figure 2.4, describes the specifications of the business strategy and business infrastructure, which is further divided into three levels:

1. The first level details the organization's stakeholders, aims, objectives and available resources.

2. The second level describes the business strategy in the form of the business mission statement, strategic goals and targets, and the way to evaluate the strategy. The modelling of business strategy is based on the strategic alignment model (SAM) (Henderson & Venkatraman, 1992)

3. In the third level, the business infrastructure is detailed with four submodels: first, the administration is the man power in the organization; second, the

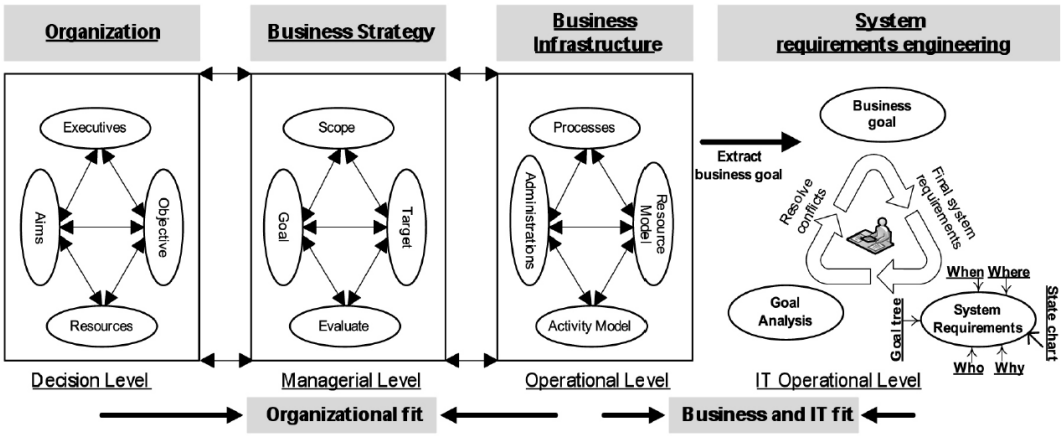


Figure 2.4: Ullah's Framework

activity model describes the organizational actors and their responsibilities; third, the resource model describes the organizational rules and regulations; and fourth, the process model defines the business workflows. To model the business process, the BPMN notation is used (BPMN, 2014).

The latter part, called *Business and IT Fit* in Figure 2.4, guides in the extraction of the system goals and the system requirements. The system goals are extracted from the analysis of the business processes that the system has to support. The final system goals are modeled as a goal tree. Then these goals are analysed in order to obtain the system requirements, which are modeled by means of UML State Chart (UML 2, 2014).

For example, applying this framework to the online-shop example, one would extract the system goals from the business processes modeled with BPMN. Thus, taking as input the *Place Online Order*, defined above and which model is shown in Figure 2.10, the goal tree shown in Figure 2.5 would be obtained. As can be seen, this diagram documents the goals leading to the execution of a business task. However, it does not relate this goals with the information detailed in other business artifact, such as the strategic goals restricting some of the business tasks.

Although this framework defines a complete process for documenting the different business information, and also guides in the derivation of the system requirements from it, it does not indicate how to document the interdependencies between the

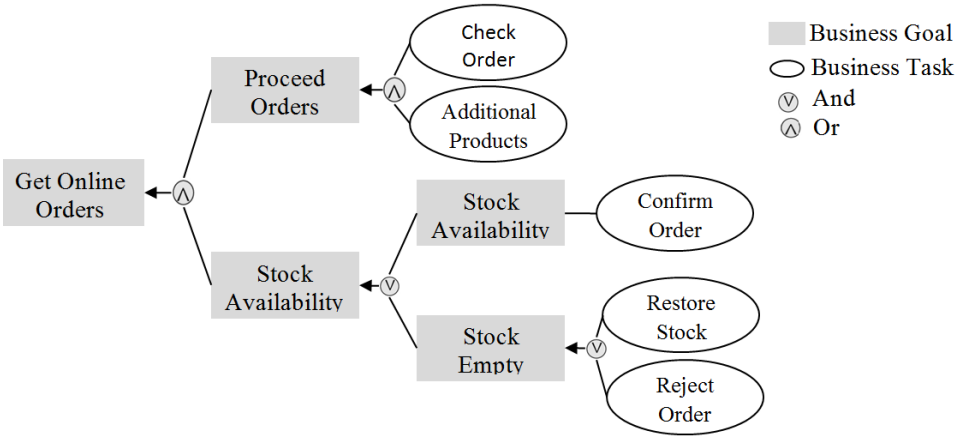


Figure 2.5: Goal tree generated applying the Ullah's approach to the Place Online Order.

different elements in each part of the process. Therefore, it definition highly depends on the support each notation (i.e. BPMN, UML and Goals) provides to the definition of these interdependencies. Thus, since the *Place Online Order* process does not document the interdependencies and the relationships with the strategic goals, because BPMN does not support it (as is detailed below), and the Ullah's framework neither documents them (as Figure 2.4 shows), these interdependencies kept implicit and are not derived to the subsequent development stages. The same happens with the documentation of the system goals and requirements. Therefore, these interdependencies will have to be identified at the system design stage analysing the generated artifact. Any misinterpretation made during this analysis may lead to design issues that are not identified until the final development stages.

Analysis and Discussion

In this section, we have reviewed different approaches analysing and documenting the business information. These approaches have been analysed in order to identify what business information they model, how they model it, and how it is used in the requirements specification. Specifically, we have focused on identify what interdependencies between business elements they can model and if they support the automatic derivation of the system requirements from the business specification. This review allowed us to meet the objectives one and seven of this thesis.

As result of this review, we have identified that normally these approaches allow one to model the enterprise's strategic goals and its business processes. Usually, the business processes are modeled to fulfil the organization strategic goals. Even some approaches, such as EKD, define links to document which goals motivate each business process. Nevertheless, they do not allow one to detail whether certain goals restrict the business processes. This information is kept implicit in the generated models. So, these models will have to be analysed in the subsequent development phases in order to identify them. Thus, for example, for architect to properly design the online-shop system, he/she will have to analyse the *Place Online Order* BPMN model, the goal tree, and all the generated requirements models in order to identify that the whole process have to be easy to adapt to changes, which is a goals constraining the process. If this interdependency is not correctly identified, it may happen that the designed system would not be sufficiently maintainable. Situation that would have to be detected and corrected at the final development stages.

In addition, we have observed that these approaches also guide in the identification of the system requirements from the business information. Thus, EKD and the Ullah's framework define how to derive the system requirements. Currently, this derivation has to be done manually since these approaches do not specify any method semi-automatizing the generation of the requirements models from the business artefacts. This not only involves an effort analysing the artifacts generated at the business phase for creating the requirements models, but also any misinterpretation made during this analysis can lead to model requirements that do not correctly align with the business. Issues that will be hauled to the final developments stages, when they are normally identified and corrected.

This thesis addresses the modeling of the interdependencies between business elements and their semi-automatic derivation to the subsequent development stages. So that, architects can easily reuse them to design a system better aligned with the business.

2.3.2 Goal Modeling Notations

A Goal is a statement about a state or condition of the enterprise to be brought about or sustained through appropriate tactics and strategies; i.e. the goals are the drivers of the organization's decisions. Goals may be formulated at different levels of abstraction, ranging from high level in terms of general concerns, to low level in terms of specific concerns. Goals can specify from functional concerns to quality o business restriction concerns.

Goal modeling is used to represent business objectives for stakeholders and to drive business and IT requirements for business analysis and execution, thereby helping to facilitate the business-IT alignment (Gao & Krogstie, 2009). The most important goal modeling techniques are Business Motivational Model (BMM) (BMM, 2013) and i* Framework (E. S.-K. Yu, 1995).

Business Motivation Model (BMM)

BMM is a metamodel of the concepts essential for business governance. Its focus is on why enterprises run their businesses the way that they do, and its underlying principle is "Businesses are driven, not by change, but by how they decide to react to change".

BMM was developed by the Business Rules Group (BRG) and in 2005 was accepted by the Object management Group (OMG) as the subject of a Request for Comment. In 2008, BMM was considered as a de-facto standard and the first specification of it was released by OMG.

The basic idea behind BMM is to develop a good business model in terms of business plans before system design or technical development is begun (Gao, 2011). The Business Motivation Model allows a business plan to be developed, communicated and managed in an organized fashion. Specifically, the Business Motivation Model does all of the following:

- Identifies factors that motivate the establishing of business plans.
- Identifies and defines the elements of business plans.

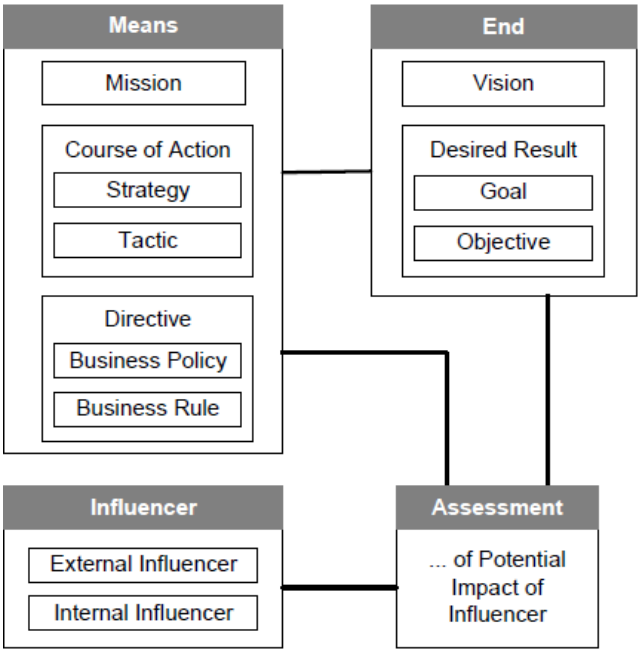


Figure 2.6: BMM Overview

- Indicates how all these factors and elements inter-relate.

As is shown in Figure 2.6, with BMM the business plans are modeled in terms of the Vision, Goals, Objectives, Mission, Strategies and Tactics, and the Influences, internal and external. The most important components are:

- *Ends.* These elements are used to indicate the future state the business wants to achieve. Ends are categorized as Vision and Desired Results, and Desired Results as Goals and Objectives.
- The *Means* detail the methods and strategy the business will employ to achieve the Ends. Means are organized into Mission, Courses of Action, and Directives. The Means does not include either the tasks necessary to exploit it, nor responsibility for such tasks.
- The *Influencer* are things that can cause changes that affect the enterprise in its employment of its Means or achievement of its Ends. Influencers may be Internal (from within the enterprise) or External (from outside the enterprise boundary).
- *Assessment.* These elements are judgments about the Influencers on the en-

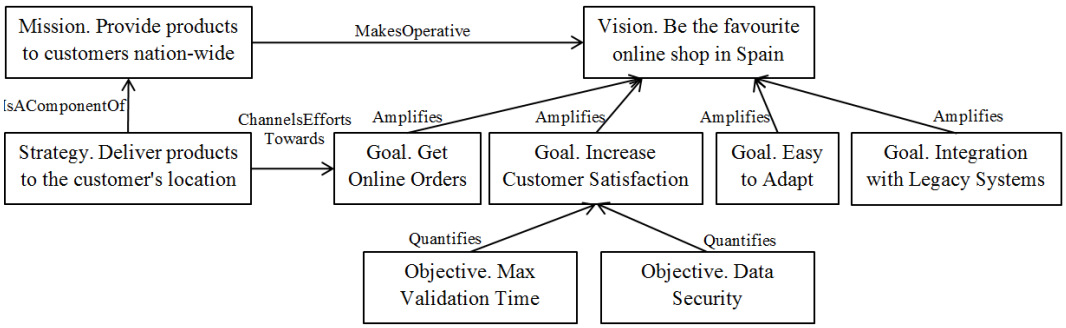


Figure 2.7: Excerpt of the online-shop business plan modeled with BMM.

terprise's ability to employ its Means or achieve its Ends

Figure 2.7 shows an excerpt of the business plan obtained applying BMM to the online-shop running example. As can be seen, this plan start defining the business *Vision* and the *Mission* that will make it operative. The vision is refined into four goals *Get Online Orders*, *Increase Customer Satisfaction*, *Easy to Adapt to Changes* and *Integration with Legacy Systems*. In turn, the goal *Increase Customer Satisfaction* is refined in two objectives *Maximun Validation Time*and *Data Security*. Likewise, from the mission, the strategy *Deliver products to the customer's location* is defined as a course of action to achieve the goals. Specifically, the model documents that this strategy supports the achievement of the goal *Get Online Orders*, as the relationships *ChannelEffortsTowards* indicates.

The strategies are subsequently operationalized in business processes, which are documented with BPMN (BPMN, 2014). However, as it is indicated in the specification (BMM, 2013), the relationships between BPMN elements and BMM are loosely defined. Therefore, documenting the relationships between strategies and goals to relationships between business processes and goals largely depends on the support provided by BPMN for this. Since BPMN cannot model them (as Section 2.3.3 details), they are kept implicit. Thus, for example, engineers cannot models with BMM, nor with BPMN, what business processes, or part of a process, of the online-shop are constrained by the *Easy to Adapt to Changes* goal. This information would have to be identified in further development stages to design a system correctly meeting all the business goals.

i* Framework

i* framework is a modeling language suitable for an early phase of system modeling in order to understand the problem domain (E. S.-K. Yu, 1995). The i* framework, as a part of the User Requirements Notation (URN), was approved in 2008 as an international standard as ITU-T Recommendation Z.151.

i* is an approach originally developed for modelling and reasoning about organizational environments and their information systems composed of heterogeneous actors with different, often competing, goals that depend on each other to undertake their tasks and achieve these goals. Its modeling language allows to model both as-is and to-be situations.

Domain documented with i* can be documented with two different models:

- *Strategic Dependency Model (SD).* This model describes the relationships between actors in an organization. These relationships are dependencies between actors, where an actor (the depender) depends on another (the dependee) to meet a goal, perform a task or turn over a resource. There are four kind of dependencies: Goal dependencies are used to represent delegation of responsibility for fulfilling a goal; softgoal dependencies are similar to goal dependencies, but they represent vaguely defined goal (such as quality attributes); task dependencies represent the need to perform a given activity by another actor; and resource dependencies require the dependee to provide a resource to the depender.

- *Strategic Rationale Model (SR).* This model determines through a "means ends" analysis how the dependencies are achieved through the contributions of other actors. The SR model provides a more detailed level of an actor for modeling the internal intentional relationships. The nodes (goals, tasks, resources, and softgoals) in this graph can be related through means-ends, task-decomposition, and contribution links.

This framework has been widely used for an in-depth analysis of an organization for various purposes, such as business process reengineering (E. Yu & Mylopoulos,

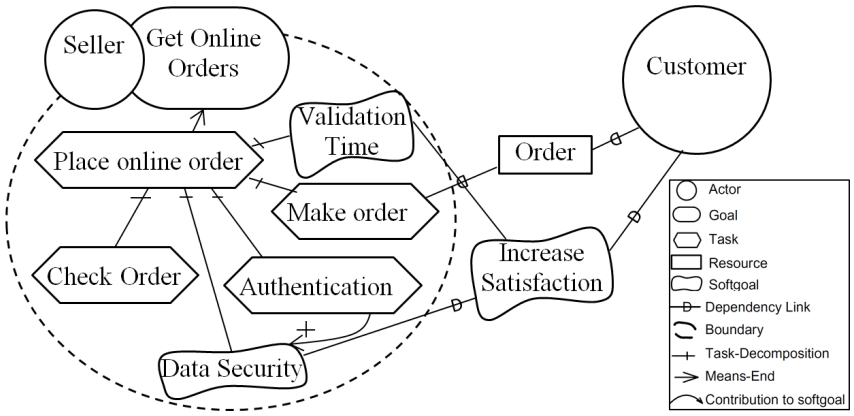


Figure 2.8: Excerpt of the online-shop modeled with i*.

1994; Grau et al., 2008) or software requirements engineering for developing information systems (E. S. K. Yu & Mylopoulos, 1994; Castro, Kolp, & Mylopoulos, 2002). It allows one to detail a number of relationships between actors and, even, to refine them in order to document how the actors fulfil them. Thus, the engineer can model if specific goals motivate certain business processes.

Using this framework to analyse the online-shop business, and specifically the *Get online orders* goal, the SR model shown in Figure 2.8 would be generated. The SR model documents that this goal motivates the *Place online order* process, which is composed of the business tasks *Check order*, *Make order* and *Authentication*; and which is also related with two softgoals (*Data security* and *Validation time*). Furthermore, the SR model can document what business tasks affect positively or negatively to the softgoals. Thus, the *Authentication* business task positively affect the *Data Security* softgoal. However, with this framework the modeler cannot document the opposite relationship, i.e. if a softgoal affects and/or contrains a business task. Thus, he/she cannot model that the *Validation Time* softgoal restrict the *Chek Order* business task. This information kept implicit in the models. Therefore, when the system is designed, this interdependency has to be identified. If it is not properly detected, there is an increasing risk of designing a system in which not the appropriate components meet the performance requirement. These issues have to be corrected as soon as they are identified, which usually happens at the final development phases.

Analysis and Discussion.

The goals of a company drive how their activities should be performed. In this section, we have reviewed the most important Goal modeling notations in order to identify which goal they model, how they model them and, especially, what relationships between goals and the business activities affected by them can be documented with these notations. The identification of the interdependencies supported by these notations allowed us to more completely achieve the objective one of this thesis.

As result of this analysis, we have identified that BMM is focused on analysing an enterprise in order to model its business plans, while i* is more oriented toward reengineering its business process or getting information for an early phase of the software engineering process.

Both notations facilitate the modeling of business goals at different level of abstraction, being able to refine them until the level where they can be completely detailed. They also define different kind of relationships to document what strategies or tactics are need to achieve them, or the business tasks motivated by them. Nevertheless, one cannot document if some specific goals affect and/or restrict some business tasks. It is also difficult to model if these tasks are already being supported by legacy systems. These interdependencies are essential to design a system correctly integrated and aligned with the business. If they are not modeled, they have to be identified at the architecture design phase at the risk of misinterpretations leading to incorrect designs.

2.3.3 Business Process Modeling Notations

Business Process Modeling is a crucial part of BDD as process models are the main artifact of the development process. Business Process Modeling has acquired a great importance in organizations, since it helps companies to communicate and to make business decisions, and employees to better understand the running of an organization. In short, it helps companies to be competitive and achieve their goals (Indulska, Recker, Rosemann, & Green, 2009).

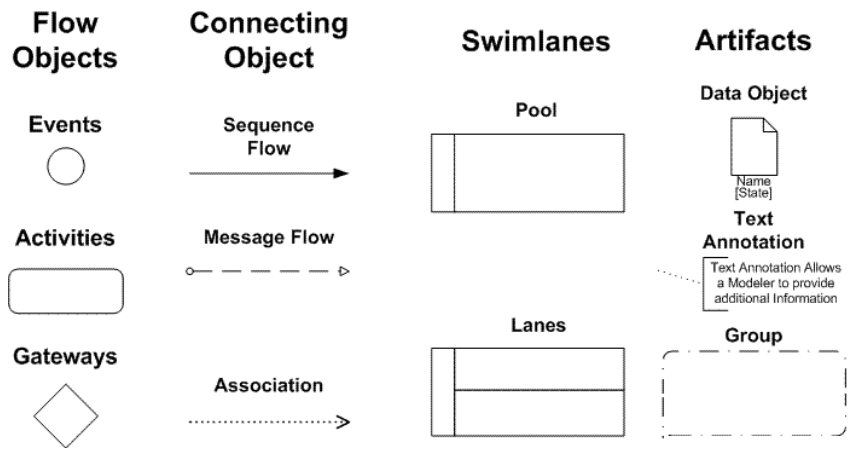


Figure 2.9: Core set of BPMN elements.

A business process is a collection of related and structured activities that produce a specific output for a particular customer (Davenport, 1993). Business processes can be modeled as a sequence of activities in a business process model. Nowadays, a number of notations have been defined for Business Process Modeling (Dumas, van der Aalst, & ter Hofstede, 2005). In this section, the most important ones are reviewed: BPMN (BPMN, 2014) and UML Activity Diagrams (UML 2, 2014).

Business Process Model and Notation (BPMN)

BPMN (Business Process Model and Notation) is a standard graphical representation for specifying business processes in a business process model. It was developed by the Business Process Management Initiative (BPMI) group in 2004 and was widely accepted. In 2005, this group merged with Object Management Group (OMG) and, since then, this notation has been maintained by OMG.

The primary objective of BPMN is to provide standard notations, which are simple and easy for developers and other business stakeholders to understand, yet able to represent complex process semantics. Therefore, it uses a standard modeling language to bridge the gap between the business model and implementation. Because of that it is very appreciated by both business analysts and IT engineers.

This notation allows companies to model and document their processes through Business Process Diagrams (BPD), which consist of the following elements (Fig-

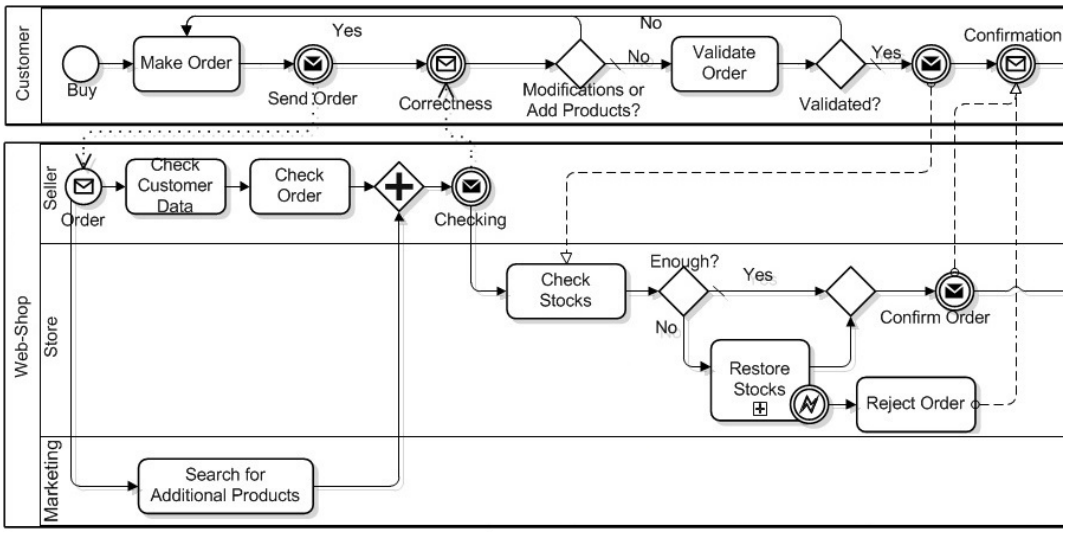


Figure 2.10: Business process to handle orders in an online-shop.

ure 2.9 shows the graphical representation of the elements):

- *Flow Objects.* These are the most important elements in BPMN, since they are used to model business process logic. They are events (indicating when a process starts, pauses, resumes or stops), activities (describing the work which must be done) and gateways (used to fork or merge paths, depending on the conditions expressed).
- *Artifacts* allow one to bring addition information into the model (such as the data required by each activity or groups of tasks that have a common characteristic among them). In this way the model becomes more readable and complete.
- *Connecting Objects.* Flow objects and artifacts are connected to each other using connecting objects, which are of three types: sequence flows (normal and default), message flows, and associations.
- *Swimlanes* allow the role or department responsible for a specific set of business tasks to be represented; they are pools and lanes.

This notation allows one to easily model any kind of process, regardless of its complexity. Furthermore, relationships between different processes or diagrams can be modeled. For example, Figure 2.10 shows an extract of the online-shop process

for handling orders. This models, in addition to document the business tasks and their sequentiality, details a relationships with the subprocess *Restore Stocks*, linking both workflows.

However, natively one cannot document the interdependencies of the processes with other business elements (such as with the business goals constraining them or with the legacy elements already supporting some business tasks). Thus, in the diagram of the process for handling orders (Figure 2.10), one cannot detail that the goal restricting the time in which an order has to be validated directly affect the *Check Customer Data*, *Check Order* and *Search for Additional Products* business tasks. Similarly, one cannot natively model that the sub-process *Restore Stocks* is supported by the legacy system *Stock Control*, which implies an integration of the system with the legacy system. Therefore, with the standard notation of BPMN, this information kept implicit in the diagrams. Therefore, it would have to be identified in a subsequent development phase.

Currently, there are different approaches and BPMN extensions especially developed to model some of these interdependencies. These approaches are only focused on business information for making business process reengineering or cannot model all the interdependencies between business goals and business processes. Therefore, to the best of our knowledge, there is no comprehensive approach managing the interdependencies between business goals and processes and deriving them to the subsequent development stages. The most important extensions related to this thesis are reviewed in Section 2.3.4.

UML Activity Diagrams

UML (Unified Modelling Language) (UML 2, 2014) is considered de facto standard for the modelling of object-oriented software systems. It consists of a number of diagrams with different goals and for different purposes. Currently, thank to the versatility of UML, the Activity Diagrams are not only used to model the flow and the dynamic behaviours of a system, but also to document business processes.

Activity diagrams show the workflow from a start point to the finish point de-

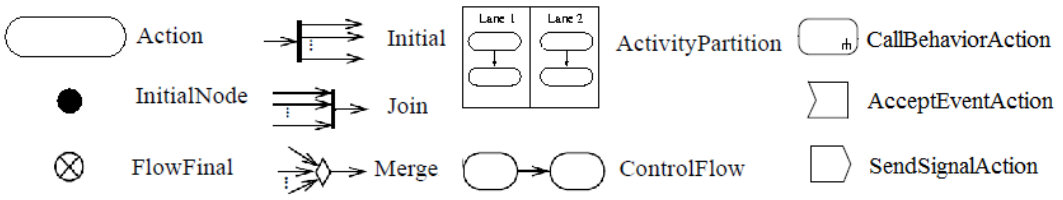


Figure 2.11: Core set of elements for modeling Activity diagrams

tailing the many decision paths that exist in the progression of events contained in the activity. The most important elements that constitute an Activity diagram are:

- *Activities*, which are the highest-level units of behaviour.
- *Actions.* These are used to represent a single step within an activity.
- *Flow Nodes* represent point at which the workflow can start or stop.
- *Control nodes.* These elements are used to fork or merge different paths depending different conditions or situations.
- *Object nodes*, which represent data and information used or generated for actions.
- *Flow edges.* These edges are used to connect action and flow nodes between them (to indicate the sequentiality of the tasks) or with object nodes (to indicate input or output information).
- *Partitions* allow one to group actions according to a specific criterion.

With this notation engineers can model organizational and computational workflows. In addition, different signal and call action can be modeled indicating relationships with other workflows. Therefore, the online-shop process for handling orders can be modeled similarly as is detailed with BPMN (Figure 2.10). However, as with BPMN, interdependencies with other kind of elements cannot be modeled (such as with business goals or with legacy systems). Thus, for example, the relationships between the goal indicating the time in which an order has to be validated and the process for handling orders would keep undocumented. Therefore, they would have to be identified in a subsequent development phase.

Currently, there are different approaches and profiles guiding the modeling of

some of these interdependencies. These approaches are usually focused on specific relationships or on concrete diagrams. Therefore, there is no comprehensive methodology managing the interdependencies between business goals and processes and deriving them to the system design stage. The most important approaches, related to this thesis, and extending UML for modeling the interdependencies are reviewed in the following sections.

Analysis and Discussion

In this section, the notations for business process modeling have been reviewed with the aim of identifying if they support the modeling of the interdependencies of the processes with other business elements (such as business goals or legacy systems) and if they can be extended to allow engineers to model them. This review has also been done to meet the objective one of this thesis.

As result of this evaluation, we have identified that although both notations allow one to perfectly model business processes at a different abstraction levels, usually BPMN is more commonly used for modeling business process at an organizational level, while Activity diagrams are used for detailing workflows at a computational level.

Both notations provide different mechanism to define relationships between business processes. Nevertheless, none of them have elements to model which strategic goals motivate each business process, nor if some goals affect and/or restrict a process, or a part of it. This information is kept implicit in the generated models. Therefore, when it is needed, it has to be identified by analysing these models. In addition, they do not have elements to indicate if any part of the process is already being supported by a legacy system. Lanes or partition (representing a legacy system) could be used for this purpose but, then, information about the roles responsible for the tasks would be lost.

Finally, both notations have already been extended by different approaches to represent specific relationships at concrete development stages. Nevertheless, to the best of our knowledge, there is no comprehensive approach managing the interdepen-

dencies between business goals and processes and deriving them to the requirements and the system design stages. This thesis aims to extend both notations and to define a comprehensive methodology providing these capabilities.

2.3.4 Modeling the Interdependencies between Business Goals and Processes

Normally, between the business goals and processes there are a set of relationships and interdependencies. Thus, a business goals can motivate a complete process, or a part of it; or may affect and/or restrict how a process, some of the defined business tasks or the flow have to be done. The former are usually goals that can be detailed to an operational level, so that one can specify concrete business tasks or processes to satisfy them. The latter are non-operational goals (such as business constraints or quality attributes) that usually are not satisfied by performing new tasks, but how the existing ones are carried out.

In the literature one can identify different approaches to model the relationships between business goals and processes. These approaches have been specified in order to achieve different aims, such as to better document the business, to perform business process reengineering or to develop information systems aligned with the business. The most important proposals, and related with these thesis, are Aburub et al. (2007), Pavlovski and Zou (2008) or Bocanegra, Pena, and Ruiz (2011).

Aburub's Approach

This approach presents a methodology for identifying and modeling business goals and for linking them with business processes. Their objective is not to design a technical system, or a product, but to understand and model the goal of the business itself with the aim of identify improvements that could be made in the business processes (Aburub et al., 2007).

First, the business processes are modeled with Role Activity Diagrams (RADs) (Ould, 2005). RAD shows the roles that play a part in the process, and their

Table 2.2: Using the Aburub's approach for relating the *Data Security* goal with the process *Place Online Order*

Process	Place Online Order
Goal	The customers and orders data must be secure.
Role	Customer
Action	For accessing to the system, customers have to authenticate

component actions and interactions, together with external events and the logic that determines which actions are carried out and when. So, it shows the activity of roles in the process and how they collaborate. Modelling the business via RADs proceeds in a bottom-up direction, identifying activities and interactions taking place in the business and representing them as a set of interacting roles.

Then, the business goals are modeled. For this, they have adapted an approach defined by Cysneiros, Sampaio, and de Melo (2001) as an immediate model for representing and decomposing goals and resolving conflicts between them. The production of goals is essentially a top-down process which starts from consideration of general goals from which sub-goals can be methodically derived. To that end, they propose the following steps to model and decompose the goals:

- Elicit goals and decompose them into subgoals.
- Define relationships between goals and subgoals.
- Identify actors who will achieve goals.
- Operationalise goals.
- Analyse positive and negative interactions between goals.
- Select which goals must be addressed to achieve process improvement.

Finally, the goals and the processes are linked with a table indicating to which processes each goal have to be applied and which roles and actions should be done in order to fulfil the goal. Thus, one can easily identify activities, interactions, and roles in the business processes that should be done to deliver or serve the goal better.

For example, in the online-shop running example, analysing the*Data Security*

goal, refined from the *Increase Customer Satisfaction* goal (as Figure 2.8 indicates), and how it is applied on the *Place Online Order* process, the Table 2.2 is defined to document how they are linked. As can be seen, this table indicates that in order to achieve the goal, the customer for placing an order has to be authenticated. This information is very useful for reengineering and improving the business processes. Nevertheless, this approach does not indicate how to document the relationships with goals that cannot be operationalised and only impose restrictions on the already defined task (such as the one indicating that the order validation has to be done in less than 0.7 seconds). If this information is not documented, it would have to be identified at the architectural design stage in order to identify the components and the functionalities that have to be designed taking the performance into account. If these interdependencies are not correctly identify, there is an increasing risk of designing a system that does not correctly meet the performance constraints and is not correctly aligned with the business. Issues that normally are not identified until the final development stages, when they are expensive to solve.

Pavlovski's Approach

Pavlovski and Zou (2008) identified that the business goals are arguably more difficult to capture in business process modeling, since the focus of the methods and notations used is the modeling of functional behaviour. To contribute to a model that provides a more complete representation of the overall business process, they proposed two new artifacts for BPMN. These artifacts model the constraints associated with a business process. Thus, these characteristics can be captured early in the systems development life-cycle. They claim that these artifacts provide an opportunity to identify, at a business level, the non-functional properties of a business process.

The two artefact defined for BPMN are shown in Figure 2.12 are:

- *Operating Condition.* This element is used to declare additional semantic constraints for the business tasks. These constraints may include security policies, organisational policies, regulatory constraints, and operational performance

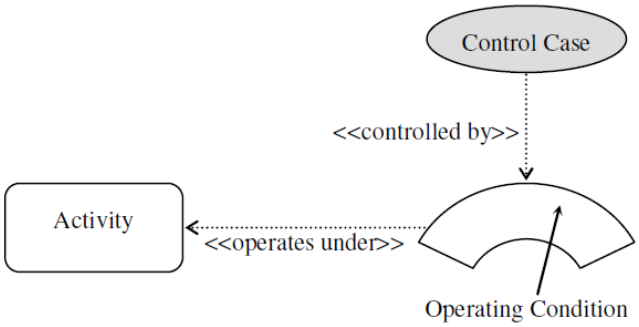


Figure 2.12: New elements defined for modeling constraints in business processes.

characteristics.

- The *Control Case* element is used to define additional control mechanisms that have to be put in place to control the operating condition. This is in order to mitigate or reduce the business risk.

This BPMN extension is very useful to model business operational restrictions in business processes and define controls to handle these restrictions. Thus, the relationships between business operational goals and business processes can be easily identify and communicated to the business expert and requirement engineers by means of the BPMN diagrams.

Figure 2.13 shows the process for handling orders of the online-shop with the *Data Security* and *Validation Time* restrictions modeled with the previously detailed BPMN elements. As can be seen, one can model the exact tasks restricted by each Goal. Thus, the *Check Customer Data*, *Check order* and *Search for Additional Products* are restricted by the goal limiting the time in which an order has to be checked. However, as the number of goals and tasks affected by them increases the readability of the diagram decreases. Thus, due to the number of tasks constrained by the *Data Security* goal, the diagram is messier. In addition, the approach is oriented to document operational restriction. This implies that, for example, the goal indicating that the system should be easy to change cannot be modeled, since it does not lead to operational restrictions. To model a more complete range of restriction, it would be desirable to be able to model non-operational restrictions or goals. Thus, all the relationships between business processes and business goals

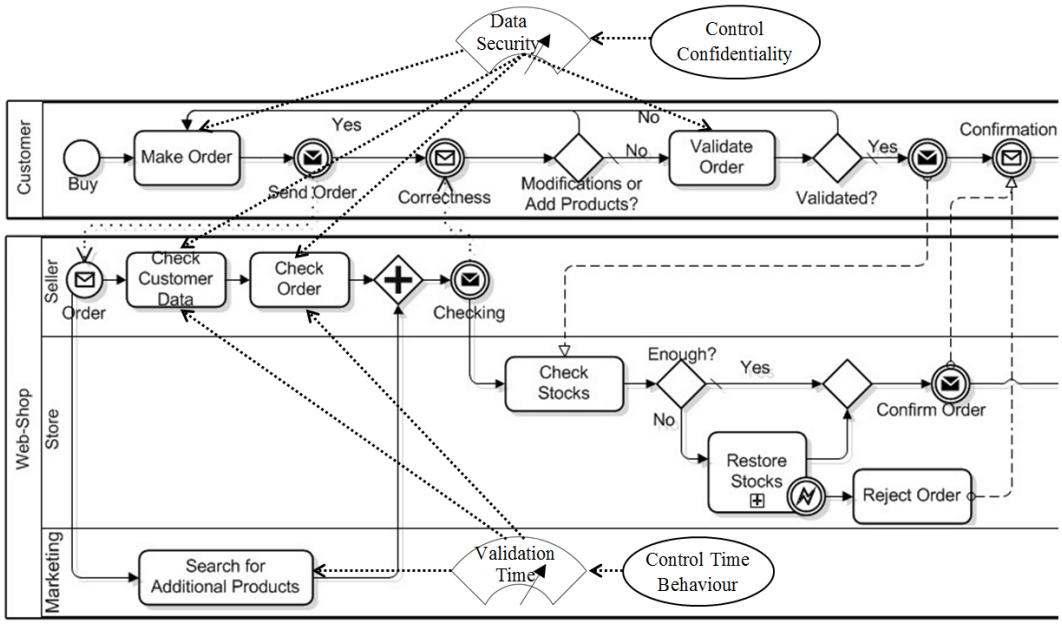


Figure 2.13: Process for handling orders annotated with Pavlovski's approach.

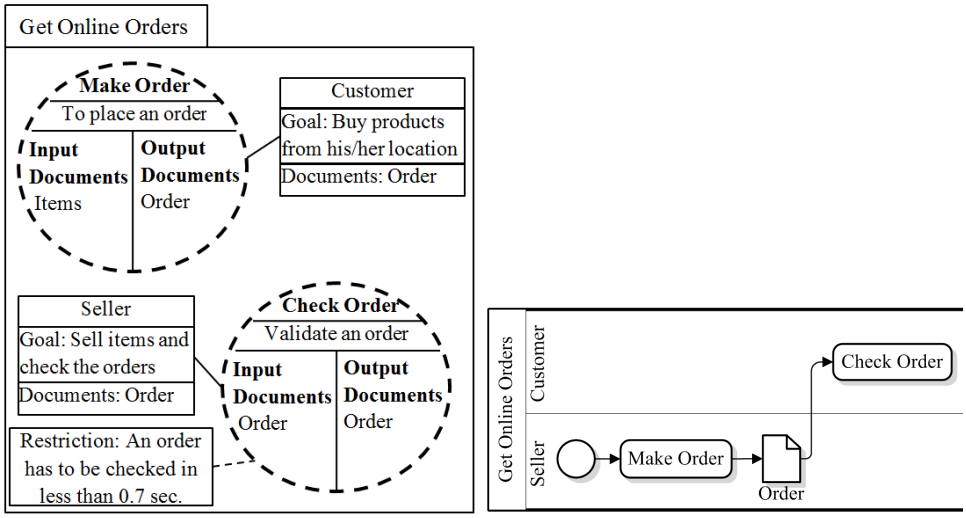
would be easily transmitted to the subsequent development stages in order to design architecture better integrated and aligned with the business, with less effort and with a lower risk of misinterpretations.

Interorganizational Business Modeling

In this approach Bocanegra et al. (2011) indicate that the business processes modeled with BPMN do not cover the documentation of business goals, legal restriction, economic restrictions or the roles' profiles. This additional information is very important to correctly document the business processes, and especially those representing interactions between two or more companies. The set of all this information is called *Business Transaction* (Papazoglou & Kratz, 2006).

Therefore, the authors define a model-driven approach for documenting business goals and transactions at different levels of abstraction with different notations.

The strategic goals are modeled using UML2 Package Diagrams. In order to be able to model all information concerning business goals, this diagram has been extended with a set of profiles for annotating if a goal is mandatory or optional and to better model the relationships between goals.



(a) Organizational View. (b) Business Process View.

Figure 2.14: Excerpt of the business analysis done to the online-shop with the Interorganizational Business Modeling approach.

A business transaction represents how a strategic goal, or a set of goals, are achieved. A transaction is modeled with two different views: the organizational view and the business process view.

- The organizational view details the interactions between the participants of an interaction and is modeled with UML2 Collaboration Diagrams. For each interaction one can model the input and output artifact, the goals pursued by the participants and/or the business rules imposed to it.
- The business process view document the business activities and the order in which they have to be done to achieve all the goals and restriction detailed for the business transaction in the organizational view. This view is modeled with BPMN. This model is automatically generated applying some ATL transformations to the organizational view.

This approach guides in the identification, analysis and documentation of business goals and business processes. They have defined a methodology and a set of models to keep both elements traceable and synchronize at different abstraction level.

Figure 2.14a shows an excerpt of the organizational view modeled to document

the interaction for placing an order of the online-shop running example. This view details that the main goal of the interaction is *Get Online Orders*, its participants are the *Customer* and the *Seller*, and that some of operational goals pursued by them are *Make Order* and *Check Order*. In addition, it details some relationships between operational goals and business restrictions. Specifically, it documents that the validation of an order has to be done in less than 0.7 seconds.

The organizational is later transformed into the business process shown in Figure 2.14b. As can be seen, the main goal is mapped to a pool (in order to document the main objective pursued by the interaction), each participant is mapped to a lane and, finally, the operational goals of each role are transformed to business task associated to him/her. However, the relationship between the operational goal *Check Order* and the time restriction is not derived to the business process. Therefore, this information is lost and if one wants to use it in a further development phase, then the organizational views have to be re-analysed and the relationships have to transferred to the models used in that phase.

Although this approach provides the architect the interdependencies at the organizational level, he/she has to identify how the different business elements have evolved in the subsequent development stages (i.e. which Use Cases and components have been derived from the *Check Order* operational goal) in order to detect which functionalities are constrained by the time restriction. Similarly, this analysis can lead to misinterpretation and to design an architecture with issues that should be solved as soon as they are detected, which usually happens at the final development stages.

Analysis and Discussion

In this section, we have reviewed the approaches focused on modeling the relationships between business goals and business processes. The aim of this analysis was to identify what interdependencies they can model, how they model them, and what relationships important for the architectural design they cannot document. This review allowed us to achieve the objective one of this thesis.

As result of this review, we have identified that these approaches model these interdependencies with different purposes, from identifying areas in a business process that have to be improved in order to better satisfy the business goals, to enhancing the business documentation and specification in order to facilitate the identification of the requirements of a future information system.

To model these interdependencies, these approaches extend existing notations for specifying the operational business restrictions that have to be met by each process. However, they do not define elements to model if a business process, or a part of it, is interrelated with non-operational business restrictions.This information still remains implicit in the generated models, so that it has to be made explicit in the further development stages. Forgetting or not correctly identifying these relationships can lead to make incorrect architectural designs, hindering the fulfilment of the business goals. Issues that would be detected at the integration and deployment stages, when they are costly to solve. In fact, these were the problems that occurred at the Teseo software factory, at the origins of this thesis, and which we aim to solve by making these interdependencies explicit from the early development stages and deriving them to the architectural design phase.

2.4 Modeling the Requirements

The analysis and modeling of the system requirements are the set of activities aimed at identify the requirement, document them and refine them into more specific requirements. Building accurate models means that engineers can guarantee the correctness of the system functional and non-functional requirements. This correctness is essential for developing a system satisfying the goals, since the requirements models are used as specifications for the designers and builders of the system (Pohl, 2010).

With the aim of facilitating and guiding in the requirements analysis and modeling, nowadays different techniques propose activities and specific notations for the identification and documentation of the requirements. This section focuses on introducing the most common notations used for modeling the functional and non-

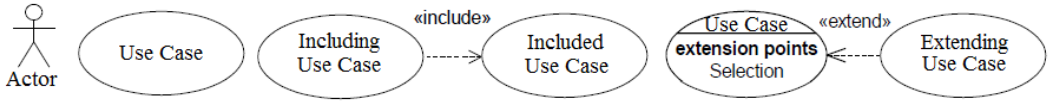


Figure 2.15: Core set of Use Case Diagram elements.

functional requirements.

2.4.1 Notations for Requirements Modeling

The notation for modeling requirements allows the engineer to perfectly document the system requirements in order to drive them to a subsequent development phase (system analysis and design). Nowadays, there is a huge number of notations/approaches for modeling the system requirements. These notations can be differentiated in the kind of requirements that they can document: functionals, non-functionals or both functionals and non-functionals.

The most important notation for modeling the functional requirements is UML (UML 2, 2014). The most important notation for modeling and analysing the non-functional requirements is NFR Framework (Chung, Nixon, Yu, & Mylopoulos, 2000). A standard and important notation for modeling both the functional and non-functional requirements is User Requirements Notation (Amyot, 2003). The support that these notations give to model the requirements and their relationships is analysed in the following sections.

UML Use Case and Activity Diagrams

The Unified Modeling Language (UML) (UML 2, 2014) is a general-purpose modeling language that was developed in 1996 by Booch, Rumbaugh, and Jacobson. Later, in 1997, was accepted as a standard of the Object Management Group (OMG) and, in 2005, was standardized by the International Organization for Standardization (ISO) as a general-purpose modeling language in the field of software engineering (ISO/IEC 19501:2005).

UML 2 is composed of fourteen types of diagrams used to document and gen-

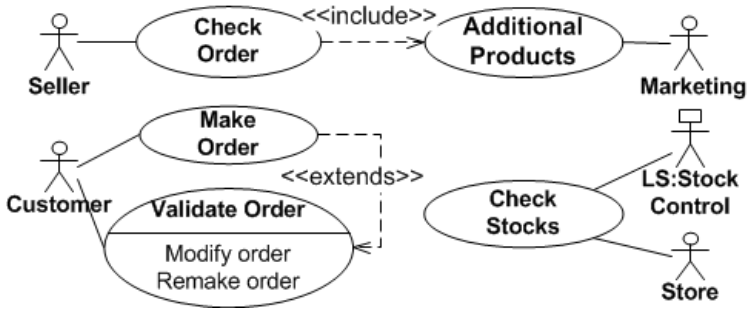


Figure 2.16: Excerpt of the Use Case Diagram extracted from the online-shop.

erate different artifact along the software development process. Specifically, for the requirements modeling, the diagrams that are normally used are the Use Case diagram and the Activity diagram.

The **Use Case Diagram** details the functionalities provided by a system. It specifies the actors of the system, their goals or functionalities that wish to perform, and the dependencies among Use Cases (Figure 2.15 shows the main elements defined to represent these concepts). Specifically, this diagram allows one to define two different relationships between Use Cases, *extend* and *include*. The *extend* relationship specifies that the behaviour of a Use Case may be extended by the behaviour of another Use Case. *Include* is a relationship implying that the behaviour of the included Use Case is inserted into the behaviour of the including Use Case. Note that the included Use Case is not optional, and is always required.

Figure 2.16 shows the Use Case diagram generated from the analysis of the online-shop process detailed in Figure 2.10. As can be seen, the defined relationships facilitate the modeling of interdependencies between Use Cases. For example, the *Additional Products* functionality is included by the *Check Order* Use Case.

The Use Case diagram is easy to understand and use, in fact it is one of the most used requirements notations in the software industry (Marinelli & Laplante, 2008), but it has some shortcomings to express other kinds of relationships. It cannot document the interdependencies between Use Cases and non-operational goals. Thus, one cannot document that the *Check Order* and *Additional Products* Use Cases are restricted by the goal constraining the time in which they have to be performed.

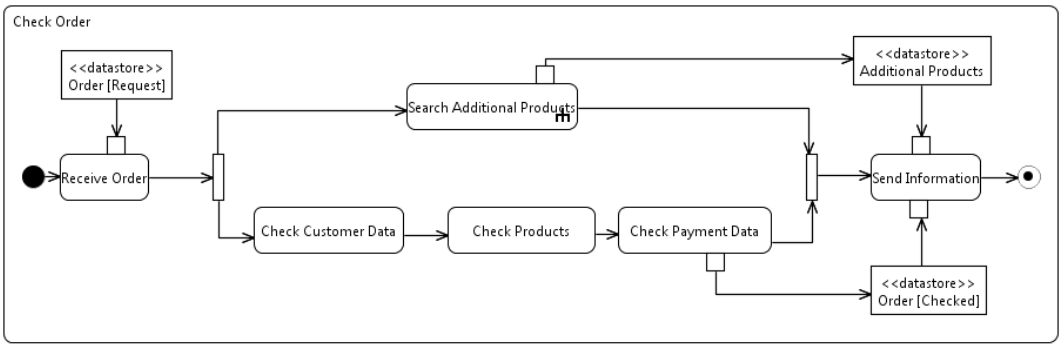


Figure 2.17: Activity diagram modeling the *Check Order* Use Case's workflow.

The **Activity Diagrams**, in addition to model business processes (as Section 2.3.3 indicates), are normally used to document the workflow that has to be followed to perform a functionality. Figure 2.11 shows the most important elements for it definition. The *CallBehaviorAction* or *SendSignalAction* are the elements used to define interdependencies between functionalities or Activity diagrams.

Figure 2.17 shows the flow of actions that have to be done during the execution of the *Check Order* Use Case. One of this action is a *CallBehaviorAction* which documents the relationships between this diagram and the *Additional Products* Activity diagram. Thus, the *include* relationship defined between both Use Cases is derived to the Activity diagrams.

This diagram allows engineers to easily document at a lower granularity level the activities involved in each Use Cases and in each relationship between Use Cases. However, as happened in the Use Case diagrams, one cannot detail if the whole diagram or specific actions are constrained by goals or quality attributes. Therefore, later during the design phase, architects will have to make an indepth analysis of the documents detailing the functional and non-functional requirements in order to take these relationships into account.

Non-Functional Requirements Framework (NFR Framework)

NFR Framework (Chung et al., 2000) is a methodology for documenting and reasoning about non-functional requirements, or quality attributes. This methodology guides in the analysis and decomposition of the system requirements expressed as

goals.

This framework is focus on the softgoals, which are those goals that are ambiguous or fuzzy, but tend to be global qualities of a software system. During the softgoal analysis, the more abstract one are decomposed to more detailed softgoals to uncover a tree structure of interconnected goals, called Softgoal Interdependency Graph (SIG) diagram.

The SIG diagram is composed of the following elements:

- NFR softgoals, which represents the high-level goals.
- Operationalizing softgoals. This element specifies concrete implementations of design mechanism that can be selected to achieve the softgoal.
- Claims, which represents domain knowledge.
- Logical (AND/OR) interdependecy links state the relationships between the more abstract softgoals to more concrete ones indicating whether all the concrete softgoals have to be achieved or only one of them.
- Bilateral interdependence links. These links details if a softgoal affect other softgoal. The type of effect may be positive (MAKE, HELP), negative (BREAK, HURT), or equivalent to the source (EQUAL). This contribution is annotated next to the interdependence link. Depending on whether the effect is direct (explicit) or indirect (implicit) the link is called *Contribution* or *Correlation*

Once the goal tree has been detailed, the different alternatives of operational softgoals (or design mechanism) meeting specific NFR softgoals are analysed, and those facilitating the fulfilment of the greater number of NFR goals are selected. Typically, this evaluation process starts from lower-level NFR and propagates through the links.

This framework facilitates the documentation and refining of non-functional requirements, along with the reasoning about the concrete operations and design mechanism that have to be implemented to achieve them. However, it does not allow one to document the interdependencies with the functionalities affected by softgoal. Therefore, when these mechanisms are applied, the architect have to make

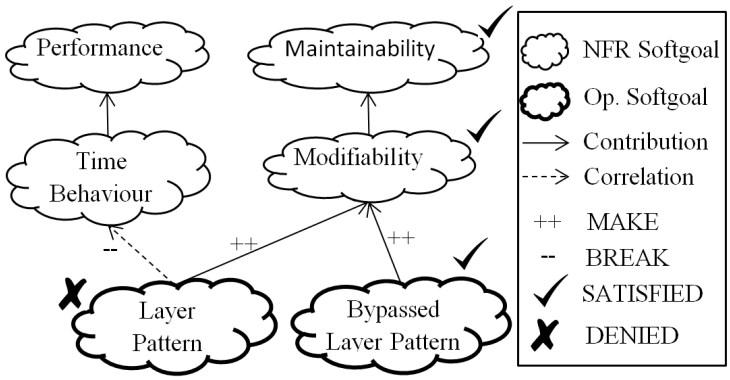


Figure 2.18: SIG of the online-shop running example.

an indepth analysis of the generated artefact to identify them.

This approach has been applied to the online-shop. Figure 2.18 shows an extract with the analysis done on the performance and maintainability softgoals. These quality attributes have been refined for detailing the design mechanism improving the system maintainability. The first one is the layer pattern, which improves the system maintainability but harms its time behaviour (due to the communication between layers). The second one is a variation of the layer pattern bypassing some layers for the functionalities constrained by the time behaviour goal. Thus, the performance is not adversely affected. Nevertheless, to apply this decision, the architect has to know the exact functionalities constrained by the time behaviour goal. Since this information is not detailed in the SIG, he/she has to make an analysis of the requirements to identify it.

User Requirements Notation (URN)

User Requirements Notation (URN) (Amyot & Mussbacher, 2003), (Amyot, 2003) is a notation for the visual description and analysis of requirements for complex and dynamic systems that, in 2003, was approved as the Z.150 by the International Telecommunications Union's Telecommunication Standardization Sector (ITU-T) (ITU-T, 2003).

URN combines two complementary views to model the non-functional (URN-NFR) and functional requirements (URN-FR). The URN-NFR is supported by the

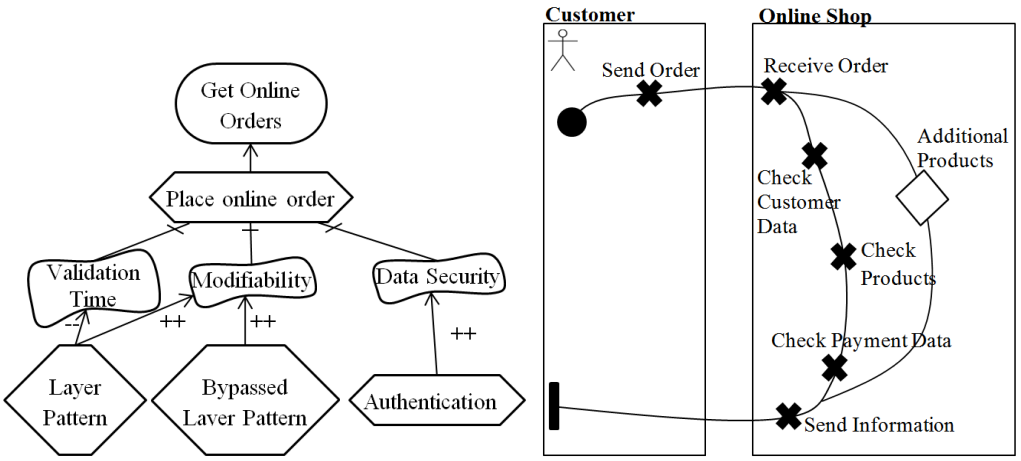
Goal-Oriented Requirement Language (GRL). GRL is a visual modeling notation based on i* frameworks that includes some concepts of NFR Frameworks (such as the classifications detailing how a goal affect to another or the evaluation mechanism). Because of this, it facilitates the documentation of the relationships between goals and among goals and concrete design options (or patterns) that can be used to achieve of them or that affect them.

Figure 2.19a shows an extract of the GRL model detailing the online-shop goals. Its main goal, *Get Online Orders*, is satisfied by the *Place Online Order* task, to which three different softgoals contribute. Also, the *Layer Pattern* is specified as a possible mechanism (or architectural pattern) to achieve the *Modifiability* softgoals but, at the same time, it hurts the *Validation Time* non-functional requirement.

The URN-FR is supported by the Use Case Maps (UCM) notation. UCM is a scenario-based notation for modeling functional requirements. These scenarios are sets of tasks, or activities, performed by an actor or the system, and paths describing their sequentiality and the logic to merge or split the path in several branches. In addition, links can be defined to encapsulate and/or reuse some models into others maps. Thus, relationships between funtional requirements can be defined

Figure 2.19b details the scenario for the functionality *Check Order* of the online-shop running example. In addition to actions that have to be done and their sequentiality, it details the relationship with the functionality *Addition Products*.

This notation, in the URN-NFR view, allows engineers to model the relationships between softgoals and among softgoals and design options. Also, the URN-FR facilitates the modeling of relationships between functionalities. Therefore, this notation facilitates the documentation of requirements models more complete, detailing the functional requirements, the non-functional requirements and the reasoning that led to each design decision made. Nevertheless, there is no concrete definition to link the functional requirements (modeled with UCM) and the non-functional requirements (modeled with GRL) (Roy, 2007). Thus, in Figure 2.19 one cannot model how the softgoal *Validation Time* constrains the tasks modeled in the *Check Order* UCM. This implies that the architect would have to analyse both diagrams to, first, evalu-



(a) URN-NFR View.

(b) URN-FR View of the *Check Order* functionality.

Figure 2.19: Excerpt of the requirements analysis done to online-shop with the URN.

ate which is the most adequate design decision that he/she has to make depending on what task, of the *Check Order* functionality, are constrained by the *Validation Time* softgoal and, secondly, to identify in which components the decision made has to be applied. Any misinterpretation made during this analysis can lead to incorrect decisions that would have to be corrected in the subsequent development stages.

Analysis and Discussion

In this section we have evaluated approaches focused on modeling functional and non-functional requirements with the aim of identifying how they meet the objective four of this thesis (i.e. to identify what interdependencies between requirements can be modeled with them).

As result of this evaluation, we have identified that UML allows one to model the system functional requirements, even at different granularity levels. In addition, the relationships and dependencies between functional requirements can be modeled. Nevertheless, it does not define any element to detail what non-functional requirements constrain each functionality. Likewise, the NFR framework facilitates the documentation of non-functional requirements and the relationships between them, but neither allows one to document the interdependencies between functional

and non-functional requirements. Finally, URN defines views to model both the functional and non-functional requirements, but it does not define concrete links to document the interdependencies between them.

The documentation and analysis of these interdependencies is very important to identify from what design mechanism is the most appropriate to achieve a quality attribute (depending on the constrained functionalities, their size or their relationships) to how to correctly apply the selected mechanism (depending on the affected functionalities). Since the previously presented notations cannot explicitly model this information, the architect must analyse indepth both artifacts to identify it.

2.4.2 Modeling the Interdependencies between Requirements

Normally, between the functional and non-functional requirements there are a set of relationships. This interdependencies detail how the non-functional requirements constrain the functional ones. As has been analysed in the previous section, the most common notations normally do not give support to model these interdependencies.

Other approaches have identified, like in this thesis, that not documenting the interdependencies between requirements increases the likelihood of integration and misalignment problems. These proposals extend the previously analysed notations for modeling these relationships. The most important approaches, which are detailed in the following sections, are Roy (2007) and Dörr (2011).

Requirements Engineering with URN

Requirements Engineering with URN (Roy, 2007) is an approach extending the User Requirement Notation specification. It defines a metamodel that integrates GRL (URN-NFR view) with an existing UCM metamodel (URN-FR view) in order to better evaluate the impact of the softgoals (detailed in the GRL model) on the functional requirements.

These interdependencies are idenfyied analysing the relationships modeled in the GRL diagram (URL-NFR view) between tasks and softgoals. Once identified, they

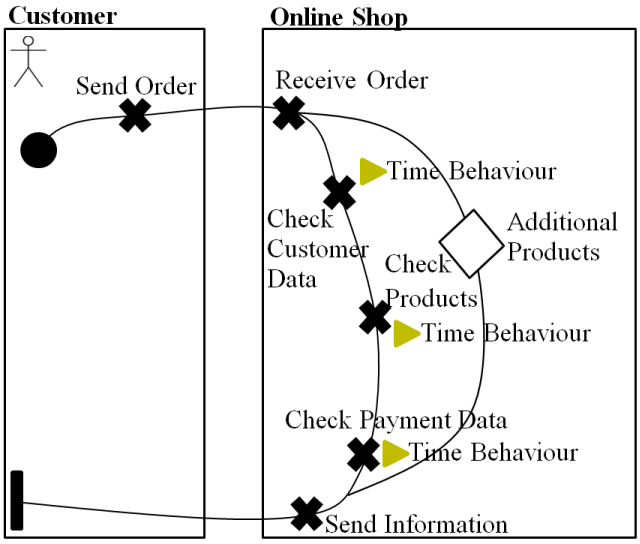


Figure 2.20: Interdependencies between the functional and non-functional requirements modeled with the *Requirement Engineering with URN* approach.

are modeled using a new concept defined by the authors, *URNlinks*, to links the specific elements modeled in the GRL diagrams with the specific actions modeled in the UCM diagrams. In addition, in order to simplify the UCM analysis based on the strategies and softgoals detailed in the GRL model, the number of links is limited to one per UCM element.

Figure 2.20 shows the *Check Order* UCM model detailing the interdependencies between their tasks and the *Validation Time* softgoal (the *URNlinks* are annotated with a yellow triangle). Specifically, the three tasks involved in the checking of an order have been annotated. Thus, these interdependencies can be taken into account to evaluate whether implement a design decision or another.

This approach is a big step forward in the specification of the interdependencies between functional and non-functional requirements; since, in addition to allow engineers to document them, the annotations defined can be reused by tools assisting architects in the architectural decision making. Nevertheless, only one relationship per UCM element can be defined. Thus, in the online-shop running example, it would not be possible to annotate that these tasks are also restricted by the *Modifiability* and *Data Security* softgoals. This implies that some interdependencies between UCMs and softgoals kept implicit. Therefore, architects would have to

identify them by analysing the URN-FR and the URN-NFR views in order to design and architecture meeting not only the *Validation Time* softgoal, but also the *Modifiability* and *Data Security* softgoals.

IESE-NFR

IESE-NFR (Dörr, 2011) is a systematic process developed in the Fraunhofer IESE focused on eliciting, documenting and analysing NFR. This process has two stages: a quality model tailoring and a non-functional requirements elicitation stage.

In the first stage, a quality model is defined for the business. This quality model is defined tailoring reference quality models or quality models from other projects. This model captures and details quality attributes and sub-quality attributes in a goal tree structure until metric can be attached to the leaf nodes. This goal tree also captures the relationships and dependencies between the quality attributes, indicating how an attribute affects to another.

In the second stage, specific values (or value domain) for the quality attributes are detailed (i.e. the values of the metrics associated with the quality attributes are specified) in order to detail the system non-functional requirements. Then, a dependency analysis is done to identify the interactions between non-functional and functional requirements (specified as Use Cases). The identifyed interdependencies are finally documented in the same artefact used to detail the Use Cases with the aim of generating a complete requirements document. When the non-functional requirements are detailed in Use Case diagrams, notes in natural language are added to the Use Cases constrained by each non-functional requirement.

Figure 2.21 shows the online-shop Use Case diagram (detailed in Figure 2.16) annotated with the non-functional requirements concerning the time in which the checking of the orders has to be done, the security for accessing customer data and the system maintainability. As it can be seen, this approach allows engineers to perfectly detail in natural language the interdependencies between Use Cases and non-functional requirements. Nevertheless, the use of natural language hinders their reutilization by tools assisting in the subsequent development stages.

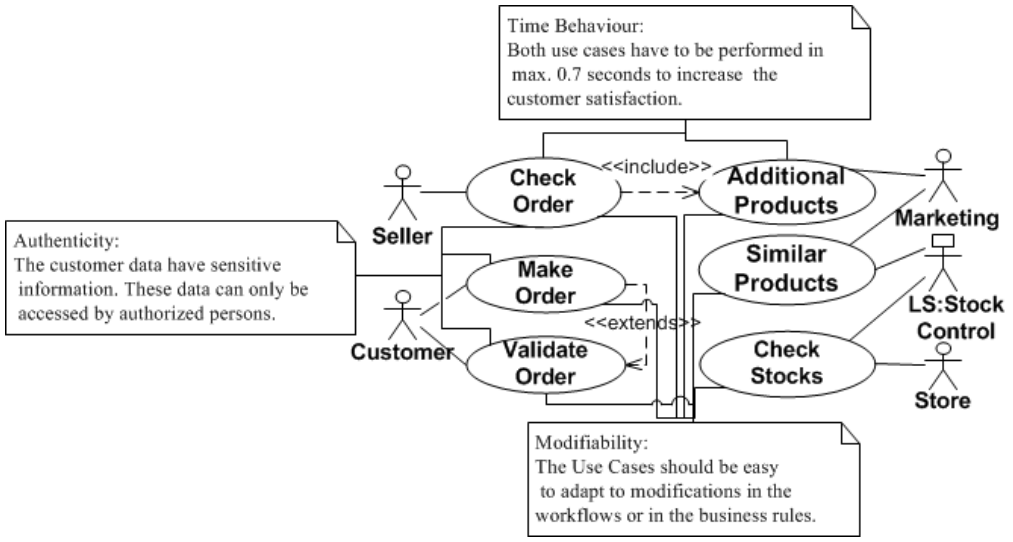


Figure 2.21: Interdependencies between Use Cases and non-functional requirements modeled with the *IESE-NFR* approach.

Analysis and Discussion

In this section, we have reviewed approaches focused on documenting the interdependencies between functional and non-functional requirements. This analysis has allowed us to know what interdependencies they model, how they model them and what relationships important for the architectural design they cannot model. In addition, we have evaluated if the interdependencies modeled are semi-automatically derived from the business information and it they can also be used by tools assisting the architect in the architectural decision making. This review enabled us to fulfil the objectives four, seven and nine of this thesis.

Requirement Engineering with URN allows the requirement engineer to perfectly detail the relationships between non-functional requirement or design alternatives and functional requirements (detailed with UCM). These interdependencies are manually derived from the relationships modeled in the GRL diagram (URL-NFR view) between tasks and softgoals. Its documentation is specially oriented to facilitate the evaluation of alternative design decisions. However, with the aim of facilitate this evaluation, this approach only permits to model one relationship per UCM element. Therefore, if several requirements constrain the same element, only one interdependency can be modeled. So, likewise architects have to analyse indepth

the URN-FR and the URN-NFR views in order to identify if there are not modeled interdependencies that have to be taken into account for designing the architecture.

IESE-NFR allows one to detail the relationships between non-functional and functional (modeled as Use Cases) requirements, regardless of the number of non-functional requirements that affect the same Use Case. To identify these relationships, quality models are tailored and used. Therefore, if this tailoring is not properly done, their identification and the business-IT alignment may be adversely affected. In addition, these interdependencies are detailed in natural language, so that they cannot be taken as input by tools assisting the architect in making decisions.

As these approaches demonstrate, the analysis of the requirements interdependencies is essential for the architectural decisions making. Furthermore, if the requirements interdependencies are detailed, the alignment of the architectural design with the business is more easily obtained. Finally, to make them even more useful to architects, this information should be modeled to be reused by tools assisting them in the decisions making process. This thesis aims to facilitate the modeling of these interdependencies using annotations that can be reused by these tools.

2.4.3 Deriving the Requirements from the Business Information

The system requirements are typically elicited by analysing the business information. Usually, this is an activity that is performed manually. Engineers analyse all the business processes and goals to identify the functional and non-functional requirements; thus, he/she identifies the requirements aligned with the business. However, the analysis and identification of this information requires a great effort. Also, since the requirements identification is subject to the engineer interpretation, is error prone.

A number of approaches aimed at semi-automate the derivation of requirements from the business information have been defined. The following sections detail the most important proposals focused on deriving the functional requirements from the business processes, since they are highly related to the goal of this thesis. These approaches are (Dijkman & Joosten, 2002), (De la Vara & Sánchez, 2009) and

(Siqueira & Silva, 2011).

Dijkman et al.'s Approach

Dijkman and Joosten (2002) introduce a technique to simplify requirements capture. The authors indicate that they have observed some similarities between Use Case models and business process models (both at the definition and specification levels). These observations lead them to the assumption that it is possible to translate business process models to Use Case models. Thus, Use Cases aligned with the business can be derived from the business process, instead of having to perform interviews in an organization.

Therefore, they defined a procedure and a set of mappings to transform business process models into UML Use Case diagrams. The most important mappings are:

- A business process role is mapped to an Use Case actor.
- A set of business tasks (*step*) is transformed to an Use Case. The authors indicate that mapping a single task to a Use Case would be to constraining. Also, mapping a procedure to a Use Case, lead to very complex Use Cases (due to the number of actors, path and interaction involved in it). Hence, they introduced the concept of *step*, as a sequence of tasks that can be performed without interruption by the same role.
- The associations between steps and roles are converted in association between the corresponding actors and Use Cases.
- The transition between tasks in a step are mapped to interactions between actions in an Use Case.
- Each alternative path of a branch in a process is mapped to an alternative path in a Use Case or to an extending Use Case.

Applying these mappings to the online-shop business process defined in Figure 2.10, one can derive a Use Case model similar to the one shown in Figure 2.22 (depending on the set of task assigned to each *step*). As it can be seen, in this case, five Use Cases and one *extend* relationship have been identified.

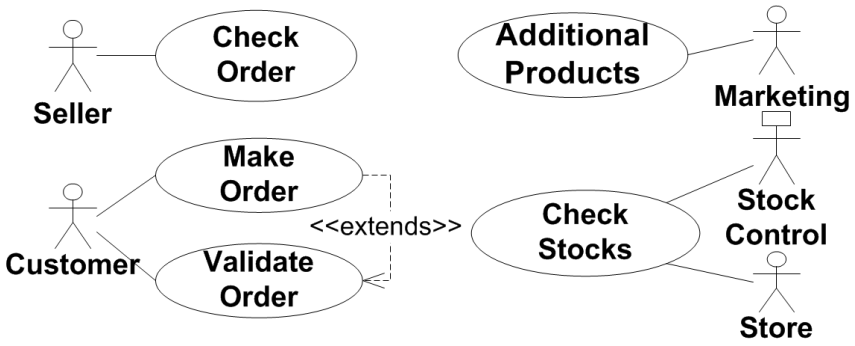


Figure 2.22: Use Case model derived from the online-shop business process using Dijkman, et al.'s approach.

The mappings defined in this approach facilitates the derivation of Use Cases from the business, generating functional requirements models aligned with the business with less effort and with at a lower risk of making misinterpretations. However, in the defined mappings, only *extend* relationships can be identified. *Include* relationships cannot be detected nor derived to the Use Case models. The identification of this kind of relationship is essential for generating better aligned requirements models. Thus, for example, as Figure 2.16 shows, there is an *include* relationship between the *Check Order* and *Addition Products* Use Cases that has not been identified. Furthermore, there are other situations in the business process models that are not treated, such as exceptions flows or sub-processes. This implies that, once the Use Case model has been generated, the business processes have to be reanalysed in order to deal with such situations.

BPMN-Based Specification of Task Descriptions

De la Vara and Sánchez (2009) identified that there is a gap between business processes and functional requirements that must be bridged in order to correctly specify the functional requirements of a system. To that end, they defined a methodological guidance to properly specify functional requirements from business processes.

This approach consists of two stages: business process enrichment and functional requirements specification. In the first stage, the business process models are analysed and labelled according to the system support that they will have. These labels are assigned to each flow object detailed in the processes, which are: *O* (out of the

system), if it will not be part of the system; *IS* (controlled by the system), if the system will be in charge of its execution; or *U* (controlled by a user), if it will be executed by a person. Next, the requirement engineer and the stakeholders have to identify the flow objects that will be executed consecutively and connect them by means of a *consecutive flow* (which is a new BPMN element defined by the authors). This allows them to control the granularity of each feature.

In the second stage, the functional requirements are elicited from the enriched business process models and specified by means of Task Descriptions (Lauesen, 2003). A Task Descriptions is a textual template whose purpose is to specify adequate system support for business tasks. The content of the templates is derived by following the structure and automation of the business processes. Specifically, each set of the business tasks connected by *consecutive flows* are specified in a Task Description. Thus, the granularity of all the functional requirements is homogeneous.

For example, assuming that all the business tasks related to checking an order, of the business process for handling orders 2.10, would be annotated with the *consecutive flows* element, then a Task Description documenting this functionality would be derived from them. Table 2.3 shows an excerpt of this Task Description. As this table shows, the use of this template allows engineers to perfectly document the business process from which the functionality was derived, the trigger starting the functionality, the input and output documents, and the sequence of actions that should be followed.

Although this approach reduces the gap between business processes and functional requirements, the derivation of the requirements is made manually. Also, the derived requirements are detailed in natural language. This specification is helpful for the requirements engineers and architects, but cannot be reused by tools assisting them in the development phases. In addition, this approach does not guide in the identification of the relationships between functional requirements. Thus, applying this approach to the online-shop business process, five Tasks Description would be derived (just like five Use Cases were identified with the previous approach). However, the relationships between them would not be detailed, which could lead to integration problems between functionalities and business-IT misalignment.

Table 2.3: Excerpt of the *Check Order* functionality detailed in the Task Description template.

Task Description: Check Order			
Business Process: Handle Orders		**Role:** Seller	
Subtasks: Decide whether accept or reject an order, get products that can be interesting to the customer			
Trigger: Order reception			
Input		**Output**	
Data Object	State	Data Object	State
Order	New	Order	Accepted/Rejected
		More Products	New
User Intention		**System Responsability**	
User Intention		System Responsibility	
1. Receive Order 3. Check Customer Data ...		2. Show new order ...	

Transforming an Enterprise Model into a Use Case Model

Siqueira and Silva (2011) indicates that although there are some approaches for requirements specifications, these approaches require an indepth domain knowledge. Also, the requirement engineers has to manually refine the business information into the requirements specifications. Therefore, they propose a set of heuristics to semi-automate the transformation of the business processes to the requirements specification using Model-Driven Engineering concepts.

The most important heuristics to transform business process models into Use Cases defined in XML are the following:

- Each *pool* or *lane* is mapped to an actor.
- Each business process is transformed into an Use Case.
- All the actors involved in the execution of the business process are participant in the Use Case.
- The business tasks defined in the business process are actions of the Use Case.
- The succession between business task reflects the actions order.
- The splitter and merger gateways in the business processes are mapped to

conditional statement in the Use Case.

This approach facilitates the derivation and specification of the Use Cases of a system, in XML, from the business processes of an organization. Thus, using the online-shop running example, a Use Case per business process would be derived. At a more detailed level, for example, from the process for handling order 2.10, a Use Cases would be generated documenting the flow of actions that should be followed, i.e. first, the customer has to make the order; second, the seller has to check the customer data; then, he/she has to check the order data, etc. Thus, a complete Use Case specification would be generated.

This approach facilitates the generation of Use Case models with less effort and reducing the risk of making misinterpretations. However, the definition of a Use Case covering a complete process can lead to very large and complex requirements, as is stated in (Dijkman & Joosten, 2002). Furthermore, this approach does not define how to identify the relationships between Use Cases. Thus, the Use Cases identified for the online-shop would be independent. This situation can lead to integrations problems.

Analysis and Discussion

In this section, we have evaluated approaches focused on semi-automatizing the derivation of requirements from the business information with the aim of identifying what requirements and what interdependencies between requirements they can derive. This review conducted us to achieve the goal seven of this thesis.

The approach defined by (Dijkman & Joosten, 2002) facilitates the derivation of Use Cases from business processes. In addition, it indicates how to identify some *extends* relationships. However, there are other relationships (such as include) or situations (such as exception flows or sub-process) that are not treated. (De la Vara & Sánchez, 2009) and (Siqueira & Silva, 2011) detail how to derivate requirements specifications from business processes, but they do no indicate how to identify the relationships between the derived requirements. The absence of approaches for eliciting these relationships leads to the generation of incomplete requirements mod-

els that have to be completed by the requirements engineer analysing the business information.

In addition, (Dijkman & Joosten, 2002) and (De la Vara & Sánchez, 2009) indicate that the Use Case granularity has to be controlled in order to not derive very complex, nor very simple requirements. However, they do not precisely indicate when a Use Case should start or end, in relation to the business process from which it is extracted. These points should be identified by the requirements engineer. Therefore, the granularity of the derived Use Cases highly depends on engineer experience.

Finally, a very important aspect is that these approaches do not indicate how to derive the relationships between functional and non-functional requirements. These relationships, or are not elicited, or are manually identified by the requirement engineers analysing the business information. Therefore, part of the effort saved semi-automatizing the derivation of the functional requirements is wasted analysing the business information to identify these interdependendencies. One of the goals of this thesis is to also semi-automate the identification and derivation of these relationships from the business models.

2.5 Basing the patterns selection in the requirements information

In the architecture design, the system requirements and their interdependencies should be analysed in order to evaluate the impact and importance of each requirement. Thus, the Architectural Significant Requirements (ASRs) can be identified. These requirements are used by the architect to design an architecture facilitating the fulfilment of both functional and non-functional requirements.

Currently, there are a number of approaches assisting the architect in the design of the architecture. Some of these approaches link quality models with architectural patterns. So that, once the ASRs have been identified by the architect, it is easier to select the architectural patterns that can be applied to achieve the ASRs. The most

important approaches, which are detailed in the following sections, are (Wojcik et al., 2006), (Bachmann, Bass, & Klein, 2003) and (Kim et al., 2009).

2.5.1 Attribute-Driven Design (ADD)

Attribute-driven design (ADD) (Wojcik et al., 2006) is one of the best known approaches in this field. In ADD, a software architecture is defined in a recursive process based on quality attribute requirements. This process guides the architect in the identification of the subsystems, the *architectural driver requirements* (or architectural significant requirements) impacting each subsystem, and in applying architectural tactics or patterns to satisfy them.

The identification of the *architectural driver* is based on the analysis of the *attribute primitives* which are the non-functional requirements that high influence in the definition of the architecture. The selection of each *attribute primitive* is based on its importance for satisfying a quality attribute and its collateral effects on other requirements. Therefore, to make this selection the architect has to have a perfect knowledge of the interdependencies between requirements.

Then, the architect, in accordance with his/her knowledge, analyses the architectural patterns and/or tactics in order to identify which of them facilitate the achievement of the architectural driver and the quality attributes. The selected patterns identified are applied by assigning responsibilities to the subsystems according to the functional requirements associated with the architectural drivers.

Although ADD allows the architecture to be defined based on quality attributes, the entire requirement analysis and decision making effort falls on the software architect. In order to reduce some of this effort, it has been proposed that one could use *reasoning frameworks* (Bachmann et al., 2003), (Bachmann et al., 2005).

A *reasoning framework* is an application which contains a quality model implementing a specific quality attribute theory. Also, it contains mechanisms to evaluate which patterns facilitate the achievement of the architectural drivers related to the quality attribute that it implements. As result, this framework indicates the set of patterns that can be applied. Then, the architect has to evaluate these patterns to

decide whether or not to apply them taking into account, inter alia, the functionalities to which apply them and other quality attributes that also constrain these features. Again, to select the pattern to apply, the architect has to have a perfect knowledge of the requirements relationships.

Applying this approach to the online-shop running example, the architect would identify, analysing the requirements, that the non-functional requirement indicating that the system has to be easy to changes has interdependencies with all the functional requirements. Therefore, it would be selected as architectural driver and used in the *reasoning framework* implementing the theory for the *Maintainability* quality attribute. This framework could indicate that the *Layer* or *Bypassed Layer* pattern can be used to satisfy it. Then, taking into account that some functionalities are also restricted by the *Time Behaviour* attribute, the architect would select the *Bypassed Layer Pattern* to not harm this requirement.

Therefore, this approach and the reasoning frameworks greatly facilitate the design of an architecture meeting the quality attribute but, for the successful completion of the process, the correct identification and analysis of the requirements interdependencies is essential.

2.5.2 Quality-Driven Architecture Development

Quality-Driven Architecture Development (Kim et al., 2009) is a systematic approach for embodying non-functional requirements into software architecture using architectural patterns and tactics. In this approach, architectural patterns are selected based on a given set of non-functional requirements, and the selected patterns are composed to produce a complete pattern. The complete pattern is then instantiated to create an initial architecture of the application into which the non-functional requirements are embodied.

A complete specification of each pattern is detailed using a UML extension called Role-Based Metamodeling Language (RBML). This extension models each pattern using two diagrams: a Structural Pattern Specification (SPS) used to define its structural details, and an Interaction Pattern Specification (IPS) used to define its

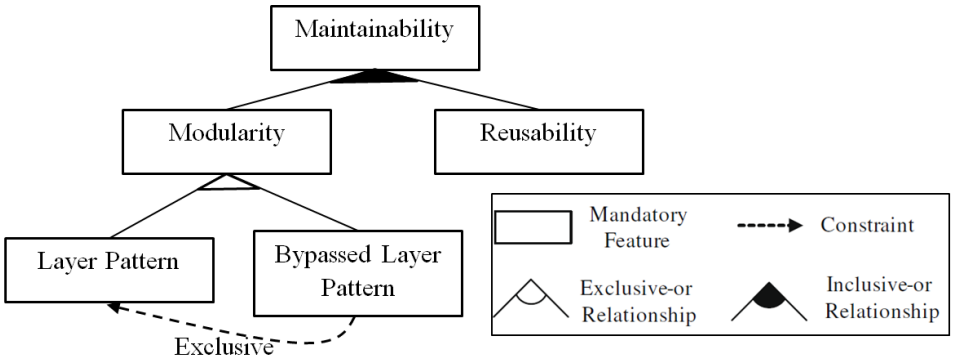


Figure 2.23: Feature model documenting the patterns for the *Modifiability* quality attribute.

behaviour.

Each pattern and its variations are detailed in a quality model based on feature modeling (Czarnecki & Eisenecker, 2000). This model, first, details the patterns for satisfying specific quality attributes and, second, documents the relationship between patterns (i.e. if two or more patterns have obligatorily to be combined, ought to be combined, or are mutually exclusive). The architect, once he/she has a perfect knowledge of the requirements and their interdependencies, uses this model to identify patterns than can achieve the quality attributes of the system and to also consider what patterns select depending on whether they must or should be combined, or excluded.

Figure 2.23 shows an excerpt of the feature model that can be used to select patterns for the online-shop running example. As can be seen, for satisfying the *Modifiability* quality attribute two different patterns can be applied (*Layer* and *Bypassed Layer*), which are mutually exclusive. In this case, because some requirements that must be modifiable must also take into account the response time constraint, the architect would select the *Bypassed Layer* pattern. For the architect to be able to make this selection, he/she has to have a perfect knowledge on the interdependencies between de modifiability requirement and the functional requirement, and between these and the *Time Behaviour* requirement. If these relationships are not modeled, he/she would have to acquire it by analysing the requirements models. Acquire this knowledge is a very complex and prone to failures activity.

Analysis and Discussion

Nowadays, there are approaches guiding architects in the architectural decisions making. In this section, we have evaluated the most important proposals in order to identify how they assist architects, what knowledge on the requirements and their interdependencies architects should have in order to correctly apply them, and if they provide any assistance or tool facilitating the analysis of the requirements. The review of this approaches and the identification of such information led us to achieve the objective ten of this thesis.

Attribute-driven design defines a process guiding the architect, first, in the analysis and identification of the architectural significant requirements and, second, in the selection of patterns meeting those requirements. The analysis and selection of the patterns is supported by *Reasoning Frameworks*, thus facilitating this step. However, the analysis and identification of the ASRs, and their interdependencies, is done manually by the architect.

Quality-Driven Architecture Development details a set of models for documenting the patterns that can be used for meeting a specific quality attribute. Although these models facilitate the architect to analyse all the patterns that can be used to satisfy specific requirements, to select a pattern the architect must perfectly know the interdependencies of the functionalities, on which to apply the pattern, with other quality attributes. Information that has to be manually identified analysing both kinds of requirements.

Therefore, to properly apply these approaches, the architect must have a perfect knowledge of the requirements and their interdependencies. If these interdependencies are not explicitly detailed, he/she has to manually identify them. This identification, first, is a complex task since it requires a great knowledge about the requirements engineering methods and, second, is prone to misunderstandings. Any misunderstandings can lead to the selection of incorrect patterns preventing, rather than facilitating, the achievement of the requirements.

2.6 Summary

In this chapter we have made a thorough review of different proposals supporting specific activities and phases of the BDD discipline. This review has revealed that they represent a very important advance in the development of IT systems aligned with the business, but there are still some areas that can be improved for getting even a better alignment with less effort.

Firstly, the methodologies centred in the enterprise modeling perfectly guide in the analysis and documentation of the business processes and goals. Nevertheless, we have identified the following areas that could be improved to better document the business information:

- The approaches supporting the analysis and modeling of the business information only allow one to document which business goals motivate each business process. They do not support the modeling of the constrains imposed by the goals on the processes.
- The approaches focused on the modeling of the interdependencies between business goals and processes do not define how to model the interdependencies with non-operational business objectives.

Therefore, when the system requirements are derived from these models, engineers have to perform a thorough analysis of the generated models in order to identify these relationships. It would be desirable to be able to model these interdependencies in order to create more complete business models.

Secondly, the methodologies guiding in the identification and modeling of the requirements facilitate the documentation of functional an non-functional requirements of the system. Nonetheless, there are some parts of these approaches that could be improved in order to detail more complete requirements models with less effort. The following are the detected areas that could be improved:

- The notations for the requirements specification natively do not support the modeling of interdependencies between functional and non-functional requirements.

- The proposals semi-automating the derivation of requirements from the business do not support the identification of interdependencies between functional and non-functional requirements. Also, there are some situation, leading to the definition of *include* or *extends* relationships between Use Cases, that they neither detail. Therefore, most of these relationships are not identified.

- The approaches focused on modeling the interdependencies between functional and non-functional requirements only allow engineers to model them in natural language. Thus, they cannot be reused by tools assisting them in the subsequent development phases. In addition, some of these proposals do not base the identification of these interdependencies on the business information. Therefore, it is more complicated to achieve the business-IT alignment.

Finally, the proposals assisting architects in the design of systems aligned with the business make use of these interdependencies; first, to identify the impact of each requirement and, second, to know the exact functionalities on which apply the architectural patterns facilitating its achievement. Since these interdependencies are not explicity modeled, architects have to deeply analyse all the generated models to identify them. This task involves a lot of effort and is prone to misinterpretation that may lead to select patterns preventing the achievement of the requirements rather than facilitating it.

Therfore, based on the areas that we have been identified that can be improved, this thesis proposes a comprehensive methodology supporting and guiding in the modeling of these interdependencies from the early development stages, and deriving them to the architectural design phase. To that end, first, we have detailed an extension of BPMN allowing the documentation of the interdependencies between business processes and business goals (both operational and non-operational); second, we have defined a set of patterns and rules semi-automating the derivation of the functional requirements and their relationships from the annotated business models; third, we have defined some UML profiles allowing engineers to model the relationships between functional and non-functional requirements; and, finally, we have detailed a metamodel and a set of ATL transformations that, using the documented interdependencies, facilitate the architectural decisions making.

Chapter 3

Modeling Business and Requirements Interdependencies

"I'm a great believer in luck, and I find the harder I work the more I have of it."

Thomas Jefferson

The explicit documentation of the interdependencies between business elements and among requirements is crucial for architects to design a system perfectly integrated and aligned with those needs and requirements. This chapter details the solution proposed in this thesis for the specification of this information. This solution guides in the definition of these interdependencies both at the business and at the requirements level, and in their derivation from the business analysis stage to the architectural design phase.

The following sections detail the proposed solution. Section 3.1 shows an overview. Section 3.2 details the process guiding in the modeling and derivation of the interdependencies. Section 3.3 specifies the contributions developed to model the interdependencies between business elements. Section 3.4 details the profiles and tools implemented to document the interdependencies between requirements. Section 3.5 specifies how to reuse the modeled interdependencies to identify the Architectural Significant Requirements. Finally, Section 3.6 explains how the defined contributions integrate with approaches guiding in the architecture design.

3.1 Overview of the proposed solution

This thesis details a comprehensive methodology for making the business and the requirements interdependencies explicit from the early development stages and for deriving them to the architectural design stage, with the aim of facilitating the definition of architectures better integrated and aligned with the business.

Figure 3.1 shows an overview of the proposed methodology. As can be seen, this figure is divided in two parts. The first part, *Design Process*, details the steps that are usually followed in the analysis and design of an IT system. To these steps, we have added three new activities, highlighted in blue in the figure, focused on the documentation and analysis of the interdependencies. These steps and activities are grouped by the software development phase in which they are performed. The second part, *Tools*, shows the notations and tools usually used during the realization of the development activities. The extensions, profiles and tools developed for supporting the new defined activities are highlighted in blue.

Specifically, to the business analysis and requirements analysis stages, two new activities focused on the modeling of the business and requirements interdependencies have been added. Furthermore, another activity has been added, to the architecture design stage, to guide the architect in the analysis of these relationships. These activities and the specific tools supporting each of them are:

- **Model the Contextual Information.** In this activity, the business processes are annotated with information on the relationships between their tasks and different elements of the business, such as the relationships between business processes and business goals, the legacy system already supporting some of the business tasks, and the business tasks that will be covered by each system functionality.

  This activity is supported by the following extensions and tools:

  - *BCPI Extension.* An extension of BPMN 2.0 defining the elements needed to model the business interdependencies.
  - An algorithm guiding in the modeling of some of the new defined elements.

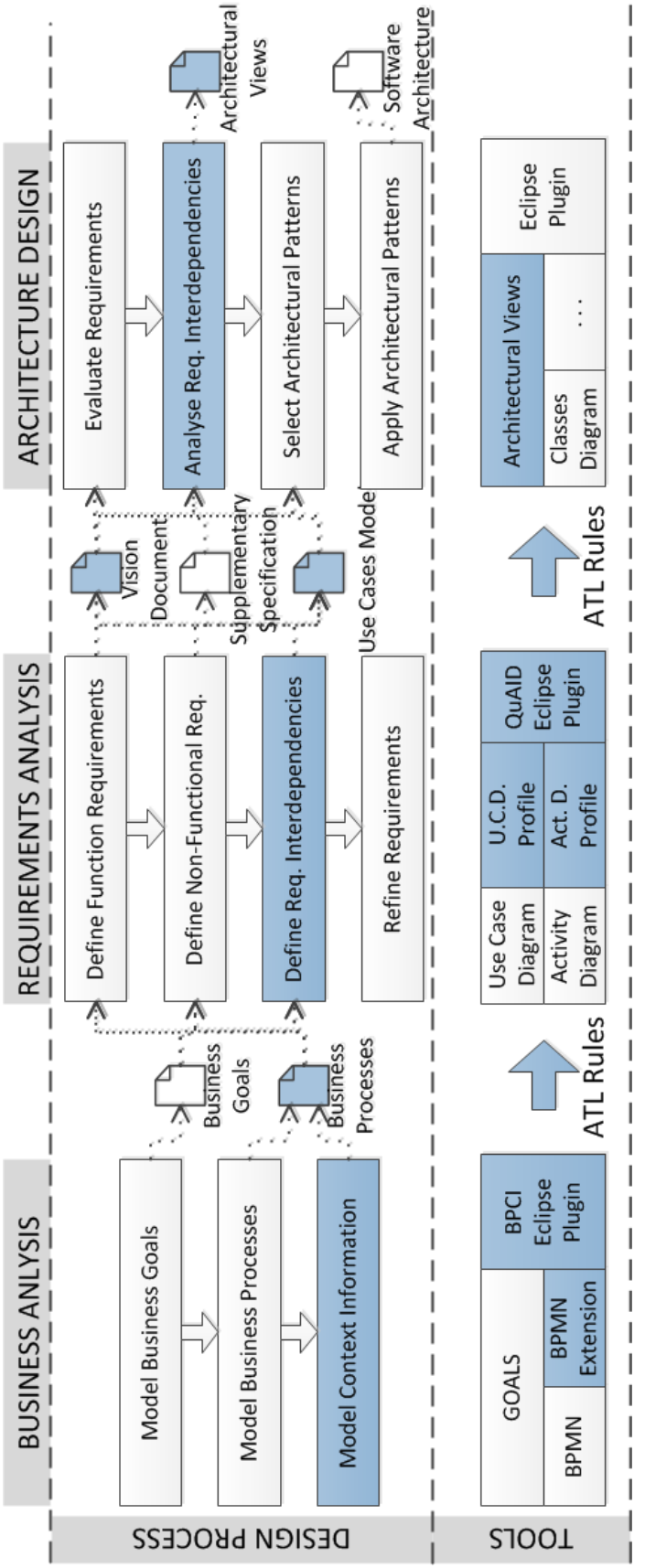


Figure 3.1: Overview of the proposed solution.

- *BCPI Eclipse Plugin.* An extension of the Eclipse plugin BPMN2 Modeler (*Eclipse BPMN2 Modeler*, 2014) facilitating the modeling of the elements defined in the BPCI Extension.

- **Identify and Define the Requirements Interdependencies.** In this activity, the relationships between the system requirements are specified. To this end, we have defined a set of patterns defining the commonest situations in the business processes (annotated with the BPCI Extension) leading to the identification of relationships between functional requirements and between functional and non-functional requirements. Once identified, these interdependencies are modeled in Use Case and Activity diagrams.

  This activity is supported by the following profiles and transformations:

  - *UC-QuAID Profile.* A profile for UML Use Case diagrams allowing engineers to model the relationships between functional and non-functional requirements.

  - *AD-QuAID Profile.* A profile for UML Activity diagrams for modeling the interdependencies between functional and non-functional requirements at a lower granularity level.

  - *QuAID Eclipse Plugin.* An extension of the Eclipse plugin Papyrus (*Eclipse Papyrus*, 2014) implementing the previously defined profiles.

  - A set of ATL rules automating the application of the patterns for deriving the systems requirements and their interdependencies from the business models.

- **Analyse the Requirements Interdependencies.** During this activity, the requirements and their relationships are analysed with the aim of identifying the impact of each requirement and, thus, discovering the Architectural Significant Requirements. This information, along with the annotated requirements models, is subsequently provided as input to other tools assisting the architect in the decisions making process.

  This activity is supported by the following metamodels and transformations:

- A metamodel (*Architectural Views*) for documenting, for each system view, the impact of each non-functional requirement on the elements of that view.
- A set of ATL transformation rules semi-automating the derivation of the *Architectural Views Model* from the annotated requirements models.

Throughout the previously defined activities the business and requirements interdependencies are properly modeled and documented in order to provide more information to architects and, thus, reduce the risk of misinterpretation during their analysis.

3.2 Process for modeling, deriving and analysing the interdependencies

This section focuses on giving specific details about the activities listed above, how they are applied during the software development, and how they integrate. To that end, the running example is used here to illustrate how they are performed and how the defined extensions/profiles and developed tools are used in each of these activities. The subsequent sections specify all the capabilities each of these contributions.

3.2.1 Modeling Context Information

In analysing the organization's business, one identifies and models information about its objectives, needs, legacy systems, and processes. As indicated above, these elements are usually documented in specific models oriented to perfectly detail their characteristics. Thus, for the online-shop, the Figure 2.7 show the *Business Goal Model* and the Figure 2.10 document the business process for placing online orders.

The objective of this activity is to detail the relationships between these elements. These relationships are annotated on the business processes modeled with BPMN. These models are used as a basis on which to model business interdependencies since

BPMN provides a number of communication facilities (BPMN, 2014) that are reused to facilitate discussion about these relationships.

In order to be able to model these relationships, we have extended the BPMN 2.0 notation. This extension, called *BPCI Extension*, defines the following elements:

- To model the interdependencies between goals and business processes and, thus, facilitate the realization of more complete business models, the *Quality Attribute* element has been defined. This element has been defined to document the business tasks constrained by each quality attribute.
- To document the interdependencies between the system and legacy systems and, thus, simplify their analysis in the subsequent development stages, the *Legacy System* element has been added. This element details which business tasks have to interact with each legacy system and when.
- To document in an early stage the business tasks that will be supported by each system functionality and, smooth their derivation to the further development stages, the *Business Use Case* element has been defined.

Below, the most important characteristics of these elements, and how they are applied to the online-shop example, are detailed. Figure 3.2 shows the process for handling orders annotated with these elements. These elements can be annotated in any BPMN model independently of its content and the ordering of its business objects.

The *Quality Attribute (QA)* element allows one to group the tasks of a process that must meet certain business quality objectives or requirements. In Figure 3.2, the online-shop process has been annotated with three relationships with business goal. The first specifies that the entire business process is related to the goal of ease to adapt. The second documents that the business tasks managing customer data (*Make Order*, *Validate Order*, *Check Customer Data*, *Check Order*) have to be secure. Finally, the third indicates that the *Check Customer Data*, *Check Order* and *Search for Additional Products* business tasks are constrained by the objective which limits the time in which they must be completed. These annotations make it easy for the engineer to see which of the process's specific tasks must fulfil each

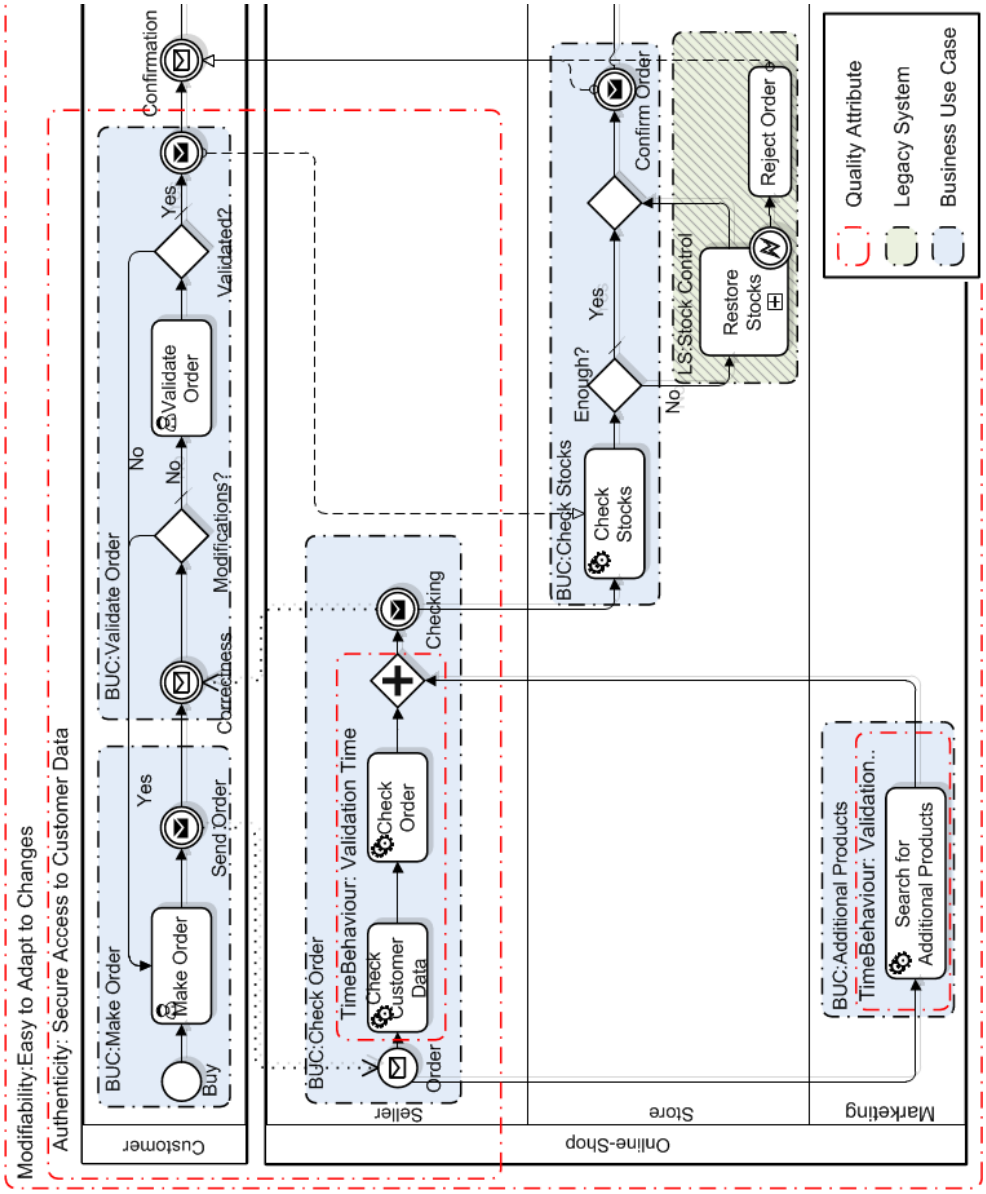


Figure 3.2: The order-placing process annotated using the defined extension.

goal.

The *Legacy System (LS)* element facilitates the grouping of the business tasks already being supported by a legacy system, even though they are modeled in a lane because a certain role is responsible for them. The resulting groupings are then used to derive the relationships between the new system and the legacy systems. Thus, these relationships can be easily analysed in the subsequent development stages to design better integrated IT systems. In Figure 3.2 for instance, this new element is used to indicate that the *Restore Stocks* sub-process is already being supported by the *Stock Control* legacy system. In this way, one can easily identify which functionalities and when (after checking the stocks) the legacy system should be invoked and, thus, design a system taking this interactions into account.

Finally, the *Business Use Case (BUC)*[1] element allows process tasks to be grouped according to the system functionalities required to support them. Besides to being able to identify the BUCs manually, a process has been defined guiding in their identification. Thus, BUCs at a similar granularity are modeled. This process is detailed in Section 3.3.2. Five BUCs are modeled in the online-shop's business process. The first BUC, for example, groups the tasks the user performs to place the initial order. These grouping, firstly, show a first approximation of the system functional requirements that can be easily discussed between the business expert and the requirements engineer; secondly, are reused in the subsequent stages to derivate the Use Cases and their relationships.

By means of the above annotations, the engineer can reflect the interdependencies between different elements of the business, documenting them and facilitating their consideration in future development phases. In addition, these annotations may also be automatically dealt by tools that aid certain development activities, as is detailed in the next section.

[1]The term Business Use Case is used here, even though it is similar to the System Use Case term (Cockburn, 2000), because it groups business tasks that should be refined in order to detail the system tasks.

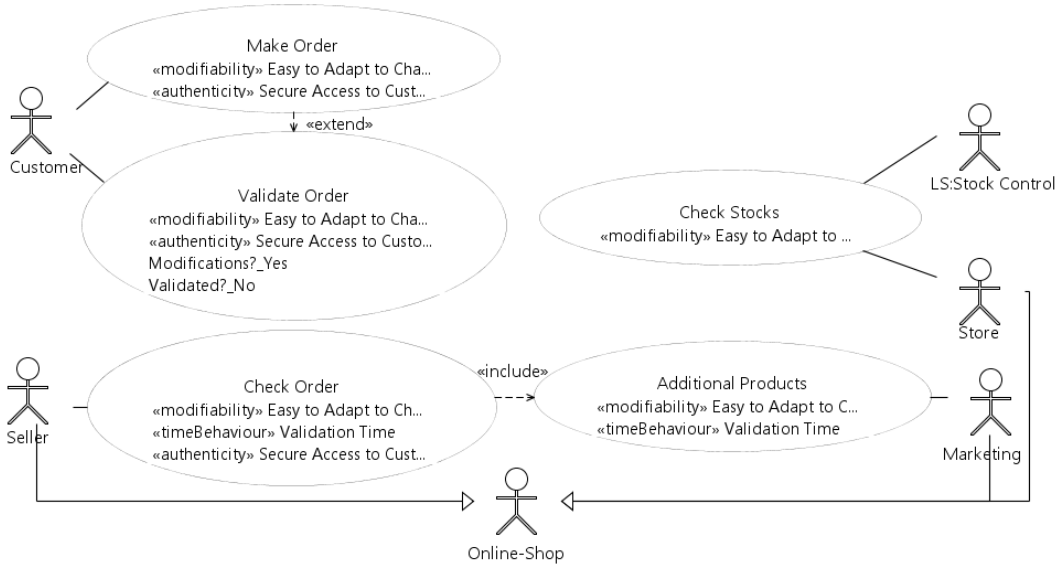


Figure 3.3: Annotated Use Case Diagram extracted from the online-shop process.

3.2.2 Define Requirements Interdependencies

The functional and non-functional requirements are defined on the basis of the information modeled in the business. The functional requirements are modeled and specified as Use Cases in the *Vision Document* and in the *Use Cases Model*, and the non-functional requirements in the *Supplementary Specification Document*.

In this activity, the relationships between Use Cases and between functional and non-functional requirements are documented. Besides being able to deduce these relationships manually, we have also defined a series of patterns to guide the engineer in their identification. These patterns are based on the information annotated in the business processes. They define the commonest situations in the annotated business processes leading to the definition of relationships between Use Cases. These patterns can identify relationships between Use Cases and actors, between Use Cases and non-functional requirements, between Use Cases and legacy systems, and include or extend relationships. Section 3.4.3 details all the defined patterns. Furthermore, in order to automate the application of these patterns, some ATL rules have been defined. This rules are detailed in the Section 3.4.4.

Applying these patterns on the annotated process, depicted in Figure 3.2, which previously was used to derive the system Use Cases from the BUCs, for example an

extend relationships would be identified between the *Make Order* and the *Validate Order* Use Cases, since the *Validate Order* BUC contains an exclusive gateway from which an alternative branch (grouped by the *Make Order* BUC) starts and the fourth patterns indicates that the BUC in the alternative branch *extends* the normal flow of the BUC containing the gateway. In addition, a relationship between the *Check Order* Use Case and the *Time Behaviour* quality attribute is identify since some actions covered by the BUC *Check Order* are annotated with this quality requirement.

Natively, the interdependencies between non-functional requirements and Use Cases cannot be modeled. Therefore we have defined a profile for the UML 2 Use Case Diagrams, called *UC-QuAID Profile*. This profile defines stereotypes that extend the *Extension Points* metaclass to enable these relationships to be modeled. It was decided to extend this metaclass because it maintains the readability of the diagram when a large number of relationships are modeled. These stereotypes were defined taking into account the ISO/IEC 25010 quality model (ISO/IEC, 2011) and a complete specification of the profile is detailed in Section 3.4.1.

Figure 3.3 shows the Use Case diagram extracted from the information annotated in the online-shop business process, illustrated in Figure 3.2, applying the defined patterns. The diagram shows five Use Cases, all annotated with the *Modifiability* non-functional requirement; the *Make Order*, *Validate Order* and *Check Order* Use Cases have to meet the *Authenticity* constraint; and the *Check Order* and *Additional Products* functionalities must also fulfil the *Time Behaviour* requirement. Thus, the relationships between each Use Case and non-functional requirement are easily documented and visualized. In addition, these annotations have been especially designed to be reused by tools semi-automating the architectural decision making and assisting in the subsequent development stages.

Afterwards, in order to be able to detail at a finer granularity the relationships between Use Cases and non-functional requirements, we have also defined a profile for UML Activity diagram, called *AD-QuAID Profile*. This extension defines stereotypes for grouping the actions that have to fulfil each non-functional requirement. This profile, and their stereotypes, is detailed in Section 3.4.1.

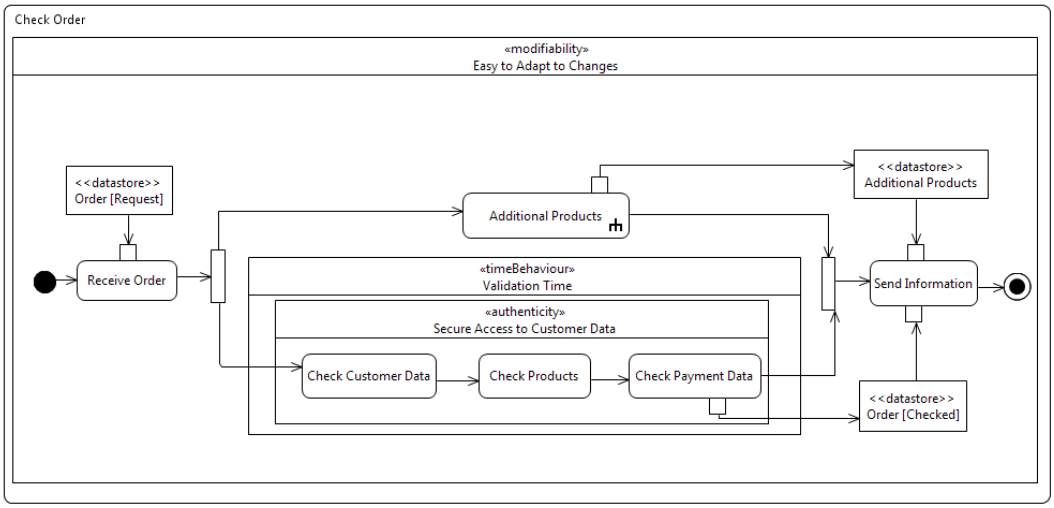


Figure 3.4: Activity diagram of the Check Order Use Case.

Figure 3.4 shows the Activity diagram for the online-shop's *Check Order* Use Case. As in the Use Case diagram, it has three relationships annotated with non-functional requirements. The first is that all of the Use Case's actions must satisfy the *Modifiability* requirement. The second indicates that just the *Check Customer Data*, *Check Products* and *Check Payment* actions must satisfy the *Time Behaviour* requirement. The third reflects that the actions managing customer data (*Check Customer Data*, *Check Products* and *Check Payment*) have to be secure.

The above patterns and profiles facilitate the identification and documentation of the relationships between Use Cases and between Use Cases and non-functional requirements to the point of detailing exactly which actions should satisfy each non-functional requirement. In this way, with little effort the architect can identify and analyse that information in the system design phase, thereby reducing the risk of failure due to the misinterpretation of some relationship.

3.2.3 Analyse the Requirements Interdependencies

In designing the system, the architect should get a complete picture of the requirements and how they relate to each other. Thus, he/she can know which requirements impact more on the system. This information is needed to select the most appropriate architectural patterns. Each pattern may affect different requirements posi-

Table 3.1: Benefits and liabilities of the architectural patterns.

Pattern	Benefits	Liabilities
Layer	Security Maintainability Adaptability Developed by multiple teams	Efficiency Development Complexity
Pipes And Filter	Maintainability Efficiency	Usability Security Reliability

tively or negatively. For example, as is shown in Table 3.1, derived from (Harrison & Avgeriou, 2007), the *Layer* pattern affects positively to the adaptability and ease of change requirements, but negatively to the performance requirement. Therefore, in having a complete picture of the system requirements, the architect can make architectural decisions seeking a balance to satisfy all the requirements.

In this activity, the architect analyses the relationships between functional and non-functional requirements through the diagrams annotated with the QuAID profiles. Thus, he/she obtains information on the different views of the system and how each quality requirement affects each view. Taking this picture as basis, the architect begins to identify which requirements are the most architecturally significant.

In order to provide this picture in a simpler way to the architect, we have defined a metamodel for documenting it. This metamodel allows one to define a number of architecturally significant views for a system, such as the one defined in (Kruchten, 1995). For each view, a set of important attributes for the view, together with their values, can be defined. Similarly, for each view the quality attributes affecting it and how they affect the different attributes can be documented. Section 3.5 details the specified metamodel.

This model can be semi-automatically derived from the annotated Use Case and Activity diagrams. Thus, the *Use Case View* can be generated from the Use Case diagram. In this view, one could consider the number of actors, Use Cases, and their relationships as its attributes. Similarly, from the annotated Activity diagram one can get information to generate the *Process View* documenting the number of

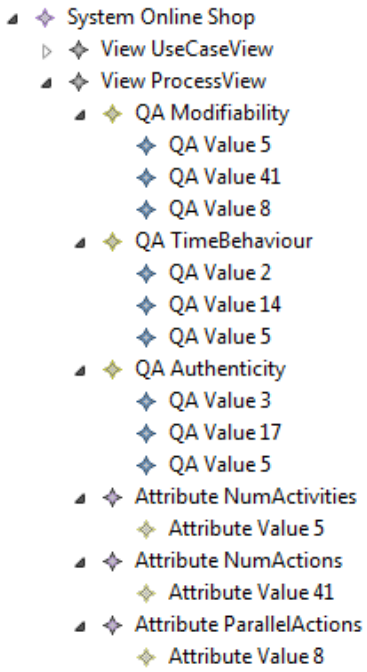


Figure 3.5: Extract of the Process View of the online-shop example.

activities, actions, parallel actions, or threads as its attributes, and how each quality attribute impact on them.

Figure 3.5 shows an extract of the *Process View* derived from the online-shop Activity diagrams. This view details that the system has five activities and forty-one actions, and eight of those actions are executed in parallel. This model also documents that this view is affected by the *Modifiability*, *Authenticity* and *Time Behaviour* quality attributes. The first one is annotated in all the activities and actions of the system; the second one is annotated in three activities, seventeen actions and five parallel action; and the latter is only annotated in two activities, fourteen actions, and five parallel actions.

This information is very useful to get a complete picture of how the non-functional requirements affect each architectural view. This information can be manually used by the architect for making decision, or provided as input to the tools assisting the architect in this process. Section 3.6 details the integration with these approaches.

If it is manually reused, in the online-shop views (figure 3.5), for example, one can easily see how the *Modifiability*, *Authenticity* and *Time Behaviour* attributes affect

the system. Taking these non-functional requirements and the patterns indicated in Table 3.1 into account, the architect could decide to apply the *Pipes and Filter* pattern, since it benefits both the *Modifiability* and *Time Behaviour* requirements. However, since the *Process View* indicates that most of the functionalities should be executed sequentially (only eight of the forty-one are executed in parallel), he/she would discard it because it only provides the desired benefits in parallel processing. Thus, given that *Modifiability* affects a greater number of Use Cases, activities, and actions, the architect may decide to meet this requirement applying the *Layer* pattern although it endangers the achivement of the *Time Behaviour* requirement.

Afterwards, using the annotated Use Case and Activity diagrams, the architect can evaluate at a more detailed level how to apply this pattern and how to select other patterns in order to meet the *Time Behaviour* and *Authenticity* requirements. The annotated Use Case diagrams provide a high level vision of the Use Cases together with the non-functional requirements that each of them has to achieve. The Activity diagrams give a finer grain view of which actions have to support each non-functional requirement. In this way, the architect can evaluate whether the desired benefits are really obtained by applying the desired pattern to that set of Use Cases and actions.

By observing the annotated diagrams of the online-shop, the architect is aware of which actions would be positively affected regarding the maintainability advantage provided by the *Layer* pattern, and which would be negatively affected by the performance liability. Thus, to achieve the desired performance, for example, a variant of the *Layer* pattern bypassing some of the layers (Harrison & Avgeriou, 2010) would be selected. Therefore, the defined models and annotations provide the architect a deeper knowledge of the requirements and their relationships with less effort and reduce the likelihood of misinterpretations

In this section, we have intuitively detailed the main contributions of this thesis and how they integrate to facilitate the modeling and analysis of the business and the requirements interdependencies in order to design IT systems better integrated and aligned with the business. The definition of this comprehensive process was the main goal of this dissertation. The following sections detail more specifically all the

capabilities of each contribution and how they meet the sub-goals of this thesis.

3.3 Modeling the Contextual Information

To model the interdependencies at the business level we have defined a set of models, guidelines and applications. These contributions are:

- The BPCI Extension for modeling what parts of the business processes are constrained by quality attributes, what are already supported by legacy systems, or what will be supported by specific functionalities of the system.
- A process guiding in the identification of the set of business tasks supported by each system functionality.
- The BPCI Eclipse Plugin, which extends the Eclipse plugin BPMN2 Modeler for implementing the previously defined extension.

These extensions and tools allow the business experts and engineers to perfectly model and document the interdependencies between the elements of the business. Moreover, these relationships are annotated so that they can be reused in the next development stage. The following subsections detail each of these contributions.

3.3.1 BPCI Extension

BPCI (**B**usiness **P**rocess **C**ontextual **I**nformation) is an extension of BPMN 2.0 allowing engineers to annotate the business tasks that are related to contextual information important for the development of a system. Thus, this information can be reused to make a more complete, and better aligned with the business, system specification and design. Figure 3.6 shows an extract of this extension with the most important elements (the whole extension can be seen in Appendix A). These elements are the following:

First, the native class extended by all the new elements is **Group**. This metaclass is extended because it allows one to group BPMN flow elements. This artifact is not constrained by restrictions of Pools and Lanes. This means that a *Group* can stretch

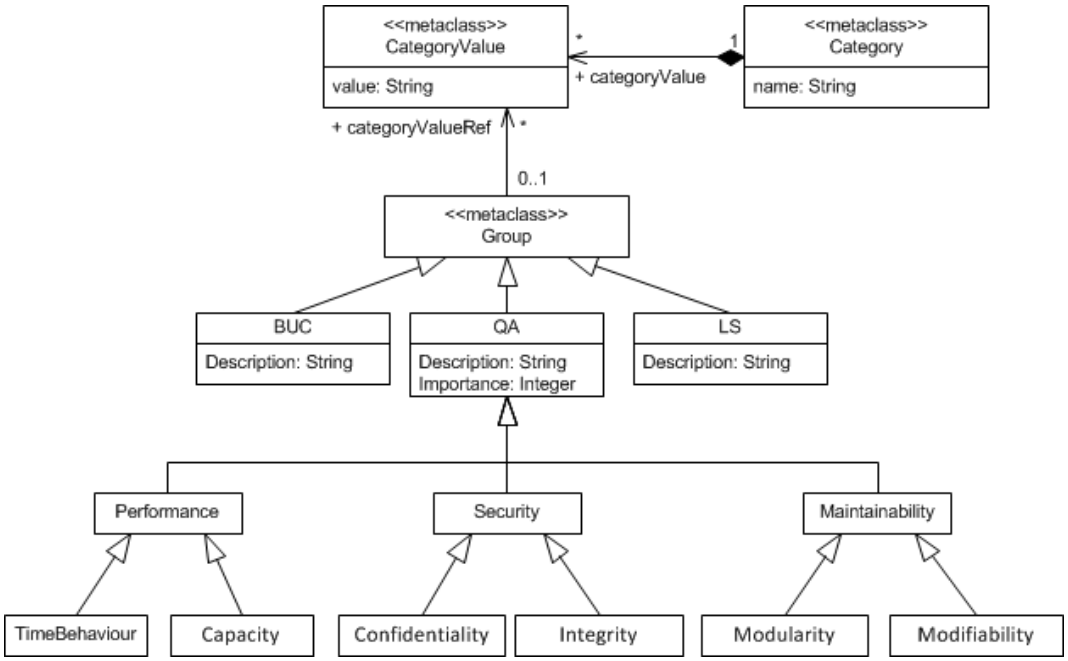


Figure 3.6: Extract of the BPCI extension.

across the boundaries of a Pool/Lane to surround BPMN elements. Simplifying the grouping of any kind of flow elements.

Moreover, the elements are grouped by a specific criterion that is documented by means of this artifact. This criterion is specified by a *CategoryValue* assigned to the attribute *CategoryValueRef* of each group. A *CategoryValue* consists of two different attributes: a *Category* and a *Value*. The *Category* is the name of a property and the *Value* is a specific value for that property. A (*CategoryValue*) is usually depicted in a group with the following nomenclature: *Category's* name, plus the delineator ":" and plus the specific value.

All the elements contained in a group inherit the *CategoryValue* assigned to that group. This value is assigned to the *CategoryValueRef* of each flow element. Therefore, these elements can subsequently be analysed and processed by transformation rules based on the *CategoryValues* assigned to them (as Section 3.4.4 details).

LS (Legacy System) extends the *Group* metaclass to group all the flow elements that are already supported by legacy systems. This element has two attributes: *Name* and *Description*. The attribute *Name* is inherited from the meta-

class and is used to indicate the legacy system's name. The attribute *Description* details the legacy system. The LS element is represented in a BPMN diagram as a rounded corner rectangle with a solid green line. Its *CategoryValue* consists of the *category LS* and the legacy system's name as value. The modeling of this information facilitates the identification of the interdependencies between the new system and legacy systems. Thus, the system design can take this relationships into account, improving its integration into the environment.

The **QA (Quality Attribute)** stereotype groups the flow elements constrained by a specific quality attribute. This element has three attributes: *Name*, which is inherited from the metaclass and specifies its name, *Description*, which details the QA specification; and *Importance*, which indicates the interest for the achievement of the quality attribute. A QA is modeled with a rounded corner rectangle with a solid red line. Its *CategoryValue* consists of the *category QA* and the value is the specific quality attribute to achieve.

In order to facilitate the modeling of more specific quality attributes, the QA element is extended with a tree of more concrete quality requirements. This tree is based on the quality model detailed in the ISO/IEC 25010 (International Standard Organization (ISO/IEC), 2011), which details a two level hierarchy defining characteristics and sub-characteristics. The sub-characteristics in turn refine the characteristics in more specific non-functional requirements. Figure 3.6 only shows the most important quality attributes or those used in the online-shop running example (Appendix A shows the complete extension). These attributes are:

- **Performance,** which details quality attributes relative to the amount of resources used under stated conditions. This attribute is decomposed in the following sub-characteristics:
  - **Time Behaviour.** Response and processing times and throughput rates of a product or system, when performing its functions.
  - **Capacity.** It specifies the maximum limits of a product or system.
- **Security.** Types and levels of authorization to system functionalities and data. This attribute is decomposed in the following sub-characteristics:

  - **Confidentiality.** Degree to which a system has to ensure that data are accessible only to those authorized to have access.
  - **Integrity.** Prevent of unauthorized access to, or modification of, computer programs or data.

- **Maintainability.** Level of effectiveness and efficiency with which a product or system can be modified by the intended maintainers. This attribute is decomposed in the following sub-characteristics:
  - **Modularity.** Degree to which a system or computer program is composed of discrete components.
  - **Modifiability.** Facility of modification of a system without degrading existing product quality.

The modeling of these elements is similar to the QA element, just only the *Category* of the assigned *CategoryValue* changes. The *Category* of these elements is the characteristic/subcharacteristic. Figure 3.2 shows the online-shop process for placing orders annotated with the *Modifiability* quality attribute. As can be seen, this annotation is composed by the characteristic name plus the specific quality restriction to achieve. Thus, in successive stages of the process detailed in this thesis, these annotations can be automatically processed by ATL transformation for identifying the interdependencies between the functionalities and specific quality attributes.

Finally, the element **BUC (Business Use Case)** extend the *Group* metaclass to group all the business tasks that have to be supported by a specific system functionality. This element has two attributes: *Name* and *Description*. The attribute *Name* (inherited from *Group*) indicates the BUC's name and *Description* details the requirement specifications. A BUC is modeled as a rounded corner rectangle with a solid blue line. Its *CategoryValue* consists of the *category BUC* and the functionality name. The modeling of this information facilitates the subsequent derivation of the system functional requirements models aligned with the business specification by means of ATL rules, as Section 3.4.4 details.

3.3.2 Process for the identification of Business Use Cases

This section details a process guiding the requirements engineer in the identification and modeling of the Business Use Cases. To that end, different rules defining what business tasks should be supported by each system functionality have been detailed.

This process is comprised of two steps: the definition of the support the new system is going to provide to each business task, and the identification of the system functionalities required to support that business tasks.

To perform the first step, the engineers and the business experts have to discuss what tasks are going to be *automated*, which are going to be performed with the *support* of the system (such as through forms, or partially automating it), and which are going to be done *manually*. Then, this information is annotated in the model. To that end, a task is annotated as *Service Task* if it is going to be automated, *User Task* if it is going to be supported, or *Manual Task* if it is going to be done manually.

In Figure 3.2, the kind of support that the system has to provide to each task is detailed. For example, the tasks *Make Order* and *Validate Order* have been defined as supported tasks, and the *Check Customer Data* and *Check Order* tasks have been annotated as automated.

The next step is the identification of the system functionalities required to support the process. For that purpose an algorithm is proposed. The algorithm ensures that Business Use Cases are identified at a proper granularity. It is based on the Cockburn's *Coffee-break* rule (Cockburn, 2000) and on the *Step* concept defined in Dijkman and Joosten (2002). Both rules indicate that a Use Case should be something that an actor can do in a single session, without interruption; and once completed, they could go to get some coffee. If the Use Case has multiple primary actors, or it takes days to complete, it's too big. The Listing 3.1 presents the pseudo-code of the proposed algorithm. This algorithm has not been semi-automated because it only details some guidelines that engineers should follow to identify the BUCs.

```
1. Begin with the first element of the business process
2. All consecutive flow elements are in of the same BUC
3. A BUC ends when:
 A. The control flow passes from one Lane to another
 B. There is a time transition between tasks. This is:
  I. There is a task marked as manual
  II. There is an intermediary event or a sub-process
  III. There is a communication with an external agent
       This indicates the start (input of data) or end
      (output of data) of the BUC
  IV. There is a task performed by a legacy system
 C. The end of the process is reached
```

Listing 3.1: Algorithm for the identification of Business Use Cases.

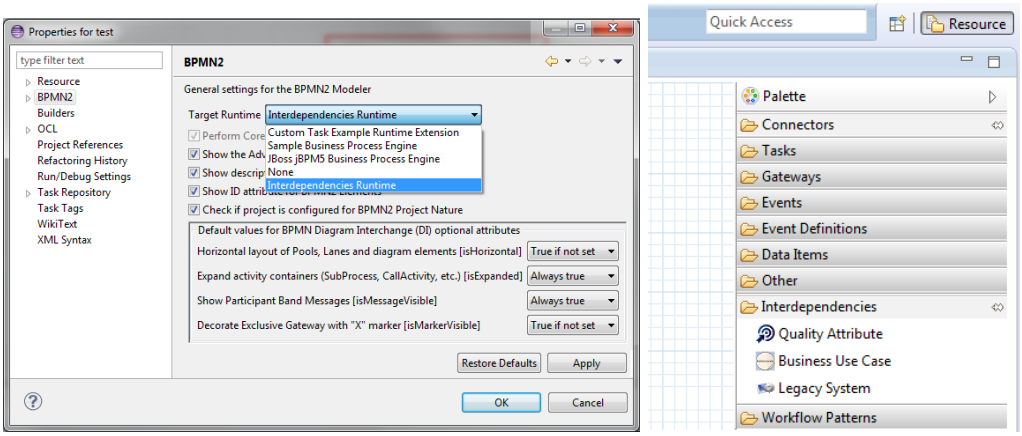
Figure 3.2 also shows the BUCs identified for the online-shop process. For instance, the *BUC:Check Order* groups the elements for checking the order until the result is sent to the customer.

Modeling the Business Use Cases in the BPMN diagrams provides the engineer with a first approximation to the system functionalities, and facilitates their discussion with the business experts. In addition, this process facilitates the definition of BUCs with a similar granularity, independently of the engineer experience.

3.3.3 BPCI Eclipse Plugin

BPCI Extension facilitates the modeling of the interdependencies between business elements. Likewise, the defined process for the identification of the Business Use Cases guides in grouping the business tasks according to the system functionalities that will support them. With the aim of facilitate the modeling and application of these contributions, we have extended the plugin for Eclipse BPMN2 Modeler. BPMN2 Modeler is a graphical modeling tool which allows creation and editing of BPMN diagrams. The defined extension is called **BPCI Eclipse Plugin**.

This extension has been developed so that its utilization is not mandatory. When a new project or BPMN diagram is created, the modeler can define in the Target Runtime whether he/she wants to use it or not, as Figure 3.7a shows. If he/she selects to use the *Interdependencies Runtime* extension, the IDE shows a new palette,



(a) Creation of new BPMN diagrams with the defined extension.

(b) Palette with the new defined BPMN elements.

Figure 3.7: Snapshots of the BPCI Eclipse Plugin for creating new diagrams and modeling the business interdependencies.

Interdependencies, for documenting the elements defined in the BPMN 2.0 extension, as Figure 3.7b depicts. Thus, once a business process has been modeled, or as it is modeled, the business expert or the requirements engineer can use the elements defined in the palette to easily annotate the interdependencies in the process.

Furthermore, for each element annotated, a properties view is presented to specify the value of the element's attributes. For a quality attribute, for example, he/she can detail its name, importance and description. Figure 3.8 shows the properties view for the *Modifiability* quality attributed annotated in the online-shop process for placing orders.

Finally, in the development of this extension some bugs of the plugin BPMN2 Modeler were identified. These bugs prevented the flow elements grouped with a *Group*, or any extension of this artefact, to inherit the *CategoryValue* assigned to the group, making their further processing by the transformation rules more difficult. These bugs were analysed, reported and helped to solve, as can be seen in the following web page `https://bugs.eclipse.org/bugs/show_bug.cgi?id=422793`.

In this section, we have detailed a set of contributions facilitating the documentation of more complete business models. Specifically, we have presented the BPCI Extension, a process facilitating the identification of the BUCs, and the BPCI Eclipse

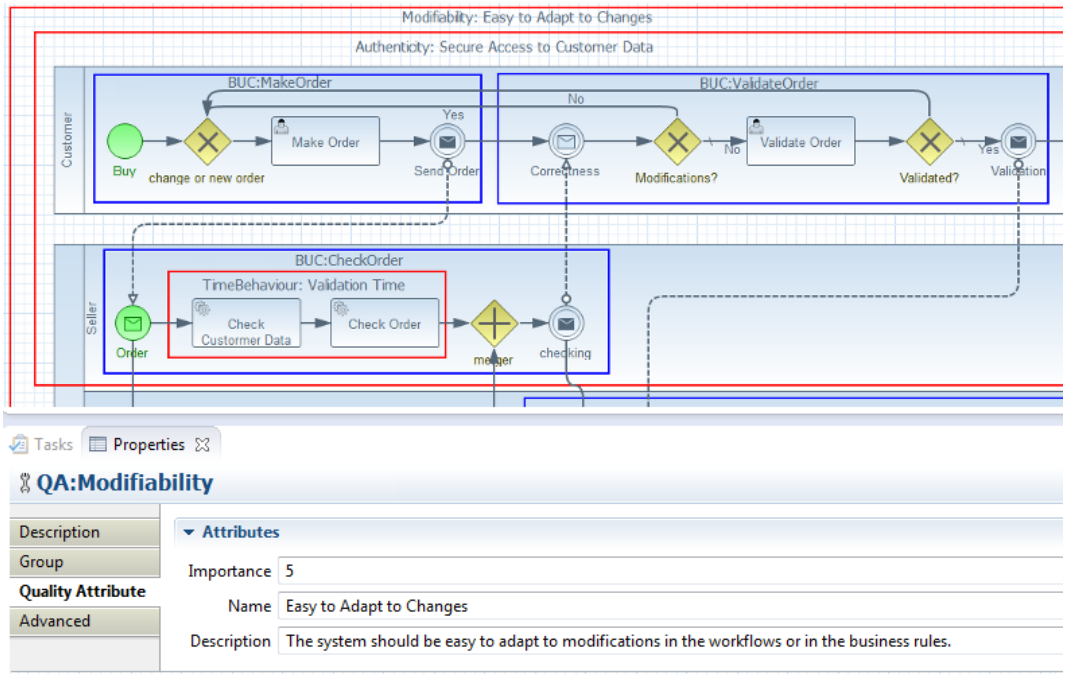


Figure 3.8: Properties of the *Modifiability* quality attribute for the Online-shop order-placing process

Plugin supporting the defined extension. The information that can be documented with these contributions is also highly useful for the subsequent development stages. The next section focuses on detailing how the requirements and their interdependencies can be semi-automatically derived from the annotated BPMN models and how the existing notations have been extended for modeling the interdependencies between functional and non-functional requirements.

3.4 Defining the Requirements Interdependencies

Once the interdependencies between business elements have been annotated, they should be derived and modeled in the requirements model. Thus, they are available for be analysed by the architect in the phase of designing the system architecture.

With the aim of modeling them, in this thesis we have defined the following elements:

- The QuAID Profiles, extending the UML Use Case and Activity diagrams mak-

ing the modeling of interdependencies between functional and non-functional requirements possible.

- The QuAID Eclipse Plugin, implementing the defined profiles as an extension of the Eclipse modeling tool Papyrus.

Furthermore, to guide and automatize the derivation of requirements and their interdependencies, the following contributions have been implemented:

- A set of patterns guiding the derivation of functional models and their relationships from business process annotated with the BPCI extension.
- A number of ATL transformations automating the application of the defined patterns.

These profiles and patterns allow the requirement engineers to automatically obtain a functional model aligned with the business and with the relationships between requirements documented. Moreover, these relationships are annotated so that they can be reused in the architecture design. The following subsections detail each of these contributions.

3.4.1 QuAID Profiles

The **QuAID** (**Qu**ality **A**ttribute **I**nter**D**ependencies) profiles have been defined to allow the requirements engineer to model the relationships between functional and non-functional requirements. These profiles have been specified for the UML diagrams more commonly used for documenting functional requirements, which are the Use Case and Activity diagrams. In addition, with these two diagrams, the functional requirements and their interdependencies can be documented at different abstraction level.

Figure 3.9 shows an extract of the profile defined for Use Case diagram (UC-QuAID profile) detailing the most important stereotypes (the whole UC-QuAID profile can be seen in Appendix B). The metaclass extended to model these relationships is **ExtensionPoint**. This metaclass has been extended in order to maintain the readability of the diagram when a large number of relationships is modeled, since

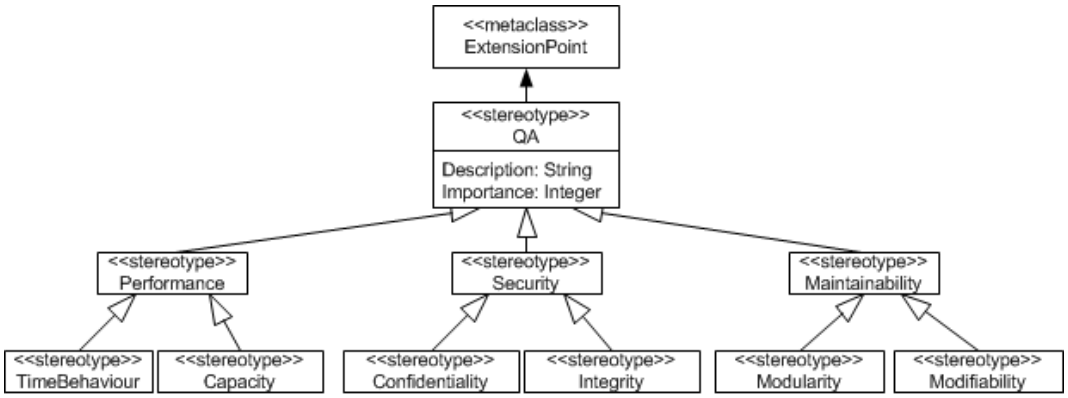


Figure 3.9: Extract of the UC-QuAID profile.

they are added to each specific Use Case that has to fulfil them without ravel the diagram. Thus, the annotated Use Cases can also be further analysed and processed by transformation rules inferring useful information for the arquitect, such as the impact of a specific requiremment (as Section 3.5 details).

The main stereotype is **QA (Quality Attribute)**, which allows one to model general interdependencies between functional and non-functional requirements. It has three attributes: *Name*, which is inherited from the metaclass and specifies the quality requirement name, *Description*, which details the QA specification; and *Importance*, which indicates the interest for the achievement of the quality attribute. A QA is modeled as an *ExtensionPoint* and is depicted with annotation $<<QA>>$ plus the name of the non-functional requirement to achieve.

Similarly to BPCI Extension, in order to facilitate the modeling of more specific quality attributes, the QA element is extended with a tree of more concrete quality requirements. This tree is based on the quality model detailed in the ISO/IEC 25010. Figure 3.9 only shows the quality requirements used in the running example. These requirements are:

- **Performance.** It details non-functional requirements relative to the amount of resources used under stated conditions. This attribute can be decomposed in the sub-characteristics *Time Behaviour* (for detailing the processing times) and *Capacity* (for specifing the maximum limits of a system).
- **Security**. This chareacteristic details the authorization levels of a system.

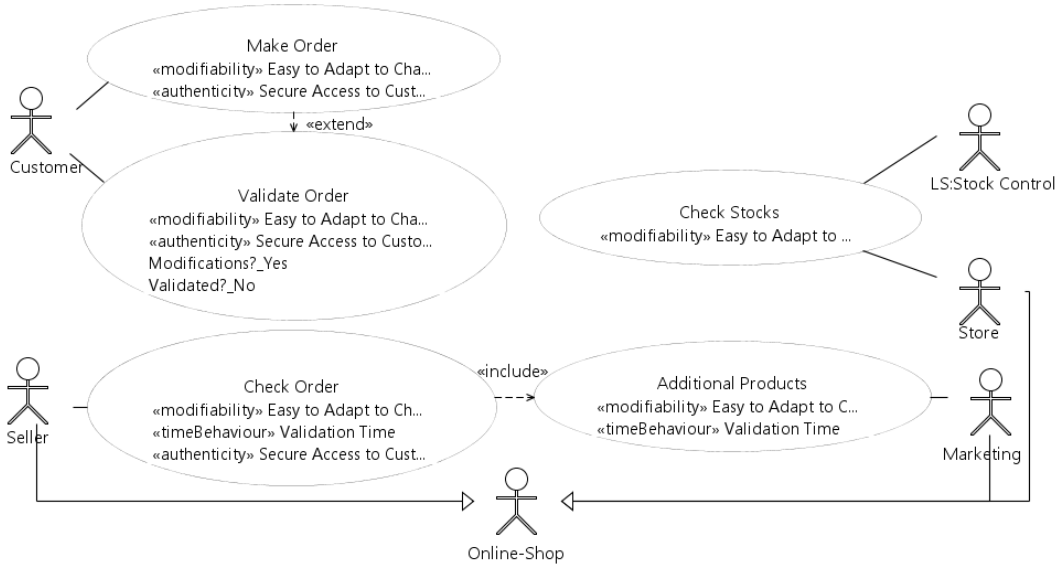


Figure 3.10: Online-shop Use Case Diagram annotated with the UC-QuAID Profile.

It can be decomposed in the sub-characteristics *Confidentiality* (for detailing which data should be only accesible for those authorized) and *Integrity* (for preventing unauthorized access).

- **Maintainability**, which is used to document the level of efficiency with which a system can be modified. This attribute is decomposed in the sub-characteristics *Modularity* (documenting the degree to which a system has to be composed of discrete components) and *Modifiability* (for detailing the facility of modification of a system).

The modeling of these elements is similar to the QA element, just only the specific quality attribute's name is used between the braces. Figure 3.10 shows the online-shop Use Case diagram annotated with the defined stereotypes. Thus, all the Use Cases have been annotated with the quality attribute *Modifiability*, since the whole business process they support had to achieve this qualitty attribute; the *Make Order*, *Validate Order* and *Check Order* have been annotated with the *Authenticity* requirement, since they manage sensitive information; and the *Check Order* and *Additional Products* Use Cases have been annotated with the quality requirement *TimeBehaviour*, since they have to be completed in a certain time for pleasing the customer.

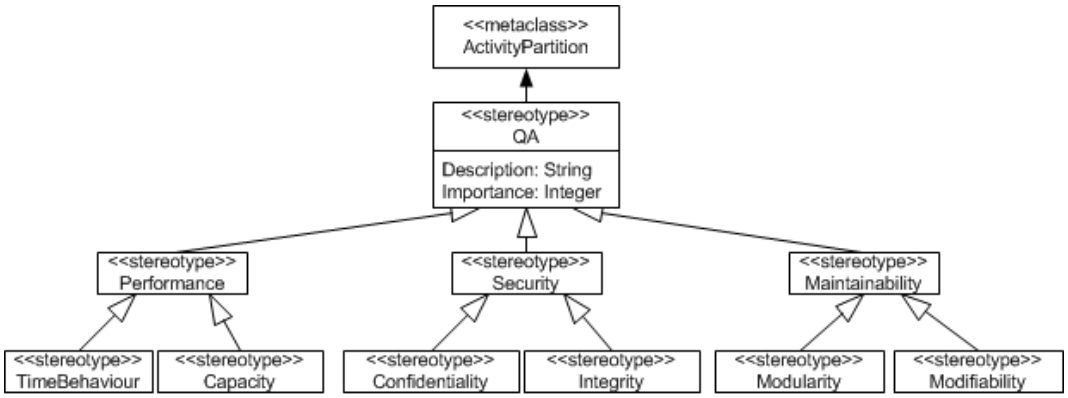


Figure 3.11: Excerpt of the AD-QuAID profile.

Figure 3.11 shows an excerpt of the profile defined for UML Activity diagrams (AD-QuAID profile). This profile is similar to the UC-QuAID profile because the same quality model (ISO/IEC 25010) has been followed for its definition. The extended metaclass is **ActivityPartition**. This metaclass has been extended because it allows grouping elements by a specific criterion. In this case, the criterion is the non-functional requirements they have to achieve. The stereotypes extending this metaclass are depicted in the Activity diagram as rectangles with a solid black line with the quality attribute's name between braces plus the non-functional requirement name. Figure 3.12 shows the Activity diagram of the *Chek Order* functionality annotated with the *Modifiability*, *Authenticity* and *TimeBehaviour* quality attributes.

All the elements contained in a partition inherit the partition's name. This value is assigned to the attribute *InPartition* of each flow element. Therefore, these elements can subsequently be analysed and processed by transformation rules based on the partitions assigned to them.

Using the QuAID profiles, the requirements engineer provides a more complete and detailed information on the interdependencies between functional and non-functional requirements. So that these relationships can be more easily analysed and evaluated by the architect. In addition, these annotations can be automatically processed by ATL transformation for identifying the impact of each non-functional requirement (as Section 3.5.2 shows).

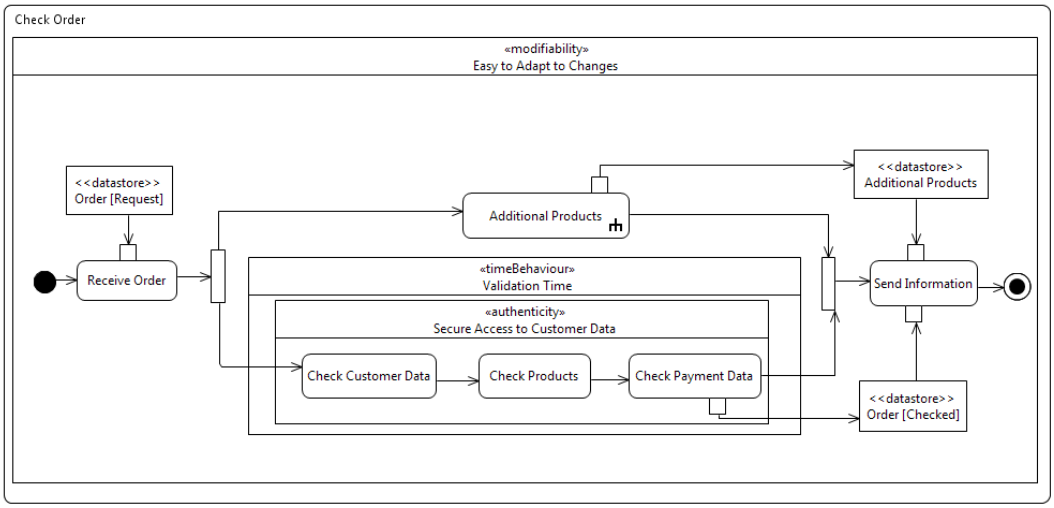


Figure 3.12: Check Order functionality annotated with the AD-QuAID profile.

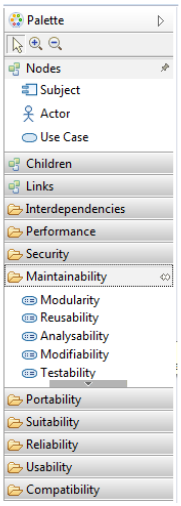
3.4.2 QuAID Eclipse Plugin

With the aim of facilitating the modeling of interdependencies between requirement with the QuAID profiles, we have implemented a tool supporting them for the Eclipse plugin Papyrus (*Eclipse Papyrus*, 2014). This tool is called **QuAID Eclipse Plugin**. Papyrus provides an integrated and user-consumable environment for editing any kind of EMF model and particularly supporting UML and related modeling languages.

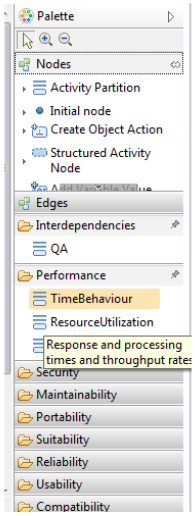
First, the profiles have been modeled with Eclipse Papyrus. Thus, they can be used and applied on the Use Case and Activity diagrams created with Papyrus.

In addition, two palettes have been implemented to facilitate the modeling of the defined stereotypes. Figure 3.13a shows the palette supporting the UC-QuAID profile, and Figure 3.13b shows the palette for AD-QuAID profile. For each stereotype specified in the palettes, a description is also shown. For example, in Figure 3.13b the description of the *Time Behaviour* stereotype is depicted. These palettes are shown when a new Use Case or Activity diagram is created so that the modeler can create *ExtensionPoints* or *ActivityPartitions* with specific stereotypes directly applied.

Furthermore, Eclipse Papyrus natively provides a properties view for each mod-



(a) Palette supporting the stereotypes defined in the with UC-QuAID profile.



(b) Palette supporting the stereotypes defined in the with AD-QuAID profile.

Figure 3.13: Palettes for modeling the elements defined in the QuAID profiles.

eled stereotype in which the requirement engineer can detail its specific attributes. For an interdependency with a non-functional requirement, the engineer can detail its description and importance. Figure 3.14 shows the properties view for the *Modifiability* stereotype applyed on the *Make Order* Use Case.

Therefore, this tool provides an environment to document the interdependencies between requirements, allowing the requirement engineer to model them or refine the ones already specified in a simple manner.

3.4.3 Patterns for Deriving Functional Models and Requirements Interdependencies

Once the business processes have been annotated with the BPCI Extension, it is possible to derive a Use Case diagrams from them. To facilitate this activity, we have defined a set of patterns guiding in the commonest situations leading to the identification of Use Cases, Actors and Interdependencies. Table 3.2 shows some of the patterns used in the running example. The most important patterns identified as part of this thesis are detailed in Appendix C. To facilitate their comprehension, a description, the source BPMN pattern, and the derived Use Case fragment are

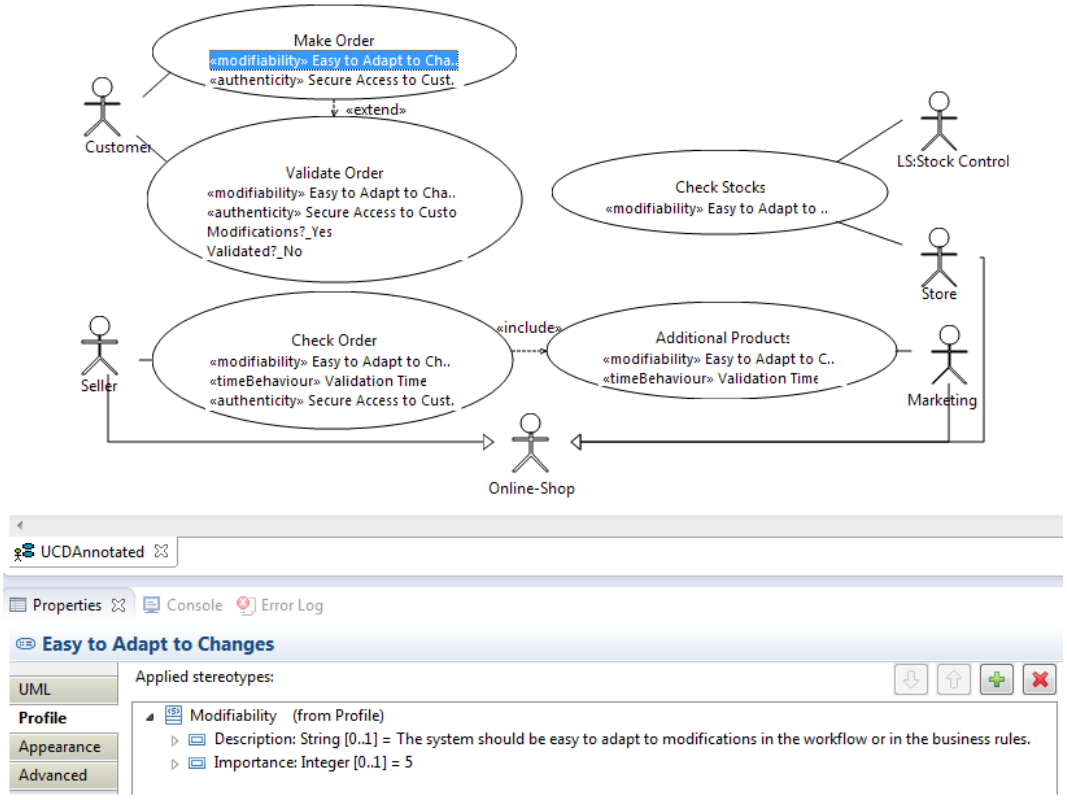


Figure 3.14: UC-QuAID Eclipse Plugin view for editing the properties of a quality attribute annotated in a Use Case.

detailed for each pattern.

First, the Actors and Use Cases are extracted. These elements are identified by applying the patterns one to three of the Table 3.2. Specifically, each BUC annotated in a process is mapped to a Use Case; and each Pool, Lane or Legacy System is derived to an Actor. As can be seen in Figure 3.14, five Use Cases (one per each BUC) and six actors (one from the *Online-Shop* Pool, four from the modeled Lanes and one from the *Stock Control* Legacy System) were derived from business process for placing orders.

Subsequently, the relationships between Use Cases and actors, between Use Cases and Legacy Systems, and *include* or *extend* relationships are derived from the annotated business processes. Specifically, in the online-shop running example, one can identify, for example, an *extend* relationship between the *Make Order* and the *Validate Order* Use Cases. This relationship is identified applying the fourth pat-

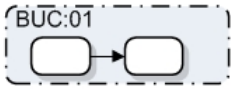
tern of Table 3.2, since the *Validate Order* BUC contains an exclusive gateway from which an alternative branch (grouped by the *Make Order* BUC) starts. Therefore, the *Make Order* Use Case extends the normal flow of the Use Case *Validate Order* when the exclusive gateway is reached.

Likewise, an *include* relationship between the *Check Order* and *Additional Products* Use Cases can be identified applying the fifth pattern of Table 3.2. This pattern is applied because the *Check Order* BUC contains a merger parallel gateway and the *Additional Products* BUC covers the branch that has to be waited. Thus, for the *Check Order* Use Case to continue its execution, the functionality covered by the Use Case *Additional Products* has to finalize.

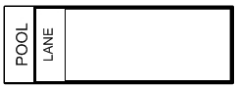
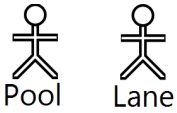
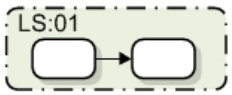
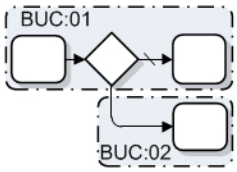
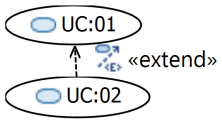
Finally, applying the last pattern defined in Table 3.2 the interdependencies between functional and non-functional requirements are derived. In the online-shop running example, since some of the business tasks supported by the BUCs *Check Order* and *Additional Products* are annotated with the *Time Behaviour* quality attribute, the interdependencies with it are derived and annotated in the corresponding Use Cases using the UC-QuAID profile.

The patterns identified and defined in this section lead to the derivation of functional models and the interdependencies with non-functional requirements from the business process. This ensures that these models are complete and aligned with the business.

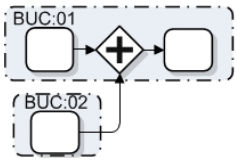
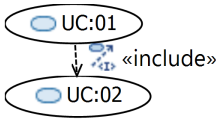
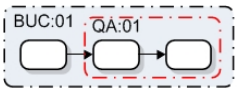
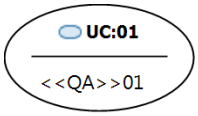
Table 3.2: Excerpt of the Patterns for Deriving Functional Models and Requirements Interdependencies.

Description	**Pattern**	**UC Fragment**
Each Business Use Case is derived to a Use Case, since both represent a set of tasks that an actor has to perform.		

(The table continues in the following page).

Description	Pattern	UC Fragment
The Pools and Lanes in the business processes are mapped to Actors in the Use Case diagram, since they represent the roles responsibles for executing the set of tasks or the Use Cases.		
The Legacy Systems annotated in the business processes are derived as Non-Human Actors in order to be able to represent interaction of the system with them.		
If there is a BUC grouping some tasks, an exclusive or inclusive gateway, and the gateway's default flow, and there is another Business Use Case grouping the tasks defined in an alternative branch, then an *extend* relationship exists between the two Use Cases, since the BUC in the alternative branch extends the normal flow to follow when the gateway is reached.		

(The table continues in the following page).

Description	Pattern	UC Fragment
When two or more control flows are merged into one by a parallel gateway and there is a BUC covering a branch and, at least, the parallel gateway, then there is an *include* relationship between the BUCs covering each branch and the BUC containing the gateway; since in order to continue the execution of the BUC containing the gateway, the other branches have to finalize.		
If some of the business tasks covered by a Business Use Cases are annotated with a quality attributes, then the Use Case derived from that BUC is also annotated with the same quality attribute since it has also to achieve that non-functional requirement.		

(End of the Table)

3.4.4 ATL Transformations Automating the Application of the Patterns

With the aim of automate the application of the patterns defined above, a set of ATL transformations has been implemented. These transformations take as input the business processes annotated to automatically generate functional models annotated with the interdependencies with non-functional requirements. Some of the most important transformation for the running example are detailed below. Appendix D shows the ATL transformations implementing the most important and representative patterns.

An important transformation, is the derivation of actors (second pattern in Ta-

```
rule lane2Actor{

  from
    lane : inMM!Lane

  to
    actor: outMM!Actor (
      name <- lane.name,
      generalization <- generalization
    ),

    generalization : outMM!Generalization(
      general <- lane.getParent()
    )
}
```

Listing 3.2: Identification of actors in the business process models.

ble 3.2). The Listing 3.2 presents a transformation mapping each lane to an actor. In addition, it creates a generalization relationship between the created actor and the actor extracted from the Pool or the Lane containing the processed lane. Thus, for the online-shop, this rules would generate the actors *Customer*, *Seller*, *Store* and *Marketing* and would model generalization relationships between the *Seller*, *Store* and *Marketing* and the *Online-Shop* actor, as the latter is derived from the Pool containing the Lanes.

The *AlternativeExclusiveGateway* ATL rule, defined in the Listing 3.3, implements the fourth pattern of Table 3.2 (i.e. the pattern for identifying *extend* relationships between Use Cases connected by and an exclusive or inclusive gateways). This rule, first, identifies the source pattern in the business process. For each detected pattern, the extending and extended Uses Cases are identified from the annotated BUCs. Then, an *Extension Point* is created in the extended Use Case. The *extend* relationship is created only if there is not another *extend* relationship already defined between the two Use Cases. Otherwise, the *Extension Point* is directly linked to the existing *extend* relationship. In the online-shop example process, this transformation is identified twice, because two different exclusive gateways are covered by the BUC *Validate Order*. The first time it is identified, an *Extension point* and an *extend* relationship are created to reflect that, during the validation of an order,

```
1  rule AlternativeExclusiveGateway {
2
3    from
4      seq : inMM!SequenceFlow ( seq.connectBUCs() and
5        if seq.sourceRef.oclIsTypeOf (inMM!
             ExclusiveGateway) then
6          if seq.sourceRef.default<>OclUndefined then
7            seq.sourceRef.default <> seq
8          else
9            false
10         endif
11       else
12         false
13       endif
14   )
15
16   to
17     ePoint : outMM!ExtensionPoint (
18       name <- seq.sourceRef.name+'_'+seq.name,
19       useCase<-thisModule.resolveTemp (seq.sourceRef.
             getBUC(),'uc')
20     )
21
22   do {
23     if (not seq.alreadyExtended(seq.targetRef, seq.
          sourceRef)) {
24       thisModule.createExtendRelationship(seq);
25     } else {
26       seq.getExtend(seq.targetRef, seq.sourceRef).
             extensionLocation <- seq.getExtend(seq.
             targetRef, seq.sourceRef).extensionLocation.
             append(ePoint);
27     }
28   }
29 }
```

Listing 3.3: Identification of extend relationships between Use Cases in the business process models.

```
lazy rule setQAUCRelationship {

  from
    uc : outMM!UseCase,
    cValue : inMM!CategoryValue

  to
    extensionPoint : outMM!ExtensionPoint (
      name <- cValue.value,
      useCase <- uc
    )

  do{
    extensionPoint.applyStereotype(profile!Stereotype.
        allInstances()->any( e | e.name = cValue.getName
        ()));
  }
}
```

Listing 3.4: Derivation of interdependencies between functional and non-functional requirements.

if a modification is required, a new order has to be placed. The second time, only the *Extension Point* is created (the *extend* is not duplicated) to reflect that if the order is not validated a new one can be placed.

To automate the derivation of interdependencies between functional and non-functional requirements some ATL transformations have been defined (implementing the last pattern of Table 3.2). These rules, whenever a Use Case is derived from a BUC, check if the BUC is restricted by some quality attribute. If it is, they identify the quality requirement and, then, it is annotated in the derived Use Case. An excerpt of these rules is shown in the Listing 3.4. The *setQAUCRelationship* rule creates a new extension point in the Use Case for representing the interdependency applying the corresponding stereotype.

These transformations, along with the ones shown in Appendix D, automate the defined patterns for generating functional models annotated with the interdependencies with non-functional requirements. Thus, it is ensured that with little effort the requirements engineer can obtain a complete, and aligned with the business, model.

In this section, we have detailed a set of profiles, tools and ATL transformations for documenting and deriving the requirements and their interdependencies from annotated business processes. The QuAID profiles and Eclipse Plugin allow engineers to model the interdependencies between functional and non-functional requirements. The patterns and the ATL rules defined semi-automate the derivation of the requirements models from the business models. Therefore, these contributions facilitate the definition of requirements models better aligned with the business, more complete and annotated with information very useful for the subsequent development stages. Specifically, the annotated information has been specially designed to facilitate the requirements analysis during the architectural design. The following section details how this information is reused to facilitate the architectural decision making.

3.5 Analizing the Requirements Interdependencies

The diagrams generated during the requirements analysis stage are subsequently evaluated by the architect to get a picture of the system without having to thoroughly analyse all the artifacts. Taking this picture as basis, the architect identifies the most Architecturally Significant Requirements leading the system design.

With the aim of facilitating the evaluation of the requirements for the architecture design, the following contributions have been proposed:

- A metamodel for documenting the impact of the non-functional requirements on the system.
- A set of ATL transformations, based on the diagrams annotated with the QuAID profiles, for semi-automatically generate the model documenting the impact of the non-functional requirements.

The model generated by means of the defined metamodel and the ATL transformations provides greater information to the architect on how many elements are affected and impacted by each non-functional requirement. The use of this information reduces the effort required to evaluate the system requirements and facilitates the architectural decision making.

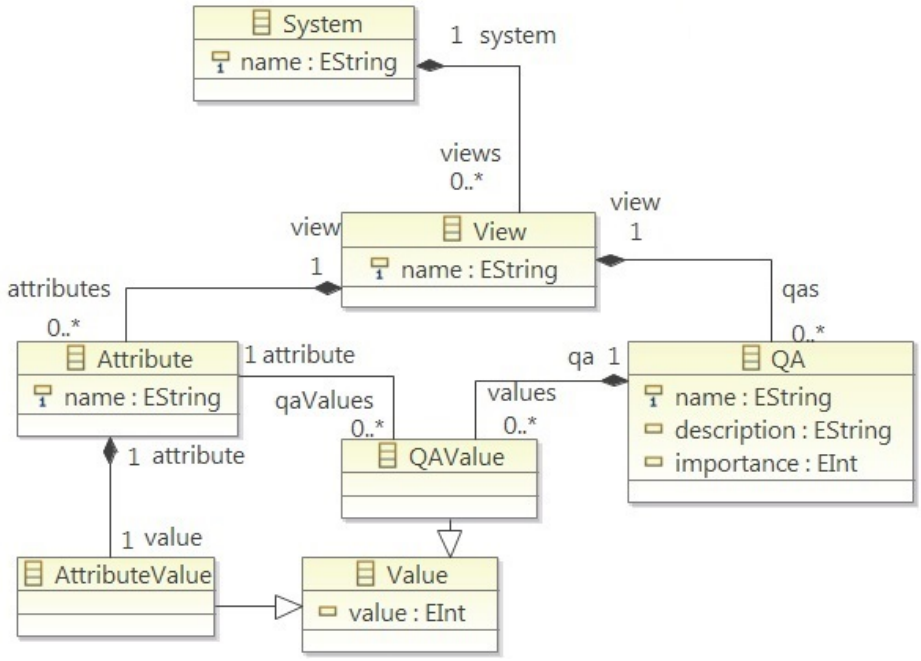


Figure 3.15: Metamodel for documenting how the quality attributes affect each architectural view.

3.5.1 Metamodel for Documenting the Impact of each Quality Attribute

With the aim of facilitating the documentation of the complete picture of a system, the author of this thesis has defined a metamodel for documenting the different architectural views of the system and how each non-functional requirement affect the elements conforming each view. The architectural views are usually used to describe the system from the viewpoint of different stakeholders, such as end-users, developers and project managers, and are used by the architect to design the system. For example, an *Use Case View* can be used to represent the system from the viewpoint of the end-users, since it details the functionalities they can perform.

Figure 3.15 shows the defined metamodel. As can be seen, a system is composed of a set of *views*. For each view, in turn, the following elements are defined:

- The *Attributes*, which detail the most important elements contained in the view. Thus, for example, an *Use Case View* can specify the Actors, Use Cases and their relationships as its Attributes. Each Attribute documents

the number of elements of that kind contained in the view by means of the *AttributeValue* property. This information can be used to infer the importance of each aspect of the view and the size of the system.

- The *QAs* constraining the attributes of the view. For each non-functional requirement, the QA element documents its name, its description and its importance. Furthermore, it is related to the *QAValue* object for detailing the number of elements of each attribute of a view affected by the quality attribute. Thus, one can easily identify the impact of each non-functional requirement on the view.

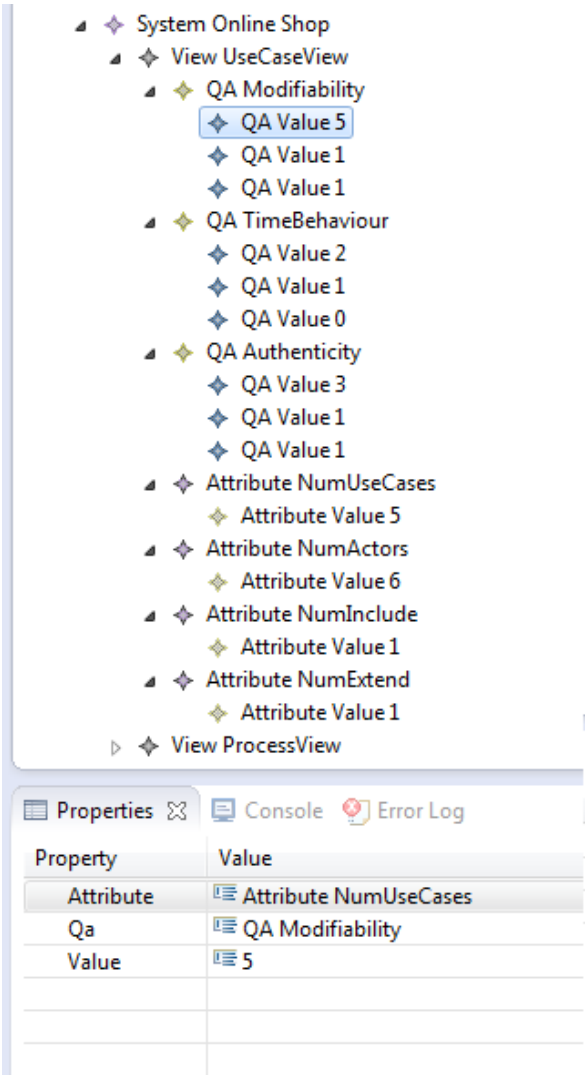


Figure 3.16: Use Case View of the Online-Shop.

Figure 3.16 shows the online-shop Use Case View modeled with the previously defined metamodel. Specifically, it indicates that the system has five Use Cases, six Actors, one Include and one Extend relationship. Furthermore, this model also documents that this view is affected by the *Modifiability*, *Authenticity* and *Time Behaviour* quality attributes. The first one is annotated in five Use Cases which in turn implies that one Include and one Extends relationships are also constrained by it. The second one restricts three Use Cases, one Include and one Extends relationship. The third one constraints two Use Cases and only one Include relationship. Thus, the architect can easyly see that the *Modifiability* has a greater impact than the other non-functional requirements, and can start making decision taking this information into account.

These models are very useful to get a complete picture of how the quality attributes affect each architectural view. In addition, assigning weights to the different attributes of each view, one can extract precise values of the impact of each quality attribute on each view, or on the complete system. This is a very important information to know what quality attributes are architecturally more significant and, thus, to begin making decisions to satisfy them.

3.5.2 ATL Transformations for Generating the Model Documenting the Architectural Views

Some of the views of the models documenting the impact of the quality attributes can be derived from the Use Case and Activity diagrams annotated with the QuAID profiles. The annotated Use Case diagram can be used to generate the *Use Case View* with information on the number of Actors, Use Cases and their relationships, along with the influence of the non-functional requirements on them.

Similarly, the annotated Activity diagram can be reused to derivate the *Process View* with information on the number of activities, number of actions or number of parallel actions, along with the influence of the non-functional requirements on them.

Likewise a set of rules semi-automating the derivation of requirements models

```
1  ...
2    from
3      model : UML!Model
4
5    to
6      ucases : QAV!Attribute(
7        name <- 'NumUseCases',
8        view <- ucv
9      ),
10
11   vUCS : QAV!AttributeValue(
12     attribute <- ucases,
13     value <- model.packagedElement->select(usecases |
           usecases.oclIsKindOf(UML!UseCase)).size()
14   ),
15 ...
```

Listing 3.5: Number of Use Cases of a System.

from the annotated business processes was defined, we have also developed a set of transformations ATL semi-automating the derivation of the Architectural Views from requirements diagrams annotated with the QuAID profiles. Some of the most important transformations for the running example are detailed below. Appendix E shows the complete set of ATL transformations deriving the Use Case View from annotated Use Case diagrams.

First, the number of elements composing the different attributes of each view has to be identified. An excerpt of the ATL rule implemented for detailing the *Number of Use Cases* attribute of the Use Case View is detailed in the Listing 3.5. Specifically, this rule obtains the number of Use Cases detailed in the Use Case diagram and details it in the appropriate view.

Subsequently, for each attribute of a view, the non-functional requirements affecting it and how many elements of each attribute are restricted have to be identified. Below an excerpt of the ATL rule implemented to obtain the number of Use Cases constrained by a specific non-functional requirement is shown. Concretely, the rule defined in the Listing 3.6 identifies those Use Cases having extensions points annotated with the stereotypes detailed in the UC-QuAID profile.

```
...
  from
    v : QAV!View,
    s : UCP!Stereotype,
    model : UML!Model

  to
    qa : QAV!QA (
      view <-v,
      name <-s.name),

    ucQAV : QAV!QAValue(
      attribute <-thisModule.attributes.at(1),
      qa <- qa,
      value <-model.packagedElement->select(elements |
          elements.oclIsKindOf(UML!UseCase))->collect(
          ext | ext.extensionPoint).flatten()->select(
          exten | not exten.getAppliedStereotype(s.
          qualifiedName).oclIsUndefined()).size()
    ),
...
```

Listing 3.6: Number of Use Cases annotaded with a stereotype.

These transformations, along with the ones shown in Appendix E, semi-automate the generation of the model documenting the impact of the quality attributes on the architectural views. Thus, this model can be used with little effort by the architect for identifying the Architectural Significant Requirements and making architectural.

This section has presented a metamodel and a set of ATL transformations facilitating the analysis of the requirements and their interdependencies and the identification of the impact of each non-functional requirement on the system. The generated models are very useful for the architect to identify the ASRs of a system and to start making the architectural decisions, or can be used as input of the tools assisting them in the making of these decisions. The next section details how the defined profiles and metamodels can be reused by these approaches.

3.6 Integration with Approaches Guiding the Architectural Decision Making

The diagrams, profiles and models defined in this thesis have been specially designed to be reused by the architect or by tools assisting him/her in the making architectural decisions process. This section focuses on how they are integrated in approaches assisting the architect.

Specifically, the following subsections detail the integration of the outcomes of this thesis with Attribute Driven Design (Bass, Clements, & Kazman, 2003), Reasoning Frameworks (Bachmann et al., 2005) and Architectural Decisions in the Development of Multi-Layer Applications (Garcia-Alonso, Berrocal, & Murillo, 2013). The integration with the two first approaches is addressed due to their importance and relevance. On the other hand, the third proposal and the contributions presented in this thesis have been specially designed to be seamlessly integrated and, hence, provide a complete support to the analysis, design and development of IT systems better integrated and aligned with the business.

3.6.1 Attribute Driven Design

Attribute Driven Design (ADD) (Bass et al., 2003), as it was specified in Section 2.5.1, is a centred software architecture design method for organizing requirements. This method defines the following steps for designing an architecture:

1. Choose a part of the system to decompose.
2. Refine the chosen part of the system according to the following steps.
   (a) Choose architectural drivers (or architectural significant requirements).
   (b) Choose architectural patterns and tactics that satisfy the architecture drivers.
   (c) Allocate remaining functionality.
   (d) Define interfaces for the instantiated elements.

(e) Verify and refine the functional requirements, quality scenarios, and constraints.

3. Repeat steps 1 through 2e.

However, this method largely depends on the architect and all steps must be performed manually by him/her by means of a thorough analysis of the requirements and the selection and application of the appropriate patterns and tactics.

The contributions presented in this thesis can be easily integrated to facilitate the ADD method. Specifically, the defined models and diagrams can be used to support the steps *2a. Choose architectural driver (or architectural significant requirements)* and *2c. Allocate remaining functionality.*

The model detailing the architectural views and the impact of the quality attributes on them (Section 3.5.1) can be used in the step 2a to facilitate the selection of architectural driver. Manually performing this step requires the architect to analyse in-depth all the system requirements, evaluate the relationships between them and obtain the impact of each non-functional requirement, to identify the most architecturally significant requirements. The model proposed in this thesis, which is semi-automatically generated, provides this information without having to analyse in-depth the requirements and their relationships.

For example, using the model depicted in Figure 3.16, the architect can easily see that the quality attribute *Modifiability* is the one that most impact on the system Use Case View. With this information, the architect can easily assess whether such a requirement is an architectural driver and then perform the next step in the ADD method to select the specific patterns of tactics for its achievement.

Finally, the Use Case and Activity diagrams annotated with the QuAID profiles can be used to facilitate the step 2c. In this step, the patterns and tactics selected to achieve the architectural driver are applied on the functionalities that have to meet it. These diagrams provide precise information on what Use Cases and actions are affected and/or restricted by each quality attribute. Using these diagrams, the architect does not have to analyse all the requirements models in order to identify those features, since they are explicitly annotated.

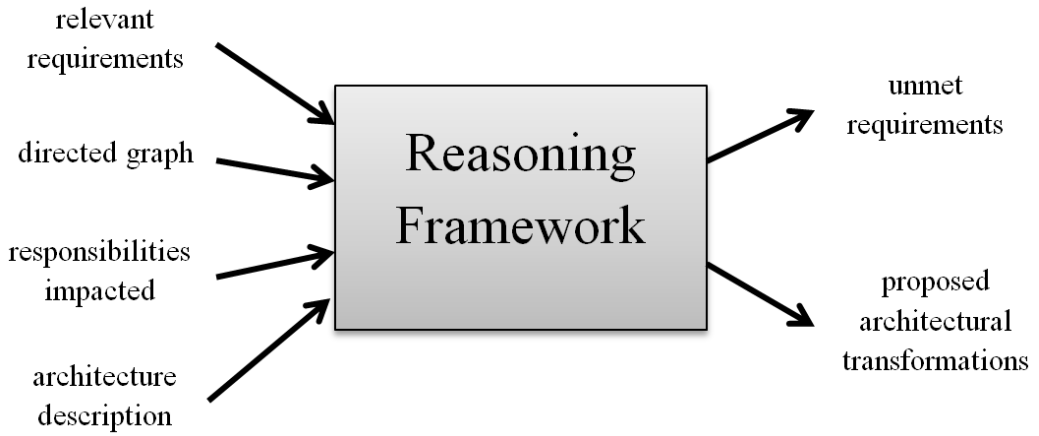


Figure 3.17: Input and output from a Reasoning Framework.

Thus, for the online-shop running example, Figure 3.10 plainly shows on which Use Cases the architect should apply the selected patterns and tactics to meet the *Modifiability* requirement.

Therefore, the integration of the models and diagrams defined in this thesis in the ADD method reduces the required effort from the architect to analyse the requirements, allowing him/her to focus on the architecture design.

3.6.2 Reasoning Frameworks

With the aim of easing the effort required by architects to select architectural patterns in the ADD method, in (Bachmann et al., 2005) the authors define structures implementing mechanisms that transform the architecture of a system with respect to a given quality attribute theory. Specifically, these mechanims support the steps 2b (*Choose architectural patterns and tactics*) y 2c (*Allocate remaining functionality*) of ADD.

A reasoning framework encapsulates knowledge relative to one quality attribute theory and its associated models. Figure 3.17 shows the inputs and outputs of a quality attribute reasoning framework. The most important inputs are the following:

- The list of quality relevant requirements (or architectural significant requirements) related to the knowledge and the quality attribute theory encapsulated by the reasoning framework.

- A directed graph that represent the system responsibilities or functionalities.
- The reponsibilities impacted by each concrete quality attribute.
- A description of the current state of the architecture design.

The most important output are the following:

- The list of relevant requirements unmet by the architecture design.
- The proposed transformations (or list of proposed architectural patterns or tactics) that might be useful in improving the architecture with respect to the unmet requirements.

The contributions presented in this thesis facilitate obtaining both the list of quality relevant requirements and the reponsibilities impacted by each concrete quality attribute.

As indicated in the previous section, the model documenting the architectural views facilitates the identification of the relevant quality requirements. The main difference is that the Reasoning Framework is based on the ISO/IEC 9126 standard (International Standard Organization (ISO/IEC), 2001), while the one proposed in this thesis is based on the ISO/IEC 25010 standard (International Standard Organization (ISO/IEC), 2011). Nevertheless, since the later standard is an evolution of the first one, the quality attributes are equal in most cases or can be easily mapped.

Finally, the Use Case and Activity diagrams annotated with the QuAID profiles facilitate the identification of the responsibilities or tasks impacted by each quality attribute, since they explicitly detail what Use Cases and actions are restricted by each non-functional requirement. Therefore, from these diagrams the architect can easily get the *resposabilities impacted* input of the Reasoning Frameworks.

Thus, for example, if the architect want to achieve the *Time Behaviour* requirement of the online-shop running example, this requirement would be provided as input to the appropriate Reasoning Framework. In addition, from the annotated Use Case diagram (Figure 3.10), he/she would easily identify that the Use Cases restricted by this requirement are *Check Order* and *Addition Products*. Furthermore,

from the annotated Activity diagrams (Figure 3.12), the architect would identify that in the *Check Order* Use Case the specific actions impacted by it are *Check Customer Data*, *Check Products* and *Check Payment Data*. Such information would be directly provided as input of the Reasoning Framework to identify which patterns or tactics apply to such actions to meet the indicated requirement.

Therefore, the Reasoning Frameworks improve the ADD method, facilitating and giving support to the architectural decision making process. Moreover, the integration of the outcomes of this thesis with Reasoning frameworks reduces the effort required by the architect to analyse the requirement in order to provide the necessary inputs to these mechanisms, facilitating the design of an architecture achieving the system requirements.

3.6.3 Architectural Decisions in the Development of Multi-Layer Applications

The authors of this approach (Garcia-Alonso et al., 2013) focus on improving and supporting the process used to develop framework based on multi-layer applications. To that end, they give support from the initial architectural decision making to the generation of part of the system source code. This support is provided by the identification and organization of the most common architectural and design decisions in the development of framework based multi-layer applications and the definition of a mechanism that allows the traceability of the decisions taken to the product developed. Also, the defined process is semi-automated by means of MDD techniques.

This work has been developed simultaneously to this thesis. Furthermore, the author of this thesis and the authors of this approach have being working closely in their definition. So that, their integration can be done in a natural way.

Figure 3.18 depicts the diagram with the most important artifact used or generated by the approach. The defined process for developing this kind of applications begins with a preliminary design, specified with Use Case and Activity diagrams, of the system to be developed. These diagrams have to be annotated with information

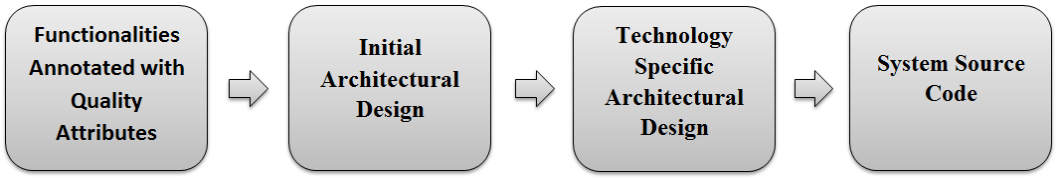


Figure 3.18: Overview of the approach Architectural Decisions in the Development of Multi-Layer Applications.

on the quality attributes restricting each documented Use Case or action in the Activities. Therefore, the main inputs of the first step of the process are the diagrams generated applying the QuAID profiles.

Subsequently, the architect has to define the initial software architecture best suited to support both the functional and the non-functional requirements. In order to simplify this step the authors have created a repository containing the commonest architectural decision, along with the ATL transformations semi-automatizing their application. The decision of whether a specific architectural decision is applied or not is based on the quality attributes annotated on the functional requirements. Therefore, depending on the annotations detailed on the Use Case and Activity diagrams, a set of possible decisions to make is automatically suggested to the architect.

Thus, for example, one of the architectural decisions made is the creation of a specific layer containing all the security actions. The application of this decision depends on whether the documented Use Cases and actions are annotated with the security non-functional requirement. For the online-shop running example, since the Use Cases *Make Order*, *Validate Order* and *Check Order* are annotated with the *Authenticity* requirement (Figure 3.10), a security layer would be suggested to be generated to make those functionalities secures. Figure 3.19 shows the Activity diagram of the Check Order Use Case once these suggestions have been applied. As can be seen, the creation of the security layer suggestion was accepted.

Thereafter, the initial architecture design is refined and tailored to the specific technologies and development frameworks that will be used in the system. To perform this refinement, first, the architect identifies the design patterns to apply. These patterns, and the associated ATL transformations, are documented in a repository,

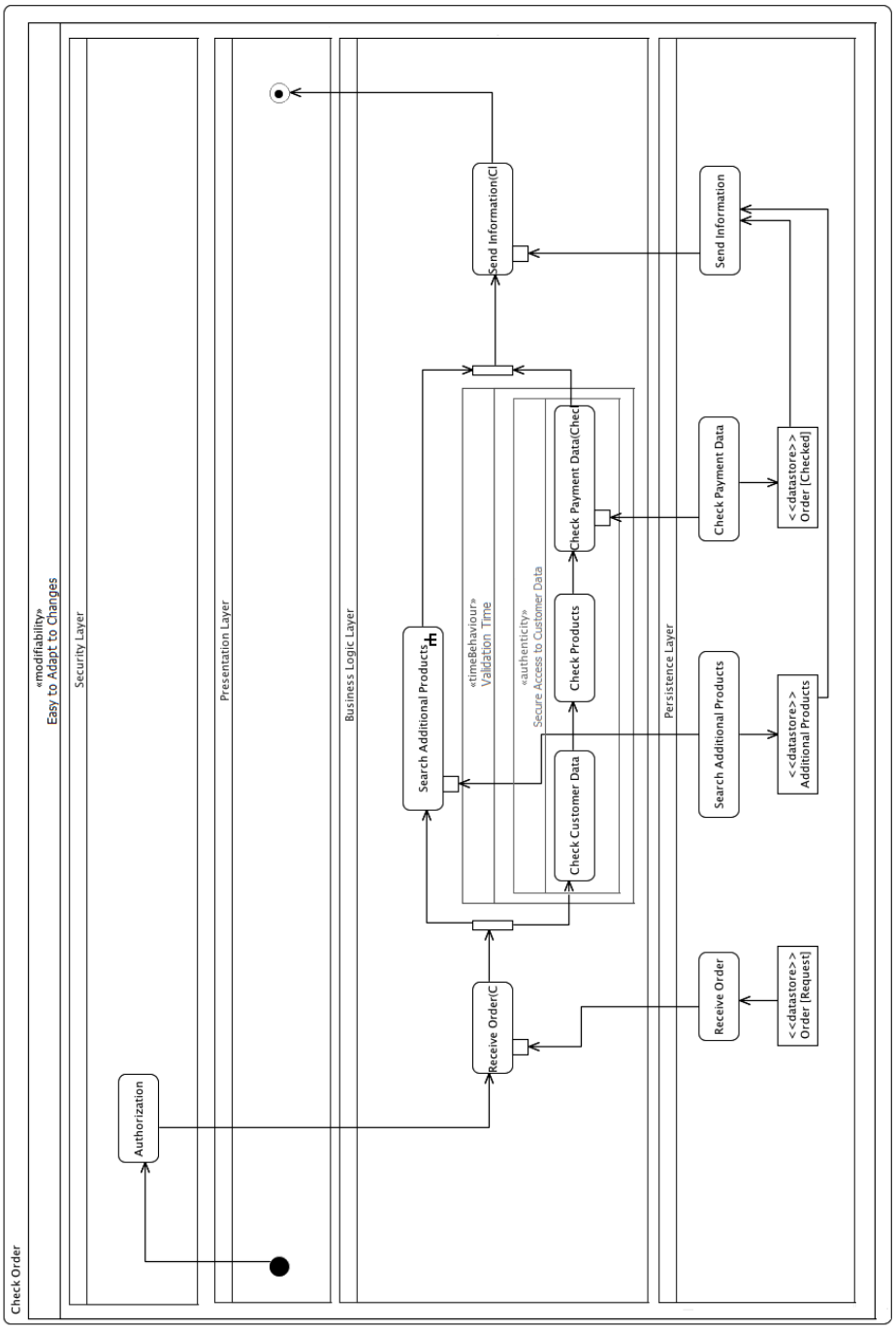


Figure 3.19: Activity diagram of the Check Order Use Case divided in layers.

as the architectural decisions, and they are selected depending on the architectural decisions already made and the annotated non-functional requirements. For the online-shop, for example, since the action *Check Payment Data* of the *Check Order* Use Case (Figure 3.19) is annotated with the *Authenticity* non-functional requirement and a security layer has been created, the *Authorization* design pattern (Fernandez, 2003) would be applied for defining new authentication actions at the beginning of the Activity diagram.

Secondly, the specific technologies or development frameworks are manually linked with the design elements. These links lead to the application of a set of ATL transformations that generate the specific architecture for such technologies. Thus, for supporting the security requirements in the online-shop, the Spring Security framework (*Spring Security*, 2014) would be linked with the generated design elements and the specific architecture for that framework would be generated.

Finally, part of the source code of the system is generated by means of a set of model-to-text transformations based on the architecture design tailored to the specific technologies.

The integration of this two approaches lead to the definition of a process that facilitates the design of multi-layer systems based on frameworks and aligned with the organization business processes and goals. In addition, the modeling of the interdependencies, between business elements and between requirements, and the semi-automation of this process reduce the likelihood of misinterpretation and the effort that the roles involved in the development process (requirements engineers, architects and software developers) devote to the development of kind of systems.

This section has presented how the outcomes of this thesis can integrate with approaches assisting the architect in the architectural decision making process. This integration reduces the effort that these tools require of architects for requirements analysis and, in turn, reduces the risk of misinterpretation that can be made during this analysis.

3.7 Summary

This chapter has presented all the contributions of this thesis. As we have shown, all these contributions are integrated in a methodology modeling and treating the interdependencies from the early development stages and deriving them to the architectural design phase.

Sections 3.1 and 3.2 show an overview of all the proposed contributions and illustrate that they are perfectly integrated by means of the online-shop running example. The main goal of this thesis was to define this comprehensive methodology.

In the other sections, all the capabilities of each contribution were presented. In Section 3.3, we detailed the contributions for documenting the interdependencies between business elements. Specifically, we proposed the BPCI Extension for modeling the business contextual information on BPMN models, a process facilitating the identification of the BUCs, and the BPCI Eclipse Plugin that supports the defined extension. These contributions allowed us to meet the sub-goals two (i.e. extend the existing notations for documenting the business interdependencies) and three of this thesis (i.e. develop tools supporting the defined extensions). In addition, they facilitate the modeling of more complete and useful, for the subsequent development stages, BPMN diagrams.

Section 3.4 detailed, first, the QuAID profiles for UML Use Case and Activity diagrams, and the QuAID Eclipse Plugin facilitating the modeling of the stereotypes defined in such profiles. The defined stereotypes allow engineers to model and document the relationships between functional and non-functional requirements. The QuAID profiles and Eclipse Plugin allowed us to achieve the sub-goals five (i.e. to develop extensions of the existing notations for documenting the interdependencies between requirements) and six (i.e. to develop tools supporting the defined profiles) of this thesis. In this same section, we also detailed a set of patterns and ATL transformations deriving the functional requirements and their interdependencies from the BPMN diagrams annotated with the BPCI Extension. These contributions enabled us to fulfil the sub-goals eight (i.e. to identify and define patterns guiding in the derivation of the requirements interdependencies) and nine (i.e. to

implement transformations semi-automating the identified patterns) of this thesis. Furthermore, they allow engineers to easily generate functional requirements models annotated with the interdependencies with the quality attributes and, therefore, better aligned with the business.

In Section 3.5, we presented a metamodel and a set of ATL transformations facilitating the analysis of the requirements and their interdependencies, the documentation of the impact of the non-functional requirements on the system, and the identification of the Architectural Significant Requirements. These contributions allowed us to achieve the sub-goal eleven of this thesis (i.e. to provide models and tools to obtain the impact of each quality attribute). The information generated with these contributions can be manually used by architects to start making the architectural decisions, or can be used as input of the tools assisting them.

Finally, Section 3.6 defined how the Use Case and Activity diagrams, annotated with the QuAID profiles, and the Architectural View models can be integrated with three different approaches assisting architects. This integration enabled us to achieve the sub-goal twelve of this thesis (i.e. to integrate the methodology, guiding in the reutilization of the interdependencies, with existing approaches assisting architects). In addition, it reduces the effort that these tools require of architects for requirements analysis and facilitates the design of architectures with functionalities better integrated and aligned with the business.

The proposed contributions provide a number of benefits for modeling, deriving and analysing the interdependencies during the busines, requirements and design phases of the software development. In the previous sections, we detailed the theorical benefits they provide. In the next chapter, we validate that such benefits are actually achieved by assessing the results of their application to three industrial projects.

Chapter 4

Validation of the proposal

"The world requires results. You do not tell others your birth pains. Show them the child."

Indira Gandhi

In the previous chapter the contributions for documenting and reusing the business and requirements interdependencies have been defined. This chapter describes the validation performed to evaluate their usefulness. To that end, they have been applied to three industrial projects, which has allowed us to get both qualitative and quantitative information on the use of the defined contributions with regard to three different characteristics: their feasibility, completeness and the effort required to perform them.

This chapter is organized as follows. Section 4.1 shows an introduction to the validation of the proposal. Section 4.2 details the characteristics, sub-characteristics and metrics defined to validate the approach. Section 4.3 presents the industrial projects in which it has been applied. Section 4.4 specifies the results of the defined metrics. Section 4.5 details a summary of the results, the validity of the results and the lessons learned. Finally, Section 4.6 presents a summary of the validation.

4.1 Introduction

This chapter shows the usefulness of the modeling and documentation of interdependencies between business elements (i.e., business process, quality attributes according to the ISO/IEC 25010 and legacy systems) and between requirements (both functional and non-functional) to facilitate the design of architectures aligned with the business. To validate the process and the contributions detailed in this thesis, they have been applied to three industrial project. Industrial projects were used instead of other validation methods (such as Controlled Experiments or Slice of Life (Shaw, 2002)) since, to properly ensure the impact and benefits of this proposal, reasonably large projects are needed. Therefore, it was more appropriate to use real projects rather than to create the complete business specifications necessary to correctly validate the approach in such Controlled Experiments or Slice of Life.

The three projects used for the validation were developed by the Gloin[1] company. This company was selected for the validation because it is a Spin-off of the University of Extremadura and it has a strong tendency towards the innovation in the development process.

In each of the projects in which the contributions of this thesis have been applied the following characteristics, or validation goals, have been evaluated: their feasibility, completeness and the effort required to perform them.

- With the feasibility characteristics, the author of this thesis evaluated whether it is possible to use the developed profiles, models and tools to model interdependencies between business elements and between requirements in real projects with medium/large complexity.
- The completeness goal was defined to evaluate if the application of the methodology, along with the defined tools, generates a complete set of business and requirements models correctly detailing all the interdependencies addressed by this thesis; as well as the impact of each quality attribute in the system.
- The effort characteristics was defined to assess the work necessary to apply

[1] http://www.gloin.es

the developed models and profiles and the amount of effort saved during the design of the system architecture.

As is detailed below, although these characteristics are still abstract and little measurable, to bring them to a more practical level, a set of more precise and measurable sub-characteristics have been defined for each of them. These sub-characteristics are used to identify qualitative or quantitatively their compliance.

4.2 Validation Characteristics and Sub-Characteristics

In this section, each characteristic is refined into a set of aspects, or properties, quantifiable. Each aspect leads to the definition of a measurable sub-characteristic. To perform this refinement, the GQM methodology was used (Basili, Caldiera, & Rombach, 1994; Van Solingen & Berghout, 1999).

GQM (Goal, Question, Metric) is an approach for defining software metrics. This approach can be used to define metrics assessing the validity of the contributions of this thesis. GQM defines a measurement model divided into three levels:

- *Conceptual level (Goal).* A goal is defined for an object, for a variety of reasons, from various points of view and relative to a particular environment.
- *Operational level (question).* A set of questions is used to define models of the object of study and then focuses on that object to characterize the assessment or achievement of a specific goal.
- *Quantitative level (Metric).* A set of metrics, based on the models, is associated with every question in order to answer it in a measurable way.

In order to be used in the validation of this thesis, this approach has been slightly adapted. In this regard, the *Goals* are the validation characteristics previously defined. The *Questions* form the sub-characteristics defined to qualitatively or quantitatively evaluate their compliance. Finally, *Metrics* have been defined for the questions/sub-characteristics quantitatively measurable. Below the questions and metrics defined for each characteristic/goal are detailed.

4.2.1 Validation Goal: Feasibility

The aim of this goal is to assess the feasibility of using the defined contributions in medium/large complex industrial projects.

To evaluate the feasibility of the contributions improving the modeling of interdependencies at the business level, the following questions and metrics were defined:

- *Question 1.1:* Is it possible to model the indicated relationships between business elements?

  Metric 1.1.1: Percentage of relationships that can be modeled with the presented contributions.

  Metric 1.1.2: Percentage of relationships that could be modeled with other approaches.

- *Question 1.2:* Is it possible to model the BUCs in business process models?

  Metric 1.2.1: Percentage of BUCs that can be modeled with the detailed proposal.

  Metric 1.2.2: Percentage of BUCs or functionalities that could be modeled with another proposal.

To evaluate the feasibility of the profiles and tools defined to improve the identification and definition of relationships between requirements, the following questions and metrics were detailed:

- *Question 1.3:* Are the modeled BUCs automatically derived as Use Cases?

  Metric 1.3.1: Percentage of Use Cases that were automatically derived.

  Metric 1.3.2: Percentage of Use Cases that may be derived with other approaches.

- *Question 1.4:* Is it possible to apply the defined patterns to identify the requirements relationships?

  Metric 1.4.1: Percentage of relationships identified using the detailed patterns.

Metric 1.4.2: Percentage of relationships that would be identified by the patterns defined by other approaches.

- *Question 1.5:* Is it possible to model the indicated interdependencies between functional and non-functional requirements?

  Metric 1.5.1: Percentage of relationships that can be modeled with defined profiles.

  Metric 1.5.2: Percentage of relationships that could be modeled or documented with the methods detailed by other approaches.

With the aim of assessing the feasibility of using the defined interdependencies for design the architecture, the following questions and metrics were specified:

- *Question 1.6:* Is it possible to generate correct diagrams of the system views from the annotations?

  Metric 1.6.1: Percentage of view or attributes that are correctly generated.

  Metric 1.6.2: Percentage of view or attributes that could be generated with other proposals.

- *Question 1.7:* Are the annotations modeled in Use Cases and Activity diagrams used during the architecture design?

  Metric 1.7: Yes/No, they are/aren't used

Finally, in order to evaluate the feasibility of using the whole methodology and the developed tools, the following questions and metrics were defined:

- *Question 1.8:* Do the users state that the method is usable?

  Metric 1.8: Average usability of the methodology (from 0 to 10) according to the users.

- *Question 1.9:* Do the users state that developed tools are usable?

  Metric 1.9: Average usability of the tools (from 0 to 10) according to the users.

4.2.2 Validation Goal: Completeness

The completeness characteristic has been defined to assess whether the presented contributions facilitate the generation of a correct and complete set of business and requirements models, perfectly detailing the interdependencies between business elements and between requirements.

In order to evaluate whether the annotated information at the business level is complete and can be used to generate a suitable set of requirements, the following questions and metrics were defined:

- *Question 2.1:* How many BUCs are not correctly identified in business processes using the defined algorithm?

  Metric 2.1: Number (and percentage) of BUCs that cannot be identified or were incorrectly identified.

- *Question 2.2:* Are additional Use Cases identified, or existing ones modified, during the requirements definition?

  Metric 2.2: Number (and percentage) of Use Cases identified/modified.

- *Question 2.3:* Are additional relationships identified, or existing ones modified, during the requirements definition?

  Metric 2.3: Number (and percentage) of relationships identified/modified.

To assess if the information detailed at the requirements level is complete and can be used to generate an architecture aligned with the business, the following questions and metrics were specified:

- *Question 2.4:* Are additional relationships identified, or existing ones modified, during the design of the architecture or in the subsequent development phases?

  Metric 2.4: Number (and percentage) of relationships identified/modified.

- *Question 2.5:* Are the non-functional requirements that most impact on the system the most architecturally significant?

  Metric 2.5: Yes/No, they are/aren't.

Finally, to validate the completeness of the whole process, the following question and metric was defined:

- *Question 2.6:* Is any additional diagram documenting the interdependencies necessary to facilitate the architectural decision making?

  Metric 2.6: Yes/No, additional diagrams are/aren't needed.

4.2.3 Validation Goal: Effort

The purpose of the Effort validation goal is to validate the time required to perform the presented methodology and amount of effort that can be saved in the design of the system architecture. To this end, the following questions and metrics were defined:

- *Question 3.1:* How much effort is needed for documenting the interdependencies between business elements?

  Metric 3.1: Increased in the effort (in man-hours and percentage) in the business analysis phase with respect to the estimations.

- *Question 3.2:* How much effort is needed for modeling the interdependencies between requirements?

  Metric 3.2: Increased in the effort (in man-hours and percentage) in the requirements analysis phase with respect to the estimations.

- *Question 3.3:* How much effort is saved in the architecture design knowing the requirements relationships?

  Metric 3.3: Decreased in the effort (in man-hours and percentage) in the architecture design phase with respect to the estimations.

- *Question 3.4:* What is the return on investment (ROI) of applying the presented methodology?

  Metric 3.4: M3.1 + M3.2 + M3.3.

4.3 Industrial Projects

This section briefly outlines the three projects in which the presented methodology has been applied. The three projects are industrial applications of medium/large size and with a medium/high development complexity due to the complexity of the business in which they had to be deployed and the technologies used.

4.3.1 Data Clean-up and Integrity (DCI)

Data Clean-up and Integrity (DCI) is a project contracted by FastTrack[2] via SGAE[3] to Gloin. The main goal of this project was to develop a software product to periodically measure the data quality of the information published by different Digital Copyright Societies. Specifically, this application had to measure the data quality of more than eighty millions of musical works published by forty societies. It facilitates the measurement of two data quality dimensions, the integrity and the consistency of the information (Mcgilvray, 2008). In the integrity dimension, the validity of the musical works data is checked, according to different business rules. In the consistency dimension, the correctness of the data of each musical work stored in the repositories of the societies are checked against to the data stored in the repository of the society owner of the musical work.

To develop this product Gloin, along with business experts, made a thorough analysis of the copyright business to identify the goals and processes that the new system had to support. The main business goals identified were: the business processes and the system had to be easy to scale depending on the workload (in order to control the processing time of each data quality assessment), the assessed data had to be strictly confidential, and the evaluation of musical works data quality had to be efficient in order to the societies' servers to absorb the workload hauled by the system. The main business processes are: to extract the complete musical works repertoire of a society, to extract incremental repertoires with the latest modifications, and assess the data quality of the repertoire of a society.

[2] http://www.fasttrackdcn.net
[3] http://www.sgae.es

This was the first project in which the presented methodology was applied. It was applied in this project because of three main reasons:

- As section 1.6.1 details, the problems and the need of developing this thesis was identified developing projects for SGAE. Therefore, it was desirable to apply the solutions generated in a project with similar problems.
- For the project to be successful, it was necessary to developed a systems fully aligned with the business. The results of this thesis improve the design of systems aligned with business.
- For extracting the repertoires of the societies, it was necessary a seamless integration with the already deployed legacy systems. The methodology detailed here improves the managing of the interdependencies with legacy systems.

4.3.2 BeeFun

BeeFun[4] is an internal project developed by Gloin applying the latest research made by employees of the company in Cloud Computing and Mobile Development. BeeFun is an application for mobile devices that, in contrast with the existing ones, allows one to send messages to a not predefined group of users depending on their contextual information(such as where they are now or what their preferences are). Thus, an evolution of the messages to group of users is achieved, making them more direct and fast. In addition, the same infrastructure can be used by Mobile Marketing Companies to send ads to users based on their contextual information. Thus, these ads will only hit the users that can be interested on them; in return, the users receive a bonus for each received message.

To develop this product, Gloin made a thorough analysis of the Mobile Apps and Mobile Marketing business in order to acquire a deep knowledge on them and being able of detailing the business goals and process that the system should support, since there were no business experts in the company having such information. The following are the most important business goals identified: the users/mobile devices contextual information is private and never had to come out of the mobile devices

[4] http://beefunapp.com/

and the system had to be efficient in the use of the mobile resources. The main business processes identified are: to gather the contextual information, to create predefined messages or ads, to send a message to a group of users and to get the rewards.

The contributions of this thesis have been applied in this project because of the following reasons:

- The developed system had to perfectly support the business processes detailed during the market analysis. The results of this thesis facilitate the development of systems fully aligned with the business.
- The quality attributes on data privacy and efficiency on the mobile resources were extremely important. The process defined in this thesis improves the treatment of interdependencies between requirements, enhancing their achievement.

Finally, in this project the presented methodology was integrated with the approach of Garcia-Alonso et al. (2013) in order to validate the use of the modeled interdependencies by tools that assist the architect in the architectural decision making process.

4.3.3 NimBees

NimBees[5] is another internal project developed by Gloin aimed at developing an API facilitating the implementation of a new kind of mobile devices apps. This API implements a new service provision model known as *People as a Service (PeaaS)*(Guillen et al., 2014). This new model converts mobile devices into Cloud Platforms providing services on the contextual information of the mobile devices and their owners (such as their personal data, the visited web pages, the places they have been, when they have been there, etc.) or inferences obtained from it (such us, for example, their preferences or likes depending on the visited web pages). Thus, the applications or services interested in these data can easily access to them calling one of the services already deployed in the cloud enabled mobile device. In exchange for this

[5] http://www.nimbees.com/

information, owners gain economic or social benefits (such as monitoring the vital signs of ill people in order to provide them a better health care). In any case, the mobile device owners always can indicate what information they want to provide and to whom.

To develop this product Gloin reused the business knowledge already acquired during the development of BeeFun. The main business goals defined for this system are: only authorized services or applications could access to the captured information and the inferences made, and the API had to be efficient in the use of the mobile resources. The main business processes identified are: to gather data from the mobile device sensors and records, to infer information from the stored data, and to provide the stored information.

The results of this thesis have been used in the development of this system because, as in BeeFun, the security and efficiency quality attributes were crucial. The contributions of this thesis facilitate the analysis of their interdependencies in order to design a system correctly achieving them. In addition, as in BeeFun, the approach of Garcia-Alonso et al. (2013) was also used in order to validate the benefit provided by the integration of the both approaches and their continued use of them by the same developers.

4.3.4 Characteristics of the Industrial Projects

Table 4.1 shows a summary of the characteristics of each project. It details the number of business processes, quality attributes, Use Cases and non-functional requirements per project. It also includes the number of UCP (Use Case Points) (Clemmons, 2006) of each project in order to show their complexity and to calculate estimations of the effort required to develop them (as UCP is the technique used in the company to calculate these estimations). Finally, the table also includes the real effort in man-hours devoted in the business analysis, requirements analysis and architecture design stages.

Table 4.1: Summary of the project characteristics.

	DCI	BeeFun	NimBees
Business Processes (BPs)	7 BPs	10 BPs	7 BPs
Business Quality Attributes (QAs)	5 QAs	4 QAs	4 QAs
Use Cases (UCs)	63 UCs	40 UCs	35 UCs
Non-Functional Requirements (NFRs)	6 NFRs	6 NFRs	4 NFRs
Use Case Points (UCPs)	629 UCPs	566 UCPs	377 UCPs
Business Analysis	1086 hours	1120 hours	650 hours
Requirements Analysis	1612 hours	1550 hours	965 hours
Architecture Design	1593 hours	1412 hours	880 hours

4.4 Validation Results

In this section, an analysis of the contributions of this thesis is done with respect to the feasibility of applying them, the completeness of the generated information and artifacts, and the effort that entails their implementation. This analysis has been made applying the questions and metrics listed above on the industrial projects.

4.4.1 Validation Goal: Feasibility

Below the results of applying the questions assessing the feasibility of the contributions are detailed.

Question 1.1: Is it possible to model the indicated relationships between business elements? Metric 1.1.1: All the interdependencies between business processes, quality attributes (according to the ISO/IEC 25010) and legacy systems were modeled in the three projects by means of the BPCI Extension. It was possible to model all relationships between business processes and quality attributes because they can be detailed at a different level thanks to defined quality attributes tree. This characteristic allowed developers of the DCI and BeeFun projects to model some quality attributes at a high level so that, at a later stage, they could be refined. *Metric 1.1.2:* The generated artifacts were also used to identify the interdependencies that could also be modeled by other approaches documenting these kind of interdependencies, such as (Pavlovski & Zou, 2008). This analysis showed that, for example, for the DCI project only the 76% of the relationships could be modeled, since some

of the quality attributes (such as *Compatibility* or *Capacity*) were not operational and this kind of requirements cannot be detailed with this approach.

Question 1.2: Is it possible to model the BUCs in business process models? Metric 1.2.1: All the BUCs identified in the business processes were modeled. However, in some processes of the DCI project (due to their size and complexity) the modeling of every BUCs, together with the rest of interdependencies, led to a decrease in the legibility of the models. Anyway, the importance and ease of modeling of these relationships was highlighted by business experts and requirements engineers. *Metric 1.2.2:* Once modeled the BUCs, they were analysed in order to identify if all of them could also be documented with other proposals, such as using the *Consecutive Flows* defined in (De la Vara & Sánchez, 2009). From this analysis, we identified that this flow would allow us to model around the 90% of the BUCs. The 10% of BUCs that could not be modeled would imply BUCs covering exception flows, a simple task, or involving communication with external agents.

Question 1.3: Are the modeled BUCs automatically derived as Use Cases? Metric 1.3.1: The defined ATL transformations allow engineers to automatically derive the system Use Cases from the modeled BUCs. In the three industrial projects in which these transformations were applied, all the BUCs were correctly derived as Use Cases. *Metric 1.3.2:* In other methodologies, such as (De la Vara & Sánchez, 2009), the Use Cases, or the functional requirements, cannot be derived automatically; the engineers have to manually derive them using some provided guides. Those approaches semi-automating the derivation of Use Case from business processes, such as (Rodríguez, de Guzmán, Fernández-Medina, & Piattini, 2010) or (Siqueira & Silva, 2011), do not facilitate the identification of BUCs for take into account the granularity of the processes.

Question 1.4: Is it possible to apply the defined patterns to identify the requirements relationships? Metric 1.4.1: The ATL transformations associated to each pattern were applied in the three projects. All transformations and patterns were properly applied. In addition, this allowed us to identify situations in which some transformations had to be refined. Thus, for example, in the DCI project we identified some situations leading to the identification of several *extend* relationships

between two Use Cases. This allowed us to adapt the patterns so that once the first *extend* relationship between two Use Cases is identified, the rest of them will only entail the modeling of the needed *extension points* and their linkage with the modeled relationship. *Metric 1.4.2:* Other existing approaches, such as (Dijkman & Joosten, 2002), detail patterns for the identification of *extend* relationships in some situations, but there are some situations not covered and other relationships that are not identified such as *include*, *generalization* o interdependencies with quality attributes. Evaluating the identified interdependencies and the patterns detailed by these approaches we estimated that they would have identified an average of the 8.5% of relationships identified by the patterns listed herein.

Question 1.5: Is it possible to model the indicated interdependencies between functional and non-functional requirements? Metric 1.5.1: In the three projects all the relations derived by applying the above patterns were correctly modeled by means of the QuAID profiles. The Use Cases interdependencies were automatically generated, by means of the ATL transformations, and the Activity diagrams relationships were manually outlined from the ones already modeled in the Use Cases. *Metric 1.5.2:* In (Dörr, 2011), the authors detail a proposal to document the requirements interdependencies associating notes in natural language to the Use Cases. Therefore, the same relationships could be modeled. However, unlike the presented method, these relationships cannot be automatically derived from the business; they have to be manually obtained and documented. In addition, as they are documented in natural language, they cannot be easily reused by tools assisting in the subsequent development stages.

Question 1.6: Is it possible to generate correct diagrams of the system views from the annotations? Metric 1.6.1: In the three industrial projects both the Use Cases and the Processes View were correctly generated to identify the impact of each quality attribute. Additional views were not created because of the effort required, in analysing all the annotated diagrams, to model them. *Metric 1.6.2:* If the interdependencies would have been detailed as notes in natural language (using the (Dörr, 2011) approach), it would not be possible to directly generate, by means of ATL transformations, nor the Use Cases View neither the Processes View.

Question 1.7: Are the annotations modeled in Use Cases and Activity diagrams used during the architecture design? Metric 1.7.1: The close collaboration between development teams and the author of this thesis has revealed us the use that architects made of the defined annotations. In every project such interdependencies were widely used, particularly the ones related with non-functional requirements identified as architectural significant. In the BeeFun project, for example, they were used to identify which Use Cases had to be efficient in the use of mobile device resources in order to decide in which kind of iOS and Android services the system has to be based on and to design the back-end of the application supporting those services.

Question 1.8: Do the users state that the method is usable? Metric 1.8.1: In every project the development teams indicated that the methodology was usable. The lower usability level was obtained in DCI. This was due to the fact that this was the first project in which the presented methodology was used, so certain situations (as the one detailed in Q1.4) were not resolved. In addition, the learning and use of a new methodology led to an impact on the development teams.

Question 1.9: Do the users state that developed tools are usable? Metric 1.9.1: All development teams indicated that the tools were usable and that they facilitated the implementation of the methodology. As in the previous question, DCI was the project in which a lower usability level was obtained due to certain bugs that were fixed in the later versions. In addition, the usability of the tools was lower than the method. This difference was analysed and we identified that it was mainly because the usability of the basic tools extended to support the contributions (BPMN2 Modeler and Papyrus) was lower for the development teams than the usability of the tools commonly used by them to model business processes and UML diagrams (Bonita BPM[6]and Enterprise Architect[7]).

Table 4.2 shows a summary of the metrics obtained applying the feasibility questions on the three industrial projects.

[6] http://www.bonitasoft.com/
[7] http://www.sparxsystems.eu/

Table 4.2: Summary of the results for the Feasibility validation goal.

Project / Question	DCI [Metrics]		BeeFun [Metrics]		NimBees [Metrics]	
Q1.1: Is it possible to model the indicated relationships between business elements?	100%	76%	100%	72%	100%	74%
Q1.2: Is it possible to model the BUCs in business process models?	100%	88%	100%	86%	100%	92%
Q1.3: Are the modeled BUCs automatically derived as Use Cases?	100%	0%	100%	0%	100%	0%
Q1.4: Is it possible to apply the defined patterns to identify the requirements relationships?	100%	9%	100%	8%	100%	8.5%
Q1.5: Is it possible to model the indicated interdependencies between functional and non-functional requirements?	100%	100%	100%	100%	100%	100%
Q1.6: Is it possible to generate correct diagrams of the system views from the annotations?	100%	0%	100%	0%	100%	0%
Q1.7: Are the annotations modeled in Use Cases and Activity diagrams used during the architecture design?	Yes		Yes		Yes	
Q1.8: Do the users state that the method is usable?	7		9		9	
Q1.9: Do the users state that developed tools are usable?	6		7		7	

4.4.2 Validation Goal: Completeness

Below the results of applying the questions assessing the completeness of the contributions are detailed.

Question 2.1: How many BUCs are not correctly identified in business processes using the defined algorithm? Applying the algorithm for identifying BUCs (defined in Section 3.3.2), all BUCs necessary to properly support the business processes were identified. This algorithm describes a set of steps to follow to identify BUCs at an appropriate granularity, so it was used by requirements engineers to manually identify when a BUC should begin or end.

Question 2.2: Are additional Use Cases identified, or existing ones modified,

during the requirements definition? In the three industrial projects some Use Cases automatically derived were refined. Some Use Cases identified from business processes were still specified at a high level, so they had to be refined and split into several Use Cases. The project in which more Use Cases had to be refined was BeeFun, with a 17,5% of them (a 3,5% more than in DCI and a 9% more than in NimBees). Analysing these results, we identified that this situation was caused because the system had to support an unknown domain for developers; and since there was not business experts guiding in the modeling of the business processes and in identification of the BUCs, when they were derived as Use Cases some of them had to be refined.

Question 2.3: Are additional relationships identified, or existing ones modified, during the requirements definition? Additional interdependencies, or modification of the existing ones, were identified in the three projects. However, most of these relationships were caused by the refinement of the Use Cases, since they led to the specification of new interdependencies between the new Use Cases and the quality attributes. Similarly, the refinement of the quality attributes (in DCI and BeeFun there were an increased from 5 and 4 respectively to 6) also led to the modification of the existing relationships with the Use Cases. Finally, as in the previous question, in BeeFun the lack of knowledge in the domain led to higher rates of modifications of relationships.

Question 2.4: Are additional relationships identified, or existing ones modified, during the design of the architecture or in the subsequent development phases? Only in BeeFun additional interdependencies were identified during the architecture design. Concretely, four new relationships were identified, which represented a 3,7% of the total detailed interdependencies. These relationships were caused because some Activity diagrams were not fully marked with the *Confidentiality* non-functional requirement that they have to meet to fully guarantee the privacy of user data; in the system designing this threat was identified and corrected.

Question 2.5: Are the non-functional requirements that most impact on the system the most architecturally significant? All non-functional requirements that were identified as architecturally significant were the ones that most impacted in architec-

tural views. In BeeFun, for example, the ASRs were *Confidentiality* and *Resource Utilization*, which were annotated in the 87,5% of the Use Cases, only some functionalities of the application backend had not to achieve these requirements.

Question 2.6: Is any additional diagram documenting the interdependencies necessary to facilitate the architectural decision making? In DCI the development teams indicated that it would be desirable to have a Class diagram annotated with the interdependencies. Thus, architects could apply the selected architectural patterns and tactics directly on this diagram to obtain the final system design. However, both in BeeFun and NimBees, this methodology was applied together with the approach of Garcia-Alonso et al. (2013) in order to also guide in architectural decisions making. In both projects the need of an annotated Class diagram was not detected, since the integrated approach guides in the selection and application of patterns, semi-automatically generating the final system design.

Table 4.3 shows a summary of the metrics obtained applying the completeness questions on the three industrial projects.

4.4.3 Validation Goal: Effort

Below the results of applying the questions evaluating the effort of applying the methodology, as well as the return of investment, are detailed.

The increase/decrease of effort in each development stage is calculated by comparing the real effort devoted to the development of each project (using the methodology) with the estimation of the effort that should have been spent according to the historical data of the company. These data show that the productivity by UCP is 14 hours, of which 1,68 hours correspond to the business analysis phase, 2,52 hours to the requirements analysis stage, and 2,8 hours to architecture design.

Question 3.1: How much effort is needed for documenting the interdependencies between business elements? The modeling of the interdependencies between business elements led to an increase of effort in all projects. In DCI and NimBees the increase was low (between 2,5% and 2,8%) and consistent with the work devoted to document this information. However, in BeeFun the increase was around 17,8% with

Table 4.3: Summary of the results for the Completeness validation goal.

Project / Question	DCI [Metrics]	BeeFun [Metrics]	NimBees [Metrics]
Q2.1: How many BUCs are not correctly identified in business processes using the defined algorithm?	0 BUCs 0%	0 BUCs 0%	0 BUCs 0%
Q2.2: Are additional Use Cases identified, or existing ones modified, during the requirements definition?	9 UC 14%	7 UC 17,5%	3 UC 8,5%
Q2.3: Are additional interdependencies identified, or existing ones modified, during the requirements definition?	26 rel 12,9%	20 rel 18,7%	6 rel 6,6%
Q2.4: Are additional relationships identified, or existing ones modified, during the design of the architecture or in the subsequent development phases?	0 rel 0%	4 rel 3,7%	0 rel 0%
Q2.5: Are the non-functional requirements that most impact on the system the most architecturally significant?	Yes	Yes	Yes
Question 2.6: Is any additional diagram documenting the interdependencies necessary to facilitate the architectural decision making?	Yes, the Class Diagram	No	No

respect to the estimations. This result was thoroughly analysed in order to identify what originated it. We identified that this increase was mainly due to the lack of interaction with business experts guiding in the analysis of the domain, rather than the introduction of the new methodology.

Question 3.2: How much effort is needed for modeling the interdependencies between requirements? The effort of using this methodology in the requirements analysis stage was lower than in the business analysis phase in all projects, even though engineers had to document and detail the interdependencies in several diagrams and at a lower level. This reduction was mainly obtained because some activities of this phase were automated by using ATL transformations. In DCI and NimBees only an increase of effort of 1,8 % and 1,6 %, respectively, with respect to the estimations was identified. Finally, as in the previous question, a significant increase was detected in BeeFun. This increase was mainly caused because the lack of domain knowledge led to the refinement of many Use Cases and interdependencies.

Question 3.3: How much effort is saved in the architecture design knowing the requirements relationships? In all projects, the knowledge of the interdependencies between requirements and with legacy systems resulted in a considerable decrease in the effort spent in the architecture design phase. In DCI, in which solely the methodology detailed in this thesis was applied, a decrease around 9,5% with respect to the estimations was identified. In BeeFun and NimBees, we detected a higher reduction of the effort (getting a decrease of 16,7% in NimBees), partly due to the integration and use of the approach guiding in the architectural decisions making. This reduction also validates that both approaches can be integrated and that the modeled information is highly useful for the tools assisting in the architectural decisions making.

Question 3.4: What is the ROI of applying the methodology? Although the incorporation of the methodology led to an increase of effort in both the business analysis and the requirements analysis phases, the benefits that it provides for architecture design make profitable the investment of that effort. Thus, in DCI and NimBees projects the ROI gotten for the three initial development phases was of 2,5% and 5,5% respectively. In DCI, the use of this methodology led SGAE to save 4,200 euros of the budget for these three development phases, which was 167,300 euros. In NimBees, Gloin saved 5,600 euros with respect to the 94,800 euros finally invested. In BeeFun, Gloin did not obtain an effective return on investment with respect to the estimations, mainly due to the large effort devoted to the acquisition of the domain knowledge. Nevertheless, the use of the methodology allowed engineers to reduce the effort devoted to the architecture design, decrementing the accumulated delay and the consequent increase of the project cost.

Table 4.4 shows a summary of the metrics obtained applying the effort questions on the three industrial projects.

4.4.4 Further Observations

This section describes additional interesting findings that were observed during the application of the presented approach in the industrial projects.

Table 4.4: Summary of the results for the Effort validation goal.

Project / Question	DCI [Metrics]	BeeFun [Metrics]	NimBees [Metrics]
Q3.1: How much effort is needed for documenting the interdependencies between business elements?	30 hours 2.8%	170 hours 17,8%	16 hours 2,5%
Q3.2: How much effort is needed for modeling the interdependencies between requirements?	29 hours 1,8%	124 hours 8,7%	15 hours 1,6%
Q3.3: How much effort is saved in the architecture design knowing the requirements relationships?	-167 hours -9,5%	-172 hours -10,8%	-176 hours -16,7%
Q3.4: What is the ROI of applying the methodology?	-108 hours -2,5%	122 hours 3,1%	-145 hours -5,5%

An important observation was the reduction of the number of bugs detected during the integration of the functionalities. In the DCI project, during the implementation and integration of the system, a reduction of 12% of bugs with respect to historical data obtained from other projects was observed. These data were analysed to identify the reasons originating this reduction. We noted that the main reduction was because the requirements relationships had been treated more carefully from the initial stages, resulting in a design and implementation paying more attention to this information.

Another important finding was the reduction of the problems arising from the integration with legacy systems. In the DCI project, during the deployment phase a reduction of 18% of the integration bugs with legacy systems, with respect to historical data, was observed. Although the integration with legacy systems were documented and treated in historical projects, it was not as clearly specified from the business analysis phase as with this approach. This led to an improvement in the treatment of these relationships and, therefore, a reduction in the number of integration problems.

These observations have shown us that the application of the methodology presented in this thesis, first, improves the specification of interdependencies and reduces the effort of designing of architectures fully aligned with the business. Secondly, it facilitates and improves the integration, deployment and maintenance of

the systems, reducing the number of bugs.

4.5 Discussion

This section summarizes the results obtained, the threats to the validity of the results and the lessons learned during the application of the methodology in the industrial projects.

4.5.1 Summary of Results

The aim of the validation done was to assess the feasibility of the methodology, the completeness of the resulting set of artifacts, and the effort needed for the application of the methodology. For each of these validation goals, we obtained statistical data used to confirm the validity and the benefits of the contributions presented in this thesis.

The results obtained evaluating the feasibility were very positive. Every interdependency between business processes and quality attributes (according to the ISO/IEC 25010) or legacy systems were correctly modeled, in addition to the BUCs needed to support each business process. Furthermore, these models were used to generate functional requirements models correctly annotated with the corresponding interdependencies. These relationships were in turn used for the identification of the architectural significant requirements and for the architectural decisions making. The only feasibility drawbacks were obtained for the usability of the methodology and the supporting tools in the DCI project (Q1.8 y Q1.9). These drawbacks were caused because DCI was the first industrial project in which the methodology and the tools were used, and some bugs were identified. These bugs were corrected for the following projects, as demonstrate the increase in the usability value obtained in them.

In general, the data collected for the questions **Q1.1 to Q1.9 strongly confirm that the elements of the proposed method are feasible**, i.e., the new defined development activities can be applied by averagely trained personnel and

the artifacts proposed can be created for real-life projects.

The results obtained assessing the completeness were encouraging. The validation results showed that the artifacts generated in each development phase were complete and very useful for the following stages. Thus, on average only 13,5% of the Use Cases and 12,8% of the interdependencies derived from the BUCs identified at the business level had to be refined or modified. These data were even better for the architecture design stage, where on average only 1,2% of the relationships had to be refined. In addition to these data, the development teams of the DCI project indicated that only one additional diagram with the interdependencies annotated, a Class diagram, would be useful for the architecture design. This deficiency was corrected in the following industrial projects integrating the presented methodology with the approach of Garcia-Alonso et al. (2013), which reuse the interdependencies modeled with this methodology for semi-automating the generation of the architectural design.

The data collected for the questions **Q2.1 to Q2.6 firmly endorse that the artifacts generated with this methodology are complete**. This methodology facilitates the documentation of a broader range of information, which is very useful for the subsequent development stages.

The results related to the effort required to apply the proposal are very promising. In the business analysis phase, only a small increase of effort around 2,65% was necessary to detail the interdependencies. Similarly, in the requirements analysis stage, only 1,6% of extra effort was required. It has to be noted that in BeeFun a higher effort was required in both phases, mainly due to the lack of knowledge in the domain. Finally, the main benefit of modeling such information was obtained in the architecture design phase, were a reduction in effort of 12,3% on average for the three projects was identified. Part of this reduction (in BeeFun and NimBees) was also due to the integration with the approach assisting in the architectural decisions making (Garcia-Alonso et al., 2013). In DCI, in which this approach was not used, a 9,5% of reduction was detected. The best ROI was obtained in NimBees with a reduction of 5,5% of the effort devoted for the three initial development phases.

Although the obtained ROI may seem small, if we extrapolate it to the revenue obtained by a company along a year, it would lead to a considerable increase in profits. For example, if this methodology is applied to Indra, which in 2013 achieved 2.914 million euros of revenues[8], and considering the ROI obtained in DCI (as this was the project in which only the proposed methodology was applied), which was 2,5%, then this company would obtain an increase in profits of 72,85 million euros. Therefore, the application of the proposed contributions allows companies to be more profitable and/or competitive.

The data collected for the questions **Q3.1 to Q3.4 strongly support the effort validation goal**, indicating that the modeling of the interdependencies between business elements and requirements reduces the effort spent in the architecture design phase.

Additionally, the treatment and documentation of such interdependencies also led to a reduction of the bugs reported due to integration problems between functionalities or with legacy systems, which in turn also decreased the time and costs of integration, deployment and maintenance of the systems.

4.5.2 Threats to Validity

The presented methodology was evaluated in three industrial projects. Data was collected to evaluate their feasibility, completeness and effort. In this section, the possible threats to validity are discussed according to the four types of possible threats reported by (Wohlin et al., 2000).

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 this thesis, this validity is related to the information

[8] http://www.indracompany.com/en/noticia/indras-profit-for-2013-was-116-million

sources used (historical data of the company, artifacts generated during the projects development and researcher's observation) and the characterization of the theory under analysis.

The main source of information used during the validation was the historical data on the productivity of the development teams. These data were obtained by analysing the effort reports generated from other projects with similar size and complexity (such as GLOCO and Geprodist).

The artifacts generated during the projects development, and from which the results of the metrics have been obtained, have been used for the documentation and implementation of the three systems (DCI, BeeFun and NimBees). These systems are functionally complete and are or have been successfully used (BeeFun, for example, and its business plan allowed Gloin to win the LaunchPad Denmark Business Plan Cup). Therefore, we can derivate that these artifact are valid and correct.

Finally, researcher expectancies are considered to having been properly addressed because both positive aspects and negative aspects of the methodological approach were discovered and reported as result of the validation. Existence and report of only positive results may suggest that experimenter expectancies may have affected the observation.

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 the assessment of this methodology, this validity is related to the metrics and data obtained answering the questions.

The data obtained during validation are objective since they were real values acquired by analysing the generated artifacts. Only the questions Q1.7 to Q1.9 contain information obtained analysing the researchers observations and the developers teams opinions. Furthermore, although the metric of Q1.7 is the result of an observation, this observation is validated by the general reduction in effort spent

in the architecture design stage of the three projects, as Q3.3 evidences. Q1.8 and Q1.9 were mainly used to obtain information on how to improve the usability of the presented methodology and the tools supporting it.

The questions measuring the effort of applying the methodology and the return on investment obtained (Q3.1 to Q3.4) are based on the comparison of the real effort data with the estimations obtained from the historical data .The accuracy of such estimations may involve a threat to the validity of the results. The reliability of the estimations is lower than the real effort data. Nevertheless, large deviations of the estimated effort are unlikely, due to experience of the company in making this kind of estimations. Slight deviations do not impact the conclusions we draw. Thus, for example, in NimBees, Gloin obtained 5,5% of ROI, some deviations would not strongly impact in this ROI.

Finally, some of the metrics defined for the questions Q1.1 to Q1.6 are values obtained from the comparison of the artifacts generated by this methodology and the information that would be modeled with other approaches. The results to these metrics are estimations calculated by the author of this thesis by the thorough analysis of the approaches detailed in Section 2. This analysis, and thereby the estimates obtained, may slightly differ depending on knowledge on each approach. Nevertheless, due to the deep analysis done, this would lead to small increases or decreases in the percentages obtained, which does not significantly affect the conclusions drawn.

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 that does not exist. In the validation of this methodology, these threats are related to truth of the metrics obtained, and the application of the methodology.

The results obtained during the validation of this methodology have been derived from analysis of the artifacts generated during the development process. The generation of these results may have been influenced by the support provided by

the researchers to the development team during the application of the methodology. Nevertheless, this support was diminishing in each project, being almost inexistent in the last project in which it was applied (NimBees). The results obtained in this project are even better than in the others. Therefore, the results obtained are correct and do not mean a threat to the validity.

Another possible threat is that due to the close collaboration between the researchers and the development teams, the contributions were applied more carefully than usual. However, this would also have led to an increase in the effort devoted to the development of the projects. Instead, the overall project efforts have been reduced, as Q3.4 shows. Therefore, this threat has not occurred.

External Validity

External validity is concerned with the generalization of the conclusions of the validation. Threats to external validity refer to the ability to generalize the results and conclusions beyond the setting of the study. In the evaluation of this methodology, this validity is related to kind and complexity of the industrial projects.

The three industrial projects were of medium/large size and complexity, but with some differences. DCI was the project in which greater business knowledge was required, but there were business experts supporting the project and providing all the knowledge needed. Also, it had to interact with a large number of legacy systems. In BeeFun the domain knowledge had to be acquired, without the help of business experts, and new technologies to development teams had to be applied. In addition, it had a high number of requirements and quality attributes very restrictive. NimBees was a medium sized project based on a known domain with technologies already used, but with very stringent requirements. Therefore, the diversity and the wide range of the projects in which the contributions were validated allowed us to solve the threat to the external validity.

A remaining threat to external validity is that many details about the participants and the artifact generated during the projects development cannot be presented due to confidentiality reason.

4.5.3 Lessons Learned

The validation of the methodology in the three industrial projects allowed us to know its strengths and weaknesses. The main strengths have been specified above: business information better documented, artifacts documenting requirements more complete, architectures better aligned with the business and a significant reduction of the effort in the architecture design stage.

Despite all the advantages identified for the proposed contributions, we also identified some weaknesses. These weaknesses were: the loss of readability of the annotated diagrams, the use of business processes at different abstraction levels, and the manual generation of Activity diagrams.

One of the main weaknesses identified during the validation of the methodology was the loss of the readability of the generated models. In the business process models, engineers had to model interdependencies with legacy systems and with quality attributes, and the Business Use Cases. This means that depending on the number of elements/interdependencies to model some processes may lose legibility, especially if they were not initially modeled taking into account that in a further step this information must be included. This loss of readability was not identified in the requirements models since less information has to be annotated in the same element/model. A solution would be the development of modeling tools allowing engineers to show/hide certain information/interdependencies depending on their needs. This solution would facilitate the generation of complete and legible models.

Furthermore, the identification of BUCs largely depends on the abstraction level of the business processes. These processes can be very abstract, detailing only business activities, or very specific, even documenting information for their implementation or execution. The algorithm defined in Section 3.3.2 facilitates the identification of BUCs at a similar granularity, regardless of abstraction level of the processes. Nevertheless, to facilitate the application of this algorithm, it would be desirable business processes at a similar abstraction level. This can be solved establishing in the methodology a common style for modeling the business processes at a specific abstraction level.

Finally, the annotated Activity diagrams have to be manually modeled. This takes some time that could be reduced, for example by using transformations to generate them. In this sense, there are some approaches, such as (Siqueira & Silva, 2011), that could be adapted and incorporated to the methodology in order to automatically generate Activity diagrams, annotated with the AD-QuAID profile, from the BUCs modeled in the business processes.

4.6 Summary

This chapter has presented the validation of the contributions of this thesis. They have been applied to three real industrial projects in which their feasibility, the completeness of the artifacts generated, and the effort required to apply them was evaluated.

This validation demonstrated that all the indicated interdependencies between business elements and between requirements can be correctly modeled with the contributions presented. Additionally, the generated models are complete and useful, since only 13,5% of the Use Cases and 12,8% of the relationships derived from the business processes, annotated with the BPCI Extension, had to be refined. Similarly, the application of this methodology entails an increase of the effort devoted to the business and requirements analysis phases, but this effort is subsequently retrieved during the architecture design stage, obtaining an overall return on investment of these three phases of up to 5,5%, according to the results of the validation. Furthermore, the author of this thesis also identified that the modeling of these interdependencies improves the integration of the system's functionalities and of the system with legacy systems. A decrease in the number of bugs identified due to these reasons was detected in the validation of the methodology.

Finally, this validation also allowed us to identify the weaknesses of the methodology. The main identified weaknesses are: when a large number of interdependencies are modeled in business processes their readability may decrease, the use of business processes at different abstraction levels may difficult the identification of the BUCs, and the manual modeling of annotated Activity diagrams entail an increase of effort.

Solutions to these weaknesses, that will be incorporated in future works making the methodology more robust, have been evaluated and presented.

In the previous chapters, we have identified the different areas that could be improved to design IT systems better integrated and aligned with the business, we have detailed the proposed contributions improving such areas, and they have been validated applying them to three industrial projects. In the next chapter, the conclusions and the future works that will be performed by the author of this thesis are detailed.

Chapter 5

Conclusions and Future Works

"Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning."

Winston Churchill

This chapter presents the main conclusions and reflections drawn after the development of this thesis. It summarizes the main contributions presented, how they improve different aspects of the software development and how they impact on other areas and companies. Finally, it also details how the development of this thesis has also contributed to the professional development of its author.

This chapter is organized as follows. Section 5.1 sums up the contributions proposed in this thesis, how they improve different areas of the software development and the conclusions drawn. Section 5.2 details the papers that have been published in the scope of this research. Section 5.3 introduces some near and future work. Finally, Section 5.4 presents a final reflection.

5.1 Conclusions

This thesis started with the aim of solving the integration and misalignment problems detected in the projects developed in the software factory Teseo (and which showed up as an open issue in which researchers were working on). Analysing their roots, we identified that they usually happened because the interdependencies between business elements and the relationships among requirements were implicit in the models. So that, in some cases, they were not properly identified, leading to architecture designs with inaccuracies that had to be fixed as soon as they arose.

This thesis was developed to tackle these problems. To that end, a set of goals were defined for modeling or analysing these interdependencies in the business, requirements and design development stages.

The goals defined for the business phase were to identify what interdependencies between business elements could not be modeled, and to propose notations and tools enabling their documentation. Reviewing the state of the art, we identified that, although there were approaches documenting the relationships between business processes and operational goals, they do not modeled the interdependencies with other elements (such as non-operational goals or with legacy systems). In order to allow engineers to also model these relationships, we proposed an extension of BPMN. This extension also facilitates the documentation of what business tasks will be supported by each functionality, providing a first approximation of the system requirements and fostering their early discussion. In order to facilitate the modeling of this information, the BPCI Eclipse Plugin was implemented. These contributions have allowed us to successfully meet the goals one to three of this thesis.

For the requirements stage, the goals established were to identify the requirements relationships that could not be modeled, and to define profiles and tools allowing their definition. Analysing the state of the art, we found that, although some approaches documented such interdependencies, these relationships were detailed in natural language. So that, the generated artifacts could not be used by tools assisting architects in the subsequent development phases. With the aim of allowing engineers to model the functionalities constrained by each non-functional

requirement, we have presented a set of UML profiles for Use Case and the Activity diagrams. These profiles were implemented in the QuAID Eclipse Plugin. These contributions, in addition to achieve the goals four to six of this thesis, allow engineers to create requirements models better aligned with the business and that can be reused in the subsequent development phases.

Moreover, for the requirements stage, we also established that we had to identify what interdependencies could not be automatically derived from the business models and we had to define patterns and rules automating their derivation. Reviewing the existing proposals, we detected that they did not define how to derivate the relationships between functional and non-functional requirements and, also, they only addressed some relationships between functional requirements. In this thesis, we have presented a number of patterns indicating the most common situations in the annotated business processes leading to the derivation of Use Cases and their relationships. These patterns were automated by means of ATL transformations, thus reducing the effort required to generate an initial but complete version of functional requirements models. These contributions have allowed us to fulfil the goals seven to nine of this thesis.

For the design phase, the defined goals were to identify how the architectural decision making could be simplified using the requirements relationships, and to define notations and a methodology guiding architects in the reutilization of these interdependencies. Analysing the approaches assisting architects, we identify that, although they facilitate the architectural decision making, architects should have a perfect knowledge of the requirements and their interdependencies. Therefore, in order to facilitate the requirements analysis, we have proposed a metamodel simplifying the identification of the Architectural Significant Requirements. In addition, both the defined profiles and the detailed metamodel were integrated with the analysed approaches, reducing the effort required by architects to analyse the requirements and the likelihood of misinterpretations. These contributions enabled us to fulfil the goals ten to twelve of this thesis.

All these contributions have allowed us to achieve the main goal of this thesis, to define a comprehensive methodology improving the design of IT systems better

integrated and aligned with the business.

The benefits provided by this methodology have been validated in three industrial projects. In every project we identified that its application is possible and allows one to generate more complete models, but it also carries an increased effort. Nevertheless, this effort is recovered with a substantial reduction of the time devoted to designing the architecture, as the ROI obtained shown. In addition, modeling and managing the interdependencies at the different stages of the software development also enhances the integration of the system functionalities between them and with the environment, improving the system maintainability and making software development companies more competitive.

The methodology here detailed is being integrated in the software process of some companies. Firstly, Gloin has already integrated the proposed activities and tools into its software processes, being more competitive, as has been detailed above. Secondly, in order to facilitate its incorporation into the software processes of other companies, we are evaluating the possibility of incorporating it to the tool developed in the Geprodist project. This tool would take as input two sets of business processes (the first one is the software process to follow, including the proposed activities for the treatment of interdependencies, and the other one specifies the business processes to be supported by the new system to be developed) in order to automate the most repetitive tasks of the project management and some of the task of the software development (such as the derivation of the Use Cases or the Architectural Views models), greatly reducing the effort and cost of managing and developing software.

Finally, the outcomes of this thesis also impact and benefit customers and/or companies for which the software is developed, since the systems developed are better aligned with their business processes and goals, providing the maximum possible benefit and making them highly competitive.

5.2 Publications

All the contributions of this thesis have been published in prestigious scientific forums. All the papers written are detailed in the following sections: the first one details those that were accepted and the second one contains those that are pending of acceptance. This information complements to the one detailed in Table 1.1, which specifies a summary of the accepted papers.

5.2.1 Accepted Papers

Below the published papers, sorted chronologically, directly related to this thesis are detailed:

- **Berrocal, J.**, García-Alonso, J., Murillo, J.M. *Hacia una gestión del proceso software dirigida por Procesos de Negocio.* Primer Taller de Procesos de Negocio e Ingeniería de Servicios. Zaragoza, Spain, September 11-14, 2007. ISSN 1988-3455. pp 72-78

- **Berrocal, J.**, García-Alonso, J., Murillo, J.M. *A suite of tools for the automation of the management of the software process.* 2nd International Workshop on tool support and requirements management in distributed projects (REMIDI'08). Bangalore, India, August 17th, 2008. Pp 14-20

- **Berrocal, J.**, García-Alonso, J., Murillo, J.M. *Automating the software process management.* 13th Conference on Software Engineering and Databases. Gijon, Spain, 2008. Pp 15-26

- **Berrocal, J.**, García-Alonso, J., Murillo, J.M. *Patrones para la Extracción de Casos de Uso a partir de Procesos de Negocio.* Actas del II Taller de Procesos de Negocio e Ingeniería de Servicios, PNIS 2009, asociado a JISBD'2009. San Sebastián, Septiembre 2009. Pp: 1-11

- **Berrocal, J.**, García-Alonso, J., Murillo, J.M. *Usando técnicas BPM para agilizar la gestión de procesos software y mejorar la alineación con el negocio.* In Proc. PNIS 2010: Tercer Taller de Procesos de Negocio e Ingeniería de

Servicios. Valencia, Spain, September 7-10, 2010. ISSN 1988-3455. Pp 14-20

- **Berrocal, J.**, García-Alonso, J., Murillo, J.M. *Lean Management of Software Processes and Factories using Business Process Modeling Techniques.* In Proc. 11th International Conference on Product Focused Software. Limerick, Ireland, June 21-23, 2010. Lecture Notes in Business Information Processing 6156 Springer 2010. Pp. 321 - 335 **(CORE B)**

- **Berrocal, J.**, García-Alonso, J., Murillo, J.M. *Facilitating the selection of architectural patterns by means of a marked requirements model.* In Proc. 4th European Conference on Software Architecture. Copenhagen, Denmark, August 23-26, 2010. Lecture Notes in Computer Science 6285 Springer 2010. Pp. 384-391. **(CORE A)**

- **Berrocal, J.**, García-Alonso, J., Murillo, J.M. *Agilizando las herramientas de gestión de proyectos.* In Proc. 1ª Conferencia Agile-Spain CAS 2010. Madrid, Spain, Junio 10-11, 2010. I.S.B.N.: 84-96737-79-2. Pp 17-28

- **Berrocal, J.**, García-Alonso, J., Murillo, J.M. *Modeling Business and Requirements Relationships for Architectural Pattern Selection.* 2013 11th International Conference on Software Engineering Research, Management and Applications Studies in Computational Intelligence, Volume 496, 2014, pp 167-181, Prague, Czech Republic. **(CORE C)**

- **Berrocal, J.**, García-Alonso, J., Vicente-Chicote, C., Murillo, J.M. *A Pattern-Based and Model-Driven Approach for Deriving IT System Functional Models from Annotated Business Models*, 2013 22nd International Conference on Information Systems Development (ISD2013) Sevilla, Spain, September 2-4. **(CORE A)**

- Guillén, J., Miranda, J., **Berrocal, J.**García-Alonso, J., Murillo, J.M., Canal, C."People as a Service: a mobile-centric model for providing collective sociological profiles" IEEE Software, 21 Nov. 2013. IEEE computer Society Digital Library. IEEE Computer Society **(JCR, 1,616)**

- **Berrocal, J.**, García-Alonso, J., Vicente-Chicote, C., Murillo, J.M. *A Model-*

Driven Approach for Documenting Business and Requirements Interdependencies for Architectural Decision Making IEEE Latin Transactions, Vol 12 (2), pp 227-235, 2014 **(JCR, 0,218)**

- **Berrocal, J.**, García-Alonso, J., Murillo, J.M. *Modeling Business and Requirements Relationships to Facilitate the Identification of Architecturally Significant Requirements.* International Journal of Software Innovation, Issue 2(1), 2014 (to be published)

5.2.2 Pending Papers

Finally, this section contains the papers that are pending of acceptance.

- **Berrocal, J.**, García-Alonso, J., Vicente-Chicote, C., Murillo, J.M. *Modeling Requirements Relationships to Facilitate Architectural Decision Making*, 2014 23nd International Conference on Information Systems Development (ISD2014) Varazdin, Croatia, September 2-4. **(CORE A)**

5.3 Future Works

Throughout the development of this thesis, and especially during its application in real industrial projects, we have identified certain areas that could be further developed in order to make the contributions more robust and to provide a greater benefit to both the software companies and the customers. The following are the most important areas identified:

- From the outcomes of the different ATL transformations one can obtain the traceability between source elements and target objects, such as between the business processes and the Use Cases derivate from them. However, it is usually necessary to refine some of the derived Use Cases in the successive development stages, losing part of this traceability. A future work is to improve the traceability between the different elements modeled and refined along the process. Thus, for example, faced a change or improvement in the business processes, it would be even easier to identify what specific functionalities should

be modified for supporting it.

- The presented contributions allow requirements engineers to detail a first approximation of the system requirements on the business processes. This approximation, in addition to facilitate the early discussion of the systems functionalities, can also be used to obtain an estimation of the costs and effort required to develop the system. This research area is already being developed in collaboration with researchers of the Central University of Marta Abreu de las Villas, defining metrics to estimate the complexity of developing a system supporting a business process and, thus, to know in the earliest phase if it is economically viable.

- In this thesis, we have detailed several proposals to make explicit the interdependencies between business elements and the relationships among functional and non-functional requirements in order to facilitate the architectural decision making process. Nevertheless, these relationships can also be reused to improve other areas of the software development. Thus, as this methodology was integrated with the tool developed in the Geprodist project, we also identified that these interdependencies could also be used to manage the allocation of tasks among distributed development teams in order to minimize the communication between dispersed teams and better coordinate the integration of the functionalities assigned to them.

- Nowadays, many software applications are developed to be deployed in specific cloud providers, since they improve the managing of the execution costs and, depending on each provider, facilitates the achievement of certain requirements and quality attributes. Another area of work that has been identified as very promising is the definition of a methodology, that using the interdependencies between requirements, would be able to suggest how to modularize a system for being deployed in multiple clouds and to indicate what are the cloud providers that should be used in each module. Thus, the greatest benefit for the customer in relation with both the costs of executing the system and the compliance with the requirements can easily be provided.

- Finally, currently we are working in a new research area proposing a new paradigm for enabling mobile devices as cloud providers, in which they can both consume and provide information using the services deployed on them. Based on this paradigm, a new system may be composed, among other elements, of services deployed in different mobile devices. In this sense, we are starting to study how the interdependencies modeled during the analysis and design of a system (with the presented contributions) can be reused to easily identify what quality restrictions (in terms of battery consumption, data transfer rate, security, etc.) must comply each service to be part of the final system.

5.4 Final Reflection

The realization of this thesis began seven years ago. In this period of time, I have come to deeply understand such complex areas of the software development as business analysis, requirements analysis and software architecture design; in which I have identified different areas that could be improved and for which I gave my best to propose innovative solutions at the researcher level (as it is evidenced by the publications obtained) and at the business level (as it is shown by collaborations with companies). This has allowed me to develop my abilities as software engineer and to improve my research and innovation skills.

Nevertheless, during this time, this is not the only activity done. Another of the most important activities has been the foundation of the Gloin company. The work in this company, although it has led to a reduction in the time devoted to research, allowed me to be even closer to the day to day work in a software company and in its real problems. This has improved my skills in teamwork, in the management of development teams, in the identification of areas in which the company can get profits and in the definition of business strategies, and in the financial management of the company. So, it has been entirely beneficial not only for my entrepreneurial side, but also the acquired skills are also very useful for my research side.

Furthermore, as co-founder of Gloin, I participated during five months in the

LaunchPad Denmark[1] program. This is a program created by the Danish Government to accelerate start-ups from all over the world. This program consists of various seminars, teaching and providing skills on the search of innovative ideas, the implementation of these ideas, the search of funding, etc., and the assistance of a business consultant, advising on the development and deployment of innovative ideas. The acquired knowledge will be very useful in all the future activities that I will undertake, independently whether they are entrepreneurial, research or personal.

Moreover, I have participated in both the application and the implementation of regional, national and european research projects, and in business innovation projects with some of the most powerful companies nationwide (such as Indra). This allowed me to know and apply all processes from how an application of a project is created to how it is justified. This is a very important knowledge that I will use in future calls to get projects to make further progress on research and innovation, or to transfer knowledge to the industry.

Finally, in these years, I have taught over five subjects to college students in different faculties of the University of Extremadura, many postgraduate courses and training courses in technology to very important companies (such as Indra, Insa or the Government of Extremadura) that has allowed me to discover the beauty of teaching.

All these activities and the work done lead me to conclude that this period of time has not only been fruitful at the researcher level, but it also provided me and strengthened some skills that will be very useful in any activity faced in the future.

[1] http://launchpaddenmark.com

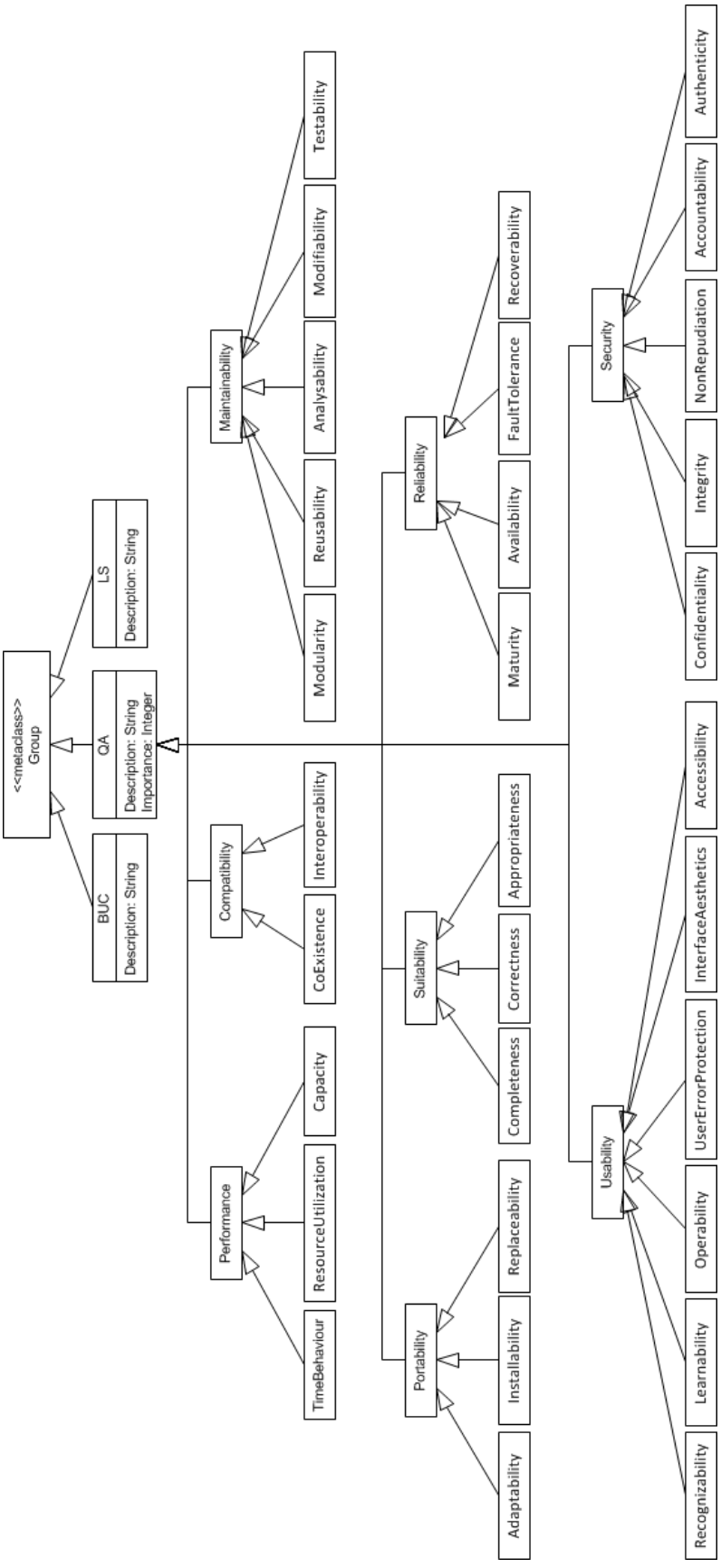
Appendix A

BPCI Extension

This appendix shows the BPCI Extension. This extension was developed for meeting the following goals:

- Allow engineers to model the interdependencies between business processes and quality attributes. This goal has been achieved adding new elements for modeling the relationships with quality attributes defined in ISO/IEC 25010 (International Standard Organization (ISO/IEC), 2011)
- Facilitate the modeling of the interactions of the new system with legacy systems. This goal has been fulfilled defining a new object to model the tasks already supported by legacy systems.
- Allow engineers to detail useful information for the following development phases. This goal has been achieved adding new elements to group the tasks that have to be supported by a functionality.

Below a diagram depicting the complete extension and a table describing the new specified objects are shown.


<<metaclass>>
Group
BUC
Description: String
QA
Description: String
Importance: Integer
LS
Description: String
Maintainability
Modularity
Reusability
Analysability
Modifiability
Testability
Compatibility
CoExistence
Interoperability
Performance
TimeBehaviour
ResourceUtilization
Capacity
Portability
Adaptability
Installability
Replaceability
Suitability
Completeness
Correctness
Appropriateness
Reliability
Maturity
Availability
FaultTolerance
Recoverability
Usability
Recognizability
Learnability
Operability
UserErrorProtection
InterfaceAesthetics
Accessibility
Security
Confidentiality
Integrity
NonRepudiation
Accountability
Authenticity

Element	**Description**
Group	The Group is the metaclass extended by the rest of objects defined in this extension. A Group provides a visual mechanism to group elements of a BPMN diagram. The grouping is tied to the CategoryValue supporting element. The graphical elements within the Group will be assigned to the CategoryValue of the Group. It has one attribute, *categoryValueRef*, which specifies the CategoryValue that the Group represents. A Group is represented by a rounded corner rectangle drawn with a dark solid dashed line.
BUC	BUC extends the Group object to define an element for grouping the business objects that are going to be supported by a functionality of a system. It has one attribute, *Description*, which specifies the goal of the functionality represented by the BUC. A BUC is represented by a rounded corner rectangle drawn with a solid dashed line and blue background.
LS	LS extends the Group object to define an element for grouping the business tasks that already are supported by a Legacy System, and with which a new application has to interact with. It has an attribute, *Description*, which details the Legacy Systems represented. A LS is represented by a rounded corner rectangle drawn with a solid dashed line and green background.

(The table continues in the following page).

Element	**Description**
QA	QA extends the Group object to define an element for grouping the business objects that are constrained by a quality attribute. Therefore, it models the interdependencies between quality attributes and business processes. It has two attributes: • *Description* which details the quality attribute. • *Importance* defining the relevance of the quality attribute. A QA is represented by a rounded corner rectangle with a red dashed line and transparent background.
Performance	This element represents a relationship with a QA detailing the performance relative to the amount of resources used under stated conditions. Performance extends the QA object, inheriting its attributes and representation.
TimeBehaviour	TimeBehaviour represents a relationship with a QA detailing the response and processing times of a system. It extends the QA object, inheriting its attributes and representation.
ResourceUtilization	This element represents a relationship with a QA detailing the amounts and types of resources that have to be used by a system. ResourceUtilization extends the QA object, inheriting its attributes and representation.
Capacity	Capacity represents a relationship with a QA detailing the maximum limits of a product or system. It extends the QA object, inheriting its attributes and representation.

(The table continues in the following page).

Element	**Description**
Compatibility	This element represents a relationship with a QA detailing the degree to which a system has to exchange information with other products. Compatibility extends the QA object, inheriting its attributes and representation.
CoExistence	CoExistence represents a relationship with a QA detailing the degree to which a product has to perform its required functions efficiently while sharing a common environment with other products. It extends the QA object, inheriting its attributes and representation.
Interoperability	This element represents a relationship with a QA detailing the degree to which two or more systems have to exchange information. Interoperability extends the QA object, inheriting its attributes and representation.
Maintainability	Maintainability represents a relationship with a QA detailing the efficiency with which a product has to be modified. It extends the QA object, inheriting its attributes and representation.
Modularity	This element represents a relationship with a QA detailing the degree to which a system has to be composed of discrete components. Modularity extends the QA object, inheriting its attributes and representation.

(The table continues in the following page).

Element	Description
Reusability	Reusability represents a relationship with a QA detailing the degree to which an asset has to be used in more than one system. It extends the QA object, inheriting its attributes and representation.
Analysability	This element represents a relationship with a QA detailing the efficiency with which the impact of an intended change has to be assessed. Analysability extends the QA object, inheriting its attributes and representation.
Modifiability	Modifiability represents a relationship with a QA detailing the degree to which a system has to be easy of modify without degrading the existing product quality. It extends the QA object, inheriting its attributes and representation.
Testability	This element represents a relationship with a QA detailing the efficiency with which test criteria have to be established for a system. Testability extends the QA object, inheriting its attributes and representation.
Portability	Portability represents a relationship with a QA detailing the facility with which a system has to be easy to transfer to another environment. It extends the QA object, inheriting its attributes and representation.

(The table continues in the following page).

Element	**Description**
Adaptability	This element represents a relationship with a QA detailing the efficiency with which a system has to be easy to adapt to different environment. Adaptability extends the QA object, inheriting its attributes and representation.
Instaliability	Instaliability represents a relationship with a QA detailing the facility with which a system has to be easy install and/or uninstall. It extends the QA object, inheriting its attributes and representation.
Replaceability	This element represents a relationship with a QA detailing the degree to which a product can replace another specified system for the same purpose. Replaceability extends the QA object, inheriting its attributes and representation.
Suitability	Suitability represents a relationship with a QA detailing the degree to which a system has to provide functions that meet implied needs when used under specified conditions. It extends the QA object, inheriting its attributes and representation.
Completeness	This element represents a relationship with a QA detailing the degree to which the set of functions has to cover all the specified tasks. Completeness extends the QA object, inheriting its attributes and representation.

(The table continues in the following page).

Element	Description
Correctness	Correctness represents a relationship with a QA detailing the degree to which a system has to provide the correct results. It extends the QA object, inheriting its attributes and representation.
Appropriateness	This element represents a relationship with a QA detailing the degree to which the functions facilitate the accomplishment of specified tasks. Appropriateness extends the QA object, inheriting its attributes and representation.
Reliability	Reliability represents a relationship with a QA detailing the degree to which a system has to be able of performing some functions for a specified period of time. It extends the QA object, inheriting its attributes and representation.
Maturity	This element represents a relationship with a QA detailing the degree to which a system has to meet needs for reliability under normal operation. Maturity extends the QA object, inheriting its attributes and representation.
Availability	Availability represents a relationship with a QA detailing the degree to which a system has to be operational and accessible. It extends the QA object, inheriting its attributes and representation.

(The table continues in the following page).

Element	**Description**
FaultTolerance	This element represents a relationship with a QA detailing the degree to which a system has to operate as intended despite the presence faults. FaultTolerance extends the QA object, inheriting its attributes and representation.
Recoverability	Recoverability represents a relationship with a QA detailing the degree to which, in the event of an interruption, a product has to re-establish the desired state of the system. It extends the QA object, inheriting its attributes and representation.
Usability	Usability represents a relationship with a QA detailing the degree to which a product or system has to be effectively used by specified users. It extends the QA object, inheriting its attributes and representation.
Recognizability	This element represents a relationship with a QA detailing the degree to which users have to recognize whether a system is appropriate for their needs. Recognizability extends the QA object, inheriting its attributes and representation.
Learnability	Learnability represents a relationship with a QA detailing the degree to which a system has to be easy to learn by specified users. It extends the QA object, inheriting its attributes and representation.

(The table continues in the following page).

Element	**Description**
Operability	This element represents a relationship with a QA detailing the degree to which a system has to be easy to operate and control. Operability extends the QA object, inheriting its attributes and representation.
UseErrorProtection	UseErrorProtection represents a relationship with a QA detailing the degree to which a system has to protect users against making errors. It extends the QA object, inheriting its attributes and representation.
InterfaceAesthetics	This element represents a relationship with a QA detailing the degree to which a user interface have to be satisfying for the user. InterfaceAesthetics extends the QA object, inheriting its attributes and representation.
Accessibility	Accessibility represents a relationship with a QA detailing the degree to which a system has to be easy of use by people with disabilities. It extends the QA object, inheriting its attributes and representation.
Security	This element represents a relationship with a QA detailing the degree to which a product has to protect information and data. Security extends the QA object, inheriting its attributes and representation.

(The table continues in the following page).

Element	**Description**
Confidentiality	Confidentiality represents a relationship with a QA detailing the degree to which a system has to ensure that data are accessible only by authorized users. It extends the QA object, inheriting its attributes and representation.
Integrity	This element represents a relationship with a QA detailing the degree to which a product has to prevent unauthorized access. Integrity extends the QA object, inheriting its attributes and representation.
NonRepudiation	NonRepudiation represents a relationship with a QA detailing the degree to which specific actions are not repudiated. It extends the QA object, inheriting its attributes and representation.
Accountability	This element represents a relationship with a QA detailing the degree to which the actions of an entity have to be traced. Accountability extends the QA object, inheriting its attributes and representation.
Authenticity	Authenticity represents a relationship with a QA detailing the degree to which the identity of a subject has to be proved. It extends the QA object, inheriting its attributes and representation.

(End of the Table)

Appendix B

UC-QuAID Profile

This appendix details the complete profile defined for modeling the relationships between Use Cases and non-functional requirements. This profile was designed to allow engineers to model any relationships with non-functional requirements according to the quality model detailed in ISO/IEC 25010 (International Standard Organization (ISO/IEC), 2011).

As the following figure details, the UML metaclass extended to model these relationships is *ExtensionPoint*. This object usually identifies a point in the behaviour of a Use Case where that behaviour is extended by the behaviour of some other (extending) Use Case. It only has one attribute, useCase, which is a reference to the Use Case owning the extension point. The Extension points are detailed in the Use Case diagram by a text string within the Use Case oval symbol specifying the name of the extension.

This metaclass is extended by the stereotype *QA*. This stereotype allows engineers to define points in the behaviour of a Use Case that are restricted or affected by a non-functional requirement. Therefore, it models the interdependencies between non-functional requirements and Use Cases.

This stereotype has two attributes:

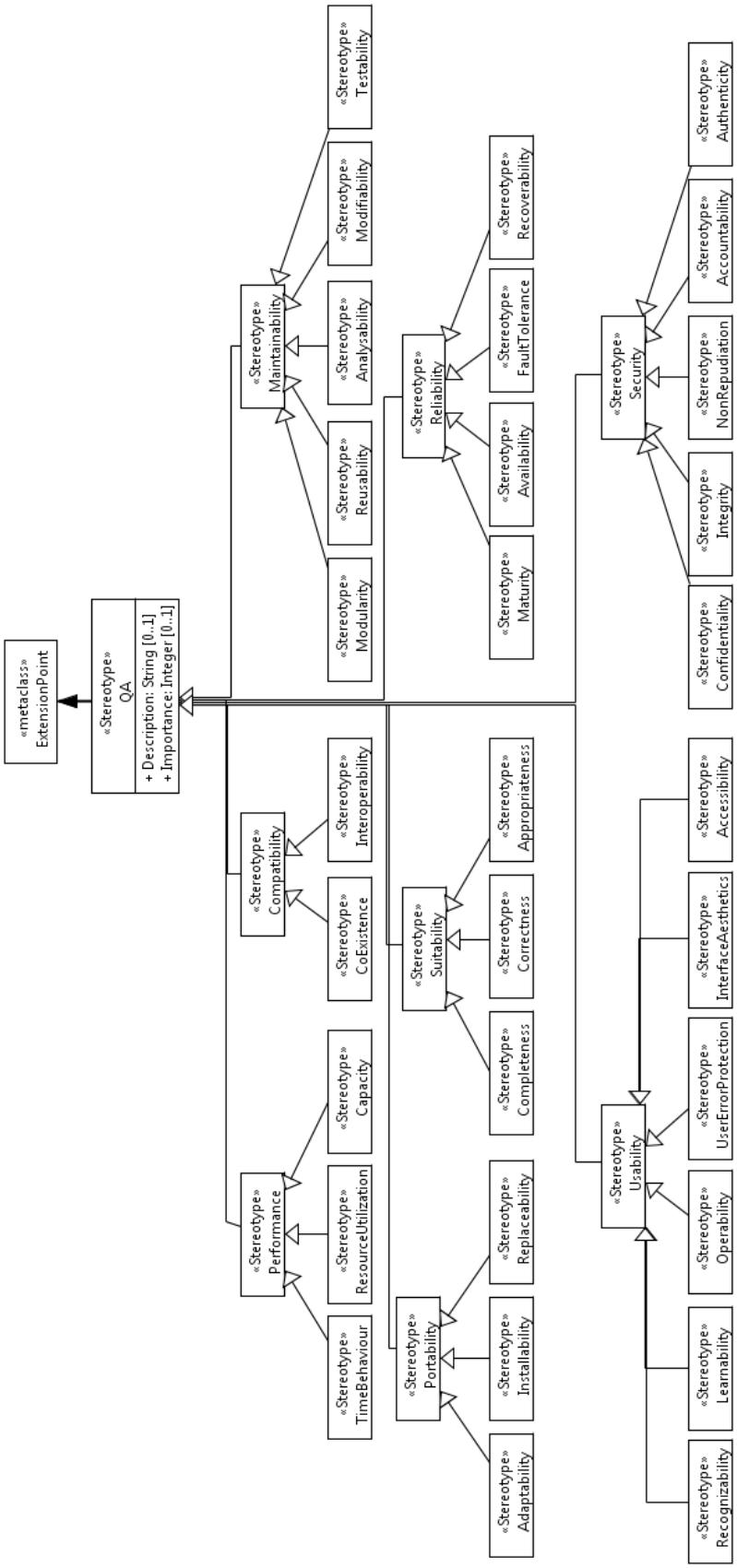
- *Description*, which details the non-functional requirement constraining a Use Case.

- *Importance*, which document the relevance of the non-functional requirement.

The QAs are modeled in the Use Cases, like the extension points, with a text string within the Use Case oval. In this case, the text always starts with the symbol <<QA>>, to indicate that the extension point details an interdependency with a quality attribute, followed by the name of the non-functional requirement affecting the Use Case (i.e. <<QA>>name).

The other stereotypes support the modeling of relationships with specific quality attributes defined in ISO/IEC 25010. The description of these elements is similar to the description of the elements defined for modeling these interdependencies in BPMN, Appendix A. In order to not duplicate this information, the description of each element is not repeated in this appendix.

These stereotypes inherit all the attributes of the object QA (i.e. Description and Importance), and they are annotated in a Use Case with the nomenclature <<*stereotype name*>> plus the name of the non-functional requirement constraining the Use Case. For example, an interdependency with a non-functional requirement restricting the processing times of a functionality would be annotated as <<*Time-Behaviour*>>*Max Response Time*.


«metaclass»
ExtensionPoint
«Stereotype»
QA
+ Description: String [0..1]
+ Importance: Integer [0..1]
«Stereotype»
Performance
«Stereotype»
TimeBehaviour
«Stereotype»
ResourceUtilization
«Stereotype»
Capacity
«Stereotype»
Compatibility
«Stereotype»
CoExistence
«Stereotype»
Interoperability
«Stereotype»
Maintainability
«Stereotype»
Modularity
«Stereotype»
Reusability
«Stereotype»
Analysability
«Stereotype»
Modifiability
«Stereotype»
Testability
«Stereotype»
Portability
«Stereotype»
Adaptability
«Stereotype»
Installability
«Stereotype»
Replaceability
«Stereotype»
Suitability
«Stereotype»
Completeness
«Stereotype»
Correctness
«Stereotype»
Appropriateness
«Stereotype»
Reliability
«Stereotype»
Maturity
«Stereotype»
Availability
«Stereotype»
FaultTolerance
«Stereotype»
Recoverability
«Stereotype»
Usability
«Stereotype»
Recognizability
«Stereotype»
Learnability
«Stereotype»
Operability
«Stereotype»
UserErrorProtection
«Stereotype»
InterfaceAesthetics
«Stereotype»
Accessibility
«Stereotype»
Security
«Stereotype»
Confidentiality
«Stereotype»
Integrity
«Stereotype»
NonRepudiation
«Stereotype»
Accountability
«Stereotype»
Authenticity

Appendix C

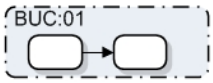
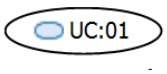
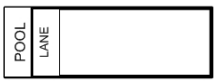
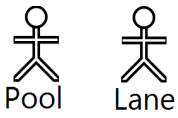
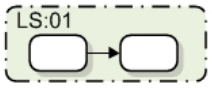
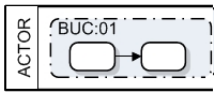
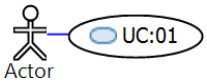
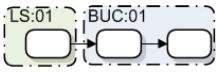
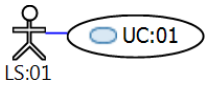
Patterns for Deriving Functional Requirements and Interdependencies

This appendix presents the defined patterns documenting the most common situations in the annotated business processes leading to the derivation of functional requirements, relationships between functional requirements, interdependencies between functional and non-functional requirements, and interdependencies with legacy systems.

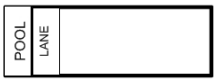
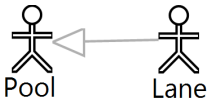
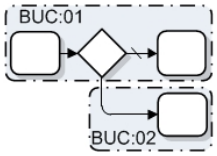
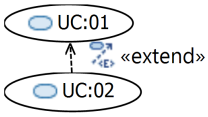
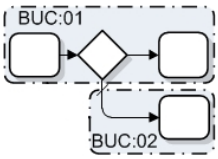
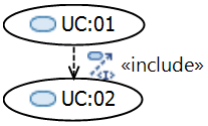
This information is detailed in a table in which each pattern is detailed in a row. For each pattern the following information is specified:

- *N.* indicates the number of the pattern.
- *Description* specifies pattern and, how and why it is applied.
- *Pattern* shows an excerpt of a business process leading to the application of the pattern.
- *Use Case Fragment* describes the result of the application of the patterns as a fragment of a Use Case diagram.

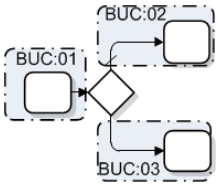
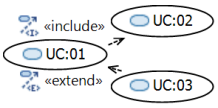
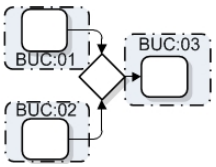
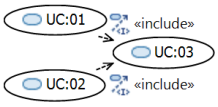
The following table complements the information presented in Section 3.4.3.

N.	Description	Pattern	UC Fragment
01	Each Business Use Case is derived to a Use Case, since both represent a set of tasks that an actor has to perform.		
02	The Pools and Lanes in the business processes are mapped to Actors in the Use Case diagram, since they represent the roles responsibles for executing the set of tasks or the Use Cases.		
03	The Legacy Systems annotated in the business processes are derived as Non-Human Actors in order to be able to represent interaction of the system with them.		
04	Each BUC must be related to the actor representing the Lane/Pool that contains it. In this way, the actor responsible for the Use Case is indicated.		
05	When a BUC is preceded or followed in the control flow by a LS, there is a relation between them. This relation indicates that the BUC must receive or provide information to the LS.		

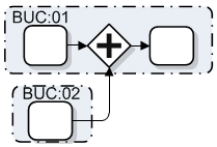
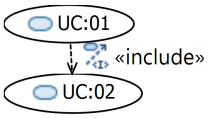
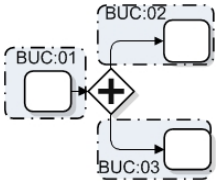
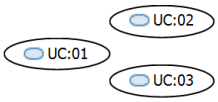
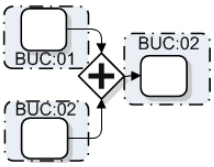
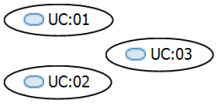
(The table continues in the following page).

N.	Description	Pattern	UC Fragment
06	If there are Lanes nested in other Lanes/Pools, then inheritance relationships between the father Lanes/Pools and the nested Lanes are derived to document hierarchical structure between the Actors derived from each Pool and Lane.		
07	If there is a BUC grouping some tasks, an exclusive or inclusive gateway, and the gateway's default flow, and there is another Business Use Case grouping the tasks defined in an alternative branch, then an *extend* relationship exists between the two Use Cases, since the BUC in the alternative branch extends the normal flow to follow when the gateway is reached.		
08	If there is a BUC grouping some tasks, an exclusive or inclusive gateway, and a gateway's alternative flow, and there is another BUC grouping the default branch, then an *include* relationship exists between the two Use Cases, since the BUC in the default branch represents the normal functionality to follow when the gateway is reached.		

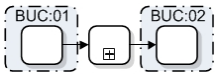
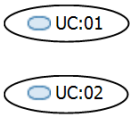
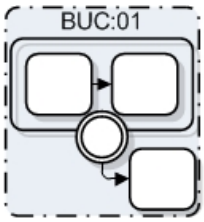
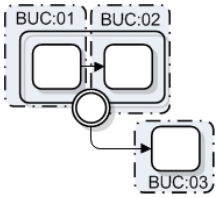
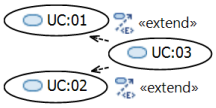
(The table continues in the following page).

N.	Description	Pattern	UC Fragment
09	If there are several BUCs connected by an exclusive or inclusive gateway, the BUC preceding the gateway is related to the BUC covering the default branch by an *include* relationship (since it represents the normal functionality to follow). Also, the BUC preceding the gateway is related to the BUC in the alternative flow by an *extend* (since it extends the normal flow).		
10	When two or more control flows are merged into one by an exclusive gateway, the BUC identified after the merge has to be related to the BUCs of each branch by means of an *include* relationship. Thus, it reflects that whenever a BUC of a branch is completed the BUC following the merge has to be executed.		

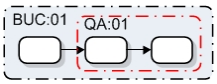
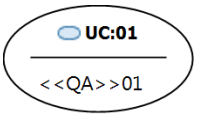
(The table continues in the following page).

N.	Description	Pattern	UC Fragment
11	When two or more control flows are merged into one by a parallel gateway and there is a BUC covering a branch and, at least, the parallel gateway, then there is an *include* relationship between the BUCs covering each branch and the BUC containing the gateway; since in order to continue the execution of the BUC containing the gateway, the other branches have to finalize.		
12	If several Business Use Cases are connected by a parallel gateway, then they have to be handled as isolated Business Use Cases; since the execution of each branch is independent of the rest.		
13	When two or more control flows are merged into one by a non-exclusive gateway, and there is a BUC covering each branch and the flow nodes following the gateway, then there is no relationship between the Business Use Cases; since the condition to start the BUC following the gateway is the completion of the BUCs in the branches.		

(The table continues in the following page).

N.	Description	Pattern	UC Fragment
14	Processing sub-processes. If in the control flow there are sub-processes, they will be treated independently. The business tasks detailed in the sub-processes will not be encapsulated in the Use Cases identified in the main process.		
15	Processing exceptions. If there is a sub-process in which the exceptions are handled, and all the sub-process's flows are covered by a single Business Use Case, then the exception is added to the Business Use Case as an exception scenario. The exception will thus be handled together with the Business Use Case.		
16	Processing exceptions. If there is a sub-process in which the exceptions are handled, and the sub-process's flows are covered by several Business Use Case, then a new BUC must be created to control the exception flow. This has to be related by *extend* relationships to the BUCs covering sub-process's flows. The exception is thus encapsulated in a BUC that extends to the others.		

(The table continues in the following page).

N.	Description	Pattern	UC Fragment
17	If some of the business tasks covered by a Business Use Cases are annotated with a quality attributes, then the Use Case derived from that BUC is also annotated with the same quality attribute since it has also to achieve that non-functional requirement.		

(End of the Table)

Appendix D

ATL Transformations for Deriving Annotated Requirements Models

This appendix shows an extract of the ATL transformations automatizing the generation of Use Case diagrams annotated with the interdependencies between Use Cases and non-functional requirements. These transformations support the most important patterns defined in Appendix C and complement the ones already defined in Section 3.4.4. Some of the patterns supported with these transformation are:

- The pattern number one, derivating Use Cases from BUCs.
- The pattern number two, mapping Pools and Lanes to Actors.
- The pattern number three, generating Non-Human Actors.
- The pattern number four and five, identifying relationships between Actors and Use Cases.
- The pattern number six, derivating *generalization* relationships between Actors.
- The pattern number seven, identifying *extend* relationships.

- The pattern number eleven, derivating *include* relationships.
- The pattern number fifteen, mapping subprocesses to Use Cases.
- The pattern number seventeen, identifying interdependencies between Use Cases and non-functional requirements.

```
1
2  -- @nsURI outMM=http://www.eclipse.org/uml2/4.0.0/UML
3  -- @nsURI inMM=http://www.omg.org/spec/BPMN/20100524/
       MODEL
4  -- @path profile=file:/D:/devPhD/eclipMT/workspace/
       BPMN2UC_UMLProfiles/UCProfile.profile.uml
5
6  module BPMN2UC;
7  create OUT : outMM from IN : inMM, IN1 : profile;
8
9  helper def : LS : String = 'LS';
10 helper def : BUC : String = 'BUC';
11
12 helper def : mRoot : outMM!Model = OclAny;
13
14 -- This helper gets the parent of the lane. The first
       element is a laneSet and the second one is the parent
15 helper context inMM!Lane def : getParent () : OclAny =
16   self.refImmediateComposite().refImmediateComposite();
17
18 --This helper check whether a CategoryValue reflect a
       Legacy System
19 helper context inMM!CategoryValue def : isLS() : Boolean
       =
20   self.refImmediateComposite().refGetValue('name')=
       thisModule.LS;
21
22 --This helper check whether a CategoryValue reflect a
       Business Use Case
23 helper context inMM!CategoryValue def : isBUC() : Boolean
       =
24   self.refImmediateComposite().refGetValue('name')=
       thisModule.BUC;
25
26 --This helper check whether a CategoryValue reflect a
       Quality Attribute
27 helper context inMM!CategoryValue def : isQA() : Boolean
       =
28   profile!Stereotype.allInstances()->any( e | e.name =
       self.getName())<>OclUndefined;
29
```

```
30 helper context inMM!CategoryValue def : getName() :
       String =
31   self.refImmediateComposite().refGetValue('name');
32
33 --This helper check if a sequence flow is annotated with
       two or mor BUC, then it connect two BUC
34 helper context inMM!SequenceFlow def : connectBUCs() :
       Boolean =
35   self.categoryValueRef->select(cat |cat.
       refImmediateComposite().refGetValue('name')=
       thisModule.BUC).size()>1;
36
37 -- This module returns the CategoryValue/BUC of node
38 helper context inMM!FlowNode def : getBUC() : inMM!
       CategoryValue =
39   self.categoryValueRef->any(cat | cat.
       refImmediateComposite().refGetValue('name')=
       thisModule.BUC);
40
41 -- This helper check whether an include relationships
       between two Use Cases has already been defined.
42 helper context inMM!SequenceFlow def : alreadyIncluded (
       owner : inMM!FlowNode, included : inMM!FlowNode) :
       Boolean =
43   outMM!Include.allInstances()->select (e | e.addition =
       thisModule.resolveTemp(included.getBUC(), 'uc'))->
       select(e | e.includingCase = thisModule.resolveTemp(
       owner.getBUC(), 'uc')).size()>0;
44
45 -- This helper check whether an extends relationship
       between two Use Cases has already been defined.
46 helper context inMM!SequenceFlow def : alreadyExtended (
       owner : inMM!FlowNode, extended : inMM!FlowNode) :
       Boolean =
47   outMM!Extend.allInstances()->select (e | e.extendedCase
       = thisModule.resolveTemp(extended.getBUC(), 'uc'))
       ->select(e | e.extension = thisModule.resolveTemp(
       owner.getBUC(), 'uc')).size()>0;
48
49 helper context inMM!SequenceFlow def : allExtends (owner
       : inMM!FlowNode, extended : inMM!FlowNode) : Sequence
       (outMM!Extend) =
50   outMM!Extend.allInstances()->select (e | e.extendedCase
       = thisModule.resolveTemp(extended.getBUC(), 'uc'))
       ->select(e | e.extension = thisModule.resolveTemp(
       owner.getBUC(), 'uc'));
51
```

```
52 helper context inMM!SequenceFlow def : getExtend (owner :
       inMM!FlowNode, extended : inMM!FlowNode) : outMM!
     Extend =
53   outMM!Extend.allInstances()->select (e | e.extendedCase
       = thisModule.resolveTemp(extended.getBUC(), 'uc'))
      ->select(e | e.extension = thisModule.resolveTemp(
       owner.getBUC(), 'uc')).first();
54
55 -- ROOT
56 rule root {
57   from
58     definition : inMM!Definitions
59   using {
60     processes :inMM!Process = inMM!Process.allInstances()
         ;
61     lanes : inMM!Lane = inMM!Lane.allInstances();
62     categories : Sequence(inMM!CategoryValue) = inMM!
         CategoryValue.allInstances();
63   }
64   to
65     model : outMM!Model (
66       name <- definition.name,
67       packagedElement <- Sequence {processes, lanes,
           categories}
68     )
69   do{
70     model.applyProfile(profile!Profile.allInstances().
         asSequence().first());
71     thisModule.mRoot <- model;
72   }
73 }
74
75 --FROM POOLS TO ACTORS
76
77 rule pool2Actor {
78   from
79     process : inMM!Process
80   to
81     actorP: outMM!Actor (
82       name <- process.name)
83 }
84 -- FROM LANES TO ACTORS
85 rule lane2Actor{
86   from
87     lane : inMM!Lane
88
89   to
```

```
90      actor: outMM!Actor (
91        name <- lane.name,
92        generalization <- generalization
93      ),
94      generalization : outMM!Generalization(
95        general <- lane.getParent()
96      )
97  }
98
99  -- FROM LEGACY SYSTEM TO ACTOR
100 rule legacy2Actor {
101   from
102     categoryValue : inMM!CategoryValue (categoryValue.
          isLS())
103
104   to
105     legacy : outMM!Actor (
106       name <- LS+':'+categoryValue.value
107
108     )
109
110 }
111 -- FROM BUC TO USE CASE
112 rule step2UC {
113   from
114     categoryValue : inMM!CategoryValue (categoryValue.
          isBUC())
115   using {
116     flowNodes : Sequence (inMM!FlowNode) =  categoryValue
          .categorizedFlowElements->select(e | e.oclIsKindOf
          (inMM!FlowNode));
117     nodesInLanes   : Sequence (inMM!FlowElement) =
          flowNodes->select(e | e.lanes.size()>0);
118     lanes    : Sequence (inMM!Lanes) =nodesInLanes->
          collect(lane | lane.lanes)->flatten()->asSet();
119   }
120   to
121     uc : outMM!UseCase (
122       name <- categoryValue.value
123     )
124
125   do{
126     -- relationships with actors
127     if (lanes.size() = 0) {
128       thisModule.getAssociation(uc,thisModule.resolveTemp
            (categoryValue.categorizedFlowElements.first().
            refImmediateComposite(), 'actorP'));
```

```
129       } else {
130         for (lane in lanes) {
131           thisModule.getAssociation(uc,thisModule.
                resolveTemp(lane, 'actor'));
132         }
133       }
134
135       --relationships with legacy systems
136       for (category in categoryValue.
            categorizedFlowElements->select(elements |
            elements.oclIsKindOf(inMM!SequenceFlow))->select(
            sequence | sequence.categoryValueRef.size()>1)->
            collect(category | category.categoryValueRef).
            flatten().asOrderedSet()){
137         if (category.isLS()){
138           thisModule.getAssociation(uc,thisModule.
                resolveTemp(category, 'legacy'));
139         }
140       }
141
142       -- relationships with quality attributes
143       for (category in categoryValue.
            categorizedFlowElements->select(elements |
            elements.oclIsKindOf(inMM!FlowNode))->select(
            sequence | sequence.categoryValueRef.size()>1)->
            collect(category | category.categoryValueRef).
            flatten().asOrderedSet()){
144         if (category.isQA()) {
145           thisModule.getExtensionPointsQA(uc, category);
146         }
147       }
148     }
149   }
150
151   -- Relationships with quality attributes
152   lazy rule getExtensionPointsQA {
153
154     from
155       uc : outMM!UseCase,
156       category : inMM!CategoryValue
157
158     to
159       extensionPoint : outMM!ExtensionPoint (
160         name <- category.value,
161         useCase <- uc
162       )
163
```

```
164     do{
165       extensionPoint.applyStereotype(profile!Stereotype.
              allInstances()->any( e | e.name = category.getName
              ()));
166     }
167   }
168
169   -- ACTOR - UC ASSOCIATIONS
170   lazy rule getAssociation {
171     from
172       uc : outMM!UseCase,
173       actor : outMM!Actor
174     to
175       association : outMM!Association (
176         name <- uc.name+'_'+actor.name,
177         memberEnd <- Sequence {src, dst},
178         ownedEnd <- Sequence {src, dst},
179         navigableOwnedEnd <- dst
180       ),
181
182       src : outMM!Property(
183         name <- 'src',
184         type <- actor,
185         association <- association,
186         upperValue <- natSrc,
187         lowerValue <- intSrc
188       ),
189       dst : outMM!Property(
190         name <- 'dst',
191         type <- uc,
192         association <- association,
193         upperValue <- natDst,
194         lowerValue <- intDst
195
196       ),
197       natSrc : outMM!LiteralUnlimitedNatural(value <- 1),
198       intSrc : outMM!LiteralInteger(value <- 1),
199       natDst : outMM!LiteralUnlimitedNatural(value <- 1),
200       intDst : outMM!LiteralInteger(value <- 1)
201     do{
202       thisModule.mRoot.packagedElement <- thisModule.mRoot.
              packagedElement.append(association);
203     }
204
205   }
206
207   -- EXTEND RELATIONSHIPS BETWEEN USE CASES
```

```
208 rule AlternativeExclusiveGateway {
209   from
210     -- select every SequenceFlow that has been annotated
          with the CategoryValue of two BUC,
211     -- the SequenceFlow's Source is an Exclusive Gateway
          and it isn't the the gateway's default branch
212     sequence : inMM!SequenceFlow ( sequence.connectBUCs()
            and
213       if sequence.sourceRef.oclIsTypeOf(inMM!
            ExclusiveGateway) then
214         if sequence.sourceRef.default<>OclUndefined then
215           sequence.sourceRef.default <> sequence
216         else false endif
217       else  false endif
218       )
219   to
220     extensionPoint : outMM!ExtensionPoint (
221       name <- sequence.sourceRef.name+'_'+sequence.name,
            --the ExtensionPoint name is the name of the
            Exclusive Gateway + the SequenceFlow name
222       useCase <- thisModule.resolveTemp(sequence.
            sourceRef.getBUC(), 'uc')
223     )
224
225   do {
226
227     if ( not sequence.alreadyExtended(sequence.targetRef,
          sequence.sourceRef)) {
228       thisModule.createExtendsRelationship(sequence);
229     }else {
230       sequence.getExtend(sequence.targetRef, sequence.
            sourceRef).extensionLocation <- sequence.
            getExtend(sequence.targetRef, sequence.sourceRef
            ).extensionLocation.append(extensionPoint);
231     }
232   }
233 }
234
235 -- An extend relationship between two Use Cases is only
     created if there isn't another one defined
236 lazy rule createExtendsRelationship {
237   from
238     sequence : inMM!SequenceFlow
239   to
240     extendsRel : outMM!Extend (
241       extendedCase <-  thisModule.resolveTemp(sequence.
            sourceRef.getBUC(), 'uc'),
```

```
242         extension <- thisModule.resolveTemp(sequence.
                targetRef.getBUC(), 'uc'),
243         extensionLocation <- sequence
244       )
245   }
246
247
248   -- INCLUSIVE RELATIONSHIPS BETWEEN USE CASES
249   rule MergerParallelBranch {
250     from
251       -- select every SequenceFlow that has been annotated
              with the CategoryValue of two BUC,
252       -- the SequenceFlow's target is a Parallel Gateway,
              it merges the branchs
253       -- and if there isn't another include relationship
              between both Use Cases
254       sequence : inMM!SequenceFlow (sequence.connectBUCs()
            and
255         if sequence.targetRef.oclIsTypeOf(inMM!
              ParallelGateway) then
256           sequence.targetRef.gatewayDirection.toString() =
                'Converging'
257         else false endif
258         )
259
260     do{
261
262       if ( not sequence.alreadyIncluded(sequence.targetRef,
            sequence.sourceRef)) {
263         thisModule.createIncludeRelationship(sequence);
264       }
265     }
266   }
267
268   -- An include relationship between two Use Cases is only
        created if there isn't another defined
269   lazy rule createIncludeRelationship {
270     from
271       sequence : inMM!SequenceFlow
272     to
273       includeRel : outMM!Include (
274         --The BUC annotated in the sequence source element
              is the include Use Case
275         addition <- thisModule.resolveTemp(sequence.
              sourceRef.getBUC(), 'uc'),
276         --The BUC annotated in the sequence target element
              is the including/owner Use Case
```

```
277             includingCase <- thisModule.resolveTemp(sequence.
                    targetRef.getBUC(), 'uc')
278         )
279     }
```

Appendix E

ATL Transformations for Deriving Architectural Views

This appendix shows the ATL transformations defined for automatically generate the model documenting the impact of the quality attributes on the attributes of the *Use Case View* according to the metamodel defined in Section 3.5.1. Specifically, the following are the documented attributes for this view:

- The number of actors.
- The number of Use Cases and how many of them are constrained by each quality attribute.
- The number of Use Cases containing *extend* relationships and how many of them are restricted by each quality attribute.
- The number of Use Cases containing *include* relationships and how many of them are constrained by each quality attribute.
- The number of Use Cases containing *generalization* relationships and how many of them are affected by each quality attribute.

```
1  -- @path QAV=/UC2Arch2/metamodels/ArchQA.ecore
2  -- @path UCP=/UC2Arch2/metamodels/UCProfile.profile.uml
3  -- @nsURI UML=http://www.eclipse.org/uml2/4.0.0/UML
4
5  module UC2Arch;
6  create outQAV : QAV from inUML : UML, inUCP : UCP;
7
8  helper def : stereotypes : Sequence (OclAny)= Sequence{};
9  helper def : attributes : Sequence (OclAny) = Sequence{};
10
11 --It returns the number of Use Cases inherited by one Use
       Case
12 helper context UML!UseCase def : getNumParents () :
     Integer =
13   if self.generalization.size()>0 then
14     self.generalization->iterate(gRel; result : Integer =
           self.generalization.size() |result + gRel.general
         .getNumParents())
15   else
16     0
17   endif
18 ;
19
20 -- It indicates if a Use Case is annotated by a
     stereotype
21 helper context UML!UseCase def : isAnnotated(s : UCP!
     Stereotype) : Boolean =
22   if self.extensionPoint->select(exten | not exten.
       getAppliedStereotype(s.qualifiedName).oclIsUndefined
       ()).size()>0 then
23     true
24   else
25     false
26   endif
27 ;
28
29 -- It indicates if the parents of a Use Case are
     annotated by a stereotype
30 helper context UML!UseCase def : parentsAnnotated (s :
     UCP!Stereotype) :Boolean =
31   if self.generalization.size()>0 then
32     self.generalization->iterate(gRel; result : Boolean =
           false | gRel.general.isAnnotated(s))
33   else
34       false
35   endif
36 ;
```

```
37
38 -- It indicates if the parents, or parents of the parents
       , of a Use Case are annotated by a stereotype
39 helper context UML!UseCase def : hierarchyAnnotated (s :
       UCP!Stereotype) :Boolean =
40   if self.parentsAnnotated(s) then
41     true
42   else
43     self.generalization->iterate(gRel; result : Boolean =
           false | gRel.general.hierarchyAnnotated(s))
44   endif
45 ;
46
47 rule model {
48   from
49     model : UML!Model
50   to
51     sys : QAV!System (
52       name <- model.name
53     ),
54
55     ucv : QAV!View(
56       system <- sys,
57       viewKind <- #UseCaseView
58
59     ),
60
61     ucases : QAV!Attribute(
62       name <- 'NumUseCases',
63       view <- ucv
64     ),
65
66     actors : QAV!Attribute(
67       name <- 'NumActors',
68       view <- ucv
69     ),
70
71     include : QAV!Attribute(
72       name <- 'NumInclude',
73       view <- ucv
74     ),
75
76     extend : QAV!Attribute(
77       name <- 'NumExtend',
78       view <- ucv
79     ),
80
```

```
81      generalization : QAV!Attribute(
82        name <- 'NumGeneralization',
83        view <- ucv
84      ),
85
86      vUCS : QAV!AttributeValue(
87        attribute <- ucases,
88        value <- model.packagedElement->select(usecases |
              usecases.oclIsKindOf(UML!UseCase)).size()
89      ),
90
91      vActors : QAV!AttributeValue(
92        attribute <- actors,
93        value <- model.packagedElement->select(actors |
              actors.oclIsKindOf(UML!Actor)).size()
94      ),
95
96      vInclude : QAV!AttributeValue(
97        attribute <- include,
98        value <- model.packagedElement->select(actors |
              actors.oclIsKindOf(UML!UseCase))->collect(uc |
              uc.include).flatten().size()
99      ),
100
101     vExtends : QAV!AttributeValue(
102       attribute <- extend,
103       value <- model.packagedElement->select(actors |
              actors.oclIsKindOf(UML!UseCase))->collect(uc |
              uc.extend).flatten().size()
104     ),
105
106     vGeneralization : QAV!AttributeValue(
107       attribute <- generalization,
108       value <- model.packagedElement->select(actors |
              actors.oclIsKindOf(UML!UseCase))->collect(uc |
              uc.generalization).flatten().size()
109     )
110
111   -- The quality attributes impacting on each view's
        attribute are identified
112   do{
113     thisModule.attributes<-Sequence {ucases,actors,
          include, extend, generalization};
114     for (stereotype in UCP!Stereotype.allInstances()){
115       if (model.packagedElement->select(elements |
              elements.oclIsKindOf(UML!UseCase))->collect(ext
              | ext.extensionPoint).flatten()->select(exten |
```

```
            not exten.getAppliedStereotype(stereotype.
            qualifiedName).oclIsUndefined()).size()>0){
116           thisModule.stereotypes<-thisModule.stereotypes.
                append(stereotype);
117         }
118       }
119
120       for(stereotype in thisModule.stereotypes){
121         thisModule.createQAView(ucv,stereotype, model);
122       }
123
124     }
125   }
126
127   -- It identifies the impact of each quality attribute in
        the defined view's attributes
128   lazy rule createQAView {
129
130     from
131       v : QAV!View,
132       s : UCP!Stereotype,
133       model : UML!Model
134
135     to
136       qa : QAV!QA (
137         view <-v,
138         name <-s.name
139       ),
140
141       ucQAV : QAV!QAValue(
142         attribute <-thisModule.attributes.at(1),
143         qa <- qa,
144         value <-model.packagedElement->select(elements |
              elements.oclIsKindOf(UML!UseCase))->collect(ext
              | ext.extensionPoint).flatten()->select(exten |
              not exten.getAppliedStereotype(s.qualifiedName).
              oclIsUndefined()).size()
145       ),
146
147       includeQAV : QAV!QAValue(
148         attribute <-thisModule.attributes.at(3),
149         qa <- qa,
150         value <-model.packagedElement->select(elements |
              elements.oclIsKindOf(UML!UseCase))->select(uc |
              uc.include.size()>0)->collect(uc | uc.
              extensionPoint).flatten()->select(ePoint | not
              ePoint.getAppliedStereotype(s.qualifiedName).
```

```
            oclIsUndefined()).size()
151     ),
152
153     extendQAV : QAV!QAValue(
154       attribute <-thisModule.attributes.at(4),
155       qa <- qa,
156       value <-model.packagedElement->select(elements |
              elements.oclIsKindOf(UML!UseCase))->select(uc |
              uc.extend.size()>0)->collect(uc | uc.
              extensionPoint).flatten()->select(ePoint | not
              ePoint.getAppliedStereotype(s.qualifiedName).
              oclIsUndefined()).size()
157     ),
158
159     generalizationQAV : QAV!QAValue(
160       attribute <-thisModule.attributes.at(5),
161       qa <- qa
162     )
163
164   -- The number of Use Case inheriting from Use Cases
        constrained by a quality attribute are identified
165   do{
166     v.qas<-v.qas.append(qa);
167
168     generalizationQAV.value<-0;
169     for (uc in model.packagedElement->select(elements |
          elements.oclIsKindOf(UML!UseCase))->select(ucs |
          ucs.generalization.size()>0)){
170       if (uc.hierarchyAnnotated(s)){
171         generalizationQAV.value<-generalizationQAV.value
              +1;
172       }
173     }
174   }
175 }
```

References

Aburub, F., Odeh, M., & Beeson, I. (2007). Modelling non-functional requirements of business processes. *Inf. Softw. Technol.*, *49*, 1162–1171. doi: 10.1016/j.infsof.2006.12.002

Alexander, I. F., & Stevens, R. (2002). *Writing better requirements.* Pearson Education.

Amyot, D. (2003). Introduction to the user requirements notation: Learning by example. *Computer Networks*, *42*(3), 285–301. doi: 10.1016/S1389-1286(03)00244-5

Amyot, D., & Mussbacher, G. (2003). Urn: Towards a new standard for the visual description of requirements. In *Proceedings of the 3rd international conference on telecommunications and beyond: The broader applicability of sdl and msc* (pp. 21–37). Berlin, Heidelberg: Springer-Verlag.

Avgeriou, P., Grundy, J., Hall, J. G., Lago, P., & Mistrík, I. (Eds.). (2011). *Relating software requirements and architectures.* Springer.

Bachmann, F., Bass, L., & Klein, M. (2003). Moving from quality attribute requirements to architectural decisions. In *Straw* (p. 122-129).

Bachmann, F., Bass, L., Klein, M., & Shelton, C. (2005, aug.). Designing software architectures to achieve quality attribute requirements. *Software, IEE Proceedings -*, *152*(4), 153 - 165. doi: 10.1049/ip-sen:20045037

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). The goal question metric approach. In *Encyclopedia of software engineering.* Wiley.

Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice, second edition.* Addison-Wesley Professional.

Berrocal, J., Garcia Alonso, J., & Murillo, J. (2014). Modeling business and requirements relationships to facilitate the identification of architecturally significant requirements. *International Journal of Software Innovation (IJSI)*, *2*(1).

Berrocal, J., García-Alonso, J., & Murillo, J. M. (2007). Hacia una gestión del proceso software dirigida por procesos de negocio. In *I taller de procesos de negocio e ingeniería de servicios* (p. 72-78).

Berrocal, J., García-Alonso, J., & Murillo, J. M. (2008). Automating the software process management. In *Jornadas de ingeniería del software y bases de datos (jisbd)* (pp. 15–26).

Berrocal, J., García-Alonso, J., & Murillo, J. M. (2009). Patrones para la extracción de casos de uso a partir de procesos de negocio. In *Ii taller de procesos de negocio e ingeniería de servicios* (p. 1-11).

Berrocal, J., García-Alonso, J., & Murillo, J. M. (2010a). Facilitating the selection of architectural patterns by means of a marked requirements model. In M. A. Babar & I. Gorton (Eds.), *Ecsa* (Vol. 6285, p. 384-391). Springer.

Berrocal, J., García-Alonso, J., & Murillo, J. M. (2010b). Lean management of software processes and factories using business process modeling techniques. In M. A. Babar, M. Vierimaa, & M. Oivo (Eds.), *Profes* (Vol. 6156, p. 321-335). Springer.

Berrocal, J., García-Alonso, J., & Murillo, J. M. (2010c). Usando técnicas bpm para agilizar la gestión de procesos software y mejorar la alineación con el negocio. In *Iii taller de procesos de negocio e ingeniería de servicios* (p. 14-20).

Berrocal, J., García-Alonso, J., & Murillo, J. M. (2013). Modeling business and requirements relationships for architectural pattern selection. In J. Kacprzyk (Ed.), *Software engineering research,management and applications 2013.* Springer.

Berrocal, J., Garcia Alonso, J., Vicente Chicote, C., & Murillo, J. (2014, March). A model-driven approach for documenting business and requirements interdependencies for architectural decision making. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, *12*(2), 227-235. doi: 10.1109/TLA.2014.6749542

Berrocal, J., García-Alonso, J., Vicente-Chicote, C., & Murillo, J. M. (2013). A

pattern-based and model-driven approach for deriving it system functional models from annotated business models. In *22nd international conference on information systems development (isd2013)*. Springer.

BMM. (2013). *Business Motivation Model.* Retrieved from http://www.omg.org/spec/BMM/

Bocanegra, J., Pena, J., & Ruiz, A. (2011). Interorganizational business modeling: an approach for traceability of goals, organizational models and business processes. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, *9*(1), 847-854. doi: 10.1109/TLA.2011.5876430

BPMN. (2014). *Business Process Modeling Notation 2.0.* Retrieved from http://www.bpmn.org/

Bubenko, J. A., Brash, D., & Stirna, J. (1998). *EKD User Guide* (Tech. Rep.). Stockholm, Sweden: Kista, Dept. of Computer and Systems Science, Royal Institute of Technology (KTH) and Stockholm University.

Capilla, R., Ali Babar, M., & Pastor, O. (2012). Quality requirements engineering for systems and software architecting: methods, approaches, and tools. *Requirements Engineering*, *17*(4), 255-258.

Cardoso, E., Almeida, J., & Guizzardi, G. (2009). Requirements engineering based on business process models: A case study. In *Enterprise distributed object computing conference workshops, 2009. edocw 2009. 13th* (p. 320-327).

Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., & Natt och Dag, J. (2001). An industrial survey of requirements interdependencies in software product release planning. In *Requirements engineering, 2001. proceedings. fifth ieee international symposium on* (p. 84-91). doi: 10.1109/ISRE.2001.948547

Castro, J., Kolp, M., & Mylopoulos, J. (2002). Towards requirements-driven information systems engineering: The tropos project. *Information System Journal*, *27*(6), 365–389. doi: 10.1016/S0306-4379(02)00012-1

Chan, Y. E., Huff, S. L., Barclay, D. W., & Copeland, D. (1997). Business strategic orientation, information systems strategic orientation, and strategic alignment. *Information Systems Research*, *8*(2), 125-150.

Chung, L., & do Prado Leite, J. C. S. (2009). On non-functional requirements in software engineering. In A. Borgida, V. K. Chaudhri, P. Giorgini, & E. S. K. Yu

(Eds.), *Conceptual modeling: Foundations and applications* (Vol. 5600, p. 363-379). Springer.

Chung, L., Nixon, B. A., Yu, E., & Mylopoulos, J. (2000). *Non-functional requirements in software engineering (the kluwer international series in software engineering volume 5)* (1st ed.). Springer.

Clemmons, R. K. (2006). Project Estimation With Use Case Points. *Diversified Technical Services, Inc. CrossTalk - The journal of Defence Software Engineering*, 18-22.

Cockburn, A. (2000). *Writing effective use cases* (1st ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Cragg, P., King, M., & Hussin, H. (2002). {IT} alignment and firm performance in small manufacturing firms. *The Journal of Strategic Information Systems*, *11*(2), 109 - 132. doi: http://dx.doi.org/10.1016/S0963-8687(02)00007-0

Curtis, B., Krasner, H., & Iscoe, N. (1988, November). A field study of the software design process for large systems. *Commun. ACM*, *31*(11), 1268–1287. doi: 10.1145/50087.50089

Cysneiros, L. M., Sampaio, J. C., & de Melo, J. (2001). A framework for integrating non-functional requirements into conceptual models. *Requirements Engineering*, *6*(2), 97-115. doi: 10.1007/s007660170008

Czarnecki, K., & Eisenecker, U. (2000). *Generative programming: methods, tools, and applications*. Addison Wesley.

Dahlstedt, A. G., & Persson, A. (2003). Requirements interdependencies - moulding the state of research into a research agenda. In *Ninth international workshop on requirements engineering: Foundation for software quality (refsq 2003)* (pp. 71–80).

Davenport, T. (1993). *Process innovation: Reengineering work through information technology*. Harvard Business School Press. Retrieved from http://books.google.dk/books?id=kLlIOMGaKnsC

De la Vara, J., & Sánchez, J. (2009). Bpmn-based specification of task descriptions: Approach and lessons learnt. In M. Glinz & P. Heymans (Eds.), *Requirements engineering: Foundation for software quality* (Vol. 5512, p. 124-138). Berlin, Heidelberg: Springer.

de Leede, J., Looise, J. C., & Alders, B. C. M. (2002). Innovation, improvement and operations: an exploration of the management of alignment. *International Journal of Technology Management*, *23*(4), 353-368.

Dijkman, R. M., & Joosten, S. M. M. (2002). *Deriving use case diagrams from business process models* (Tech. Rep. No. TR-CTIT-02-08). Eindhoven, The Netherlands: University of Twente.

Dörr, J. (2011). *Elicitation of a complete set of non-functional requirements.* (Unpublished doctoral dissertation). University of Kaiserslautern.

Dumas, M., van der Aalst, W. M., & ter Hofstede, A. H. (Eds.). (2005). *Process-Aware Information Systems : Bridging People and Software through Process Technology.* Wiley-Interscience.

Eclipse BPMN2 Modeler. (2014). Retrieved from http://eclipse.org/bpmn2-modeler/

Eclipse Papyrus. (2014). Retrieved from http://www.eclipse.org/papyrus/

Fazal-Baqaie, M. (2009). *Ensuring quality in business-driven development of it systems using workflow patterns* (Tech. Rep.). Paderborn 33100, Germany: University of Paderborn.

Fernandez, E. B. (2003). Layers and non-functional patterns. In *In: Proc. of chiliplop 2003.*

Gao, S. (2011). *High level modeling and evaluation of multi-channel services* (Tech. Rep.). Trondheim, Norway: Norwegian University of Science and Technology.

Gao, S., & Krogstie, J. (2009). A combined framework for development of business process support systems. In A. Persson et al. (Eds.), *The practice of enterprise modeling* (Vol. 39, p. 115-129). Springer.

Garcia-Alonso, J., Berrocal, J., & Murillo, J. M. (2013). Architectural decisions in the development of multi-layer applications. In *The eighth international conference on software engineering advances* (pp. 214 – 219). ACM.

Grau, G., Franch, X., & Maiden, N. A. M. (2008). Prim: An i*-based process reengineering method for information systems specification. *Inf. Softw. Technol.*, *50*(1-2), 76–100.

Guillen, J., Miranda, J., Berrocal, J., Garcia-Alonso, J., Murillo, J. M., & Canal, C. (2014). People as a service: A mobile-centric model for providing collective

sociological profiles. *IEEE Software*, *31*(2), 48-53.

Harrison, N. B., & Avgeriou, P. (2007). Leveraging architecture patterns to satisfy quality attributes. In F. Oquendo (Ed.), *European conference on software architecture* (Vol. 4758, p. 263-270). Springer.

Harrison, N. B., & Avgeriou, P. (2010). How do architecture patterns and tactics interact? a model and annotation. *Journal of Systems and Software*, *83*(10), 1735-1758.

Henderson, J., & Venkatraman, N. (1992). *Strategic alignment: a model for organizational transformation via information technology.* Oxford University Press, USA.

Herrmann, A., Paech, B., & Plaza, D. (2006). Icrad: an integrated process for the solution of requirements conflicts and architectural design. *International Journal of Software Engineering and Knowledge Engineering*, *16*(6), 917-950.

Indulska, M., Recker, J., Rosemann, M., & Green, P. F. (2009). Business process modeling: Current issues and future challenges. In P. van Eck, J. Gordijn, & R. Wieringa (Eds.), *Caise* (Vol. 5565, p. 501-514). Springer.

International Standard Organization (ISO/IEC). (2001). *Informational technology – product quality: Quality model. International Standard ISO/IEC 9126.*

International Standard Organization (ISO/IEC). (2011). *ISO/IEC 25010:2011 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models* (Tech. Rep.). Geneva, Switzerland: ISO/IEC.

ITU-T. (2003). *Recommendation z.150, user requirements notation (urn) – language requirements and framework.*

Jackson, M. (1995). *Software requirements & specifications: a lexicon of practice, principles and prejudices.* New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.

Kearns, G. S., & Lederer, A. L. (2000).

The Journal of Strategic Information Systems, *9*(4), 265-293.

Kim, S., Kim, D.-K., Lu, L., & Park, S. (2009, August). Quality-driven architecture development using architectural tactics. *J. Syst. Softw.*, *82*, 1211–1231. doi: 10.1016/j.jss.2009.03.102

Koehler, J., Hauser, R., Küster, J., Ryndina, K., Vanhatalo, J., & Wahler, M. (2008). The role of visual modeling and model transformations in business-driven development. *Electronic Notes in Theoretical Computer Science*, *211*, 5 - 15.

Kotonya, G., & Sommerville, I. (1998). *Requirements engineering: processes and techniques*. J. Wiley.

Kruchten, P. (1995). The 4+1 view model of architecture. *IEEE Softw.*, *12*(6), 42–50. Retrieved from http://dx.doi.org/10.1109/52.469759 doi: 10.1109/52.469759

Lauesen, S. (2002). *Software requirements: Styles and techniques*. Addison-Wesley.

Lauesen, S. (2003, March). Task descriptions as functional requirements. *IEEE Softw.*, *20*(2), 58–65. doi: 10.1109/MS.2003.1184169

Loucopoulos, P., & Kavakli, E. (1995). Enterprise modelling and the teleological approach to requirements engineering. *International Journal of Intelligent and Cooperative Information Systems*, *4*, 45–79.

Marinelli, V., & Laplante, P. A. (2008). *Requirements engineering: the state of the practice revisited* (Tech. Rep.). Pennsylvania, USA: Penn State University.

Mcgilvray, D. (2008). *Executing data quality projects: Ten steps to quality data and trusted information (tm)*. Burlington, MA: Elsevier.

Mitra, T. (2005). *Business-driven development*. IBM WebSphere Developer Works.

OMG. (2014). *Object management group*. Retrieved from http://www.omg.org/

OpenUP. (2013). *Open unified process*. Retrieved from http://http://epf.eclipse.org/wikis/openup/

Ould, M. (2005). *Business process management: A rigorous approach*. BCS Learning & Development Limited.

Papazoglou, M. P., & Kratz, B. (2006). A business-aware web services transaction model. In (Vol. 4294, p. 352-364). Springer.

Pavlovski, C. J., & Zou, J. (2008). Non-functional requirements in business process modeling. In *Proceedings of the fifth asia-pacific conference on conceptual modelling - volume 79* (pp. 103–112). Darlinghurst, Australia, Australia: Australian Computer Society, Inc.

Pohl, K. (2010). *Requirements engineering: Fundamentals, principles, and tech-*

niques (1st ed.). Springer Publishing Company, Incorporated.

Regnell, B., Paech, B., Aurum, A., Wohlin, C., Dutoit, A., & Dag, J. N. O. (2001). Requirements mean decisions! - research issues for understanding and supporting decision-making in requirements engineering. In *In requirements engineering, proc. 1st swedish conference on software engineering research and practice (serp 01* (pp. 49–52).

Rodríguez, A., de Guzmán, I. G. R., Fernández-Medina, E., & Piattini, M. (2010). Semi-formal transformation of secure business processes into analysis class and use case models: An mda approach. *Inf. & Soft. Tech.*, *52*(9), 945-971.

Roy, J. (2007). *Requirement engineering with urn: Integrating goals and scenarios.* University of Ottawa (Canada).

Shakil Khan, S., Greenwood, P., Garcia, A., & Rashid, A. (2008). On the impact of evolving requirements-architecture dependencies: An exploratory study. In Z. Bellahsène & M. Léonard (Eds.), *Advanced information systems engineering* (Vol. 5074, p. 243-257). Springer Berlin Heidelberg. doi: 10.1007/978-3-540-69534-919

Shaw, M. (2002). What makes good research in software engineering. *for Technology Transfer (STTT). Springer Berlin / Heidelberg*, *4*, 1-7.

Singh, S., & Woo, C. (2009). Investigating business-it alignment through multi-disciplinary goal concepts. *Requirements Engineering*, *14*, 177-207.

Siqueira, F., & Silva, P. (2011). Transforming an enterprise model into a use case model using existing heuristics. In *Model-driven requirements engineering workshop* (p. 21 -30).

Spring Security. (2014). Retrieved from http://docs.spring.io/spring-security/site/index.html

Stahl, T., Voelter, M., & Czarnecki, K. (2006). *Model-driven software development: Technology, engineering, management.* John Wiley & Sons.

Stolfa, S., & Vondrak, I. (2006). *Mapping from business processes to requirements specification* (Tech. Rep.). Trier, Rhineland-Palatinate, Germany: CUniversitat Trier.

Traetteberg, H., & Krogstie, J. (2009). Enhancing the usability of bpm-solutions by combining process and user-interface modelling. In J. Stirna et al. (Eds.), *The*

practice of enterprise modeling (Vol. 15, p. 86-97). Springer Berlin Heidelberg.

Ullah, A., & Lai, R. (2011). Modeling business goal for business/it alignment using requirements engineering. *Journal of Computer Information Systems*, *51*(3), 21–28.

UML 2. (2014). *Unified Modeling Language 2.0.* Retrieved from http://www.uml.org/

van Lamsweerde, A. (2004). Goal-oriented requirements enginering: a roundtrip from research to practice. In *Requirements engineering conference, 2004. proceedings. 12th ieee international* (pp. 4–7). doi: 10.1109/ICRE.2004.1335648

Van Solingen, R., & Berghout, E. (1999). *The goal/question/metric method: A practical guide for quality improvement of software development.* McGraw-Hill Higher Education.

Vernadat, F. (1996). *Enterprise modeling and integration.* Springer.

Verner, J. M., Cox, K., Bleistein, S. J., & Cerpa, N. (2005). Requirements engineering and software project success: an industrial survey in australia and the u.s. *Australasian J. of Inf. Systems*, *13*(1).

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2000). *Experimentation in software engineering: An introduction.* Norwell, MA, USA: Kluwer Academic Publishers.

Wojcik, R., Bachmann, F., Bass, L., Clements, P. C., Merson, P., Nord, R., & Wood, W. G. (2006). *Attribute-driven design (add), version 2.0, technical report cmu/sei-2006-tr-023* (Tech. Rep.). Pittsburgh, Pennsylvania, USA: Software Engineering Institute.

Xu, L., Ziv, H., & Richardson, D. (2005). *Towards modeling nonfunctional requirements in software architecture.*

Yu, E., & Mylopoulos, J. (1994). Using goals, rules, and methods to support reasoning in business process reengineering. In *System sciences, 1994. proceedings of the twenty-seventh hawaii international conference on* (Vol. 4, p. 234-243). doi: 10.1109/HICSS.1994.323491

Yu, E. S.-K. (1995). *Modelling strategic relationships for process reengineering* (Unpublished doctoral dissertation). University of Toronto, Toronto, Ont., Canada, Canada.

Yu, E. S. K., & Mylopoulos, J. (1994). Understanding 'why' in software process modeling, analysis and design. In *In proceedings sixteenth international conference on software engineering* (pp. 159–168).

Zikra, I., Stirna, J., & Zdravkovic, J. (2011). Bringing enterprise modeling closer to model-driven development. In P. Johannesson et al. (Eds.), *The practice of enterprise modeling* (Vol. 92, p. 268-282). Springer Berlin Heidelberg.