

A method for Requirements Management and modeling



Universiteit Twente
de ondernemende universiteit

Master Thesis Business Information Technology
University of Twente

Date
April 2, 2008

Author
Wilco Engelsman

Graduation Committee
Dr. P.A.T. van Eck
Dr. M.E. Jacob
Dr. Ir. H. Franken
Dr. Ir. H. Jonkers

BiZZdesign

Title page

Title: Method for Requirements Management and modeling

Author: W. Engelsman

Student number: s0120324

Date: 2 April 2008

Track: BIT Mixed

University: Universiteit Twente, Enschede

Company: BiZZdesign B.V.
Colosseum 21
7521 PV Enschede
www.BiZZdesign.nl

Graduation Committee: Dr. P.A.T. van Eck (1st supervisor)

Dr. M.E. Jacob (2nd supervisor)

Dr. Ir. H.M. Franken (external supervisor)

Dr. Ir. H Jonkers (external supervisor)

Summary

Motivation

Requirements Engineering as a scientific discipline has matured over the last decade, the process itself is rather well understood and led to numerous of techniques and models. Lately, the term requirements management has become more important. Requirements Management adds traceability to the requirements engineering process and artefacts. Although requirements management is a fairly well understood discipline, in practice companies still have questions and issues how to deal with requirements. BiZZdesign identified this need and initiated a design study to develop a new method for Requirements Management.

Design Goal

BiZZdesign already completed some work in the area of Requirements Management and the main result from this initial study was a proposed modeling domain. What BiZZdesign needed next was a requirements method, based on the scientific literature to acquire the requirements and afterwards model the requirements using the proposed modeling domains. Therefore the design goal of this study was:

"To develop a method for requirements management, the method will comprise of requirements acquisition techniques, specification techniques and modeling techniques in the context of EA and SOA. Requirements modeling will be based upon the proposed modeling domain (see [66])."

Approach

To realize the design goal we needed a comprehensive understanding of the subjects at hand, therefore we needed to acquire knowledge about the subjects under investigation. Therefore we investigated the requirements engineering, the requirements traceability and enterprise architecture literature. After solving these knowledge problems we were able to provide the basic requirements for a requirements management method.

Results

Using the requirements provided we started to design a method for requirements management. In this method we proposed a basic framework, comprising the following steps:

- Problem investigation
- Investigate solution alternatives
- Solution validation

For each of these steps we provided a basic approach, the problem investigation phase leads to the business requirements. During the solution alternatives phase we investigate possible alternatives and specify the requirements for these alternatives. These system requirements for these possible solutions should implement the business requirements from the problem investigation phase. After specifying the solution the solution needs to be validated and shown correct.

A second important goal was to incorporate the already available modeling domain; we were able to use this modeling domain after providing a few adjustments. We introduced the concepts of **business requirements** and **system requirements** as a concrete specialization of a requirement. We also introduced some of the concepts found in **goal oriented requirements engineering**. We designed our **business requirements** as goals that should be realized by the **system requirements** of a proposed solution.

Recommendations

The main goal of this thesis was to provide insight how a RM could look like and use it as a **blueprint** for a **handbook** about RM. We do feel that the method presented here is a complete and logical structuring about how the scientific field discusses RM, however the method is completely based on theory and the only validation is done by **showing how to use the method** on a small example. Therefore we can suggest the following course of action:

- Empirical based knowledge gathering, find out the exact desires from the practical field and what techniques are currently used for RM;
- Validate the method using larger scale field trials;
- Focus on the integration of the different RE process models and views on RE;
- Pay special attention on the service-oriented section in the validation effort. This section is based on the least amount of scientific literature;
- Use this input to launch a new iteration of the design cycle used during this research;
- Implement the method by writing the handbook;
- Evaluate the implementation;

Preface

This document is the result of eight months of hard work and will conclude my master programme Business Information Technology at the University of Twente. This is also the end of a ten year uphill battle against the Dutch educational system. This is something I have been looking forward to for quite some time. As most people know, getting to this point where I am now was not easy. I can only speculate about the reasons why it has taken me this long. Was it because of a misinformed headmaster who informed my parents at what level I should enter secondary school, or was it because I wasn't ready? All that matters now is that I have completed my final assignment.

It wasn't possible to successfully complete this assignment with the help and supervision from a number of persons. Firstly I would like to thank Henry Franken for giving me this opportunity to perform this assignment and renewing his support in the form of a job offer. Secondly I would like to thank the remaining members of my graduation committee, Henk Jonkers of BiZZdesign, Pascal van Eck and Maria Jacob from the University of Twente, for their supervision and insights.

I would also like to mention that this document would never be completed without the excellent work climate at BiZZdesign. It was a joy going to work on a daily basis.

Lastly I want to thank my parents for their continuous and unconditional support throughout these past years, supporting every decision I made to enter yet again a new school or university and providing the financial security I needed so I could continue my Endeavour.

March 2008,

Wilco Engelsman

Table of Contents

1. Introduction	11
1.1. BiZZdesign	11
1.2. Conceptual research design.....	13
1.3. Technical research design	23
2. literature review	25
2.1. Requirements Management	25
2.2. What are Requirements	25
2.3. Requirements engineering overview.....	27
2.4. RE for bespoke systems	29
2.5. RE for COTS selection	31
2.6. RE for services	33
2.7. Requirements Traceability	35
2.8. Requirements Management and Enterprise Architecture.....	37
2.9. Requirements for a RE method.....	37
2.10. Conclusion.....	39
3. A RE method	41
3.1. A RE framework.....	41
3.2. RE approach for problem investigation.....	44
3.3. Investigating solution alternatives.....	46
3.4. Solution validation	69
3.5. Conclusion	70
4. Requirements modeling in Architect.....	72
4.1. Requirements traceability modeling framework.....	72
4.2. Requirements modeling concepts.....	75
4.3. Requirements Viewpoint.....	80
4.4. Conclusion	81
5. Tool support	83
5.1. Requirements for tooling support.....	83
5.2. Current tooling support.....	83
5.3. Conclusion	85
6. Method validation	87
6.1. Example case	87
6.2. A new business service	87
6.3. Selecting a customer management application	99
6.4. Mapping of solution properties to the requirements	102
6.5. Conclusion	104
7. Comparison to related concepts	105
7.1. Positioning of the method in the layered BPM model.....	105
7.2. BPM and the RM method.....	106
7.3. Positioning in development methods	107

7.4. Alternative RE methods	109
7.5. Conclusion	112
8. Conclusion and discussion	114
8.1. Result	114
8.2. Discussion	114
8.3. Relevance	115
8.4. Conclusion and further work	115
8.5. Recommendations	117
Appendix I. Goal Oriented RE	128
Appendix II. RE for bespoke systems	136
Appendix III. Requirements Engineering and COTS selection	153
Appendix IV. RE and services	158
Appendix V. Requirements Traceability	166
Appendix VI. RM AND EA	173
Appendix VII. SOA	178

List of figures

Figure 1: layered BPM model.....	12
Figure 2: requirements positioning.....	17
Figure 3: engineering cycle	17
Figure 4: applied research framework by Verschuren [160].....	21
Figure 5: spiral research model	22
Figure 6: different phases of requirements engineering [69]	27
Figure 7: conceptual model (adapted from [125], refined using concepts from [66]).....	36
Figure 8: Requirements allocation model (adapted from [125])	35
Figure 9: framework for RE method.....	42
Figure 10: spiral model of the framework.....	43
Figure 11: business requirements approach.....	44
Figure 12: process selection framework.....	47
Figure 13: RE for bespoke systems	48
Figure 14: RE for COTS selection.....	52
Figure 15: overview service oriented RE	56
Figure 16: RE at the service consumer	57
Figure 17: SLA process [26].....	60
Figure 18: RE for in-house services or externally.....	62
Figure 19: Specifying or selecting	63
Figure 20: RE at the service provider	64
Figure 21: sla process [26]	68
Figure 22: architecture life-cycle model	73
Figure 23: motivation layer	74
Figure 24: realization layer.....	75
Figure 25: example goal tree.....	84
Figure 26: example business process	84
Figure 27: context diagram	85
Figure 28: subject domain data model	85
Figure 29: snippet of a goal tree.....	89
Figure 30: modeled concerns of the customer.....	90
Figure 31: modeled concerns of senior Management.....	91
Figure 32: business requirements related to stakeholder concerns.....	91
Figure 33: stakeholders, concerns and business requirements.....	92
Figure 34: snippet of an example business process	93
Figure 35: available services	94
Figure 36: mapping from business requirements to system requirements.....	96
Figure 37: snippet of modelled requirements for a portfolio Management service.....	97
Figure 38: modeling the design decision.....	98
Figure 39: requirements model for the portfolio management service	99
Figure 40: stakeholders, concerns and business requirements.....	100
Figure 41: choosing application x.....	101
Figure 42: requirements model for the customer management application	102
Figure 43: an EA process model [71]	108
Figure 44: pre and post re traceability [58].....	168
Figure 45: framework for traceability [116]	171
Figure 46: Zachman framework.....	175
Figure 47: TOGAF ADM.....	175
Figure 48: IAF framework	177
Figure 49: continuous improvement and refinement	177
Figure 50: layering in SOA [105] 161].....	180

List of tables

Table 1: mapping of the ways onto aydin's model.....	20
Table 2: overview RE activities, techniques and methods.....	30
Table 3: overview COTS selection methods.....	32
Table 4: cots techniques.....	33
Table 5: overview SORE methods.....	34
Table 6: overview techniques.....	35
Table 7: stakeholder.....	75
Table 8: stakeholder concern.....	76
Table 9: business requirements modeling concept.....	77
Table 10: design decision.....	78
Table 11: design alternative.....	78
Table 12: requirements modeling concepts.....	79
Table 13: case modeling concepts.....	80
Table 14: design artifact.....	80
Table 15: stakeholder concerns view.....	81
Table 16: modeling concepts related to the phase of the RC method.....	82
Table 17: request for insurance scenario.....	94
Table 18: mapping of the RM onto the EA method.....	109
Table 19: mapping onto EAA approach.....	109
Table 20: summary of analyzed goal methods.....	134
Table 21: overview scenario based methods.....	152
Table 22: overview traceability usage.....	170

1. INTRODUCTION

Requirements Engineering as a scientific discipline has matured over the last decade, the process itself is rather well understood and led to numerous of techniques and models. Lately, the term requirements management has become more important. Requirements Management adds traceability to the requirements engineering process and artefacts. Although requirements management is a fairly well understood discipline, in practice companies still have questions and issues how to deal with requirements. This report describes a method that will guide practitioners through the requirements engineering process and show how to incorporate traceability into that process. It also describes tool support and how to model requirements. A major characteristic of this method is that it tries to not only provides guidelines for dealing with requirements when developing custom software, but also provides guidelines when selecting standard components or how RE works in a service oriented organization.

The structure of this chapter is as follows. Chapter 1.1 introduces BiZZdesign, the company where this research takes place, section 1.2 describes the conceptual design and background information and section 1.3 describes the technical research design.

The second section, the way of thinking, describes the conclusions about the topics under investigation, in our case, Requirements Management and RM in Architecture. The third section of this document, the way of working, describes the process of the RM and provides guidelines for this method. The fourth section, the way of modeling, explains how to model the requirements and finally the last section, way of supporting, describes how this method can be supported through the use of tools.

1.1. BiZZdesign

BiZZdesign operates at the junction of process, organization and information technology [19]. BiZZdesign works with effective and proven methods and tools, to improve businesses. This is supported by tools, consulting and training. Its core activities are designing, improving and arranging business processes, from an architectural level to process management level. These activities are part of a total business engineer approach (including, among others, EA, BPE, BPM, etc.). For the BPM approach BiZZdesign uses the BPM layered-model (see Figure 1).

We will use the work of [85] [71] to explain this model. The model comprises of three horizontal layers (strategize, design and execute) and two vertical control layers (implementation and governance).

Strategize

The focus at "the strategy" level lies on creating strategic guidelines. These guidelines will be translated into more detail in the lower levels. For example, some of the activities could be to determine the type of governance, quality standards and to create an Enterprise Architecture.

Design

Models are used to (re)design organizations; these describe the various layers (e.g. business, application and technology) at different abstraction levels. Making the relations between these layers explicit will help to identify and solve the possible lack of coherence within the EA and possible alignment problems.

Execute

The execution layer refers to the realization of the proposed changes. The new architecture will be included and used in the daily routines. This may lead to new products or services, possibly during new business processes, using new applications and information.

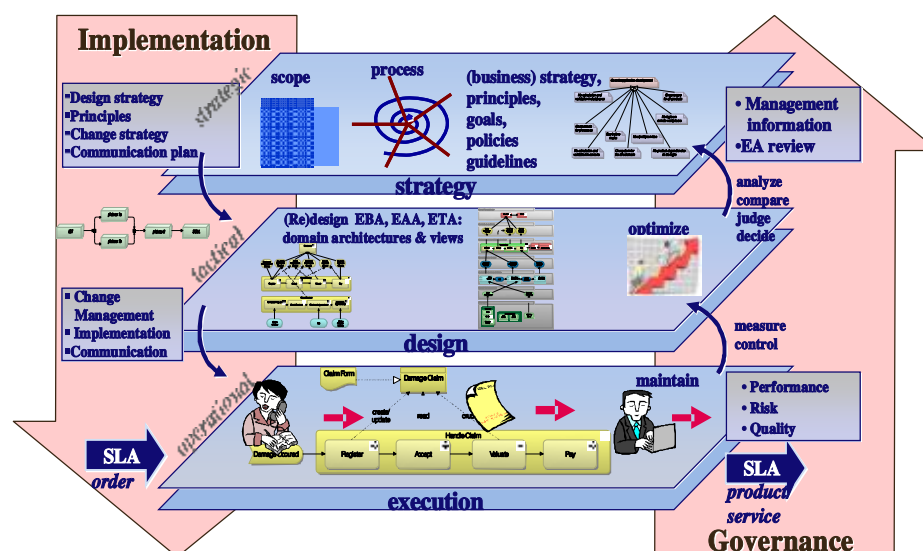


Figure 1: layered BPM model

The above-mentioned horizontal layers covering the change process must be complemented with control mechanisms facilitating its management and monitoring.

Implementation

Implementation is generally, but not as pure waterfall process, executed top-down and will result in the implementation of the architecture. The intention is to make the EA actionable, or, in other words, to incorporate and use architecture in the working practices in order achieve the business goals driving the change.

Governance

Inflicting control from the operational level towards the strategic level (i.e., bottom-up) can be translated into the governance processes.

The governance layer is a bottom-up control layer, control is imposed from the operational level to the strategic level. This will enable the continuous improvement of the EA, while closing the loop initiated in the implementation.

Governance begins at the operational level with tasks such as measuring and monitoring of performance, risks and quality indicators (e.g. service-level-agreements). At the tactical level, information coming from the operational level can be processed by means of complex analyses (e.g. cost benefits, performance analysis, etc). This may lead to relevant management information, and can serve and motivate the decision making process at the strategic level.

The applications developed by BiZZdesign can also be placed within this model. The application "Architect" is meant to model Enterprise Architectures. The "BiZZdesigner" tool is used to model business processes, this is done on the tactical level. The tool "RiskManager" is used on the operational level and is used to monitor financial risks.

1.2. Conceptual research design

In the previous section we have introduced this research, thus providing an initial scope. It is now possible to conceptually design the project.

A conceptual design provides a steering function during the project design and execution. If the conceptual design covers the aspects required it is possible to determine the concrete research activities that should be done during the executing phase [160].

1.2.1. Design goal

Customers of BiZZdesign increasingly expressed the need for a method that enables them to work with, and manage, requirements. Competitors of BiZZdesign (e.g. Telelogic) already have a RM tool and method to assist their customers. This research will not just focus on requirements engineering, but also adds traceability to the requirements engineering process. The method will not limit itself to RE for bespoke systems, but will also describe what RE for COTS selection looks like and how in a service-oriented environment RE can be used.

Another new aspect of this method is that it also includes ways to support the RE process using tools and model the requirements.

Therefore the goal for this research is:

To develop a method for requirements management, the method will comprise of requirements acquisition techniques, specification techniques and modelling techniques in the context of EA and SOA. Requirements modelling will be based upon the work done by Sebastiaan Hoogeveen [66].

The results from this design are applicable for two main stakeholders. For BiZZdesign the main results will be a better understanding of RE and RT and a method design that can be used to write a handbook about service-oriented RM. This handbook is interesting for the customers of BiZZdesign. This handbook will enable them how to deal with the issues related to RM.

1.2.2. Background

This design study is about developing a method for RM. In this chapter we will give a brief introduction of the concepts of this research. The main concepts in this research are Requirements Engineering and Requirements Traceability. In this thesis we will take the view that RM is [48]:

RM is the systems engineering activity that is concerned with finding, organizing, documenting and tracking requirements for systems with its focus on maintaining traceability between requirements.

This definition can be split into two separate concepts. The activities that are concerned with finding, organization and documenting requirements are called *requirements engineering*. Tracking requirements is known as *requirements traceability*.

1.2.2.1. Requirements Engineering

RE is about getting from problems to the possible solutions [14]. In this thesis we will show that there are different types of solutions that can solve problems and that we do not need to limit ourselves to a particular solution type. Therefore we define a solution as [170]:

A *solution* is a system that provides desired services

When we analyze this definition more closely we see that the central topic of a solution is a system. To fully understand what a solution is we also need a definition of a system [170].

A *system* is a set of elements with relationships among them that have emergent properties.

A system can be a new information service to customers, new business processes, new work procedures, supporting software systems and application services that support business processes.

Because RE is about bridging the gap between problem and solution, two different views on RE emerged. The first view on RE is *problem oriented*. Problem oriented RE is about problem investigation, to investigate and determine what the actual problem is. Problem oriented RE involves finding and documenting the problematic phenomena before thinking of how to solve that particular problem. A key concept

within this is: understanding the actual problem is required before one starts to think of solution alternatives.

The second view sees RE as solution specification. *Solution oriented RE* is about designing and describing system behavior and about showing which alternative best solves the problem.

When we combine these two views on RE we can use the following definition [27]:

RE is investigating and describing a problem domain and requirements and designing and documenting the characteristics for a solution system that will meet these requirements.

Requirements are collected in a *Requirements Specification (RS)*, Wieringa [168] provides the following definition for a RS:

An RS is a description of the relevant business objective(s) and of the desired way of working to reach these objectives. A RS must not refer at all to the system under development but only to the desired way of working and the business objectives: It only says what work is done and why this is done.

The definition given above clearly takes the viewpoint that requirements are part of the problem domain, because we will discuss both views in this thesis, we will also give a definition of a RS that clearly falls into the solution domain view [130]:

An RS is a document that contains the requirements; this specification defines the product and can be used as a contract to build the product.

If we try to relate these two views, we can argue that *problem oriented RE* and *solution oriented RE* come together in *architecture* [169]. If we investigate a certain problem, we could determine that certain stakeholders experience that the service delivery to customers is insufficient to realize certain business goals. A solution to this problem could be new business processes, a new delivered service to the customer, alignment of existing business processes to this new service and a new supporting software system. Together these different solutions form the architecture of an overall solution. If we look at enterprise architecture (EA), this solution is either an architectural element within the architecture or a combination of multiple architectural elements. The emergent properties of the architectural elements combined should solve the identified problem.

When we look at RE for software solutions we see three different types of RE. Requirements engineering for bespoke systems, RE for commercial-off-the-shelf-software (COTS) and RE for services. RE for bespoke systems is the classical requirements engineering, whereas RE for COTS emerged around a decade ago. The latest trend within organizations is Service-Oriented-Architecture, this has led to the first steps for service-oriented requirements engineering. We will discuss these variants in more detail in the way of thinking section of this thesis.

1.2.2.2. Requirements Traceability

Requirements traceability has received a fair amount of attention in the scientific community in the past decade and it is a demand in a number of quality standards (e.g. CMM(I) and U.S. Department of Defense (DoD) standard 2167A). In CMMI(I) we find RM at level 3 and RE at level 3. RT can be described as the ability to follow the life of a requirement in a forward and backwards direction. For example, tracing back a requirement from whom it spawned is *backwards traceability* and tracing the design artifacts that spawned from a requirement is *forward traceability*. This leads to the following definition [57, 59]:

RT refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases).

1.2.3. Positioning of requirements in Enterprise Architecture

When we position requirements in the global context of Enterprise Architecture (EA) we can see the following [95]. EA provides a global scope and context to RM. Requirements are no longer specified just for a single system, but a multitude of systems all aligned to reach the common organizational goals. In this way the architecture provides an additional set of requirements. High level business goals act as general requirements and scope. For every relevant solution realization project the relevant goals are selected and made more concrete. These requirements can be based on the architectural principles or requirements/constraints concerning interoperability with existing systems in the architecture.

Within the actual solution realization projects (design projects) the requirements engineer designs the system requirements and together with the architectural requirements the systems are specified.

We can find requirements on the business level; here they specify products, services, processes, etc. These solutions usually need some sort of software support and therefore provide a scope what the underlying software systems should accomplish. What we see here is a goal driven requirements process. Systems are not seen as single entities but they implement the systems on the higher layers in the EA. Both RM and the EA hereby drive the design of these systems.

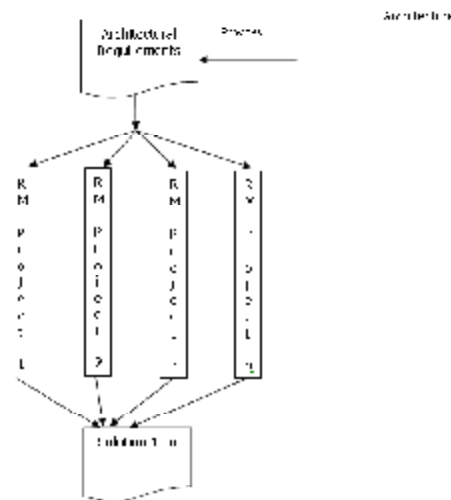


Figure 2: requirements positioning

In Figure 2 we have positioned the influence of architectural requirements on the various parallel running projects. In this section we will look more closely at the activities in these projects and how architectural requirements influence these activities.

Problem solving happens in a logical framework [172] (see Figure 3). What we see here is that RE is from getting from problem investigation to a solution design. This design needs to be validated, to see if it has closed the identified gap properly. The influence of the architecture is to provide additional information, requirements and constraints to the problem solving process. For example, when the requirements for a certain solution are specified the architectural requirements can mandate that the system under development needs to have an interface to existing systems.

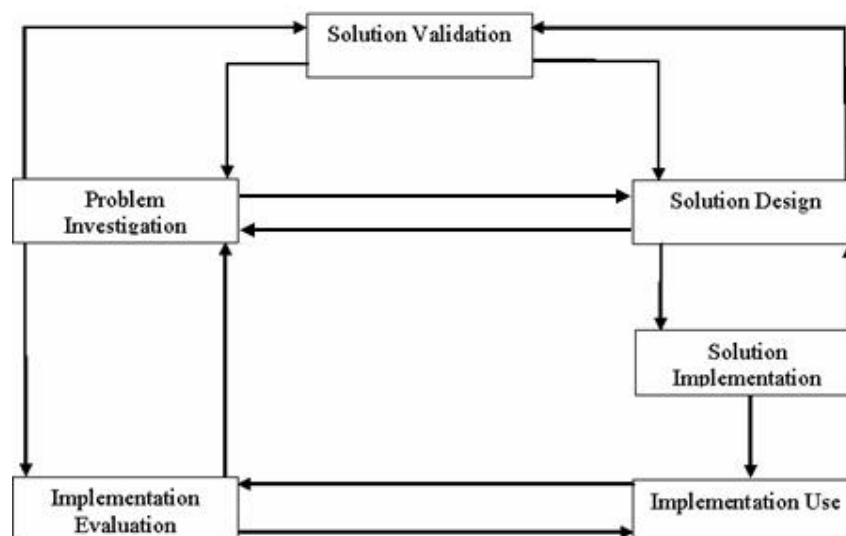


Figure 3: engineering cycle

1.2.4. Aspects of a method

Within this research we are interested in literature about requirements engineering, requirements traceability, business process management, SOA and enterprise architecture. We will use these findings to create a method, as specified in the goal of this research. But firstly we need to make clear what a method is. We will use the definition provided by [29]:

A method is an approach to perform a systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities with corresponding development products.

This definition provides a wealth of information what a method should contain, it specifies that a method is an *approach*, based on theory (way of thinking), that contains guidelines (directions and rules) and that it should be structured in a systematic way.

To address the aspects provided by the definition above we will use the work of Aydin [15] and Wijers [175], mostly adopted from [71] and [62].

In [15] Aydin proposed a generic framework, which stems from general method engineering theory. According to Aydin a general method consists out of three elements: a philosophy, a framework and supporting tools and techniques. The philosophy part will cover all the basic principles, assumptions and constraints concerning the requirements method, thus philosophy defines the scope and steers the constituent steps of the method.

A framework is a collection of templates, rules and styles through which the various elements (such as products, deliverables and process steps) of the method can be positioned and classified. The tools and techniques support the activities specified by the method and contribute to the physical realization of the results following the application of the method. This framework can be used to structure the method initially, but it is too abstract to be of any use for a concrete method. Therefore [175] [152] performed some additional research has been undertaken to complement this framework. This framework is often used as a framework for developing, understanding, and structuring modeling methods. This additional research has led to the six ways, the way of thinking, the way of working, the way of modeling, the way of supporting, the way of communicating and the way of controlling.

The way of thinking

The way of thinking is in the literature also referred as paradigm, underlying perspective or philosophy [175] [62]. It articulates the assumptions on the kinds of problem domains, solutions and modelers. The way of thinking should answer the 'why question', resulting in the explanation of the remarkable aspects of the method. In my thesis this will cover the chapters explaining what requirements engineering, requirements traceability, enterprise architecture and SOA are. More precisely, we will investigate these topics more extensively using a literature study to identify the state of the practice, main concepts and techniques.

The way of working

The way of working describes a structured manner to apply the method, it consists of the tasks possibly to be performed during a development process, the decomposition of tasks into subtasks and decisions, the possible order of tasks and decisions and the informal guidelines and suggestions on how to perform tasks and how to make decisions [175].

In the context of this thesis one can think of the structuring of the execution of the method in terms of tasks, order of execution and guidelines or heuristics for these tasks. We will select practical to use techniques, describe how to use them to gather requirements and document them accordingly.

The way of modeling

The way of modeling defines the concepts of the language(s) that will be used to denote, document, analyze, visualize or animate the architectural requirements descriptions. This is based largely on the work provided by [66] and will be using ArchiMate modeling concepts and extensions to the ArchiMate modeling concepts, as ArchiMate does not support all the required concepts.

The way of controlling

The way of controlling deals with managerial aspects of a development project, providing a mechanism to control the way of working. It includes planning and plan evaluation. Planning and plan evaluation are realized by dividing development processes into phases with checkpoints and baselines. The way of controlling interacts with the ways of modeling and working. The controlling mechanism intends to influence the operational activities (working and modeling) by steering measures, which leads to a certain feedback of information. Where upon new steering measures are possibly desirable. One possible way in controlling is providing guidelines in assigning the roles to different persons within the requirements engineering process.

The way of supporting

Tools should support information systems development, which in itself can be understood by distinguishing a way of modeling, working and controlling, all embedded in a specific way of thinking. In this chapter we will provide guidelines how the available tools (i.e. Architect or BiZZdesigner), or new tools can support the RE method.

The way of communicating

The way of communicating describes how the meaning of the abstract concepts defined by the way of modeling will be conveyed to the different stakeholders, for example in terms of textual or graphical notation.

Usually, the way of communicating uses a graphical notation. It may very well be the case that different methods are based on the same way of modeling, and yet use different graphical (or textual) notations [62]. ArchiMate may also play a role here, as it provides a standardized notation. Although this "way" is not within the scope of this research.

In Table 1: mapping of the ways onto aydin's model we will show the mapping between Aydin's model and the six "ways" (adopted from [71]).

Aydin's Model elements	Ways REF
Philosophy	Way of thinking
framework	Way of working
	Way of controlling
Tools and techniques	Way of modeling
	Way of supporting
	Way of communicating

Table 1: mapping of the ways onto aydin's model

1.2.5. Knowledge problems

In order to solve the dilemma how to solve the design goal of this project we need to ask ourselves what knowledge we need to know before we can think of solving the design problem. Therefore this section will describe a number of research questions addressing this problem.

After analyzing the goal and initial literature of this research the following main research question can be composed.

What RM method can be created based on the scientific literature, structured around the layered BPM model and incorporates requirements modeling.

To answer the main research question, we have composed a number of research questions which cover the concepts found in the literature. The first four questions are part of a literature survey to review the state-of-the-art within these topics. The second groups of questions are the actual research questions.

Literature Survey

- What is Requirements Engineering?
- What is Requirements Traceability?
- What is the role of RM in EA?

Research Questions

- How does a method for RM globally look like
 - What framework can we use when we perform RM within EA?
 - What are the steps/phases within this method?
 - How do these phases look like?
 - What guidelines can we provide?
 - How can traceability be incorporated within this method?

- What are the fundamental differences between classical RE, RE for COTS selection and service oriented RE?
- How do you position this method within the BiZZdesign layered model?
 - How do the activities within RM relate to the 5 areas of the layered model?
 - What activities are performed at the strategize level?
 - What activities are performed at the design level?
 - What activities are performed at the execute level?
 - What activities are performed at the implementation level?
 - What activities are performed at the governance level?
- How can requirements be modelled?
 - How do you model requirements in an architecture using the proposed meta-model?
 - What changes need to be made to the proposed meta-model?
 - What level of abstraction should be used?
- How can this method be supported through the use of tools?
 - Which existing tools within BiZZdesign can be used?
 - Which other existing tools can be used?
 - Which tools need to be developed to support this method?

1.2.6. Research Model

We have designed this research using a research framework by Verschuren and Doorewaard [160] (see Figure 4). This research framework gives a schematic overview of the research goal. It also shows the steps needed to reach that goal. It helps to communicate the type and nature of the project and its results.

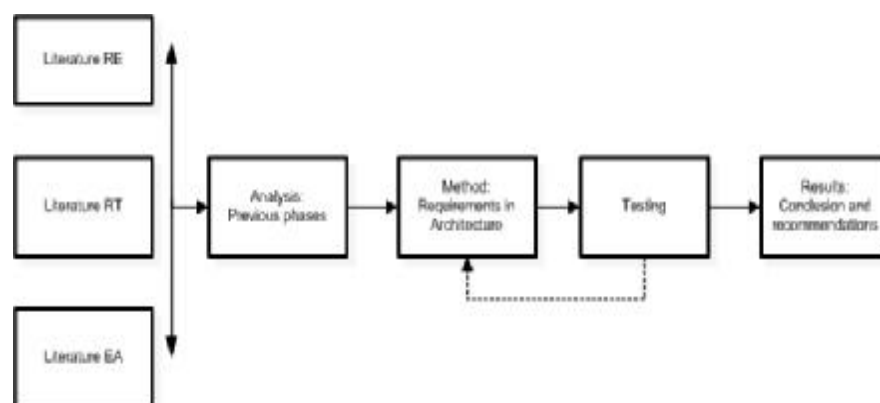


Figure 4: applied research framework by Verschuren [160]

At first we will review the literature on requirements engineering, requirements traceability, enterprise architecture and existing work about requirements in architecture. We will use this literature review to determine the key concepts used in

requirements management and architecture. After identifying the concepts (methods, techniques for said subjects) we will order and place them in the layered BPM model in use by BiZZdesign, the next phase comprises of a RM method. This will demonstrate how to elicit, analyze, prioritize and validate requirements. The third phase describes how to introduce and realize RT. Finally we will describe how to model these requirements using the proposed meta-model for requirements modeling.

The process of creating this method is not a waterfall like process. It is rather a evolutionary-iterative process where after each cycle the stakeholders will validate the method. Therefore, in the process of developing a method for RM, we will take a spiral approach (see [24],[149] and [71] for examples of such spiral models and Figure 5).

Spiral approach for RE

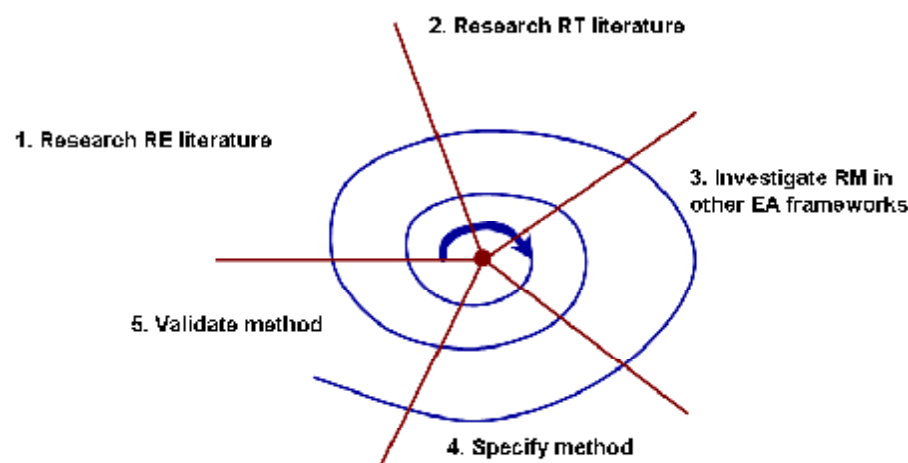


Figure 5: spiral research model

The elaboration of this research method starts in the middle. We start reviewing existing literature about RE, RT. The next step is to investigate what has been done about RM in existing EA methods and frameworks. The fourth step is to actually create the method and the final step is to validate this method. After each cycle we have gathered all comments, learning experiences, etc and refined the method. These first results of method validation will be used to initiate another cycle of refinement and validation. After a number of cycles we have refined the theoretical aspects of the method with input from practice. After this phase the method not only meets the demands dictated by the scientific literature, but is also applicable to use in practice.

1.2.7. Research validation

The result of this research is a method based on the combination of requirements engineering, requirements traceability, layered BPM model and Enterprise Architecture, which will answer the question how to deal with requirements on an architectural level.

Validation will take place on the following levels:

- the graduation committee will evaluate the thesis;
- an example case will be used to demonstrate the usefulness of the method;
- the method will be tested using an e-mail based survey (although this is out of the scope of this thesis), results will be used for implementing the result in a handbook;

1.2.8. Scope definition

This research will be based on existing methods and models available in the literature. For every relevant concept we will review the models and methods. After reviewing we will select the most appropriate model to use and where necessary combine them. It is not the goal of this research to create a new method or model for elicitation, validation, tracing, modeling, analyzing requirements, etc.

Although we will review existing popular enterprise architecture frameworks, we will only try to find out how they deal with requirements and traceability. It is not the goal to develop modeling concepts that are compatible with those frameworks. Requirements traceability and modeling will be introduced using ArchiMate modeling concepts, based on an internal proposed meta-model [66].

Since the goal of this research is creating a practical handbook, we will focus less on formal methods and notations in requirements engineering. We will refer the reader to them, but not cover them extensively.

In this research we will concentrate only on the way of thinking, way of working, way of modeling and the way of supporting.

1.3. Technical research design

After designing the direction of the research, as done in the conceptual research design, it is time to design how the research should be realized. This paragraph describes how to obtain the answers for the defined research questions in the previous chapter. It describes what material and sources should be utilized and which strategy has to be pursued.

1.3.1. Research Material

To answer the research questions it is a necessity to examine research material. During this research we will review the scientific literature about RE, RT and RM to determine the techniques and methods available. A major topic within RE that is largely neglected by scientific research is stakeholder identification and role

assignment. We will look into the interdisciplinary fields of systems engineering and enterprise engineering to see how these topics there are covered.

We will also use the scientific literature and internal BiZZdesign documents to determine the available EA frameworks. Within BiZZdesign there were a number of research projects that lead to handbooks about EA. These handbooks describe EA frameworks and the BPM model used by BiZZdesign. We will use these handbooks and adapt the relevant material from it.

1.3.2. Research Strategy

According to Verschuren [160] the most decisive factor in the technical design is choosing an approach. The approach, research strategy, covers the whole of related decisions about the way the research should be executed.

There are five types of research strategies [160]:

- survey
- experiment
- case study
- grounded theoretical approach
- literature research

This research is a combination of two of them. The method will be elaborated using a literature research strategy. The method will be validated by applying it in practice using one, or more, case studies. We will use a case study to get feedback on the method from practice, in order to prevent that the method will only be created as a theoretical exercise.

2. LITERATURE REVIEW

In this chapter we will discuss the results from a number of literature reviews (these can be found in the appendices of this document). Before we start to design a method, the concepts need to be investigated. Firstly we will discuss the topic of RE. We identified a number of sub-concepts in RE. We will investigate problem-oriented RE, RE for bespoke systems, RE for COTS selection and lastly service oriented RE (SORE). The second concept within RM is RT. We will also discuss the results from investigating the literature about RT.

In this thesis, we do not only look at RM as a stand-alone activity, but in conjunction with EA. Therefore we have investigated some of the most popular EA frameworks. In the last section we will discuss the derived requirements from these reviews.

In section 2.1 we will discuss requirements management, section 2.2 describes the concept of requirements. Section 2.3 provides an overview of the concept requirements engineering. In section 2.4 we discuss RE for bespoke systems, in section 2.5 we discuss RE for COTS selection and in section 2.6 we will discuss service oriented RE. Every section will be structured as follows. We firstly describe the essence of the topic and then discuss the techniques.

2.1. Requirements Management

As mentioned in the introduction of this thesis RM is an umbrella activity that describes various topics on how to deal with requirements. It is now time to provide a more concrete description for RM. In this thesis we will take the view that RM is [48]:

RM is the systems engineering activity that is concerned with finding, organizing, documenting and tracking requirements for systems with its focus on maintaining traceability between requirements.

This definition can be split into two separate concepts. The activities that are concerned with finding, organization and documenting requirements are called *requirements engineering*. Tracking requirements is known as *requirements traceability*.

2.2. What are Requirements

Before we start to discuss the requirements engineering process we will need to describe what a *requirement* actually is, we will provide the following definition from [27]:

Requirements are the effects that the client wishes to be brought about in the problem domain.

Robertson and Robertson [130] state that a *requirement* is:

A requirement is something that the product must do or a quality that the product must have.

The first definition describes the requirement from the problem domain (this is the domain in which a system is going to be used) viewpoint, the second definition emphasizes less on the problem domain and more on the capability needed for a solution.

In this thesis we see a number of different requirements as important to solve our problems. *Business requirements* [166] are the requirements why something is being built; they describe the benefits for the customers and the business. Business requirements describe in an abstract manner what should happen (i.e. negate the experienced problems) to solve the problem without specifying the solution type. *Business requirements* describe the desired effects the solution should cause in the problem domain which leads to reducing the gap between what there is and what is experienced.

Business requirements serve as an input when we start to think about what solutions are possible to solve our problem. Business requirements are refined into *system requirement* [102]. System requirements are more concrete than business requirements; they are already more constrained by the chosen solution type, where in essence different solution types can solve the same problem. When we speak of system requirements we can already distinguish between *functional* and *non-functional* requirements. *Functional requirements* can be met by certain behavior of the new solution system and *non-functional* requirements describe how well the solution system meets these requirements (i.e. without non-functional requirements the system can be built as cheaply as possible, because it does not need to adhere to a certain quality standard).

It is important to notice that *system requirements* are designed [84]. This is so because we design the solution and the properties of these solutions are our requirements. We will illustrate this with an example. After we have identified a business problem and stated the business requirements we have concluded that we need both a new tailor made software system and new work procedures to achieve our business requirements. The requirements engineer uses the identified business requirements as an input document to determine the requirements of the proposed solutions. He needs to *refine* the business requirements into requirements of two proposed solutions. He therefore designs and structures the required properties of the solutions to match the stated business requirements.

Another well known classification is *user requirements* [166]. User requirements are usually captured in the form of use cases or scenarios (an instance of a use case). These describe the tasks or business processes a user will be able to perform, or wants to realize with the system (user goals).

Our view is that software based systems, although not always, help realize or completely realize the before mentioned business solutions. We can also find the functional and non-functional requirements at the software system levels.

Constraints are another type of well known requirements. These are also properties of the proposed solution, but these limit the solution space of the analyst (e.g. the system shall be designed according to the latest service oriented principles).

2.3. Requirements engineering overview

In the previous chapter we introduced a definition of RE and we explained the different views on RE in the introduction section of this thesis. The first part of the definition describes that RE is involved with investigating and describing the environment in which the envisioned system is supposed to create desired effects (the so-called problem domain). The second part of the definition describes that RE is also about designing and documenting the behavior of the envisioned system.

Our view in this thesis is, as mentioned, that RE is about getting from a problem to a solution. In this thesis we therefore will use a requirements engineering method that describes techniques ranging from problem description to solution specification and add traceability to the process. Understand that problem analysis and solution specification need not be sequentially related. In general, problem analysis and solution specification proceed jointly, with the engineer spending most, but not all of her time on problem analysis at the start of the process, and spending most, but not all of her time on solution specification towards the end of the process [169]. If we relate that concretely to our method, the requirements engineer has during problem analysis already an idea what possible solution types he could use to solve the problem (e.g. only a business solution, business solution and software, etc).

In [69] Hull *et al* also distinguish between requirements in the problem domain and requirements in the solution domain (see Figure 6). Requirements in the problem domain are requirements that the stakeholders want to achieve through use of the system, with no reference at all to a possible solution. System requirements in the solution domain state how the system will meet the stakeholder requirements, without any reference to a design (see Figure 6: different phases of requirements engineering).

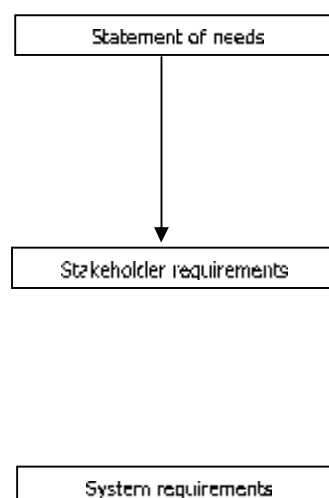


Figure 6: different phases of requirements engineering [69]

2.3.1. Describing the problem

In this section we look at RE as a problem solving activity. The literature mentions a number of RE styles that can be used in this view (see [169]). The literature recognizes goal-oriented RE as a way to look at RE in this context. In Appendix I we provide a review of the GORE literature.

When we only look at the first part of the provided definition, RE is about problem analysis. This view on RE comes from systems engineering and is about investigating and documenting a problem domain [14]. Within this view the requirements engineer describes the experienced problematic phenomena, the relations between these phenomena, why this is seen as problematic and who experiences these problems. Problem oriented RE abstracts from every possible solution type and is therefore suited to use as a RE technique for different types of solution systems.

Wieringa provides us [169] with information about what a Requirements Specification (RS) should contain when RE is seen as problem analysis, a RS in this view describes:

- the problematic phenomena;
- the cause relationships between these phenomena;
- by which norms these phenomena are problematic;
- which stakeholders have these norms

When can therefore use the following definition for a RS [168]:

A RS is a description of the relevant business objective(s) and of the desired way of working to reach these objectives. A RS must not refer at all to the system under development but only to the desired way of working and the business objectives: It only says what work is done and why this is done.

If we look closely at this definition and compare it to the reviewed GORE literature (see Appendix I and [159]) and especially the section where van Lamsweerde describes the needs for goals we argue that goals can be used to specify the needs of a system. We therefore will use goals as a representation of the business requirements discussed in section 2.2), in [159] we can find the arguments why goals are needed.

2.3.2. Describing the solution

As mentioned in section 2.3 the scope of RE does not only lie in problem analysis, but also describing possible solution alternatives. We will again use the work of Wieringa [168].

When using the view of solution specification a requirements specification consists of [169]:

- A specification of the context in which the system will operate.
- A list of desired system functions of the system.
- A definition of the semantics of these functions.
- A list of quality attributes of those functions.

Therefore the definition provided in the introduction section of this document is valid:

A RS is a document that contains the requirements; this specification defines the product and can be used as a contract to build the product.

2.4. RE for bespoke systems

RE for bespoke systems is the classical requirements engineering (see Appendix II for an extensive literature review). In this branch of RE the requirements engineer specifies the needs for a system that will be constructed from scratch. Although most of the RE literature does not mention that it is intended for bespoke systems only, it is assumed the reader can distinguish between the concepts.

If we look at the summarized literature (see Table 2 for an overview), RE for bespoke systems has received a large amount of attention in the scientific literature. The main activities in RE for bespoke systems are elicitation, analysis, negotiation, specification, validation and verification. The end-goal of these activities is to get a complete and correct RS. In order to assist the requirements engineer the literature suggests the use of a number of methods. These methods vary from goal based methods, to scenario based and to a collection of techniques to specify system behavior.

Within elicitation the main techniques are interviews, surveys, group sessions, workshops, goal identification techniques, introspection, viewpoints, and scenarios (e.g. see [34]). Requirements analysis is an extension of requirements elicitation. This is where the requirements engineer tries to refine the requirements and derive additional requirements from the existing ones (see [27]). Techniques the requirements engineer can use are structured analysis (see [13]), goal analysis (e.g. see [155]), problem framing (see [72]), scenario analysis (e.g. see [122]), object oriented analysis (e.g. see [73]) and checklists. Because some requirements are more important than others it is important to prioritize these (see [16]). The main activities here are discussing the requirements with all stakeholders and prioritizing them. Techniques the requirements engineer can use are prioritization workshops, pair wise comparisons, weightings schemes, QFD, etc. The next step is to document the requirements. In this phase the external behavior is specified and the specification is written (e.g. see [27], [174]). Some well known specification techniques are subject domain modeling (ERD), data flow diagrams, state transition tables, context diagrams, goal modeling, etc (see [173]).

	Main Activities	Techniques	Methods
Elicitation	Identify stakeholders, record their desires, identify constraints, structure requirements, identify existing systems and derive requirements from them, investigate architectural requirements	Interviews, surveys, group sessions, process model analysis, organizational documents analysis, goal identification, viewpoints, introspection, scenarios, Volere stakeholder identification, prototyping, protocol analysis, investigate business events	ACRE, I*, F3, ORDIT, FSA, inquiry cycle, CREWS, SCRAM, RESCUE, VORE
Analysis	check for necessity,	Structured analysis, goal	I*, F3, ORDIT, FSA, inquiry

	check for completeness and consistency; check for feasibility	analysis, object oriented analysis, problem framing, scenario analysis, checklists	cycle, CREWS, SCRAM, RESCUE, NYAM
Negotiation	Discuss the requirements with all stakeholders, prioritize the requirements, agreeing on the requirements	EMAP, SIBYL, ROI calculation, pairwise comparisons, 100 dollar techniques, QFD, voting schemes, win-win, Prioritization workgroups	QFD
Specification	Transform objectives into system functions Specify behavior Write the specification	Document writing techniques, modeling techniques (ERD, dataflow, goal modeling, context diagrams, etc)	KAOS, GBRAM, NFR, NYAM
Validation, verification	Review specification, develop prototypes, write manuals, write test cases, perform acceptance tests	Simple checks, document reviews, formal validation, prototyping, scenarios, functional testing, writing manuals	GQM, CREWS, RESCUE, SCRAM, VORE

Table 2: overview RE activities, techniques and methods

2.4.1. Selection of techniques

We have identified five main phases in RE (see [21], elicitation, analysis, negotiation, specification and validation / verification. In this section we will choose activities and techniques discussed in Table 2: overview RE activities, techniques and methods.

2.4.2. Elicitation

Within elicitation the activities stakeholder identification and finding out their desires are key goals within the elicitation phase. Appropriate techniques to use here are plain interviewing techniques, surveys, workshops, scenario creation (using business events) etc. A technique for stakeholder identification is the Volere stakeholder identification technique [2]. This technique has been proven to work in RE tools and is from the well known Atlantic system guild. In the introduction section of this thesis we explained the role of architectural requirements. They are basically described in a document or model containing the principles, organizational goals and requirements imposed from the architecture. Investigating this document during elicitation will also provide additional requirements (see [95]).

2.4.3. Analysis

The main concerns of analysis are to refine the elicited requirements. This can be done through both analyzing and modeling. Some very popular modeling techniques are structured analysis (see [13]), object oriented analysis (see [73]). Scenarios and goal refinement trees can be used to analyze the initially elicited requirements (see [155]). Goal analysis is a promising technique to use because goals and business requirements are largely interchangeable. If non-functional requirements need to be analyzed the NFR framework is a well known and accepted method.

2.4.4. Negotiation

The negotiation phase contains three main phases that are important. Requirements discussion, requirements prioritization and agreement over the requirements are the steps involved. A few suitable light-weight easy to use techniques are the weighting;

scoring / voting techniques (e.g. see [16]). These techniques are highly accepted and easy to use. For a more heavy weight technique both QFD and win-win (see [24]) are accepted techniques to use.

2.4.5. Specification

Globally speaking a specification contains all the results from the previous phases. The first set of activities are, specifying the functional, non-functional and constraints.

In specifying external behavior system requirements need to be transformed in system functions. A specification should also contain a context diagram and a data diagram (see [169]). We recommend the Volere template to specify the requirements. This template is both recommended by [149] and [170].

2.4.6. Validation / Verification

For validation the easy to use techniques like document reviews are suitable to use. If more detailed validation is required scenarios or scenarios combined with prototyping are well accepted techniques. For verification the easiest and best way is to organize an acceptance test.

2.5. RE for COTS selection

Requirements Engineering for COTS is different than classical requirements engineering (see Appendix III for a detailed literature review). Much less emphases is placed on requirements elicitation and specification. The requirements engineer elicits a rough set of requirements and then looks at the market to find suitable products. After he identifies the products he analyzes them to compare them to the elicited requirements. This analyzes is also used to identify additional requirements. When a number of products have been identified the requirements engineer must select the best option. We have discussed a number of methods and techniques and we will summarize them in the table provided below. All methods argue that the requirements specification should be compared to the product features. Most methods use some sort of goal-oriented analysis. The methods only differ in how and how detailed the comparison of the wants to the offered products. The evaluation criteria can be either functional based or non-functional based and the requirements should be organized according to their desirability. Table 3 summarizes these methods.

	Re phases	Main goal	Core Techniques	Advantages	Disadvantages
OTSO	None	COTS selection	Evaluation and Analyzing through well defined criteria	Thorough, Proven	Disregards RE
PORE	elicitation	Refinement of COTS product options	Knowledge engineering, MCA, requirements elicitation	Based on empirical research, uses proven RE techniques	Research prototype, ill defined compliance process
CARE	Elicitation	Map system goals	Goal oriented	Based on existing	No systematic

	, analysis	to component goals	(nfr i*, agent oriented, knowledge oriented	RE methods	mismatch process
CRE	Elicitation analysis	Select COTS products based on non-functional qualities	Goa oriented (NFR framework)	Uses well known RE techniques	Very time intensive
CMT	Elicitation / analysis	Resolve selection conflicts	Goa oriented, utility oriented, quality models	Very systematic, use of metrics	Not empirically tested

Table 3: overview COTS selection methods

2.5.1. Selection of techniques

In RE for COTS selection the specification activity has been replaced with a match-making process. Therefore we suggest using well known techniques from the elicitation and analysis phases (see [106]). We already established that interviewing stakeholders using structured interviews, surveys, group sessions etc is a fine way to elicit initial stakeholder requirements.

Both goal analysis and scenario analysis can be used as well to refine the initial requirements set.

After eliciting an initial set of requirements they need to be prioritized. Prioritization is important, because COTS solutions are developed for a general public and most likely will not meet the requirements exactly (again, see [106]). Techniques that we can use are the same techniques we suggested in chapter 2.4.4.

After identifying the initial requirements the market must be scanned for possible ready to use solutions. Techniques that can be used are, searching in house libraries, search the internet, search magazines and journals, visit trade shows and contact vendors (e.g. see [81]).

After identification of possible candidates the requirements engineer needs to analyze these candidates. Techniques that the literature suggests are, multi criteria analysis, analyzing quality aspects of the software, testing the solution and conflict analysis (e.g. see [3, 4, 5]).

Because a standard build product most likely does not fit the needs exactly there are mainly two options. Firstly adapt to the product or negotiate a change. Techniques that we can use are standard negotiating techniques or business engineering techniques to adapt.

Aspect	Techniques
Initial requirements Identification	Stakeholder surveys, interviews, scenarios, goal analysis, etc
Prioritization	Pair wise comparisons, 100 dollar techniques, voting schemes, win-win, prioritization workgroups, etc
Product identification	Searching in house libraries, search the internet, search magazines and journals, visit trade shows and contact vendors.
Product analysis	Multi criteria analysis, analyzing quality

	aspects of the software, testing the solution and conflict analysis.
--	--

Table 4: cats techniques

2.6. RE for services

This section will provide conclusions about the service oriented RE literature. A review of the SORE literature can be found in Appendix IV.

Requirements engineering for services does differ for both service consumers and service providers (e.g. see [153] [61]). The requirements engineer for a service consumer is responsible for identifying the *functional aspects* the service needs to address, or in other words, the needs the service needs to address. This comprises identifying needs and addressing these needs by selecting and orchestrating already available services (e.g. see [8]. If the available services are not there, the requirements engineer could contact the service provider. The focus of these activities is in determining what services are needed in the organization and help the organization adjust to these services. Therefore the requirements engineer is not only responsible for determining the functional needs, but also aligning the processes to the offered services (see [153]). Both [35] [8] mention that services need to address the *business requirements*; therefore we see again a certain goal oriented way of working.

In the case of a solution provider, the requirements engineer is responsible for the service functionality, service design [153] [185] and the activities concerning how clients can access the services (i.e. making service level agreements, defining policies, etc) [61]. The focus lies on determining the need in the market, to create a service that is applicable to exploit in multiple organizations. Therefore it seems that the requirements engineering activity for service providers, in the case of an outsourced service provider, should start with some sort of market scan to determine the needs for services in general [61] [111]. These activities describe the functional requirements for the service for multiple organizations. Remember, a service provided by an external service provider should be developed as general and standardized as possible. The next step is to functionally design the service. This design is suitable to use for the designers of the service [153] [78, 132] [177]

The process of specifying is less project-like than requirements engineering for custom systems. Business objectives and goals gradually shift over time and services need to be adjusted accordingly [8] [153] [61] [76]. One way of measuring the performance of services is through the use of key process indicators (KPIs) within the process models [8]. A number of authors mention that is required to leverage service-oriented computing to a mainstream practitioner's level. To bridge the gap between high-level business services and lower-level software services. This requires an accurate capture and representation of business services at the business processes level [184] [12] [132] [78].

The non-functional requirements are hardly discussed in the literature, but these probably will discuss the availability, performance, security and reliability of the service. The output document is a service level agreement that contains what the service should do and how well it should be done (e.g. [153] [61]).

Another reoccurring event is to use viewpoints for the RE specification and service identification. Another important aspect of the RE process is the continuous monitoring of the service. The service provider and the consumer need to define metrics that they both agree upon. It is the task of the consumer to provide the provider with results from these metrics. If we compare SORE to RE for COTS and traditional IT services, we can see that the outline of the SORE process behaves as a combination of RE for COTS development and IT service management. The service provider invents requirements based on market demands, but after service delivery he is responsible for maintaining the service performance.

Method	Analysis Techniques	RE phases	Main Goal	Perspective	End result	Advantages	disadvantages
SREM	Goal driven	Elicitation/Analysis	Service identification from legacy systems	provider	Identified service	Complete service elicitation method	Unclear, untested, naive
Intentional Perspective	Goal driven	Elicitation/Analysis/ Specification	Service identification and design	Provider	Identified service and behavior design	Reasonably complete approach	Untested
TROPOS	Goal driven	Elicitation/Analysis	Service identification and design	Provider	Identified Service and design	Reasonably tested approach	Not service specific
Service Blueprinting	-	Design technique	Design the service	Provider	Service design	Common service (in general) design approach	Not yet proven to be of any use in RE/SOA
BPM	Process Analysis	Elicitation/Analysis	Specify, analyze, (re)design processes	Consumer Provider		Proven and popular technique	Not RE specific
IBM / SOAD/SOMA	Goal + Process driven + stakeholder interviews	Elicitation/Analysis	Method for SOA development cycle	Provider	Service specification requirements, service and design	Different viewpoints to identify services	Based on white papers, less scientific validation
RE for e-services	Evaluate, goal oriented, process analysis	Elicitation/Analysis/ Specification	Understand/Analyze e-service + develop service blueprint	provider	Service Specification	Viewpoints that describe the needs, tasks, IT	Untested

Table 5: overview SORE methods

2.6.1. Selection of techniques

In service oriented RE we do not only investigate and specify the requirements for a service, but also perform service identification and matching. Therefore we see three groups of techniques (see Table 6). Classical techniques to derive functional, non-functional requirements can be used again. Examples of these are stakeholder elicitation, scenario usage, goal analysis, standard prioritization techniques etc. But because of the different philosophy behind the service paradigm we also need to identify services and compose them into a composite service. A straightforward

technique to do this is to analyze the business processes and group them into service specific tasks.

Service specifications can be made by using a combination of techniques, firstly from requirements engineering viewpoints can be used to specify the service (see [185]). Techniques we can use in these viewpoints are value specification techniques (e.g. value chains or e3 value analysis). E3value might be a promising technique, and BiZZdesign is already investigating this technique for general purposes. Technical viewpoints for a service can be realized using BPEL and WSDL techniques.

Another aspect is that the requirements engineer is responsible for the aspects surrounding a service. Some cases describe adapting to delivered services by process redesign, developing the SLA, etc.

Monitoring is also seen as an active component in refining the service requirements. Techniques from services marketing include, gap analysis, benchmarking, performance measurements (e.g. see [113])

Main task	Selected techniques
Determine functionality	Standard elicitation techniques, scenarios, goal analysis, business process analysis, market scans, value analysis,
Design	Value specifications, process design, BPEL and WSDL descriptions, service composition
Control aspects	SLA specification techniques, monitoring techniques, gap analysis, benchmarking, performance measurements

Table 6: overview techniques

2.7. Requirements Traceability

When we analyze the reviewed literature on RT we distinguish a number of different traceability types (see Appendix V). The most common distinction is that of pre-RS / post-RS and forward/backward traceability (see [57, 60, 59, 58]). This means tracing the requirements back to either their stakeholders or to the design implementation. . Forward traceability is tracing the requirement to the resulting requirements or artifacts and backwards traceability is tracing from the requirement to its originating sources. A third distinction between traceability types is tracing between requirements and tracing between a requirement and other artifacts.

In order to realize pre RS traceability we will need to include traceability in the RE process. To realize this we will provide guidelines in our method to suggest what information should be recorded and when. Another important distinction is the different usages of traceability. Pre RS traceability can only be realized through high-end usage of traceability. High-end traceability focuses more on capturing design rationale and design decisions. Another very important requirement to enable proper traceability is tooling support. Tooling should not only be used to create advanced versions or cross-references and matrices but a tool should supplement the entire RM process [124]. An example of such a tool is PRO-ART [116].

2.7.1. Requirements traceability models

We will provide two conceptual models for traceability in RE. These traceability models will be used in the RE process in the "way of working" section. These models will focus on the traceability concepts introduced earlier in this section, these models describe requirements traceability from stakeholder to the requirements specification and where requirements are allocated in the solution systems. These models are based on the work of Ramesh and Jarke [125] and refined using the work of [66].

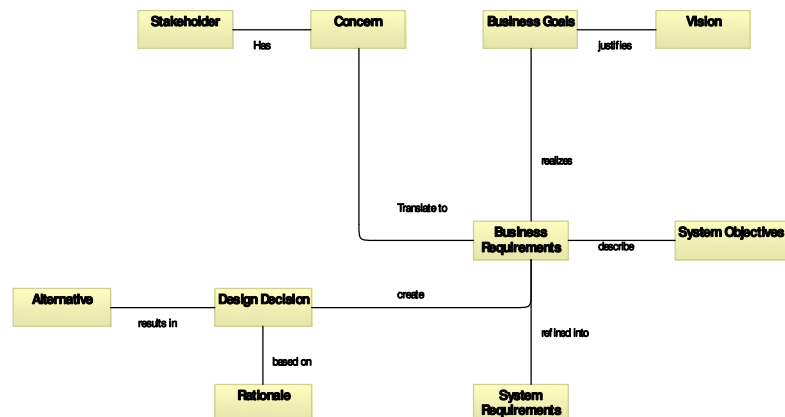


Figure 7: conceptual model (adapted from [125], refined using concepts from [66])

This model can be explained as follows. The *business requirements* describe what should be done to address the *system need*. *System requirements* are a refinement in terms of solution alternatives to realize the *business requirements*. *Requirements* can generate *conflicts* with each other and *design decisions* are needed to resolve these conflicts. Design decisions are based on *rationale*. *Stakeholders* have a certain interest or concerns with requirements and make decisions to resolve conflicting requirements.

The second model describes the design allocation mappings of requirements. This model is only applicable when we decide to use software based solutions. In this model the requirements represent the centre of all relations. *Requirements* map to *system functions*, whereas *system components* satisfy requirements and requirements are allocated to system components. Requirements are used to *drive* the *design* of systems.

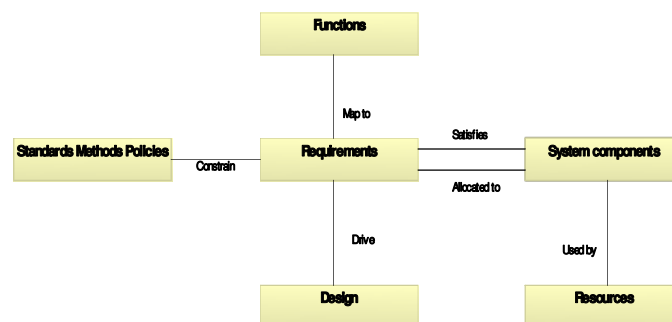


Figure 8: Requirements allocation model (adapted from [125])

2.8. Requirements Management and Enterprise Architecture

When we compare the different frameworks with each other (see Appendix VI for a description of these frameworks), especially TOGAF 8.1 and IAF (see [149][102]), we can distinguish explicit RM activities. Both frameworks have in common that they refine requirements from business requirements to more concrete system requirements. The IAF even provides general guidelines about what information to capture to create a limited form of requirements traceability.

According to IAF business requirements answer the why question. These describe the added value to organization. In TOGAF we find the business requirements in the architectural vision phase. Business requirements should not be mistaken with requirements for business systems at the business architecture. In the business architecture we find solutions describing products, services, processes that contribute in realizing the business requirements.

When we analyze the different phases in IAF that describe the requirements we see an interesting recurring pattern. It begins with identifying the business requirements, answering the why question. This can be seen as problem investigation. After this phase they transform the requirements into more concrete system requirements. System requirements are part of solution designing. They then move to solution alternatives. These are solutions that provide similar functions and they try to selection which solution fits best.

Basically this is the design phase from the engineering cycle (see introduction). The engineering cycle is about describing and investigating the problem, find a solution for this problem and determine how well this solution actually solved the problem. We can distinguish the following phases within the engineering cycle [170]:

- problem investigation: What is the problem
- Solution design: Which solution alternatives are available
- Solution validation: Which alternative best solves the problem
- Solution implementation
- Solution evaluation: How well did it solve the problem?

IAF clearly contains the first three steps of the engineering cycle. If look close at TOGAF and IAF how they deal with requirements, we can conclude the following:

- Requirements start from high level goals and business requirements.
- Requirements need to be refined into system requirements for solutions.
- This refinement becomes more concrete when it traverses the business, application and technology layer (also described by [18], but in too little detail to grant it its own section).
- The engineering cycle can be used as a control framework for RE in EA.
- Traceability is realized through recording rationale, conflicts, decisions and alternatives.

2.9. Requirements for a RE method

In this section we will derive a number of requirements for our method. These requirements are based on the result of the literature study about RM and RE related to EA. We will start with providing general requirements for this method and refine these into more detailed requirements.

2.9.1. General requirements

Below we will summarize the general requirements for the method. These requirements are based on our view on RE and the results from the literature reviews.

- Include the aspect that RE is both about problem investigating and solution specification [169];
- Show the differences between RE for bespoke systems, service-oriented RE and RE for COTS selection (see Appendixes II,III,IV);
- Show how traceability is realized from problem investigation to solution specification;
- Method should be usable independent of EA, but the relation between the two must be shown;
- Use a framework to abstract from the different process variants;

2.9.2. Requirements for the overall framework

In the discussed frameworks for EA we can already see a general implicit RM method. All the discussed frameworks provide us with a way to discuss why. A good way to answer the why question is to provide business requirements. Business requirements are high level goals that describe the need for a certain system. The Zachman framework does not mention business requirements, but in TOGAF and IAF this distinction is explicitly made.

- The method should begin with higher level requirements and work its way down to more refined system requirements;
- The concepts of business requirements and system requirements should be made clear;
- The method should be as solution type independent as possible (possible to specify requirements for business solutions and the systems that support it);
- The overall framework should work with the concepts described in Figure 7;
- The framework should lead to a design decision where alternatives are rejected and a final solution is chosen;

2.9.3. Requirements for RE for bespoke systems

RE for bespoke systems does not need an extensive requirements list for our method. This process is well described in the literature; we can summarize the work of existing models, methods, etc.

- Highlight the importance of the specification;
- Include the activities, elicitation, analysis, negotiation, specification, validation and verification
- Describe these activities;

2.9.4. RE for COTS selection

Because the literature for RE does not mention a process model for COTS selection (at least one where all agree upon) we propose our own process model for COTS selection. We reviewed the existing literature and methods to determine the main concepts used.

- Use well known techniques for initial elicitation;
- Highlight the matchmaking aspects of COTS selection;
- Include prioritization;
- Use the market place to refine the initial requirements;

2.9.5. Service-oriented RE

The literature about service-oriented RE is even less complete than the literature about COTS selection. Therefore we have analyzed a number of service-oriented development methods and RE methods to derive a number of requirements.

- Make a distinction between the service consumer and the service provider;
- show the interactions between the service consumer and the provider;
- show the different nature of service consumer oriented RE and service provider oriented RE;
 - consumer oriented RE focuses on determining service need, service identification, monitoring of existing services and adapting to delivered services;
 - provided oriented RE focuses on developing generalized services based on market demands and adjusted according to changing market needs;
- End result is a SLA;
- SORE has the properties of both COTS oriented RE and IT services management. The SORE process models should highlight these.
- Use viewpoints for the service specification;
- Include the value aspect in the specification;
- Use business processes to bridge the gap between business services and software services;

2.9.6. Requirements Traceability

In this section we will describe the traceability requirements. These are based on an extensive review of the RT literature and Figure 7, Figure 8.

- Incorporate traceability explicit in the method [116] ;
- Show the concepts of pre specification traceability (rationale, conflicts, decisions and alternatives) [57, 60, 59, 58] ;
- Show the concepts of post specification traceability;
- Show forward and backward traceability [57, 59] ;
- Traceability can only be realized through method and tool support [124] ;
- Traceability should be based on the concepts introduced by [66];

2.10. Conclusion

In chapter 2.9 we presented a number of requirements for the parts of the method we should design. We have separated the requirements into five different categories. We have chosen these categories on the results from the literature review.

The goal of the method is to guide practitioners through the requirements engineering process and suggests what information to record. We will also select a number of techniques that are well known in the literature and easy to integrate in the BiZZdesign portfolio of methods and tooling. In the next chapter we will design our method based on the requirements from section 2.9.

3. A RE METHOD

In this chapter we will describe our RM method. The method will be based upon the conclusions drawn in chapter 2.9. The method will start with specifying business requirements and refine it into system requirements. We will provide guidelines for specifying business requirements using problem-oriented RE techniques. Guidelines for system requirements will be based on techniques from RE for bespoke systems, RE for COTS selection and service oriented RE. The method itself will provide guidelines how to realize traceability and the engineering cycle will be used to provide us with a control framework.

The main goal of this chapter is to describe how a method for RM could look like; the actual method will be discussed in the forthcoming handbook and will use the results of this design study as a basis. This chapter will discuss what the steps within this method are, how these steps look like and how traceability can be realized throughout this method.

In section 3.1 we will provide a conceptual control framework to organize our method. In section 3.2 we propose a process model for getting the business requirements; in section 3.3 we describe how to specify the desired solutions and in section 3.4 we show how to validate the requirements.

3.1. A RE framework

In order to develop a method that does not make a distinction what solution we want, a control framework is needed. We have chosen the engineering cycle as our conceptual framework for our method. In the proposed method we will only focus on the first three steps of the engineering cycle [80], which is known as the design cycle. We have extended these first three steps accordingly to the IAF and TOGAF frameworks. We have also argued that RE is both about problem investigation and solution specification and that these views come together in (enterprise) architecture. Therefore we have designed a method that starts with problem analysis; these problems are translated into business requirements. These business requirements need a solution and therefore a solution investigation is required. The next step in this method is solution specification. In this step the *business requirements* are made more concrete, accordingly to [8] [166] (See Figure 9: framework for RE method).

This step takes the view that RE is about specifying desired behavior or a solution. In this step we design the system requirements [84]. After this step the solution needs to be validated and if it did not solve the problem a new iteration is required. Although we explicitly separated problem-oriented RE and solution oriented RE, in practice this distinction is probably not this strict [170]. We have used this way of thinking to structure our method.

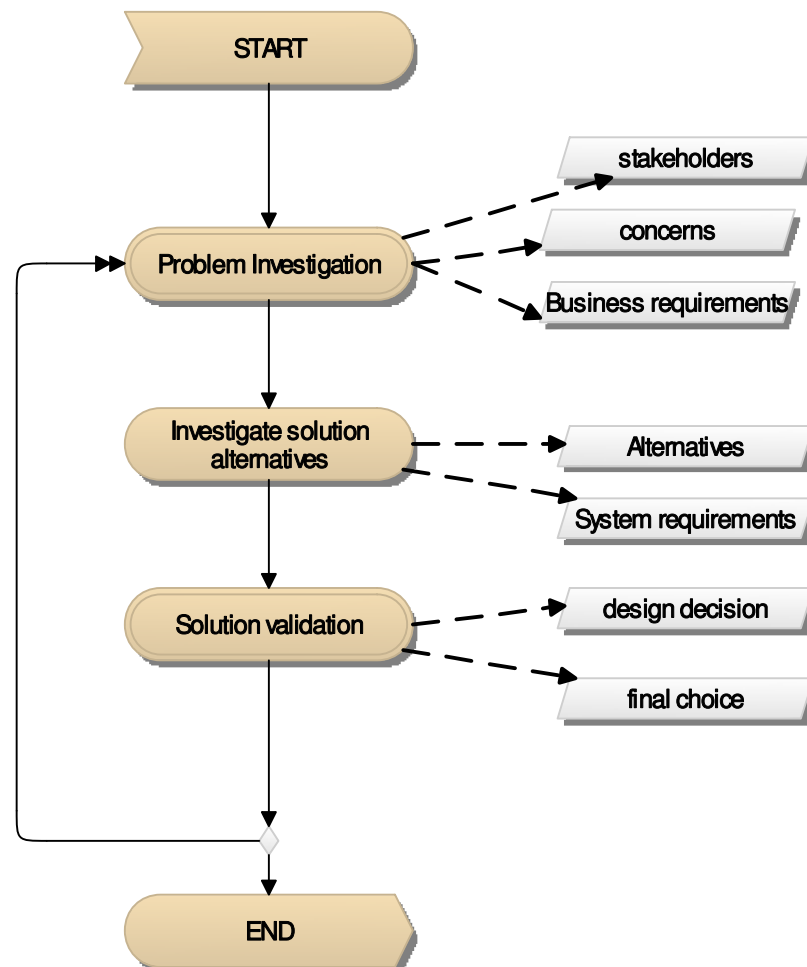


Figure 9: framework for RE method

This framework acts as a control framework because it provides a structured way of working for the different RE types. Requirements engineering for different solution types require different steps (e.g. the nature of these types are different from each other and therefore require a different process model). A second argument why this is a control framework is that this framework provides steering. It has a pre-defined set of steps and contains a feedback loop to re-initiate the RE processes. In short, it provides process guidance and links the business requirements (the objectives) to the system requirements (the behavior).

The process model described above can also be interpreted as a spiral model [71]. In this spiral model we start with investigating a business problem and end with a proposed solution, every cycle of the spiral model refines this solution into a more concrete and usable solution.

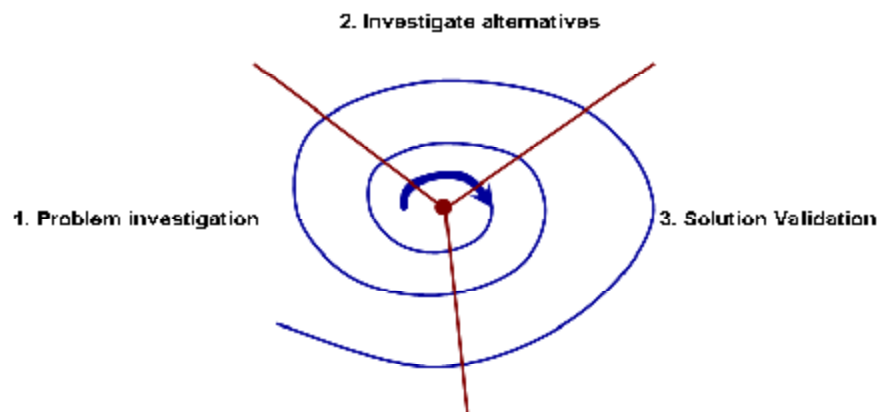


Figure 10: spiral model of the framework

3.1.1. Problem investigation

In this chapter we use RE as a problem investigation technique, we use the standard RE process model to guide us through the steps of problem investigation.

In this model we identify the stakeholders, find out what their desires are, document their desires and validate these desires. The end result of problem investigation and specifying the desired effects are the business requirements. These requirements describe what should happen to solve a certain problem [165], describe the organizational goals, etc.

Since we use the concept of problem chaining, this step can be executed multiple times and every iteration leads to a more concrete desired effects (i.e. for every specified solution it is possible that we need to refine that solution into multiple solutions, therefore we need to investigate the problems that arise from that solution so we can solve the next step).

For traceability purposes we recommended to record the stakeholders, their concerns and the priority of their concerns and the business requirements with their priority. This is based on the work provided by [66] [125] [102]

3.1.2. Investigate solution alternatives

In the previous step we have identified the problem and their stakeholders, and specified the desires that should solve the problem. In this step we start to look for possible solutions that are available to solve our problem. In thesis we only treat specifying solution alternatives concerning business processes, business services or software based systems like application services, tailor made software, commercial software or application services. To realize traceability in this context we have to record the design decisions made, who was responsible for this decision, alternatives, rationale where it was based on, etc. According to [167] solution specification is an important activity during this phase. In section 5.3 we propose process models for specifying solutions.

In the previous chapter we have described a way to specify the desired needs to solve a certain problem. In the next chapters we will describe how these desired needs are fleshed out into more detail. Remember, although we explicitly written

down the steps of problem analysis and solution specification sequentially, in practice this does not need to be the case. In this chapter we provide similar process models (alternatives to the process model used in the problem analysis phase), but they explicitly focus on solution specification. It is entirely possible that the problem analysis and solution specification happen at the same time (probably when the same person is responsible for both problem analysis and solution specification).

3.1.3. Solution validation

In the solution validation phase the different solution alternatives are compared and analyzed. The main goal is to determine which solution best implements the business requirements [102]. IAF [102] suggests using scenarios and prioritized requirements to determine which alternative fits best. In chapter 3.4 we will describe a validation approach with recommended techniques.

3.2. RE approach for problem investigation

In this section we will describe how to find and analyze the business requirements (see Figure 11:). The business requirements serve as a why for the system requirements [102] [182]. These are the high level goals and objectives that belong to the architectural elements. We have used goals as the representation as business requirements. The definition of business requirements itself (as explained in chapter 2.2) already suggests that a goal notation is applicable to use. Business requirements provide the why for systems. According to [89] [101] [159] goals provide an excellent way to explain the why.

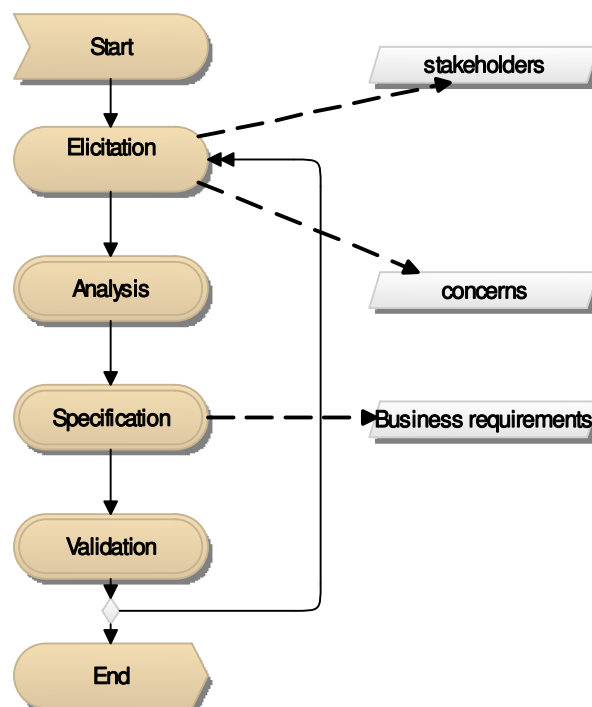


Figure 11: business requirements approach

3.2.1. Elicitation

The main goal of eliciting business requirements is to identify the *why* for the system requirements [102][149]. During this phase we want to find stakeholder goals, organizational goals, etc. We will also investigate the architectural requirements. The architectural requirements document contains requirements, principles and organizational goals for the relevant RM project.

3.2.1.1. Activities

- stakeholder identification;
- Interview stakeholders and record their concerns [159];
- Investigate organizational goals and objectives [10] [156] [159];
- Investigate architectural requirements document [95];
- Investigate business processes [10];
- Use business scenarios to refine the concerns [10] [156] [159] [133];
- Record stakeholders and their concerns (based on [66] and RT literature);

3.2.1.2. Techniques

These techniques have been selected according to suggestions of the reviewed work from [53], [102] [141, 149] [108] [159].

- Volere stakeholder identification [2];
- Baseline stakeholder identification [137];
- Goal identification;
- Interviews;
- Surveys;
- Business scenarios;
- Process inspections;

We will at least use a combination of goals and scenarios as suggested by [133] [121]. Scenarios are used to elicit more goals. Business processes also provide us with context and goals.

3.2.2. Analysis

Elicited concerns can still be vague, incomplete or just incorrect. In this step we will refine these activities. We will check for completeness, necessity and correctness.

3.2.2.1. Activities

- Check for completeness, necessity and correctness;
- Refine goals and concerns;
- Prioritize concerns;
- Transform concerns into business requirements;
- Record analysis results (goals, rationale, decisions, priority of requirements)

3.2.2.2. Techniques

- Goal refinement trees;
- Business process modeling techniques;
- Business scenarios;

- Prioritization techniques;

3.2.3. Specification

Specification is about documenting the business requirements. Wieringa suggested [169] that a specification should contain the business objectives and how these business objectives are reached. A second element of this specification is: the stakeholders and their concerns. We will therefore specify the business objectives (our goals) and how we are going to reach these goals (refinement of goals)

3.2.3.1. Activities

- Specify stakeholders and their concerns;
- Specify business requirements;
- Specify constraints;
- Specify non-functional business requirements;
- Record all process traces, models created, rationale behind decisions, conflicts, priority, etc;

3.2.3.2. Techniques

- Guidelines about writing requirement documents [84];

3.2.4. Validation

Business requirements validation is about checking if the recorded requirements have been correctly specified [27].

3.2.4.1. Activities

Firstly to check if the written requirements are written down correctly without any ambiguity then discuss them with the stakeholders [27].

- use simple checks to review the requirements specification;
- use document or design reviews to discuss the requirements with stakeholders within the project to receive constructing criticism;
- discuss business scenarios to validate the goals;
- compare specification to organizational documents, etc;

3.2.4.2. Techniques

There are a number of techniques that can be used for validation [27], [83].

- Interviews;
- Workshops;
- Scenarios;
- document reviewing techniques;

3.3. Investigating solution alternatives

In this section we will describe the RE process for system requirements. In this thesis we will either use RE for services, RE for COTS selection or the classical RE for

bespoke systems. The main goal of this phase is to investigate the different solution alternatives and create specifications for these [167].

For each variant we propose a process model. During each step of the process models we propose activities and techniques that can be used to gather and specify the requirements. The goal of this section is to provide process models and guidelines for the different possible solutions within an EA.

To try to integrate these different solution alternatives we provide a framework (see Figure 12) to RE in this context. In general the requirements engineer has three options. He either wants to specify the functionality of a custom system, he wants to select a commercial off the shelf system or specify a service (either at the consumer side or the provider side). Solution alternatives can occur at different levels. The requirements engineer can either specify for buy or make, leading to a design decision between either COTS selection or specification of a bespoke system. It is also possible to find solution alternatives when we chose to specify a bespoke system. We can think of alternative specifications where in each instance the product contains different functionality or quality requirements.

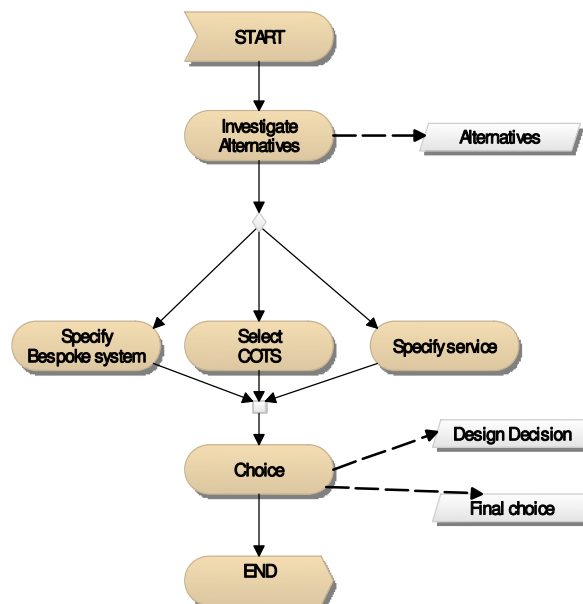


Figure 12: process selection framework

3.3.1. RE process for bespoke systems

After analyzing the literature we can derive the following process model for RE when we take the viewpoint that we are specifying custom made systems. This is based on the models proposed by [21] [83]. As mentioned in the traceability section of this thesis we will incorporate RT within this process model. We will describe what information to capture to realize pre-RS traceability.

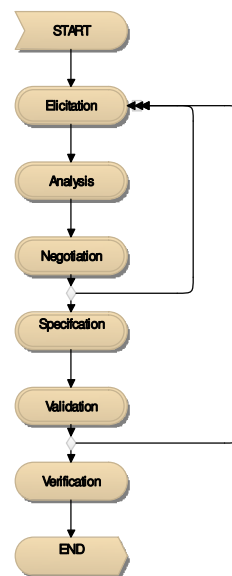


Figure 13: RE for bespoke systems

3.3.2. Elicitation

The goal of elicitation in this context is to elicit the system requirements. These system requirements should be traceable to business objectives and to system components [34].

3.3.2.1. Activities

- Stakeholder identification;
- Elicit requirements from stakeholders;
- Elicit requirements from the business requirements;
- Investigate architectural requirements document [95];
- Determine constraints;
- Create scenarios that describe system usage;
- Structure requirements;
- Record process traces (stakeholders, rationale, interview transcripts, scenarios etc)

3.3.2.2. Techniques

Within the literature we can distinguish a number of techniques [53, 74]

- Volere stakeholder identification [2];
- Interviews;
- Surveys;
- group sessions;
- scenarios;
- prototyping;
- business process inspection;

3.3.3. Analysis

During the analysis phase we will refine the elicited requirements, check for necessity, completeness, consistency, etc.

3.3.3.1. Activities

The main activities within analysis are:

- review requirements for necessity, completeness, consistency and feasibility;
- Analyze scenarios;
- Create models for further analysis (e.g. data models about the subject domain, context diagrams, use case diagrams, etc);
- Record process traces (link refined and derived requirements to their parent requirement, record created models. They provide rationale for the analysis results)

3.3.3.2. Techniques

Some of the techniques that can be used are:

- goal analysis (see[155]);
- scenario analysis (see [145]);
- structured analysis (see;[136])
- object-oriented analysis (see [73]);

3.3.4. Negotiation

The main goals of negotiation are to resolve conflicts between the stakeholders prioritize the requirements and create a mutual understanding of between the stakeholders about the requirements.

3.3.4.1. Activities

- discuss requirements with the stakeholders;
- prioritize the requirements;
- agree on the requirements;
- record design decisions and rationale (record conflicts between stakeholders, the decisions made, rationale where they are based on and alternatives, and the requirements priority)

3.3.4.2. Techniques

In 0 we reviewed a number of techniques that can be used. We will summarize them below. According to the RE literature these techniques are accepted to use.

- ROI calculation (determine which set of requirements is more favorable in financial terms);
- Pair wise comparisons;
- Prioritization workgroups;
- 100 dollar test (give every stakeholder "100 dollars" and let him divide it among the requirements);

- Voting schemes (each stakeholder is allowed to cast his vote on a number of requirements);
- Weightings(weight each requirement from a specific stakeholder);

3.3.5. Specification

A requirements specification for a system that needs to be build from scratch should at least contain [169]:

- A specification of the context in which the system will operate.
- A list of desired system functions of the system.
- A definition of the semantics of these functions.
- A list of quality attributes of those functions.

When we look at the available specification literature we see that the Volere template comes highly recommended [149] [170]. We will make the above specification description more concrete using the Volere template.

3.3.5.1. Activities

- Specify project drivers. In our case these are the business requirements. We have already identified the stakeholders and recorded their concerns;
- Specify project constraints;
- Specify functional requirements;
- Specify non-functional requirements;
- Specify system context;
- Specify list of system functions;
- Specify data requirements (e.g. subject domain model, UML diagram of the problem domain, etc);
- Record scenario, priority, motivation, rationale, stakeholder of the specific requirements and possible conflicts for traceability purposes. This should be supported by a tool;

3.3.5.2. Techniques

- Document writing techniques like [84];
- OOA or SA for specification models;
- NYAM for the specification models [174];
- Volere requirements specification template [130];

3.3.6. Validation and verification

Requirements validation is checking the requirements for consistency, completeness and accuracy.

3.3.6.1. Activities

The activities here can be divided into two main categories [27]. Check if the requirements are written down correctly and check if the specified requirements are correct. The activities we suggest are:

- Use simple checks to review the requirements specification. Check if the requirements are documented in a readable and understandable manner. See [84] chapter 13 for guidelines.
- Organize document and design reviews with the stakeholders [27];
- Develop a prototype of the system and use scenarios to analyze this prototype [27] [144];
- Write test cases (scenarios, that describe system usage);
- Write the user manual;
- Perform acceptance tests after system deliver;
- Record process traces. If certain aspects where not correct. Record it and why.

3.3.6.2. Techniques

- simple checks;
- document reviews;
- develop a prototype;
- scenario analysis;
- write the test cases;
- write the user manual;
- acceptance testing;

3.3.7. RE process for COTS selection

When RE is used to specify the requirements for systems that are bought off the shelf, the process model is a little different. The focus is less on specifying a complete requirements specification, but more on selecting and discovering requirements and matching them against your needs [3]. This process model is based on a combination of multiple models and provides the essence which all models have in common.

We will use the process model proposed here [3] as a basis.

The first step is still requirements elicitation [3] [106] [4], but this is less important than the traditional elicitation phase. The goal is to derive an initial set of requirements and scan the market for possible products.

The second step is to prioritize the requirements [97] [5]. Not all requirements are just as important and offered products will not always provide a complete match, therefore the requirements engineer needs to have an understanding of the most important requirements.

The third step is to identify and analyze the available products on the market. This is based on [4] and [81]. This step leads to another iteration of requirements elicitation. The identified products provide additional insight and lead to a better understanding of wanted product features [98]. After each iteration, the amount of possible matches will decrease, in the end only a few possible matches remain [98,

97, 106]. One of the possible selection criteria could be selecting on non-functional requirements [4].

When the requirements engineer has chosen a suitable product, two options remain [5]. In most cases the identified product does not completely adhere to the stated needs, therefore one of the options is to adapt the business processes or negotiate a change to the selected product.

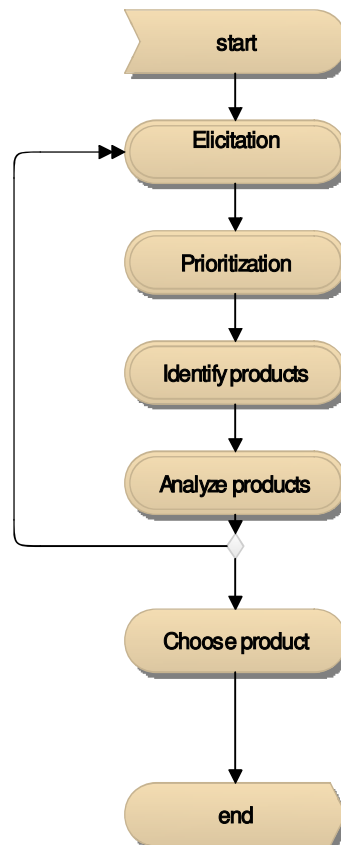


Figure 14: RE for COTS selection

3.3.8. Elicitation

As mentioned in the previous section, elicitation for COTS selection is less important than elicitation for bespoke systems. The idea behind elicitation is to gather an initial set of requirements and scan the market for products.

3.3.8.1. Activities

- Identify stakeholders;
- Analyze business requirements;
- Elicit requirements from stakeholders;
- Investigate architectural requirements document [95];
- Develop scenarios that describe system usage;

- Record process traces (stakeholders, concerns);

3.3.8.2. Techniques

There are a number of techniques the requirements engineer can select from, these are the same as elicitation for bespoke systems:

- stakeholder interviews, surveys, group sessions, card sorting;
- scenarios;

3.3.9. Prioritization

When looking at the market place for commercial-off-the-shelf solutions it is not always possible to find an exact match. Therefore the requirements engineer needs to determine whether which requirements are more important than others.

3.3.9.1. Activities

- Analyze the requirements;
- Use a prioritization technique to select the right requirements;
- Record rationale, alternatives and priority of the requirements ;

3.3.9.2. Techniques

To prioritize the requirements the requirements engineer can use a number of different techniques [50, 16] [25]:

- ROI calculation (determine which set of requirements is more favorable in financial terms);
- Pair wise comparisons ;
- Prioritization workgroups;
- 100 dollar test (give every stakeholder "100 dollars" and let him divide it among the requirements);
- Voting schemes (each stakeholder is allowed to cast his vote on a number of requirements);
- Weightings(weight each requirement from a specific stakeholder);
- Boehm's Win-Win technique (an iterative conflict-goal management technique for stakeholders);

3.3.10. Identify products

The main goal of this phase is to identify the products that can be used as a solution for our problems.

3.3.10.1. Activities

In this phase of the selection process the requirements engineer looks at the market place and identifies a number of products that fit the high level requirements. This phase is driven by evaluation criteria (such as the business requirements, non-functional requirements).

The activities in this step are: [4]

- COTS candidates' identification;
- Clarify COTS;
- Generate list of COTS products;

3.3.10.2. Techniques

Typical techniques for finding products are [81]:

- search in house libraries;
- search the internet;
- search magazines and journals;
- visit trade shows and conferences;
- contact vendors;

3.3.11. Analyze products

The analysis phase has a number of different goals. The first main goal is to assess if the product meets the requirements, the second goal is that through analysis of the products the requirements engineer can find additional requirements [4] [98, 97, 106].

3.3.11.1. Activities

The literature mentions a number of activities that take place in this phase [81][106] [97][6] [3] .

- **Define analysis criteria.** In this step the requirements engineer defines the criteria that he uses to analyze the selected product. These criteria can be based on functional requirements and non-functional requirements. Non-functional requirements help determining when multiple products fit the selected functional requirements (i.e. select on quality).
- **Compare product features to the requirements and analysis criteria.** During the analysis phase the requirements engineer compares the selected products to the selection criteria. It is wise to define patterns to assess in what manner products fulfill the requirements (e.g. a requirement is completely fulfilled, partially fulfilled, provides an additional feature or just completely doesn't fulfill a requirement at all)
- **Analyze non stated requirements.** In this step the product has given the requirements engineer additional insights in possible requirements. He could jump back to the elicitation phase and derive new requirements from this additional product feature.
- **Record their compliance.** In this step the requirements engineer records for traceability purposes the manner of compliance of the different products.
- **Record analysis results.** This phase is an extension to the previous step. The requirements engineer records all results.

3.3.11.2. Techniques

There are a number of techniques the requirement engineer can use in this phase:

- knowledge engineering techniques;

- multi criteria analysis;
- assess the product quality through comparing non-functional requirements;
- write test cases to conduct requirements tests;
- Conflict analysis; determine if certain requirements conflict with product features.

3.3.12. Choose product

The main goal of this phase is to select the product that best fits our requirements.

3.3.12.1. Activities

These activities are based on [3, 4, 5, 81, 98, 97]

- Select product;
- Adapt to product, it is not always possible to get a product that perfectly addresses the requirements. One can think of changed business processes, other work procedures, etc;
- Request changes to the product or adapt the product yourself;
- Use the requirements specification from the developer for the system requirements;
- Record alternatives, record design decisions, record rationale;

3.3.12.2. Techniques

- Business Engineering to adapt to the product;
- Negotiating techniques;
- Multi-Criteria analysis;

3.3.13. Service-oriented Requirements Engineering

One of the most important distinctions between the previous process models and the process model for service oriented RE is that the SORE model comprises of both a model for the service consumer and the service provider (see Figure 15). This is based on the assumption of the different nature of RE for the consumer and the provider [153] [61]. We propose to use the classical RE model for both the provider and the consumer, since the literature does not mention any process guidance. We will then assign the techniques from the reviewed methods in the activities of the process model. We will extend the classical RE process model with both the negotiation and monitoring concepts reviewed from the IT service management literature. Another major difference between classical RE and service oriented RE is that SORE is less project-like than classical RE. In this case RE is more like constantly monitoring the service and adjusting where needed [153] [61] [76]. We also distinguish between when a provider delivers services only to in-house consumers and services to the market place.

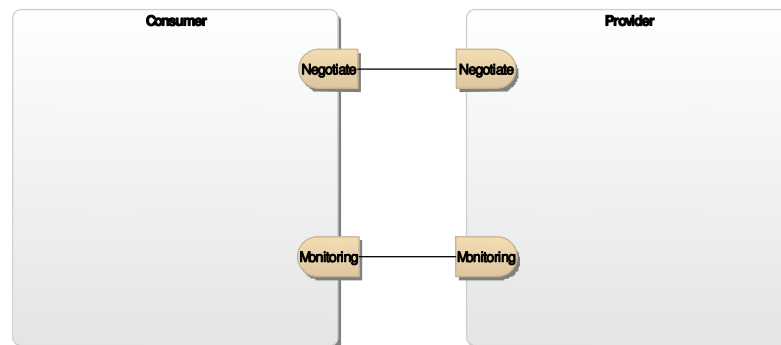


Figure 15: overview service oriented RE

In Figure 15 we see the two interaction moments of SORE. The service consumer and the service provider interact with each other. The interactions are negotiating and service monitoring.

3.3.14. RE process for service consumers

In this section we will describe the activities of the organization when it assumes the role of service consumer (demand side). We will use the classical RE process model, extended with the phases negotiating and alignment. We have extended the classical process model with these two phases based on [153]. The main goal of this process is the elicitation of a composite service. If a requirements engineer cannot compose the composite service of available services he needs to contact the service provider to deliver the additional functionality.

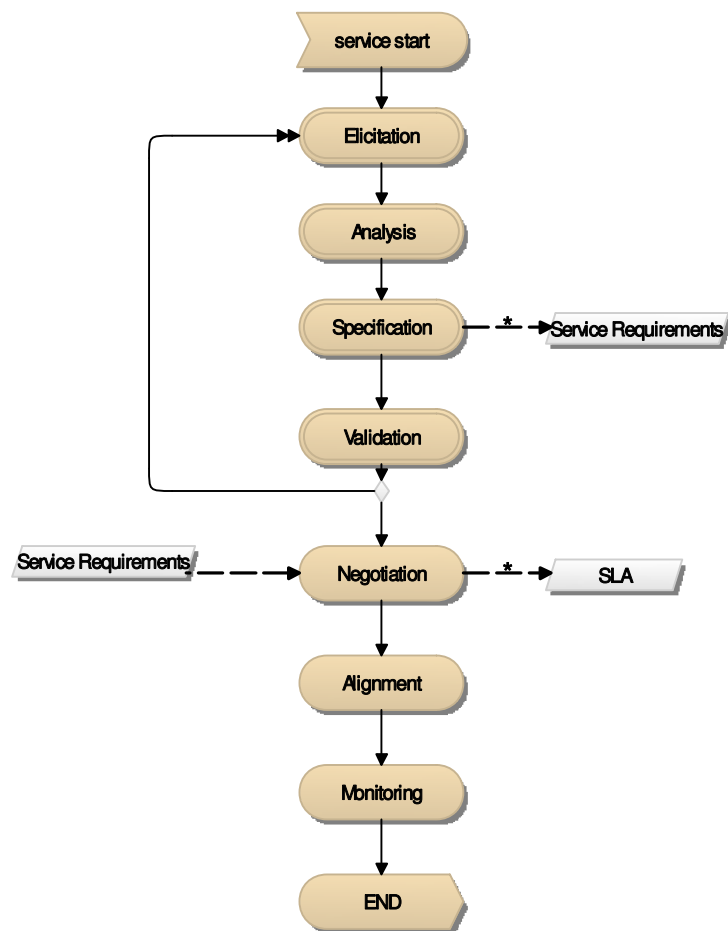


Figure 16: RE at the service consumer

3.3.15. Elicitation

Elicitation for a service-consumer does not only contain eliciting stakeholder needs, although this is still seen as an important activity [94] [184] [12]. In this phase we will identify the needs and we will identify the services. We will use both stakeholder elicitation and scenario usage to scope composite service behavior.

3.3.15.1. Activities

- Identify stakeholders;
- Elicit stakeholder requirements [177] [184] [12];
- Investigate business requirements;
- Investigate architectural requirements document [95];
- Use scenarios to derive requirements [184] [12];
- Record process traces;

3.3.15.2. Techniques

There are a number of techniques the requirements engineer can use to determine the IT need.

- Stakeholder interviews;
- Stakeholder workshops;
- Surveys;
- Business scenarios/ use cases;
- Goal identification / analysis techniques;

3.3.16. Analysis

The analysis phase not only serves the same purpose as in the classical RE process model, but also comprises of service identification and composition. Elicited requirements need to be refined, checked for feasibility and prioritized. We argue that prioritization is an important activity. Just as with RE for COTS selection the service provider might not be able to deliver a service that meets the exact needs of the consumer. An understanding about which requirements are most important helps in the later negotiation process. Another resemblance with RE for COTS selection is that there might be a number of available services that more or less deliver the same functionality. Selection of a service could then be made on the best match or on the best non-functional requirements. See chapter 3.3.7 for more information about component selection.

3.3.16.1. Activities

- check for necessity, completeness, consistency and feasibility of elicited requirements;
- Business process analysis [132] [78] [164] to identify the various activities (each (part of) process provides a service) [77];
- Discuss current services and processes with the stakeholders [111];
- Examine existing legacy applications and group their functionality (to identify if the legacy systems can already provide the required service functionality);
- Model the services with their operations;
- Examine either the service provider portfolio [111] for re-useable services or a service library (registry);
- prioritize the service requirements;
- record process traces (stakeholders, priorities, motivation ,etc);

3.3.16.2. Techniques

- goal analysis;
- requirements analysis;
- business process analysis;
- scenario analysis;
- service library analysis;
- provider portfolios analysis;

3.3.17. Specification

We argue that the RS on the service consumer side is technology independent; it says what work should be realized and how it is realized. A specification should at least contain the composite service and the identified component services with their operations. A Process models can be used to explain these operations in more detail.

One of the models of the specification should also contain the composite service. It should also contain how well we expect the service to perform (non-functional requirements)

3.3.17.1. Activities

- specify functional requirements;
- specify non-functional requirements;
- specify needed services and their operations;
- Specify business process models to provide context;

3.3.17.2. Techniques

Techniques for specifying the needed services are:

- generic document writing techniques;
- Service contract templates;

3.3.18. Validation

The RE literature does not specify distinct validation techniques for SORE. We will use standard RE validation techniques instead.

3.3.18.1. Activities

Firstly to check if the written requirements are written down correctly without any ambiguity.

- use simple checks to review the requirements specification;
- use document or design reviews to discuss the requirements with stakeholders within the project to receive constructing criticism;
- use scenarios to validate the needs
- check if the requirements are compatible with the business requirements

3.3.18.2. Techniques

- Document reviews;
- Scenarios;

3.3.19. Negotiation

The negotiation phase becomes of importance after the requirements engineer has identified a need for a certain service that is not met by current a delivered service.

The main goal of negotiation is to come to an agreement between the service provider and the service consumer [153]. The end result will be a SLA [153] [61]. According to the semantic service literature SLA negotiation can be done completely automatically. Although according to [30] this is not yet the case in practice, therefore we will focus on human negotiated SLAs.

3.3.19.1. Activities

When we combined the work provided by [26] and [150] we can suggest the following activities that should be undertaken in this phase.

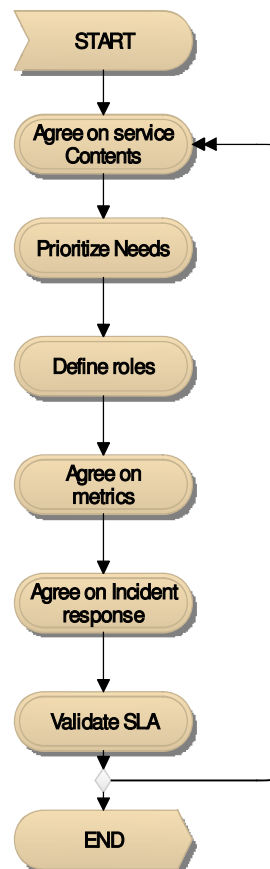


Figure 17: SLA process [26]

- agree on what should be supported;
- prioritize IT needs;
- define roles;
- agree on metrics to measure performance;
- agree on the effort that should be undertaken in case of an incident;
- validate the SLA;
- record process traces, decisions made and why

3.3.19.2. Techniques

- Prioritization techniques;
- Validation techniques;
- Negotiating techniques and skills;

3.3.20. Alignment

According to [153] delivered services do not always address the requirements stated by the consumer. Therefore the service consumer needs to adapt their business processes to match service functionality. Since the literature suggests that business processes need to be adapted, we will use the BPM method used by BiZZdesign in this phase [151].

3.3.20.1. Activities

The main activity in this phase is to adapt to the delivered service [153]. Activities in this step are [151]:

- Analyze the current situation;
- Redesign the current situation;
- Migrate between to the to-be situation;

3.3.20.2. Techniques

Some techniques that can be used are [151]:

- Interviews;
- Surveys;
- document study;
- visual inspection / introspection;
- Workshops;

3.3.21. Monitor service

In this phase the service consumer measures the actual service performance and provides the provider with results from the measurements. Another goal is to identify the need for new services.

3.3.21.1. Activities

The literature at least recommends performance measurements [8]. This can be assigning KPIs to supporting business processes, service availability quality, etc.

- Measure service performance using the pre-defined metrics;
- Communicate results to the service provider;
- Identify new service needs;
- Start a new iteration of the RE process;

3.3.21.2. Techniques

The services marketing literature suggests a number of techniques that are applicable to use in monitoring services [113].

- Gap analysis, analyze how customers perceive the delivered service quality;
- Benchmarking, compare the service quality to similar services in the market;
- Performance measurements (e.g. revenues, market share, marketing costs, overhead costs, profits, ROI, consumer attitudes complaints, etc);

3.3.22. RE approach for service providers

RE for service providers is somewhat different than for service consumers. The main tasks of consumers are to determine service needs and adapt to them. From the service provider viewpoint the main tasks are to develop a service that adheres to market demand and to functionally design a service [61] [153]. In the development context of this method a service can either be a business service to the customer

(the organization itself acts as a service provider) or an application service, where an external company, or the IT department, acts as a service provider. These business services are realized through business processes and application services support these business processes [77]. In this section we will show how to use RE for services and how these and their corresponding business processes can be used as an input for application service discovery.

In order to find activities and techniques for SORE we have not only looked at the RE literature (see [113]) but also the services marketing literature and the IT services management literature.

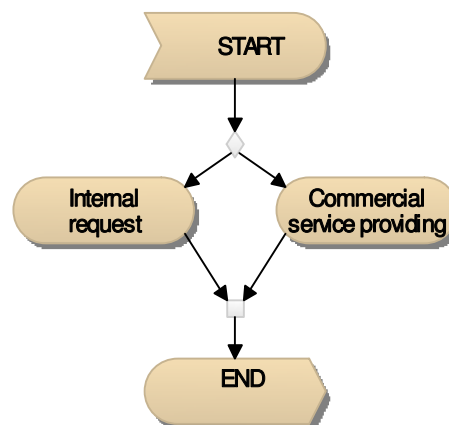


Figure 18: RE for in-house services or externally

In Figure 18 we demonstrate that there are basically two paths for a service provider to choose from (technically a combination of the two is also possible). In our case the service provider can respond to an internal request by a service consumer. In this case the service provider probably receives a pre-defined service specification from the service consumer. This will contain a list of functional and non-functional requirements, perhaps a composite service description, models from supporting business processes, etc. The nature of the RE process will again behave like a match making process. The service provider will check his portfolio if there are services that can be reused in this context or examine legacy systems to check if they can be used to address the required service functionality. If the service provider believes he can provide the requested services, or close enough, he will initiate a negotiating process to with the consumer to reach an agreement about possible service reuse. Another option is that the required functionality is not available and that a service needs to be created. RE in this context will be either solution specification or the provider will just buy a service, or application, of the shelf (see chapter 3.3.7).

When we see RE for commercial service providing (externally located) we can distinguish the following. If we compare RE for service providers with development of COTS products we do see a number of similarities. Both try to develop generalized products for a market. In [3] the author mentions a process model for developing COTS based systems. They start with inventing requirements and then prioritize these requirements. In our process model we find inventing requirements in our

market scan and elicitation activities and prioritization is included in the analysis phase. It is important to notice that RE for service providers is not the same as RE for COTS development [153]. It behaves like a combination of early COTS development but later in the RE process it behaves more like the traditional IT services discipline. A service provider delivers a service and is responsible for realizing the contents agreed upon in the SLA. In our RE process we will emphasize these two characteristics (see Figure 20). Another important distinction is that the service provider can either select a COTS product and provide it as a service or just specify a service himself (see Figure 19). For selecting a COTS product we refer to chapter 3.3.7.

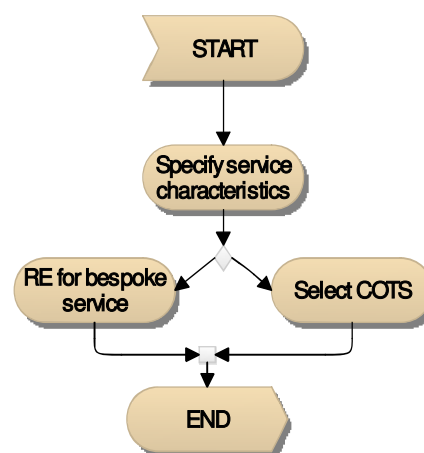


Figure 19: Specifying or selecting

As shown in Figure 19 the service provider has the choice to either select an off the shelf product or create a new one (assuming the need for a new service is identified). Because of the different and intangible nature of a service even after a COTS selection the service provider needs to specify service related behavior (e.g. the added value of a service, service design and realization, service intent, etc). We will argue that the service characteristics will be the basis for the service realization. Therefore our service-oriented RE approach will focus on specifying these service characteristics. We will mention techniques and the way of working for both an internally located provider and an (external) commercial service provider. For an overview of the RE process, see Figure 20.

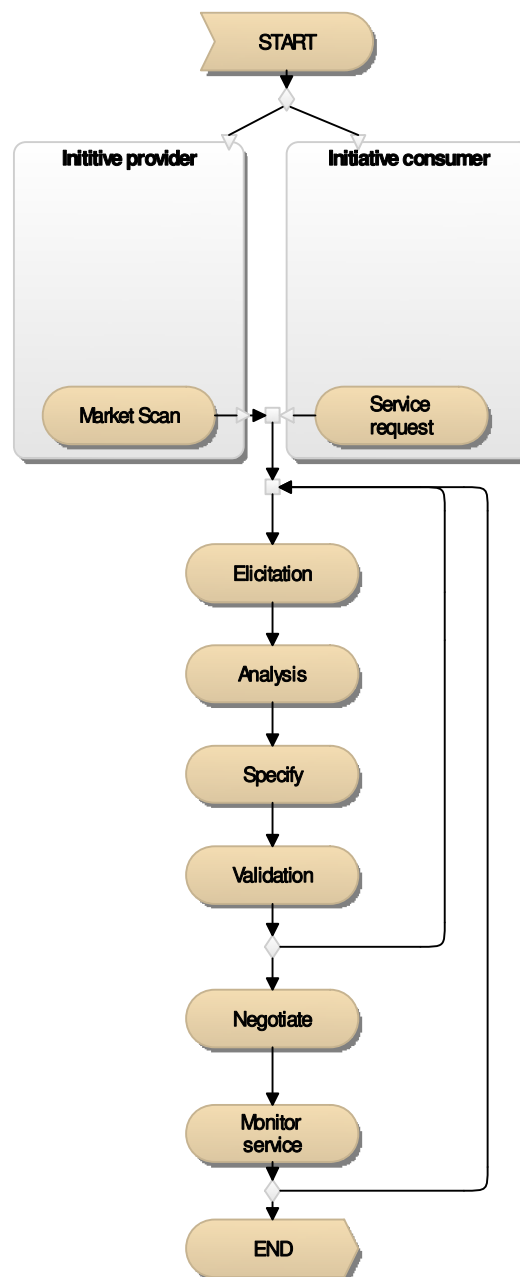


Figure 20: RE at the service provider

3.3.23. Initiative provider

When the initiative in developing a service comes from the provider itself, the RE phase should start with a market scan (as suggested by [61] and [153]) But it neglects to mention what the activities are. We have used services marketing literature to find these activities [113]. This is not intended as a detailed description of identifying suitable markets; it just outlines the basic activities. For more information we refer to [113].

3.3.23.1. Activities

- Define the market; the definition of the relevant market involves specifying the customer group (an important stakeholder) to which the company is seeking to market its services;
- Identify market segments;
- Select target market segments;
- Prioritize market segments;
- Determine characteristics of the market segment;
- Determine key success factors;
- Determine relative importance of key success factors;

3.3.24. Initiative consumer

When the service consumer initiates the request for a service, the RE process is a little different. The consumer has provided a service description and a way of working (the composite service and business processes). The provider then looks at his own portfolio to determine if he has the available services in his portfolio. Another option is to examine the available legacy systems and analyze them to see if they are capable of providing the requested service. We will discuss these variations in more detail in the sections below.

3.3.25. Elicitation

The main goal of elicitation is twofold. In the case of an external commercially operation service provider the results of the market scan are analyzed and serve as an input for the requirements inventing process. The second option is that the provider operates on behalf of a request done by a service provider. The input document will then be the service requirements document written by the consumer.

3.3.25.1. Activities

The main goal of elicitation for providers is to analyze the results from the market scan, organizational goals [8] [132] [78] [88] [114][184] [12] provided by the service consumer and stakeholder goals to determine high-level functionality of the service. In our case the business requirements are the stakeholder goals and organizational goals (provided by the service consumer).

Possible activities for a commercial provider could be (inventing requirements):

- Invent requirements based on the results of the market scan;
- Invent requirements based on organizational goals;
- Invent requirements from stakeholders (customers, internal developers, marketing consultants, etc);
- Develop scenarios that describe service usage to find more requirements;
- Record stakeholders, concerns, goals, etc;

3.3.25.2. Techniques

- Workshops;
- Stakeholder identification techniques (e.g. [2]);
- Interviews;

- Surveys;
- scenarios, use cases;

3.3.26. Analysis

During the analysis phase the initially elicited requirements need to be refined and checked for completeness and consistency. A second important activity is prioritization of requirements. During the analysis phase we also identify services based on the elicited requirements. In the case when we have received a requirements document from the consumer we can try to map the identified services of the consumer to the service portfolio of the provider. The third step is to look for component services already at the provider's disposal to realize the intended service needs. In functionally designing the service value analysis is suggested to be used (e.g. [153]).

3.3.26.1. Activities

- Check requirements for consistency, completeness, etc;
- Prioritize requirements [5];
- Identify component services based on the requirements;
- Try to compose component services into a composite service;
- Business process analysis and design;
- Analyze added value to the customers;
- Analyze service delivery quality (non-functional requirements);
- Record prioritized requirements, analysis results, design decisions, rationale during the analysis, record who was responsible for the decisions made.

3.3.26.2. Techniques

- Value chain analysis (see [113]);
- E3value analysis (see [54]);
- Goal Analysis (see [8] [132] [78] [88] [114][184] [12]);
- Business engineering/BPM (see [151]);
- Scenarios (e.g. see [122]);

3.3.27. Specification

A specification for an application service and a business service deviates a little in their level of abstraction. A specification for an application service also needs to take the technological perspective into consideration. To specify a service we will use multiple viewpoints, as suggested here [55]. Viewpoints [82] are an accepted technique within RE. In order to specify a service we will use the mission statement from services marketing. A mission statement [113] provides the high-level specification of goals of the service and provides the intended service customers. As suggested by [55][113] we also need a value oriented viewpoint. This can either be e3value models for a more formal specification, or just a general description of the added value to the customer. A service blueprint [153] [113] provides the general design of the services. In the case of specifying a business service the blueprint can be passed along to the business processes designer to design the business processes that need to support the business service. When we specify an application service an

information systems viewpoint is needed. We will use BPEL and WSDL for this viewpoint, as suggested in [55].

3.3.27.1. Activities

The main activities in this phase are a combination of service specific specification activities and general requirements engineering specification activities.

Generic specification activities

- Specify project drivers.
- Specify service constraints;
- Specify service requirements;
- Specify non-functional requirements;

Service specific elements:

- Specify a service mission; a service mission describes the high-level goals and objectives of the mission, the intended service customers and service philosophy, etc [113];
- Create value models, either generic value descriptions or more formal models (e.g. e3value models);
- Specify the service blueprint (This specifies the generic design of the service. A blueprint can also be used to demonstrate the business processes design)
- Specify supporting business processes (i.e. the business processes needed to realize the service or the business processes the service supports);
- Specify the service in language application service designers can understand for application services (e.g. BPEL and WSDL);
- Record scenarios, priority, motivation, rationale, stakeholder of the specific requirements and possible conflicts for traceability purposes. This should be supported by a tool;

3.3.27.2. Techniques

In specifying a service we can use the following techniques.

- Mission statement [113]
- E3value [54, 55, 56, 171, 185]
- Value chain models[113] [118]
- Business engineering techniques [151]
- Service blueprinting [113] [183]
- BPEL [55]
- WSDL [55]

3.3.28. Validation

The RE literature does not specify distinct validation techniques for SORE. We will use standard RE validation techniques instead.

3.3.28.1. Activities

Firstly to check if the written requirements are written down correctly without any ambiguity.

- use simple checks to review the requirements specification;
- use document or design reviews to discuss the requirements with stakeholders within the project to receive constructing criticism;
- use scenarios to validate the needs
- check if the requirements are compatible with the business requirements

3.3.28.2. Techniques

- Document reviews
- Scenarios (in combination with e.g. prototypes, stakeholder discussions, etc)

3.3.29. Negotiation

The main goal of negotiation is to come to an agreement between the service provider and the service consumer [153]. The end result will be a SLA [153] [61]. According to the semantic service literature SLA negotiation can be done completely automatically. Although according to [30] this is not yet the case in practice, therefore we will focus on human negotiated SLAs.

3.3.29.1.1. Activities

When we combined the work provided by [26] and [150] we can suggest the following activities that should be undertaken in this phase.

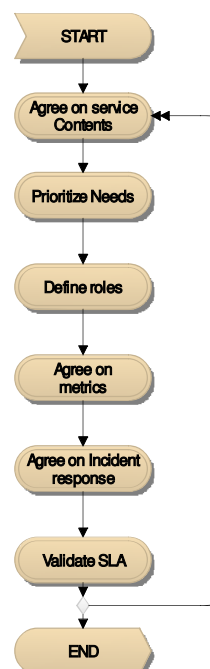


Figure 21: sla process [26]

- agree on what should be supported
- agree on metrics to measure performance
- agree on the effort that should be undertaken in case of an incident
- record the results from the negotiation

3.3.30. Monitor service

Service monitoring is an important activity for the service provide. Based on consumer feedback and market scan activities the service provider can adapt their services to the changing market [153] [61] [3].

3.3.30.1. Activities

The literature at least recommends performance measurements [8]. This can be assigning KPIs to supporting business processes, service availability quality, etc. These KPIs are the output from the previous phase of negotiating.

3.3.30.2. Techniques

The services marketing literature suggests a number of techniques that are applicable to use in monitoring services [113].

- Gap analysis, analyze how customers perceive the delivered service quality
- Benchmarking, compare the service quality to similar services in the market
- Performance measurements (e.g. revenues, market share, marketing costs, overhead costs, profits, ROI, consumer attitudes complaints, etc)

3.4. Solution validation

In this phase we have specified a number of solution alternatives and now it is time to choose the best solution. Validation is to show which solution is expected to reduce the gap between the experienced problems and the desires [170].

3.4.1. Activities

We have based these activities on the work of [170] and [102].

- compare the specified system requirements to the business requirements;;
- use scenarios to describe different situations to examine how the solution reacts to the changed situation[102] [170];
- use scenarios to describe solution usage on prioritized requirements [102];
- investigate financials of each possible solution;
- investigate possible new problems the solution causes (problem chaining);
- Organize a validation session with the stakeholders;
- For traceability purposes, record the alternatives, the decision made, who was responsible for this decision.

3.4.2. Techniques

- stakeholder analysis;
- sensitivity analysis;
- trade-off analysis;
- scenarios;
- validation session [71]

3.5. Conclusion

The method presented in this chapter has been organized using the engineering cycle [167] (in accordance to [102] [141]). The method starts with specifying business requirements which specify the need for systems [141] [102] [167]. After identifying the business requirements these need to be refined into system requirements. These requirements are designed according to the solution type. For this section we have presented three requirements engineering process models. The process model for classical RE is based on the work of [21] [83]. The process of COTS selection has been based on the available literature on COTS selection techniques (an overview can be found here [3]). RE for COTS selection starts with identifying a few baseline requirements. These are refined as more products are found. When we use RE in a service-oriented environment we see RE split off in two separate branches. RE at the consumer side, which mostly focuses on determining IT need and adapting to standard delivered services. RE at the provider side focuses mostly on specifying and designing standard services that address the market needs. The method accounts for the different nature of the RE process when we specify bespoke systems, services and COTS selection. We have positioned these variants in an over-arching generic framework that resembles the engineering cycle. This framework incorporates the important output elements required for traceability purposes. It identifies the relevant stakeholders, records their concerns, derives requirements and it specifies solution behavior. We also account for the fact that we have different solution alternatives (more than one solution can solve the problem). We record these alternatives and the arguments why the chosen solution was better than the alternatives.

When we look closely at the different proposed process models we can distinguish the following:

When we see RE as a selection activity for COTS products we tried to highlight the match making aspects. We used initial stakeholder requirements as an input for a market survey and then use the market to derive additional requirements. We based the initial elicitation on well known RE techniques.

When we reflect on service-oriented RE we can distinguish the following:

We distinguished between service consumer and service provider and the interactions between the two parties. The RE processes for the consumer and the provider behave quite different. We suggested that RE for service consumers is about identifying service needs, looking at available services to compose the composite service and when needed contact the service provider with a service specification. This service specification is used as a negotiation basis between the provider and the consumer. The RE process for a provider is different in the sense that it is based on market demands (or a consumer request). The provider invents

requirements, then specifies the service and then negotiates with consumers about the service quality and performance. The end result is then a SLA and the next activity is service monitoring. This is where the consumer provides feedback on service performance to the provider based on the defined metrics in the SLA. We used the well known RE concept of viewpoints to specify our service. Those viewpoints are based on the concepts of what the business can understand and why service designers can understand. The method also incorporates the well known concepts in the EA literature of business requirements. We use the business requirements to determine the why for the architectural elements. These are used to design the system requirements.

4. REQUIREMENTS MODELING IN ARCHITECT

In this chapter we will discuss the modeling concepts related to method presented in chapter 3. This work is largely based on the concepts introduced by Sebastiaan Hoogeveen, which can be found here [66]. We did make a few modifications to the work proposed by Sebastiaan Hoogeveen. The major change is the introduction of business requirements and that system requirements implement them. Sebastiaan Hoogeveen argued that this distinction was not needed in order to model requirements. We felt differently about this, it is true that business requirements and system requirements are both a type of a more abstract requirement. But it is important to show the relation between the two. System requirements are designed to realize business requirements, therefore separating these two concepts enable us to distinguish between the different semantics of the two.

Section 4.1 describes an extension to the ArchiMate framework. Section 4.2 introduces the modeling concepts related to the method and section 6.3 maps the method presented in chapter 5 to the modeling concepts.

4.1. Requirements traceability modeling framework

Sebastiaan Hoogeveen [66] has provided us with models that provide traceability in EA. He introduces an architecture lifecycle model that provides forward and backwards traceability for the enterprise architecture.

This way we can add motivation for the architectural model and show how the different elements within the architecture are realized. The motivation layer of the new framework provides us with the design decisions, the needs and rationale behind the architecture, the architecture layer is the normal ArchiMate framework and the realization layer provides us with all the information how the architecture is realized.

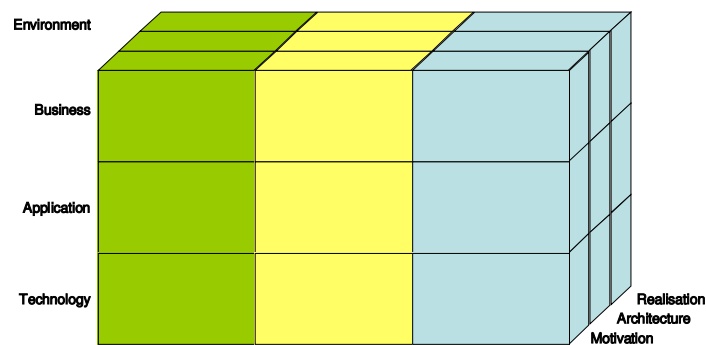


Figure 22: architecture life-cycle model

4.1.1. Motivation layer

The motivation layer contains three aspects (see Figure 23):

- stakeholders; these are the stakeholders and their interests
- conditions; these represent the conditions where the architecture must comply to
- Rationale, the decisions that were made during the architecture design

The stakeholders-domain contains the stakeholders and their concerns. Concerns are the experience phenomena or desires of a stakeholder. Within the conditions domain we can find the already developed principles domain. Principles are the architectural principles and goals where the architectural elements must comply to. We also find the *business requirements* domain here. The *business requirements domain* stretches all the way from the business layer down to the technology layer. These describe in business terms the functionality of the systems in the business (e.g. information service to the customer), application (software systems supporting business processes). The business requirements are the result of *problem investigation and analysis*. They provide motivation for systems within the architectural model.

The original model did not explicitly address business requirements. Since our method does recognize the difference between business requirements and system requirements we have included the business requirements domain. We have placed these business requirements in the motivation layer because they describe the needs for certain systems.

There is a small overlap between the principles domain and the requirements domain. The architectural goals within the principles domain can be seen as high level requirements, lower level architectural elements requirements implement these architectural goals. If we investigate the similarities even further *principles* can also describe *constraints* and our *business needs* of the system.

The *design decisions domain* describes the decisions that were made during the architecture design. This domain provides the rationale behind certain choices that led to the architectural elements within the architecture. The design decisions domain contains the concepts *design decision*, and *alternatives*.

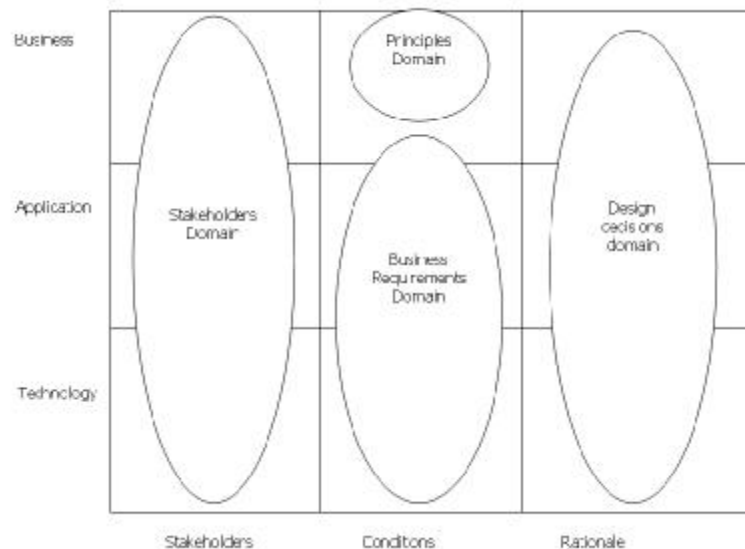


Figure 23: motivation layer

4.1.2. Realization layer

Just as the motivation layer the realization layer contains domains that cover multiple layers. Within the realization layer we distinguish the following aspects (see Figure 24):

- **control**, controlling information that has direct influence on architecture realization
- **time**, to model how the realization is realized in terms of time and planning
- **execution**, to model who is responsible for the creation, or parts, of the architecture

The existing programs and projects domain are part of the control aspect and are classified into the business layer. We added the *Solution realization domain* to control aspect. The design domain covers the business, application and technology layers. Within this domain we find the detailed *system requirements*, *solution designs* and *cases* that provide control to the realization of the architecture elements.

The aspect time contains the period domain. This domain describes the different phases within a projects life-cycle. The execution domain contains the execution domain. This domain contains the concepts of the stakeholder who is responsible for the creation, or parts, of the architecture. The executor is a role for one of the stakeholders.

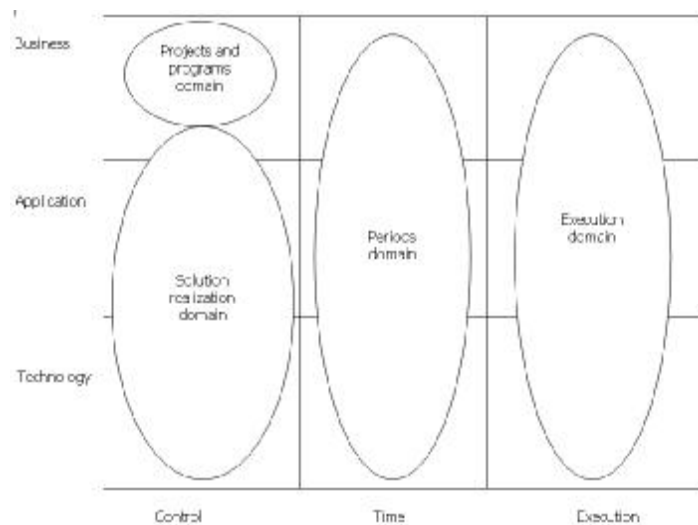


Figure 24: realization layer

4.2. Requirements modeling concepts


In [66] [71] we can find a number of modeling concepts related to requirements modeling, we will discuss these concepts structured around the before introduced layers. Section 6.2.1 discusses the modeling concepts from the motivation layer and section 6.2.2 discusses the modeling concepts from the realization layer. We will only discuss the new domains; existing domains are out of the scope of this design document (see [71] for more information). These concepts are added to the ArchiMate meta-model used by Architect.

4.2.1. Stakeholder domain

In the stakeholder domain we model the identified stakeholders and their concerns. The stakeholder domain contains the following concepts:

- stakeholder
- concern

These concepts are defined in the tables below.

Stakeholder	
Definition	Someone within the organization who has some stake with a certain element within the architectural definition
Visualization	
Attributes	Name Authority
Rules and guidelines	Stakeholders are defined as concretely as possible, if applicable with the names of the persons or

	departments.
Relationships	The stakeholder has an association relationship with a concern.
example	CEO H. Jansen Sales department

Table 7: stakeholder


Concern	
Definition	The concern of one or more stakeholders
Visualization	
Attributes	Description Weight
Rules and guidelines	A concern that is linked to the architectural elements must be indivisible, so it is clear which concern serves those elements.
Relations with other concepts	A concern has a relationship with both the stakeholder and the business requirements. The relationship with the stakeholder is an association relationship and with the relationship with the business requirements is a realization relationship originating from the business requirement.
example	Low operational costs (CEO concern) Uniform IT systems (IT department concern)

Table 8: stakeholder concern

4.2.2. Business requirements domain

The business requirements domain is a refinement of the principles domain. The business requirements are the goals for the systems to be developed. They describe the need for a system. These business requirements implement in their turn the high-level organizational and architectural principles. The modeling concepts have already been created by the principles domain. We will use the work from [71] to demonstrate these modeling concepts. We will model business requirements as goals.

Business requirements	
Definition	Business requirements describe the why for architectural elements. These can be goals of the stakeholders, revised architectural goals, revised organization goals etc.


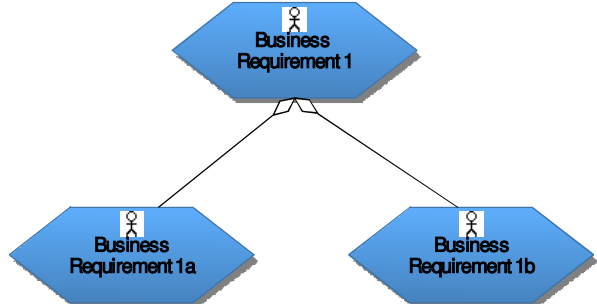

Attributes	Priority; Description; Type;
Visualization	<p>The business requirement concept can be visualized as follows:</p> 
Guidelines	<p>Decomposition: add more detail to a business requirement by introducing one or more sub requirements.</p>  <p>Naming: assign a clear, compact and striking name. Within the description (documentation) you can define the goal, ideally SMART (Specific, Measurable, Accessible, Realistic and Time-bounded). Number the goals.</p>
Relations with other concepts	The business requirement has a realisation relationship with a concern and a realization relationship with the system requirements. Business requirements can have an aggregation relationship with other business requirement.
Example	<p>PRO-FIT wants to improve the service quality while reducing costs. Goals related to customer service are:</p>  <p>For each goal you can add a description, for example (D1.3.): "Realize a 'customer profile' for every individual customer that is up-to-date at all times.</p>

Table 9: business requirements modeling concept

4.2.3. Design decisions domain

This domain contains the following elements:

- Design decision
- Design alternative


Design decision	
Definition	A decision during the architectural development. To keep things synoptic, it is advisable to only model decisions with a certain impact.
Visualization	
Attributes	Decision Responsible Rationale
Rules & Guidelines	The design decision must be formulated as short and concretely as possible.
Relations with other concepts	Design decisions justify one solution, which leads to design alternatives (the dismissed alternatives) and it is based on rationale. The relationship between the decision and the alternatives is an association relationship.
Example	Applications communicate using formal defined interfaces Systems A and B are located at location C.

Table 10: design decision


Design alternative	
Definition	Represents a possible alternative during architectural development. Only alternatives that are not part of the eventual architecture are modeled this way.
Visualization	
Attributes	Alternative Scenarios
Rules & Guidelines	Describes the state of the model as it would have been when this alternative was chosen. Try to use descriptions that discuss the difference with the eventual decision, because models change regularly.
Relations with other concepts	Design alternatives have a relationship with the design decisions, scenarios that were used during the validation process. Scenarios describe system usage compared to the requirements. The relationships are association relationships.
Example	Systems A and B on location D

Table 11: design alternative

The requirements domain contains the following concepts

- requirement
- case
- design artifact



System Requirement	
Definition	A concrete and verifiable demand for the artifact this requirement refers to. This demand is suitable for a more refined version of the architecture model or the architecture itself.
Visualization	
Attributes	Priority Description Type
Rules & Guidelines	A requirement describes the situation as it should be, as concrete as possible without implementation details. If needed use cases and design artifacts can be linked to a requirement.
Relations with other concepts	Requirements are linked to cases and design artifacts. A case can be anything from a scenario describing system usage to test cases. In the solution realization phase the requirement can also be linked to system components and application functions. The relationships with system functions, components and business requirements are realization relationships. The relationships between the system requirements and the cases are association relationships. It is also possible for system requirements to aggregate a higher level system requirement.
Example	The technology layer shall have an uptime of 99.99% Applications shall run on the existing UNIX platform

Table 12: requirements modeling concepts

Case	
Definition	A case is a certain use of the artifact. There are different cases e.g. test cases, use cases or abuse cases.
Visualization	
Attributes	Category Description

Rules & guidelines	The case can be assigned a number or a name. Use a scenario to describe the case. If needed cases can be related to each other. The relationships are association relationship.
Relations with other concepts	Cases are related to requirements and each other.

Table 13: case modeling concepts


Design artifact	
Definition	A design artifact is a design that spawns from the requirements and that entails detailed information about the realization of the corresponding requirements.
Visualization	
Attributes	description
Rules & guidelines	In most cases this will be a reference to an artifact from an external tool.
Relations with other concepts	A design artifact can either be documents created during the RE process (e.g. diagrams) or software design diagrams. It is related to the system requirements through a realization relationship.

Table 14: design artifact

4.3. Requirements Viewpoint

Based on experience with modeling the requirements we will refine the list of viewpoints a little. The requirements view is entirely based on the motivation view. In this view we can model the stakeholders, concerns, (business) requirements, design documents, decisions and alternatives.

4.3.1. Requirements view

In the requirements view it is possible to see who has what desire, the business requirements and the system requirements. This viewpoint is based on standard RE literature (e.g. [21] [83], [170]) and the RT literature (e.g. [57, 60, 59, 58]). Gotel argued about the importance of demonstrating who contributes to what. In this view we see the elicited concerns and their prioritization. In this view we can assign concerns to stakeholders, model the business and system requirements and link them together. Business requirements provide the why for systems. These contribute to provide the motivation for architectural elements. Using a cross-reference view it is easy to show which system requirements implement the business requirements. The most important relation in this view is how system requirements implement the business requirements. This view adds verifiability [147] to the

architectural element (it helps to increase the understanding of which system requirement realizes which organizational goal).

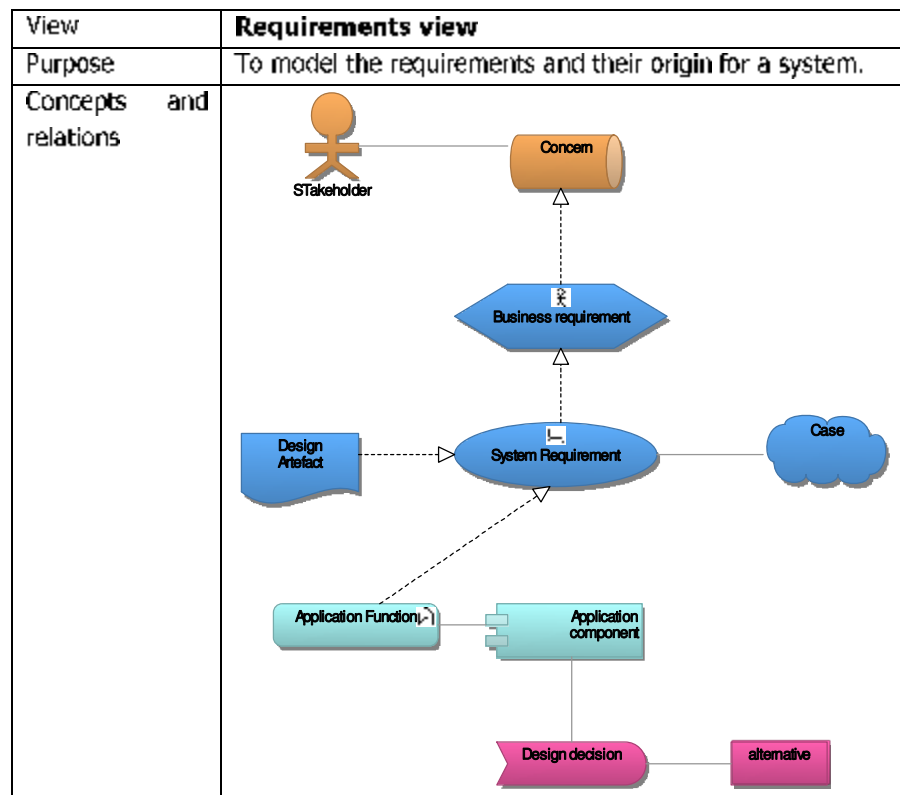


Table 15: stakeholder concerns view

4.4. Conclusion

In this section we will describe how these modeling concepts relate to the method described in chapter 3.1. We have seen that our method started with a problem investigation phase. In this phase we decided to identify the stakeholders, record their desires and analyze these desires. This has led to the modeling concepts of a stakeholder, his prioritized concern and the derived business requirements.

In the second phase of our method we investigate the solution alternatives. In this phase we transform our business requirements into system requirements. These system requirements have a priority. In this phase we specify the solution alternatives and validate each of these alternatives to the original requirements specification. An example of a solution alternative is when we decided to specify a bespoke system, scan the market for a system and need to choose between either the specified solution or the selected solution (note: this is only an example it is also possible that we only specify alternatives for bespoke systems or we record all alternatives we found during the market scan for COTS systems). During the investigate alternatives phase we create models (e.g. design artifacts), scenarios, test cases etc.

During the solution validation phase we evaluate the available specified solutions, this leads again to design decisions, based on rationale which leads again to design alternatives.

Problem investigation	Investigate solution alternatives	Validation
Stakeholder	System requirements	Design decision
Prioritized concern	Use cases	Chosen element
Business requirements	Test cases	Alternatives
	Design artifact	

Table 16: modeling concepts related to the phase of the RE method

The table clearly demonstrates that design decisions can be found in both the solution alternatives phase and the validation phase. The only difference is the scope. In "investigate solution alternatives" the scope is at the requirements level. Certain requirements get disregarded or receive a higher priority (based on a certain rationale). In the "validation phase" the scope is at the solution alternative level. Here the alternatives are the different possible solutions and we have to decide which one suits our requirements the best.

Since this is a requirements engineering method, we did not include the concepts from the projects, periods and execution domains.

5. TOOL SUPPORT

In this chapter we will discuss how the method can be supported. In chapter 4 we introduced the modeling concepts used for the concepts of the method itself. In this chapter we will discuss how the current tooling set of Bizzdesign can be used for some of the proposed techniques in the method.

Remember, the literature (e.g. [116]) suggests that a method needs to be support with suitable tooling. In section 5.1 we will discuss the general requirements for tooling support, in section 5.2 we will discuss how the current tools can be used and in section 5.3 we will provide conclusions about the current tooling.

5.1. Requirements for tooling support.

In this section we will briefly mention the global requirements for tooling support.

This is not intended as a full requirements specification for a RM tool. We will only discuss what features at least should be used and what features we can already implement in the current tooling support. When we look at [116] we see a prototype requirements tool. This tool enables working with informal requirements, more formal requirements models and it records process execution. When we map these properties to the proposed method in chapter 3 we can derive the following conclusions:

- Support for textual requirements and use of cross referencing between the requirements.
- Be able to model more formal requirements models (pro-art uses data flow diagrams, data dictionary and an ER model).
- Record the output of the requirements negotiation process (decisions, rationale, and alternatives).
- Tooling should record the output of all process execution and provide traceability links (scenarios, requirements, models, etc)

5.2. Current tooling support

In this section we will discuss how the available tools within BiZZdesign can be used to support the requirements management process.

5.2.1. Models during problem investigation

In chapter 3.2 we discuss a way of working to elicit the business requirements. One of the analysis techniques we propose there is through the use of goal analysis. It is possible to draw rudimentary goal trees in Architect.

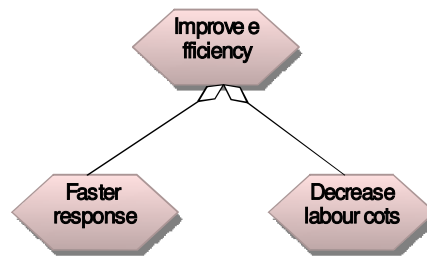


Figure 25: example goal tree

Although this kind of goal modeling is sufficient to build basic goal trees, it lacks some of the common concepts in GORE. The distinction between AND/OR relations is already present in the current release of Architect. A distinction not yet available is that of soft-goals and hard-goals. Soft goals are usually NFR or high level goals (e.g. improve efficiency in Figure 25: example goal tree).

During the method we also use scenarios to analyze goals and derive additional business requirements. Currently there is no support for actual scenario usage. In [66] the author suggests a modeling concept for a case (scenario, see also chapter 4). It is possible to use as a limited scenario representation, but it lacks the analytical properties a scenario has (e.g. triggering events, description, result, links to goals and actors)

In our method we also use the concept of a business process as an analytical model. Both BiZZdesigner and Architect are able to draw business process models.



Figure 26: example business process

5.2.2. Models during solution specification

We have already established in section 2.3 that a specification for a bespoke system should contain system context, list of functions, semantics of these functions and the quality attributes of these functions. If we look closely at the Volere template [130] we see two models that we can support through the use of Architect and BiZZdesigner.

5.2.2.1. Context diagram

A context diagram shows the communication between the system under development and the environment. This can contain persons, organizational departments or other entities depending on the level of detail. In Architect a model can be created that resembles this definition.

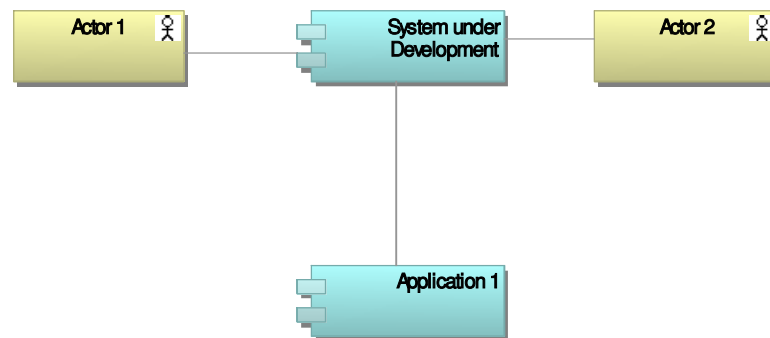


Figure 27: context diagram

5.2.2.2. Data diagram of the subject domain

A data model of the subject domain specifies the essential subjects, business objects, entities, etc that are relevant for the system under development. In BiZZdesigner we can draw this model using the standard UML notation. We refer to [73] for more information about OOA.

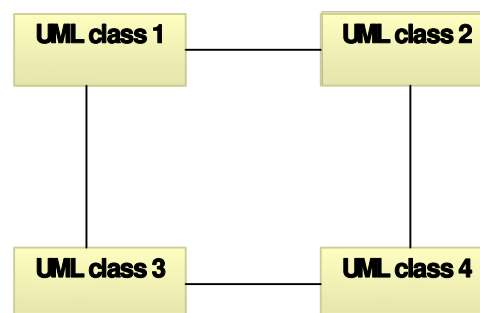


Figure 28: subject domain data model

5.3. Conclusion

After looking at the current tooling options within the BiZZdesign tool repository we can conclude that some basic well known RE models are fully supported within the available tools. This enables the customers of BiZZdesign to at least use Architect and BiZZdesigner during their RE process. Still there are a number of techniques that are not (yet) supported. Currently scenarios (use cases, test cases) can only be created during the modeling phase as a simplified representation of a proper scenario. Currently it is also not possible to create WSDL descriptions using BiZZdesigner or Architect (as suggested in the service requirements chapter). In chapter 3.3.22 we also recommend to create value models during service analysis. Currently these are not possible to model using Architect, but in the near future this support will be added to the BiZZdesign portfolio.

The current BiZZdesign toolset lacks the proper method and traceability support the RE literature recommends. The introduced concepts by [66] add traceability on a

higher abstract level. It provides the architectural model with motivation and reasoning (it supports the method framework presented in 3.1).

6. METHOD VALIDATION

In this chapter we will conduct a basic validation of the proposed method. We will provide an example case and provide an illustration to show how the method works. The example will discuss two parts of the method, we will specify a service and we will show how COTS selection takes place.

Section 6.1 will provide us with an example case which discusses the context of the RM method, in section 6.2 we will validate the service-oriented part of the method and in 6.3 we will select a new customer management system.

Both chapter 6.2 and chapter 6.3 are structured the same way, we will start with problem investigation, solution specification, solution validation and finally we will show how the end model shows us forward and backward traceability.

In section 6.4 the solution properties are compared with the results from the literature review in chapter 2 and in section 6.5 we will discuss the results from the validation attempt.

6.1. Example case

PRO-FIT is an average financial service provider, specialized in different insurance packages, such as life insurances, pensions, investments, travel insurances, damage insurances and mortgages.

In the last years PRO-FIT went through a structural change process, the result of which is that all business processes are consistent up the department level. However, the financial branch is one of the most dynamic and the senior management of PRO-FIT is now aware of new developments and threats, which require PRO-FIT to think of new ways to deal with these new challenges.

6.2. A new business service

During the identification of new developments and threats the senior management of PRO-FIT became aware of the new Service-Oriented way of thinking. A market analysis identified a number of opportunities; one of them is a differentiation strategy for their insurance services using modern technology.

During the past few months the customer support at PRO-FIT identified a number of problems as well. Customers are complaining about the lack of insight in their insurance portfolios, competitors offer new internet based solutions where customers can request all kinds of information about their insurance portfolios.

6.2.1. Problem investigation

In chapter 3.1.1 we proposed a problem investigation phase. We have also argued that this phase will result in a list of the identified stakeholders, their concerns and

the prioritized business requirements. In section 3.2 we have provided a process model to gather these results. In this section we will follow this process model, use a number of techniques (some more detailed than others) and model the results according to the proposed modeling concepts in chapter 4.2.

6.2.1.1. Elicitation

In the elicitation phase we start by identifying the stakeholders. There are a number of techniques we can use to identify stakeholders. A well known technique is the Volere stakeholder identification technique [2]. For more information about this technique we refer to the originating article. We assume that after applying the Volere stakeholder identification technique we have the following stakeholders:

- customers of PRO-FIT
- senior management of PRO-FIT
- customer service department of PRO-FIT

The second step is to record the stakeholders desires. There are a number of techniques that can be used to do so. For simplicity reasons we will assume that the desires have been collected through surveys, interviews, etc. Again, application of these techniques is not relevant for this validation. Only the results and the relationships between them are of any relevance. Secondly, we will also provide a small list of concerns, again because we just need to demonstrate feasibility.

After surveying the customers of PRO-FIT we can identify the following concerns:

- customers lack the possibility to gain insights in their current insurance portfolio through modern technology;
- customers find that current ways to manage their insurance portfolio is not optimal;
- customers find that they should be able to submit claims more easily through the internet;

The senior management of PRO-FIT has the following concerns:

- it is harder to attract new customers, insurance sales have dropped;
- existing customers leave or do not buy new insurances from PRO-FIT;
- differentiation is needed to attract new customers;

Customer services of PRO-FIT:

- they experience heavy workload in managing customer insurance portfolio's;
- they experience customer dissatisfaction;

A second way in identifying goals is examining organizational documents and goals. The relevant organizational goals, principles, general requirements for this RM project are stored in an architectural requirements document. After investigating this document we see the following goals:

- Be a reliable business partner to the customers;
- Development should be done using the service orientation paradigm;

6.2.1.2. Analysis

The elicited concerns are not yet complete. We can try to derive additional concerns using goal refinement trees (a way to transform a concern into a goal is to find an objective that negates the concern) or use a business scenario. Scenarios can be used to describe a way of working that addresses certain business goals. In this section we will transform the concerns into goals and refine them using refinement trees.

Transformation of concerns to goals:

Customer:

- Gain better insight in portfolio;
- Better management of bought insurances;

Senior management:

- increase sales;
- current customers should not leave PRO-FIT;
- attract new customers through product differentiation;

Help desk:

- decrease workload;
- increase customer satisfaction;

In the picture provided below (see Figure 29) we will show a snippet of a goal tree. We constructed a goal tree to refine the goals. We asked WHY questions to elicit more abstract goals and asked HOW questions to elicit more concrete goals.

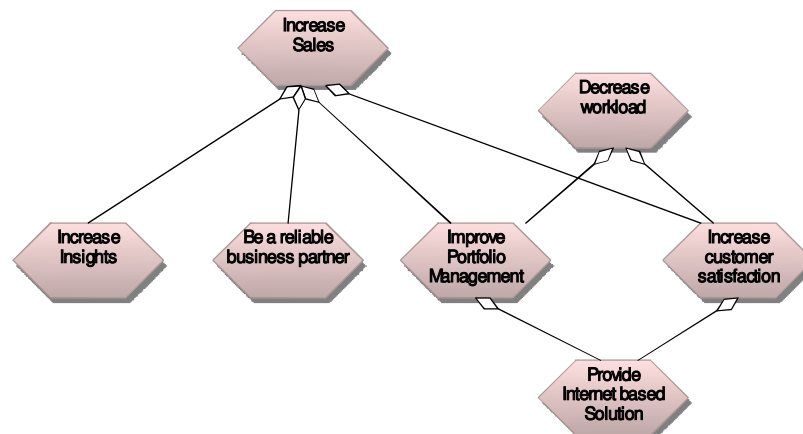


Figure 29: snippet of a goal tree

During the analysis phase we also prioritize the goals. The results from this analysis phase should be recorded. We now have prioritized goals, based on a certain rationale.

6.2.1.3. Specification

During this phase we specify all the work done up until now. During this phase we will write down the stakeholders and their concerns, the functional business requirements, business constraints and the non-functional business requirements. During this validation example we will see those steps as trivial and will not discuss them any further.

6.2.1.4. Validation

During this phase the specified business requirements need to be validated against the stakeholders concerns. The techniques to use these are trivial in nature. We will not discuss these in the example. After validation we have a validated requirements specification. It is possible to record the results of this phase.

6.2.1.5. Modeling the results

After the problem analysis phase we now have the stakeholders, their prioritized concerns and the business requirements. We can now use Architect to model these concepts. It is arguable that modeling is a form of specifying the requirements. But after actually working on this example we found that the models have the tendency to grow quite quickly. Therefore we will only focus on the relevant modeling aspects. Of course in the end the end-user of the method is free to choose how he specifies the requirements.

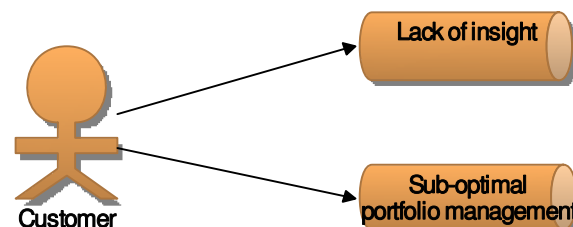


Figure 30: modeled concerns of the customer

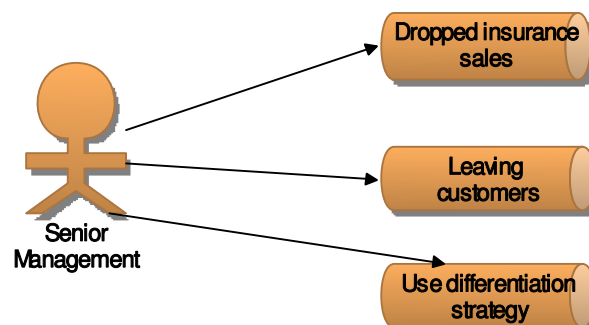


Figure 31: modeled concerns of senior Management

When we model the business requirements, we will use a modified goal representation. As mentioned before, we have chosen the goal notation because goals explain the why very well. Business requirements provide by definition the why for systems. In business requirements we can also distinguish the types functional, non-functional and constraints.

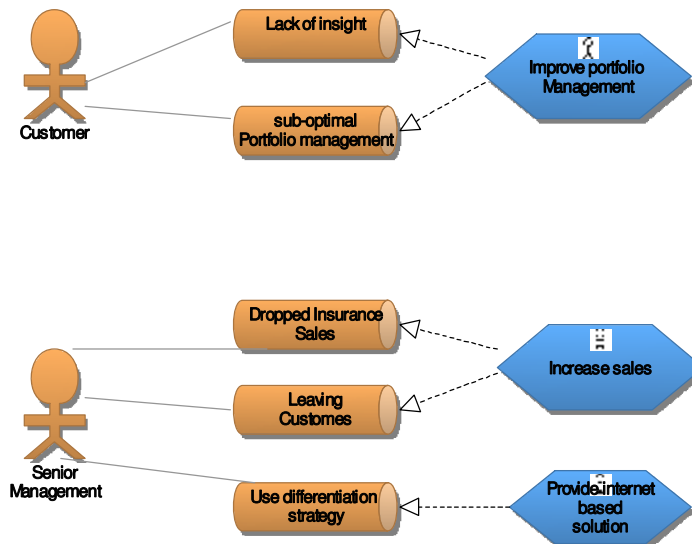


Figure 32: business requirements related to stakeholder concerns

Using a goal notation for the business requirements (see Figure 33:) provides us with an excellent way to reason about the why for architectural elements. It also provides us why we decided to build a certain element.

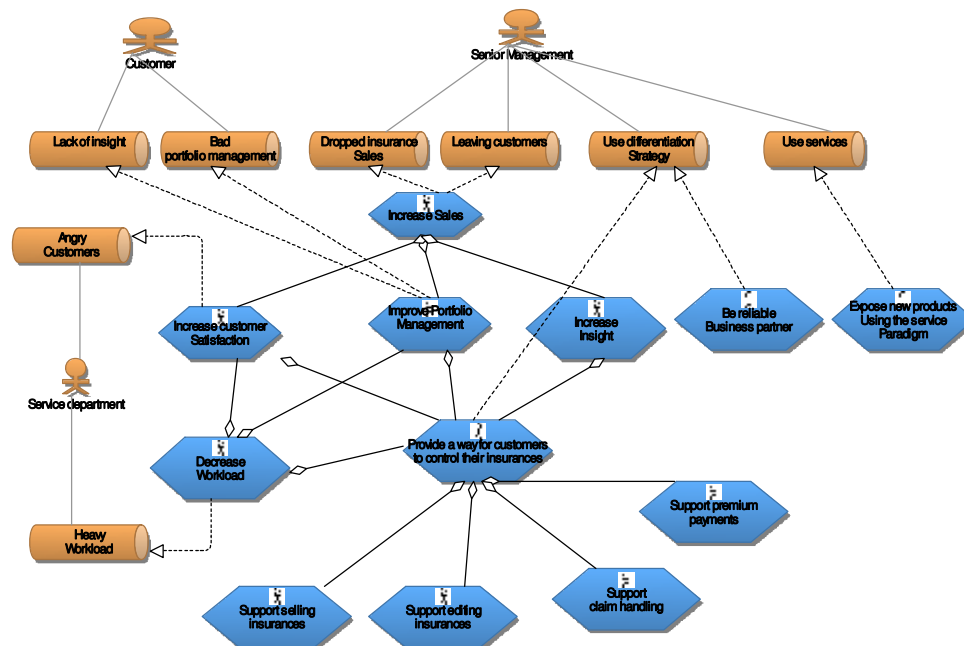


Figure 35: stakeholders, concerns and business requirements

As we can see here a goal notation for business requirements provides us with a structured way to show us why a certain architectural element should be there. It also gives us an overview of the problem under investigation; it shows who has what desires, the objectives to be reached and how they should be reached.

6.2.2. Investigate solution alternatives

In this section we will provide an example that will discuss the application of the solution specification part of the method. In this context we will specify the behavior of an insurance portfolio management service.

6.2.2.1. Elicitation

In this section we will use our proposed method to specify a business service. This business service provides the customers with an "insurance administration" option. This business service abstracts from any technology related issues. We will only specify the service-related concepts. The context of this specification is that the organization itself acts as a service provider. We therefore need to identify requirements for this service. Again we assume that the right stakeholders have been identified, and some sort of market scan has been performed (or knowledge about the market is available). We will now identify the system requirements. These are based on the business requirements, a refinement of these business requirements (e.g. stakeholders could be asked what the requirements are for a certain solution type), organizational goals, etc. The two techniques that we will use are stakeholder elicitation and organizational goal analysis. We will then elicit a number of requirements for the business service (again, this is only for illustration purposes only. A complete list of requirements is not needed).

We still have a list of stakeholder concerns; we have transformed these into business requirements. These were, improve portfolio management, and increase insight into the insurance portfolio.

We can then use these business requirements and organizational goals to identify system requirements for the business service. We present a selection of the following system requirements:

- 1) The service shall provide the customer with the option to view the status of his insurance portfolio;
- 2) The service shall provide the customer to handle claims;
- 3) The service shall provide the customer to view premium payments;
- 4) The service shall provide the customer with the option to buy new insurances from PRO-FIT.
- 5) The service must have a guaranteed availability of 99,6%.

6.2.2.2. Analysis

In this phase we will refine the elicited requirements. We will also try to look at how we can realize the service. We will investigate our business processes to elicit additional requirements and construct a scenario to check if our elicited requirements are correct.

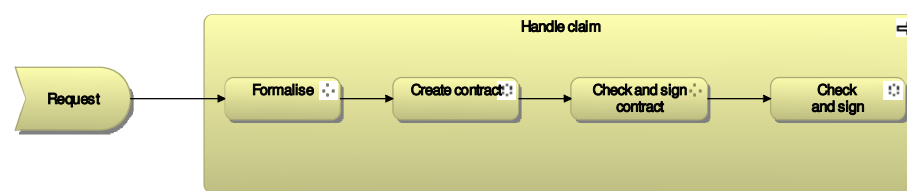


Figure 34: snippet of an example business process

In Figure 34 we see the business process "close contract". After analyzing this business process we can refine requirement #2 into:

- 1) The service must be able to formalize the contract;
- 2) The service must be able to create the contract;

We will now try to refine these requirements even more by using a scenario that describes "close contract process".

Event name	Input	Scenario	output
Request for insurance	Consumer request	After a (potential) customer initiates a request for a new insurance policy the first step is to formalize the contract. Contract formalization is realized through an intensive negotiation process with the customer, insurer and an	Signed contract

		intermediary. After the formalization phase the contract is drawn up, signed and send to the customer. The customer has to sign the contract as well and send it back.	
--	--	--	--

Table 17: request for insurance scenario

After analyzing this scenario we can refine requirement #2 even further:

- 3) The service must show the customer the status of the of the negotiation process.

Requirements prioritization also takes place during this phase. Because we work with such a small example and because of the practical nature of prioritization we take this step for granted.

The next step is to identify component services we can use to offer this new service to the customer (see Figure 35 for an overview of existing services). We can find existing services easily by investigating the existing architecture model. The architecture provides us with required design information.

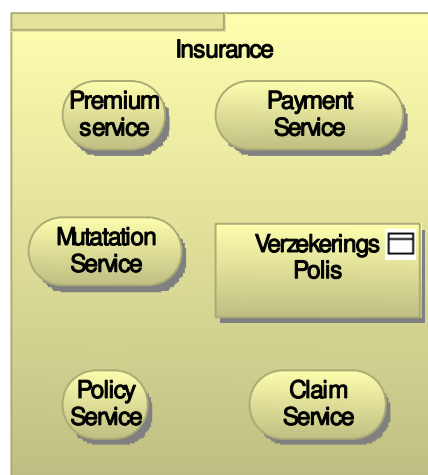


Figure 35: available services

Remember, we have identified the following requirements. We will use these requirements to elicit component services.

- buy new insurances;
- edit insurances;
- claim handling;
- premium payment;

If we compare these functional requirements to the available services we see we can create a composite service through combining the existing services. Our portfolio management service will use the:

- Premium service;
- Mutation service;
- Payment service;
- Claim service;

6.2.2.3. Specification

Again, we find that the specification phase is somewhat out the scope of this example. We refer to the Volere template [130] for more information about a requirements specification. The recommended specification techniques have been validated in [185]. After this phase we assume to have a requirements specification with the identified stakeholders, business requirements, requirements, data model of the service, value models and business process models.

6.2.2.4. Validation

During this phase the specified business requirements need to be validated against the stakeholders concerns. The techniques to use these are trivial in nature. We will not discuss these in the example. After validation we have a validated requirements specification. It is possible to record the results of this phase.

6.2.2.5. Negotiation

The negotiating phase has been validated in [26]. In a modeling example like this it is nearly impossible to validate the accuracy of process steps. We will therefore only show this step for illustration purposes only. Another possible problem is the fact that PRO-FIT just specifies a SLA and the customer just has to accept it. For illustration purposes only we will assume this is not the case.

The first step is where the consumer and the provider have to agree on the contents of the SLA. We will use the non-functional requirement as an example. PRO-FIT specified that in order for them to be a reliable business partner their service needs to have an availability of 99%. The consumer in this case does not agree with the 99% and they settle for an availability of 99.6%. After agreeing on the needs the second step is to prioritize these needs (see 3.3.4 for more information about prioritization). The third step comprises of defining the responsibility of who has to perform what. Bouman [26] mentions that this phase is particularly important because it is not always clear what the responsibilities of the consumer are for the provider (e.g. consumer is responsible for providing accurate measuring data).

The fourth step within this phase is to agree on the metrics used. One technique that can be used is the GQM technique. We have agreed upon on an availability of 99.6%. Through usage of the GQM technique we have agreed on the metric that availability is measured from the time the customer notices the incident and experiences the down-time. During the fifth step we have to agree on the appropriate incident response, PRO-FIT agrees with the customer that the proper response to reported down-time is to guarantee the service re-availability within 8 working hours.

The results mentioned here are recorded in the SLA document and after validation used as a binding contract between PRO-FIT and the customer.

6.2.2.6. Monitoring

In chapter 6 we have argued that monitoring is an important aspect in service oriented RE. Monitoring is not only used to measure the service performance against the SLA drawn in the previous phase but also to find out new demands from the market.

In the previous phase PRO-FIT and the unnamed customer agreed on service availability metrics how to measure service performance. In this phase we will show how the monitoring phase can be used to refine the requirements to changing customer demands. In chapter 6.3 we have suggested to use benchmarking, gap analysis or performance measurements as techniques to monitor service delivery performance.

The unnamed customer and other customers have been using the management portfolio service provided by PRO-FIT for quite some time now. The measurements provided by the customers do not indicate any problems with service delivery performance. In order to gain more insight in how the portfolio management service performs compared to competitors the requirements engineer performs a benchmark analysis. The results of this benchmark clearly show that similar services provided by competitors contain more features with higher service quality standards. Therefore another iteration of the RE process is required to refine the current portfolio management service.

6.2.2.7. Modeling the results

After specifying the solution properties we now have the following relevant concepts:

- system requirements
- design artifacts (requirements analysis models, etc)
- A use case (scenario) describing system usage.

We will now show some of the options we have to model these relevant concepts. Firstly we will model the system requirements and link them to the relevant business requirements.

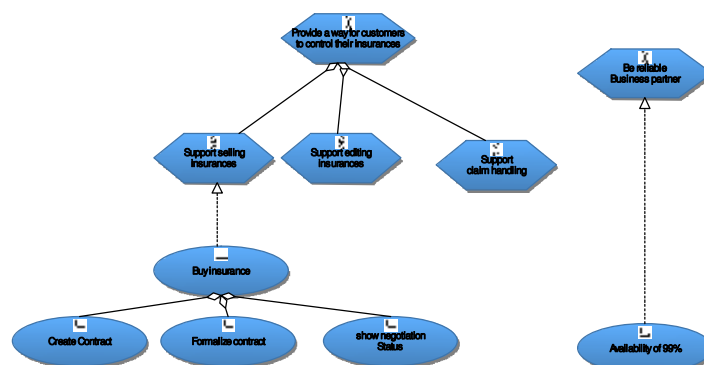


Figure 36: mapping from business requirements to system requirements

The second step of is to model the relevant artifacts created during the RE process. We will then link the relevant concepts to each other (see Figure 37).

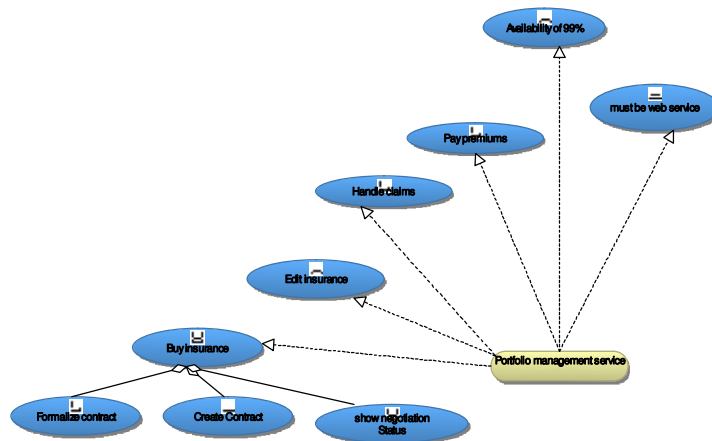


Figure 37: snippet of modelled requirements for a portfolio Management service

6.2.3. Solution Validation

In the previous phase we have constructed a number of requirements specifications (explicitly or implicitly). During this phase we will explain the validation phase to pick the best possible solution. In this example we take the view that there are a number of service specifications and designs. We assume that during the solution validation phase the requirements engineer organized a validation session with the stakeholders and that the financials have been investigated. For illustration purposes only we will assume that the different requirements specifications differ on the service delivery quality. As a concrete example we will use the availability of 99% requirement. During this phase we have three different requirements specifications. We assume we have a requirements specification that specified the availability of 98%, one with the availability of 99% and the availability of 99.9%. After an initial financial analysis (i.e. ROI) it was concluded that specification with 99% availability was the most profitable. This result was then discussed in a stakeholder validation meeting and therefore decided that the specification with 99% availability had to be implemented.

6.2.3.1. Modeling results

After the solution validation we now have a decision about which alternative is best, two solution specifications that were rejected and a final choice. The final choice is modeled as the normal architectural element.

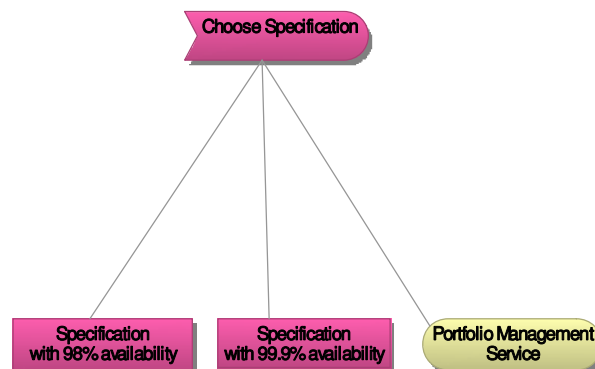


Figure 38: modeling the design decision

6.2.4. Traceability for the Portfolio Management Service

We will show that the method leads to requirements traceability through analyzing Figure 39 . We will show that forward and backward traceability is realized. We see forward traceability at work with the concern "use differentiation strategy" from senior management. This leads to the business requirement "Be a reliable business partner". We implement this goal through a non-functional system requirement "availability of the service must be at least 99%".

It is also easy to demonstrate backward traceability as well, the system requirement "formalize contract" is a sub function of "buying new insurances". We see that buying new insurances implements the business requirement "offer a web-based service for portfolio management"

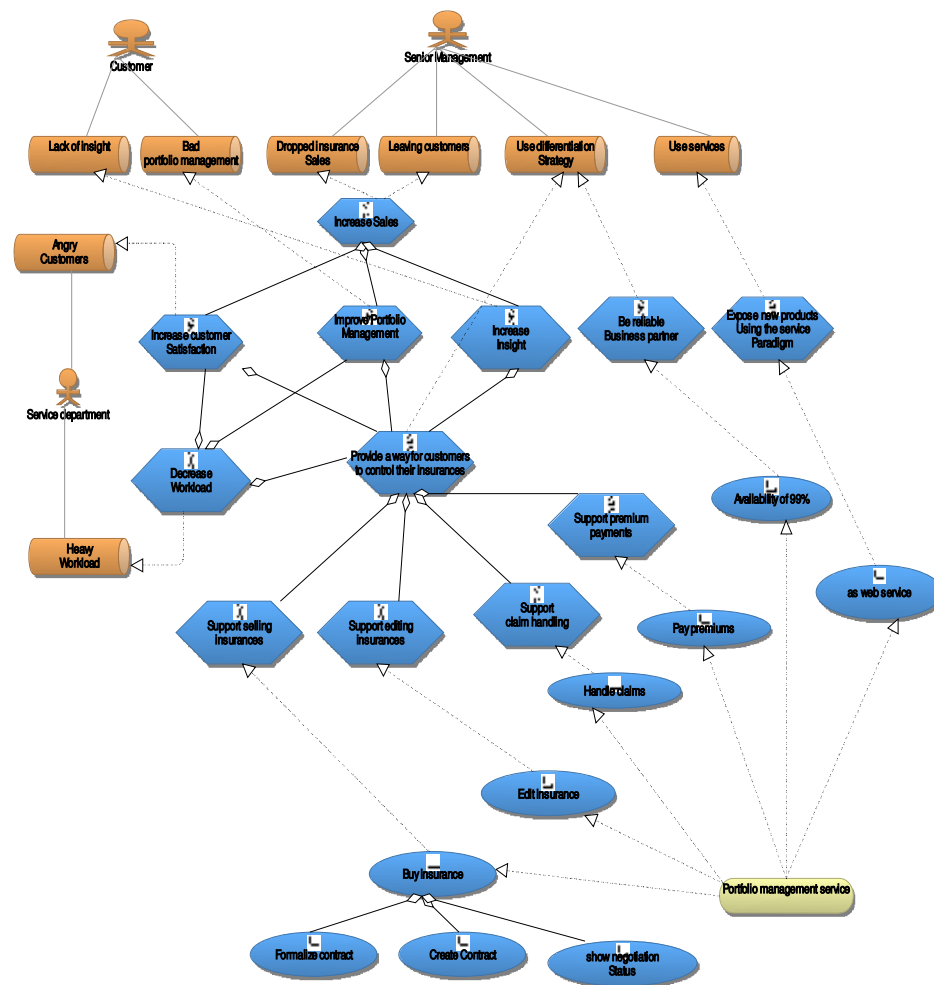


Figure 39: requirements model for the portfolio management service

6.3. Selecting a customer management application

In this section we will validate another part of the method. In the previous example we have shown how to use the method for a service. In this section we will discuss how to use the method for COTS selection. Because we already demonstrated large parts of the method, we will focus on the COTS selection only.

We will again use the PRO-FIT case, more precisely their effort in selecting a customer support application.

After the problem investigation phase we have identified the following stakeholders and their concerns:

Senior Management:

- Customer satisfaction below desired levels
- Buy of the shelf before making it ourselves

IT department:

- We do not want silo's

Customer

- Our personal information always seems incorrect or unknown

After investigating the organizational goals through analyzing the architectural principles we also found the relevant organization goal "never ask for known information"

Using the same techniques as in section 6.2.1 we have refined these into the following business requirements:

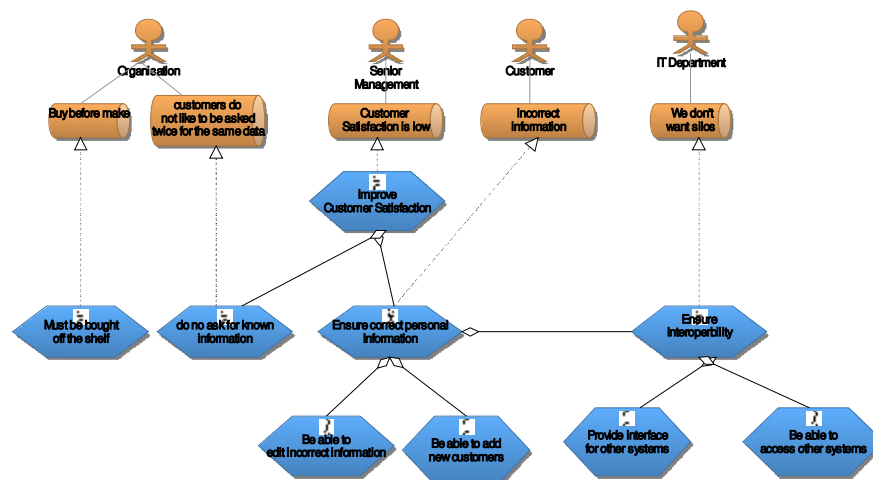


Figure 40: stakeholders, concerns and business requirements

During the investigating solution alternatives phase we will refine the concepts from Figure 40 into a preliminary set of system requirements. We have already demonstrated the applicability of some of the techniques in section 6.2.1, we will therefore only discuss the results.

After refinement we have the following set of requirements:

- Show customer information;
- Edit customer;
- Insert new customer;
- Interface with insurance acceptance system;
- Customer data interface;

One of the concerns of senior management was that they wish to buy before making it themselves. Therefore after searching the internet we have identified three customer management applications that we can use:

- Customer Management Application of company X
- Customer Management Application of company y
- Customer Management Application of company z

The first step in analyzing the found products is to determine the evaluation criteria. In our case, we will base them on the functional requirements and the price of the systems.

- The application must be able to show our customer information
- We must be able to edit customers
- We must be able to insert a new customer
- And we want an interface with our insurance acceptance system.
- We want the cheapest product that complies with the requirements.

All applications are able to show customer information, insert and edit customers. The differences lie in the 4th requirement. Because company x is also the provider of our insurance acceptance systems, they already have an interface readily available we can use. Company y doesn't provide an interface at all and company z wants to develop an interface for his product for an additional fee. Therefore the product from company x will be used as our customer management system.

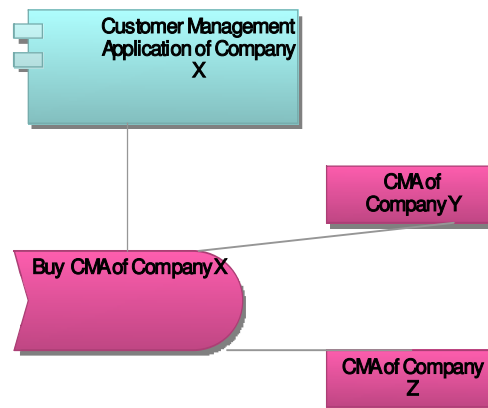


Figure 41: choosing application x

6.3.1. Traceability for the customer management application

We have already demonstrated that traceability is an important aspect in RM and argued that a RM method should incorporate traceability. However we have not yet shown how we realized traceability. We defined traceability as "the ability to describe and follow the life of a requirement, in both a forwards and backwards direction". In this section we will show that our method results in both forwards and backwards traceability through method and modeling support.

For this example we use the PRO-FIT case again, more precisely their customer support application (see Figure 42). We will show that after modeling it is possible to show where the requirements came from and where they were implemented.

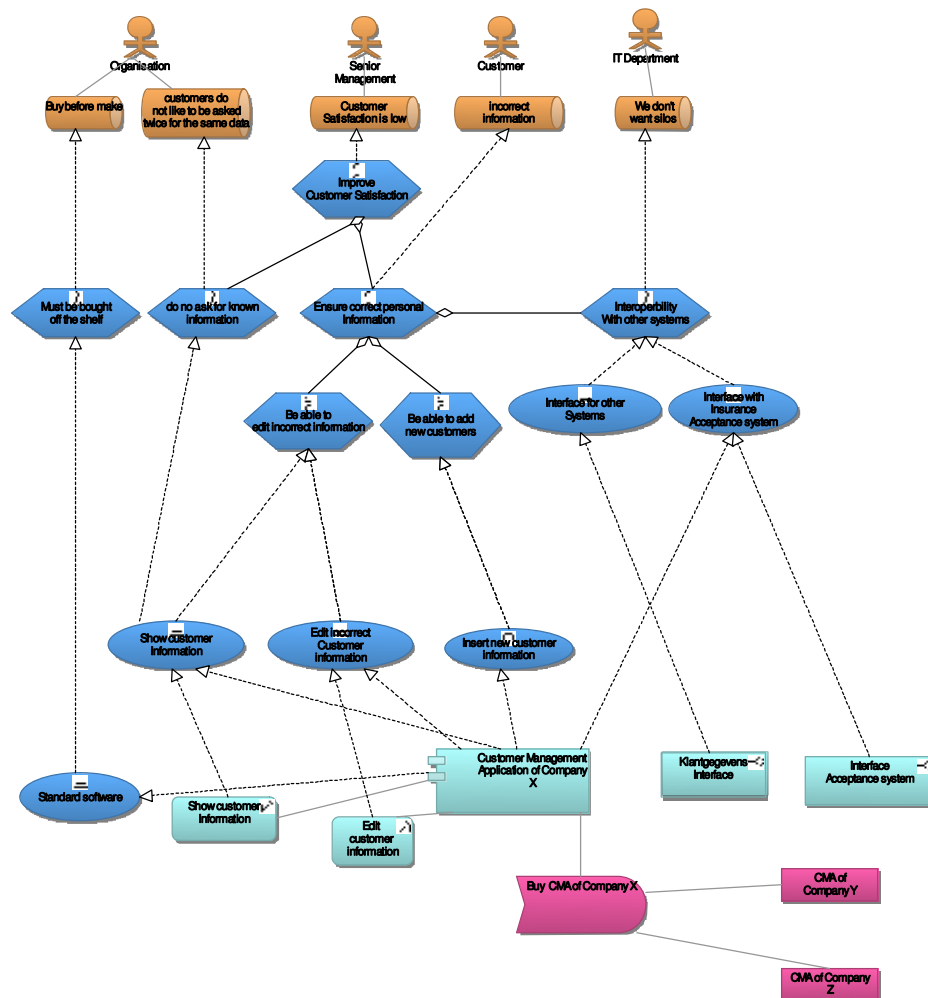


Figure 42: requirements model for the customer management application

When we look at Figure 42 we can see backward traceability. Backward traceability is to determine where the requirements spawn from. When we look at the requirement "interface for insurance acceptance systems" we see that it spawned from the business requirement "interoperability with other systems". This directly spawned from the concerns issued by the IT-department that they don't want to work with isolated silo applications.

Forwards traceability shows how requirements are implemented in either system components or system functions (see Figure 8). We will start with the business requirement "need COTS system that ensures correct customer information". This is refined into the business requirement "be able to edit incorrect information". Where as this is *implemented* by the system requirement "edit customer information" and "show customer information". Both these requirements are yet again *implemented* by the edit customer flow chart and the system that was bought of the shelf.

6.4. Mapping of solution properties to the requirements

In this section we will discuss how the proposed solution properties can be mapped against the requirements from chapter 5. One of the overall requirements was that

the method should start with high-level business requirements. In this section we discussed how we could translate organizational goals and stakeholder concerns to high level business objectives. During this phase of the method we are still (in theory) abstracted away from any possible solution implementation.

We have also discussed the requirements for realizing traceability. We have concluded that in order to realize traceability we should include it explicitly in the RE process. During this phase we have recorded the pre-RS information. We suggested recording all process information (e.g. stakeholders and their concerns, etc). This enabled us to model the stakeholders, their prioritized concerns and the high level business requirements.

We have also shown how to specify the system requirements for an information service that will be provided to the customer. We have used the earlier business requirements as an input for the system requirements. These system requirements implement the business requirements. We have also used the service provider process model. We assumed that PRO-FIT in this case is the service provider and the customer the service consumer. We therefore made an explicit distinction between the consumer and the provider roles.

We have also shown the different interaction points between the provider and the customer, being service negotiation and monitoring. The service provider process model also shows that we specified generalized services based on market demands and is adjusted according to changing market needs. We have also shown that the RE phase ends with a SLA.

We have also argued that service-oriented RE used properties of both RE for COTS and IT services management. In order to elicit requirements for a service we used the term "inventing requirements". This is gathering requirements for a system that is aimed at pleasing market demands, based on expected needs. Both [4, 7] discuss this process. We have also used the IT service management literature to include the SLA specification phase. More information about this can be found here [26].

During the requirements specification phase we have used the value aspect of a service to specify it. This was realized through the concept of a viewpoint.

We have also demonstrated that RE is both about problem investigation and specifying solution behavior. We started the method with recording the stakeholder concerns, analyzing these and translate them into business requirements. The second part of the method we have specified solution behavior. According to our discussion in section 2.3.1 we should have recorded the experienced problematic phenomena, the relations between these phenomena, why this is seen as problematic and who experiences these problems. If we look at our business requirements/ goal models from the problem investigation phase we see that this is true. We have recorded the experiences and through goal refinement it is possible to see the relations between the phenomena and why this is experienced.

We have also shown that we have implemented the requirement that the method should lead to a design decision that results in a final choice and alternatives that did not make it. We assumed that in the previous phase we had two additional specifications (it is also possible that the alternatives are completely other products

and that the design decision is a choice between a product, service, and business processes, etc)

6.5. Conclusion

During this example we have shown that it is possible to map method properties to the requirements from chapter 5. We do have a few comments and suggestions about the current modeling domain. Modeling all the requirements is an intensive task. Even in our trivial example with only a few requirements and concerns we had to limit the amount of requirements. We therefore suggest limiting the modeling of requirements to a certain level of abstraction.

A second point of further investigation is the pre-defined modeling views. The pre-defined views are not that hard to use. Although we do suggest to adapt existing views in Architect

After writing this example and modeling the results from the RE method we suggest that requirements modeling in Architect should be only used to enrich the Architect model. It is not applicable to use as a requirements repository. A dedicated tool that supports the entire RE process could be used to store the output from the requirements engineering process. Relevant information should then be modeled in Architect to enrich the architectural model.

Another point of discussion is the level of abstraction used in the requirements domain. In [66] the author introduces the concepts of case and design artifact. It might be more interesting to create new concepts based on the models that can be created using Architect or BiZZdesigner (e.g. data domain models). These models could be implemented as a new view (or adapt a view) in Architect. We suggest implementing these views after a validation round with the customers of BiZZdesign. A fourth point of discussion is the lack of a modeling approach. During execution of the method and modeling the results we found that a pre-defined modeling approach, to integrate requirements modeling in an EA context, is required. Therefore we added a modeling approach in the modeling section of this thesis. The fifth thing we noticed during the modeling phase is that the business requirements domain and the solution requirements domain could be merged.

It is also difficult to distinguish when to classify something as a business requirement or system requirement. When we are talking of high level goals, it is easy to classify it as a business requirement. We found the following rule of thumb a good guideline to use. When a business requirement can be implemented by a requirement that can be executed by a person we start speaking of system requirements (e.g. the business requirement "To support selling insurances" can be implemented by the system requirement "buy insurance").

Lastly, but not least, because we had to work with a pre-defined modeling domain which lacked some of the needed semantics we had to find and apply these semantics as well. When we look at the GORE literature we see that the majority of the newly defined semantics matches the GORE literature. The only major difference is the absence of the distinction between hard and soft goals (see Appendix I). We therefore suggest that after a more thorough validation the model domains should be further improved.

7. COMPARISON TO RELATED CONCEPTS

In this chapter we will position our method in a number of frameworks and system development methods. This is done to show the relevancy of the method and where it can be used. In section 7.1 we show how the method relates to the existing BPM model used by BiZZdesign. Section 7.2 demonstrates how the method can supplement their BPM method and in section 7.3 we position our method in existing development methods and compare it to the Volere and i*/ tropos methods.

7.1. Positioning of the method in the layered BPM model

In section 1.1 we introduced BiZZdesign and their layered BPM model. We will quickly summarize the model here and position the method in this framework. The layered BPM model comprises 3 horizontal layers (strategize, design, execute) and two vertical layers (implementation and governance).

Strategize

Steering and giving focus to an organization begins with identifying new developments and opportunities. These developments and opportunities are translated into policies, goals, guidelines and principles for the business and the IT. Business systems are a way to realize these goals; these systems should create both strategic alignment between business and IT and act as an enabler to realize organizational goals. Requirements for business systems ultimately need to adhere to these high level business goals.

Design

In the EA framework model design represents the instrument through which one can redesign organizations. These models describe the various organizational layers at different detail and abstraction levels. In design requirements engineering for business systems starts, problems are identified and possible solutions for these problems are identified. Based on the type of solution different techniques can be used and lead to different requirements models and a specification of the system.

Execute

This is the phase where the planned solutions are realized and the incorporation of the newly developed solutions into daily use. Depending on the solution type this

may result in new products, services, business processes, applications and information.

Implementation

Once the decision to improve has been taken, the implementation of this decision may begin. This starts by identifying what needs to be changed, namely the translating of the decision into concrete requirements specifications. Identifying the needs is the domain of requirements engineering. To position the engineering cycle within a known framework we argue that the steps of problem investigation, solution design, solution validation and solution implementation fit within the implementation element of the layered BPM model. We can also place RE onto this framework. As mentioned before RE is about getting from problem investigation to solution specification. RE can be found in the steps problem investigation and solution design of the engineering cycle. Requirements engineering basically comprises the steps of requirements elicitation, analyses, specification and validation. Requirements validation is when we either show that the stated requirements are correct or that the delivered solution meets the requirements. The latter is also known as requirements verification.

Governance

After a solution has been implemented, the implementation needs to be evaluated. Implementation evaluation happens in the governance layer of the BPM model. During the governance cycle the solution is constantly evaluated to check if it still adheres to the business requirements and organizational goals. The governance process for business systems is mainly directed at monitoring the performance of the systems. For example, it is common for business systems composed out of services to use KPIs [8] to monitor the performance of a service to check if the services still comply with the created SLAs. Another function of governance for business systems is to check if they still adhere to the business goals and strategy. When they no longer adhere to those goals a new iteration of change is needed to develop new, or modify the systems. This will lead to a change impulse that initiates another RE cycle.

7.2. BPM and the RM method

In the BiZZdesign portfolio we can also find their BPM/BPE method. This method is an approach to analyze and redesign business processes. In this method we can either see BPM as a technique for analyzing and specifying a new solution (business processes) or use business process analysis as a technique for requirements analysis. Business processes can serve as a context for systems, they can be used to derive goals, identify stakeholders (roles associated to the activities in a business process, etc). There are a number of situations in our method where BPM can be used as a technique to adapt.

- After the initial problem analysis, our method tries to be as solution independent as possible. New or improved business processes can be seen as a solution that can solve the experienced problems;

- Adapting to COTS software, as mentioned in the COTS selection section of this thesis, we mentioned BPM/BPE as a technique that can be used by the organization to adapt to standard delivered software ;
- Adapting to services, the same line of reasoning applies here, services offer generalized functionality, therefore the consumer might need to adapt to the offered functionality

7.3. Positioning in development methods

To demonstrate the added value of the proposed method we will compare our method to a number of well known development methods and RE methods. In section 7.3.1 and 7.3.2 we position our method in the well known DSDM and RUP development methods. In section 7.4.1 we compare our method to the well known VOLERE method.

7.3.1. DSDM

DSDM [128] is a framework containing steering mechanisms for Rapid Application Development. It is used in the development of information systems. DSDM acknowledges that projects are restrained by time and resources, and that requirements should be able to change. The framework suggests that after conducting a feasibility study and a business study a functional prototype should be identified, created and reviewed. After the next step it suggests to design the prototypes and afterwards implementation.

Although DSDM does mention requirements and a functional prototype it lacks how to get to a functional model. It does not provide process guidance how to get to a well defined requirements specification.

7.3.2. RUP

RUP is refinement of the generic Unified Process and developed by Rational Software Corporation/IBM . The main reason behind RUP was to tackle the problem of ad hoc requirements, ambiguous requirements, etc. RUP can be divided into four main phases:

- Inception phase, during this phase project feasibility and scope is determined. The end goal is a go/no-go decision based on assessed risks and cost projections.
- Elaboration phase, during this phase functional requirements are specified (using use cases) and the system architecture is development.
- Construction phase, during this phase the system is created.
- Transition phase, during this phase the system is validated by the stakeholders through besting. After implementation a lessons learned document is created.

In RUP requirements engineering starts at the inception phase and ends in the construction phase. The end document is the specification. If we look closely at the provided documents by IBM we can distinguish the same behavior as with DSDM. RUP does mention some of the RE activities and end-products, but again it lacks a detailed method for RE. Our method provides a more detailed RE process model,

with an explicit link between system requirements and their business objectives. We also abstract from techniques used and provide information when it is possible to use what technique.

This design study has provided BiZZdesign with a foundation of both proven RE knowledge and cutting edge Service-Oriented RE knowledge.

During the solution implementation phase special attention should be given to the proposed service oriented aspects of the method, as this is an opportunity in the RE field.

Another important contribution to practice is that the method distinguishes itself by now only providing a RE process model, but it also provides practice with a modeling technique. This modeling technique can be used to visually realize traceability in Enterprise Architecture models.

7.3.3. BiZZdesign EA method

An important activity is the positioning of our RM method in the general EA method used by BiZZdesign (see [71] chapter 3 and Figure 43). We will firstly summarize this process model and then position our activities for RM within this model.

The EA vision phase documents the architecture's drivers; these are expressed in long term goals and drivers. In this phase organizational goals are identified, important stakeholders are identified and their concerns are recorded. In the second phase the current situation and the desired situation are described (incorporating the stakeholder concerns). These descriptions describe either the EBA, EAA, ETA layers of the architecture framework. In the third phase the differences between the current state and the desired state are analyzed and results in a list of the functionality to be realized, documentation of the choices that have been made and the results from the different analysis. During the migration planning phase the wishes and functionality are broken into smaller parts and assigned to the projects that will realize them and during the EA maintenance phase results in the grounds for a new iteration of the cycle.

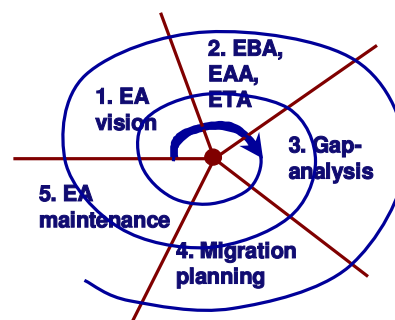


Figure 43: an EA process model [71]

Our method is also a cyclic method (see Figure 10) consisting of the phases problem investigation, specification of business requirements, investigate alternatives, specify business requirements and validate the solutions.

RM method	EA Method
-----------	-----------

Problem investigation	EBA, EAA, ETA
Investigate solution alternatives	
Validate solution	

Table 18: mapping of the RM onto the EA method

We argue that our method takes place in the relevant EBA, EAA and ETA sections of the EA process model. The reasoning behind this is as follows. If we zoom in on the EAA section of the process model and the application domain we see the following approach:

- Goals
- Design principles
- Stakeholders
- Data domain/application domain/coherence
- Document, validate and iterate

This is again the basic problem solving and design framework. We can map our three phases exactly on these steps.

RM method	EAA approach
Problem Investigation	Goals, Design principles, Stakeholders
Investigate Alternatives	Data domain/application domain/ coherence, document
Solution Validation	Validate and iterate

Table 19: mapping onto EAA approach

7.4. Alternative RE methods

There are already a number of requirements methods available. In this section we will discuss some of them. We have selected the Volere method because of its acceptance in the literature, we have selected CMMI because it provides best practices and we compare the method to I* because it handles with relevant goal concepts and modeling concepts.

7.4.1. VOLERE requirements method

One very well known RE method is the Volere specification method [130]. This method starts with a project blast off to assess the risks and to determine the business objects of the system under development. It describes a custom process model, very much similar like our bespoke systems process model.

It also uses a variation of the engineering cycle (although with custom naming conventions) and in more much more detail. But the major process steps (i.e. eliciting, analyzing, specification, validation, designing, implementation and evaluation) are there. It also provides a wealth of guidelines and techniques. The Volere method seems to focus much more objectives than our proposed method. But

in general it looks like a solid method. However, the Volere method is not entirely suitable to use in the context we position our method and does not meet all the requirements we have drawn. We will highlight a few of them:

- We already had a modeling domain. Our method is aimed at supporting the already available requirements modeling domain. Integrating this with the Volere method would have been a time consuming task.
- We use requirements as a form to create traceability in architectural models. We therefore did an extensive background research into (requirements) traceability. We have drawn up context models that can be used based on our modeling domain and the state of the art in the RT literature. This is not included in the Volere process model.
- The Volere method predates the service-oriented paradigm and does not include this in their method.
- It does not establish the influence of EA on the RM process, as we described in the introduction of this chapter and during the literature review phase.
- We also established that business requirements (system need) directly influences the system requirements. For traceability purposes we have provided a way to link these business requirements to the system requirements (we argued that system requirements implement the business requirements). This is much less extensively covered in the Volere method.

After careful consideration we rejected the Volere process model as an option to use as a basis for our method. Our proposed method suits the requirements much better than the Volere method. However we were able to use parts of the global idea and techniques in our method.

7.4.2. I* / TROPOS

I* [180] is a technique that focuses on modeling and reasoning support for early phase requirements engineering. It tries to capture the understanding of the organizational context and rationales that lead up to systems requirements. It consists of two main modeling components. The Strategic Dependency (SD) model is used to describe the dependency relationships among various actors in an organizational context. The Strategic Rationale (SR) model is used to describe stakeholder interests and concerns, and how they might be addressed by various configurations of systems and environments [179] [180].

I* can be used for both early and late phases of RE. During the early requirements phase I* is used to model the environment of the system to be, it facilitates the analysis of the domain by allowing the modeller to diagrammatically represent the stakeholders of the system, their objectives and their relationships [87]. During the late phases the I* models are used to propose the new system and the new processes and evaluate them on how well they meet the functional and non-functional needs of the users.

Tropos is an extension framework for I*, it is a requirements driven agent oriented development methodology [28] [87]. Tropos guides the development of agent-based systems from the early requirements analysis through architectural design and detailed design to implementation [87].

I* has quite a lot in common with our proposed modelling framework. The SR uses the same concepts as our method and modeling domain. They model the stakeholders, goals, soft goals and tasks. Goals are realised by tasks and soft goals mention how well a task should be executed. Their early phase requirements process maps in general pretty well to our stakeholders and business requirements domains. Their distinction between early and late requirements is almost the same as our distinction between business requirements and system requirements. Our business requirements are the needs of the organization, stakeholders and system requirements are designed to implement these needs. The main question again, why not use I*/Tropos instead of developing our own method? We will provide a few arguments why we think our method fits our requirements better:

- Although the modeling concepts used in i* match our concepts, theirs are much more detailed. Therefore their models have the tendency to grow even faster out of control.
- Our business requirements are focused on both organizational goals and stakeholder needs. They explain why an organization needs a system and why other stakeholders require this system. This is less clear in i*.
- In our method we do not explicitly specify for software systems, TROPOS and i* take a software based solution for granted.
- TROPOS does not take COTS selection in consideration.
- Although TROPOS is redesigned for services, we believe it is not suitable to use. Our review of the literature suggested different activities and specification techniques were required (e.g. monitoring activities for a continuous RE cycle or using value specifications).

7.4.3. CMMI

CMMI [148] is a process improvement approach that provides organizations with guidelines for effective processes. There are two different models available, development and acquisition and a third one (for services) is under development. CMMI provides process guidelines for RE, either for development or COTS selection. The process guidelines for bespoke systems specification resembles the process introduced in this thesis and the Volere requirements process. It starts with identifying customer needs and refining these into product requirements.

The proposed RE process model for acquisition identifies customer needs as well and tries to refine these into contractual requirements. These are specified requirements the bought product should adhere to.

The CMMI process models are widely accepted process models based on best practices. Why not use the CMMI process models instead of developing our own method? We can answer this question by reusing some of the arguments used in discussion the Volere model.

- Firstly, we already had a modeling domain. Our method is aimed at supporting the already available requirements modeling domain. Integrating this with the CMMI practices would have been a time consuming task;
- Secondly, we use requirements as a form to create traceability in architectural models. We therefore did an extensive background research into (requirements) traceability. We have drawn up context models that can be used based on our modeling domain and the state of the art in the RT literature. This is not included in the CMMI models;
- The described acquisition process model is quite different than we concluded after analyzing the scientific literature. This provides us with a reason to develop our own process model for COTS selection;
- Service orientation is still under development for CMMI, therefore no requirements process model had been defined (March 2008);
- Our method provides better integration by using an abstract framework where we place, needs identification, solution design/selection and validation.
- Because traceability was an important aspect of this research, our process models are aimed at generating the information needed to realize traceability from system functions to stakeholders and vice versa.
- We also show the role of EA in requirements engineering; this is not there in the CMMI process models.

Another difference between our method and the suggested CMMI approaches is that CMMI provides a set of activities and desired outcomes (goals). Our method provides a more detailed approach how the set of activities interact and the relation between the end products of every step (we therefore provide process guidance, where CMMI provides outcome guidance).

The most important requirements activities at CMMI are at level two and three. Requirements Management activities are defined at level two and Requirements Development at level three. If we look at the goals of these activities and compare them to our method we can see the following.

In Requirements Development the main goals are to develop customer requirements, develop product requirements and analyze and validate the requirements. Our method does use slightly different terms, our business requirements include the term customer requirements (CMMI defines this as the needs for a system). These customer requirements are then refined into product requirements, this is more or less the same we suggest when we refine our business requirements into system requirements. The last primary goal is the analysis and validation of requirements. This is also done in our method.

If we compare our method to the Requirements Management section of CMMI we do see a small gap. We clearly meet the goal "maintain bidirectional traceability of requirements". The goals "manage requirements changes" and "obtain commitment to requirements" are partially met or not met at all (the latter was left out of the scope intentionally of this thesis).

7.5. Conclusion

In this chapter we mapped the proposed method to the existing BiZZdesign portfolio. We have argued that requirements engineering and architecture design overlap, if we look closer, we see our RM method as a detailed subset of the existing EA process model. RM drives the design of the architecture and the EA therefore provides a scope and context for RM. The current state of this method does not yet contain the elements on how to control a method, how to setup RM projects, etc. In the future we need to look on how to extend the method with these elements and how the existing tools like the grip card and Risk manager can support those new aspects. We also compared our method to well known RE methods to demonstrate the need to develop a new method. Comparison with Volere, i* / Tropos and CMMI showed that although they were somewhat usable, they did not always map to our conclusions drawn in chapter 2.9.

8. CONCLUSION AND DISCUSSION

In the previous chapters we have presented the aspects of RM, developed these aspects into a method, showed how to model the requirements, and showed how tool support can supplement the method, provided a validation and discussed the position of the method in related concepts. In this chapter we will discuss the results, the relevance, derive general conclusions and provide a number of recommendations. Section 8.1 will discuss the result of this design study, section 8.2 provides a general context to the results, we will show relevance in 8.3, and conclude this thesis with a conclusion and recommendations in sections 8.4 and 8.5.

8.1. Result

In section 1.2.1 we presented the main goal of this thesis, to develop a method for RM, using previously developed modeling concepts. And in section 1.2.4 we specified the high level structure of this method. The method should be based on solid theory, provide process models, provide modeling concepts and discuss tooling support. In chapter 2 we provide the conclusions of an extensive literature review about requirements engineering, requirements traceability and how to position RM to EA. Based on these reviews chapter 2.9 provides us with requirements for the method under development. In chapter 3 we present a method based on these requirements. It positions the summarized views on RE in an overall framework which is based on the design cycle of the engineering cycle. Using this framework we structured the method around problem investigation, solution specification and solution validation. The method explicitly mentions the modeling concepts provided at the start of this design research. In chapter 4 we structure these concepts into modeling domains and views and provide a modeling approach to guide the process of requirements modeling. Chapter 6 discusses possible tooling support for the method. We review the existing BiZZdesign tools and map these to the techniques mentioned in the method. In chapter 7 we provided a basic validation (through an illustration) of the method by investigating experienced problems and specify a solution. We showed that the method implemented the requirements by explicitly mapping solution properties to stated requirements.

8.2. Discussion

The results from this design study can be used by BiZZdesign as a basis for further validation and refinement. In 1.3.2 we proposed to use a spiral model for this

method design. We were able to finish one large full circle. We used the results from our illustration example to refine the method a little. The first validation attempt provided us with a modeling approach, some refined relationships in the modeling chapter. The question now is how to proceed further. We have planned a number of e-mail interviews. These interviews can be used as a more concrete validation. It is possible to gather more data about preferable techniques and novelty aspects of the method (i.e. where should we focus on the most).

After a successful proper validation we can begin with solution implementation. Our solution implementation is writing a handbook about requirements engineering and using the method in practice. Another important aspect is to keep monitoring the changes in the RE field and EA field. New concepts may arise and the method needs to be adjusted accordingly. This is the solution evaluation phase.

8.3. Relevance

We will now discuss the relevance of this method to practice and science. We will start with showing the relevance for the scientific field.

Although the main goal of this design was not to create new knowledge, we were able to structure and integrate existing knowledge. We were able to show the relevance of the two views on RE in one method. RE is not just about problem investigation or solution specification. It is about both. The question is which phase is done in more detail.

The last contribution to the scientific field is that we used the little available knowledge in service oriented RE, supplemented it with techniques from services marketing and the IT service management field and proposed a structured way of working in the context of the service-oriented paradigm.

Our main goal was to introduce a new artifact, the artifact being a RM method. Therefore it must have some use to practice and provide a novel way of working. We argue that the proposed method is of use in practice because it provides complete requirements engineering process support. We show the practitioner how to determine the intended need of the system and specify a solution for it. It uses the state of the art from the requirements engineering field. It shows the differences between RE for bespoke systems, RE for COTS selection and how these two relate to the latest service oriented paradigm. It also integrates the latest techniques and concepts in the RE field (e.g. goal oriented RE). Using and positioning the latest developments from the RE field we create a novelty aspect for the method itself. Another novelty feature is the combination of modeling techniques to model the requirements to enrich architectural models with the results from the RM phase. Another practical advantage of this model is that it can be used in conjunction with existing system development methods (e.g. DSDM and RUP). Both DSDM and RUP mention the RE process and objectives but do not provide any guidance for it. It is possible to use our method in existing system development projects because we built our method around the idea of a logical problem solving process. This logical problem solving process can also be found in either RUP or DSDM.

8.4. Conclusion and further work

In this thesis we have presented a method, suggested a way how to deal with requirements, suggested a modeling approach and showed how the method can be used in a practical validation example. We have also shown the relevance of this method by comparing and positioning it to existing (system) development methods. What we see is that our method can be used as a detailed zoom on the different phases in RUP, DSDM and the EA method used by BiZZdesign.

When we look back at the design goal in chapter 1.2.1 we see that our method has provided us with a way to acquire requirements, how to specify the solution and how to model requirements.

Our literature review provided us with insight into the fundamental differences between classical RE, RE for COTS selection and service oriented RE. This understanding enabled us to decide on a framework to position RM in, define the steps required in this method and describe how these steps can be executed. We also showed that traceability can be incorporated by recording the process information and linking the relevant concepts to each other.

We were also able to answer the research question on how requirements could be modeled. We proposed a modeling approach, refined the originally proposed meta-model by separating the requirements domain into business requirements and system requirements. We also argued that when modeling the requirements the modeler should use some form of abstraction to limit the level of detail. For example a good level of abstraction is "The user should be able to request new insurances". A less workable level of abstraction is "when user x presses the submit insurance button, the system shall provide x with an overview of her request".

This thesis also discusses how the method can be supported using the BiZZdesign toolset, either using existing tooling to produce goal trees, process models, context diagrams, etc or the newly proposed modeling domains within Architect. We also showed that for a complete way of supporting a requirements management tool is recommended.

8.4.1. Limitations and future work

When we look critically at our work we see a first design of a RM method. It has a solid basis, but it is still not entirely complete. For example, we designed a method for requirements management in general, but the focus lies more on functional requirements. It would be interesting to investigate the role of non-functional requirements on business – IT alignment. Non functional requirements have an important role; they say how well a solution must perform. The role of non-functional requirements becomes increasingly more important in certain situations where the correctness of information, the speed of the information handling, etc is desired. In this case the business requirements dictate that information should be correctly and safely, should be handled quickly and the system requirements implement this through security requirements and performance requirements.

A second point of improvement is the modeling domain. The concepts introduced work well, but we could increase their semantics a little further. At the moment we use aggregation relationships to refine higher level requirements and goals. But as the GORE literature (e.g. see [155]) recommends, or/and refinements could be desired. Combined with the increased research into non-functional requirements we could also introduce hard/soft goals into the modeling domain.

Modeling the requirements has also proven to be quite burdensome. Models tend to grow quickly out of control; therefore relying on the models only for a complete specification is not desired. A dedicated RM tool combined with a pre-defined requirements document could be a solution to this particular problem.

Lastly, this method only describes the process steps for RM, but how to setup a RM management project is not considered in this thesis. For the actual implementation of the method this is could be another interesting subject to discuss.

8.5. Recommendations

The main goal of this thesis was to provide insight how a RM could look like and use it as a blueprint for a handbook about RM. We do feel that the method presented here is a complete and logical structuring about how the scientific field discusses RM, however the method is completely based on theory and the only validation is done by showing how to use the method on a trivial example. Therefore we can suggest the following course of action:

- Practice based knowledge gathering, find out the exact desires from the practical field and what techniques are currently used for RM;
- Validate the method using larger scale field trials;
- Focus on the integration of the different RE process models and views on RE;
- Pay special attention on the service-oriented section in the validation effort. This section is based on the least amount of scientific literature;
- Use this input to launch a new iteration of the design cycle used during this research;
- Implement the method by writing the handbook;
- Evaluate the implementation;

8.5.1. Suggestions for Implementation

- Due to time constraints, we have not included the way of controlling. During the implementation of this method it might be a topic of consideration. We can think of a detailed project inception phase, how to assign resources to the method, position risk assessments, etc.
- For complete process support a RM tool should be developed to create a requirements repository and a customized specification.
- Improve the modeling concepts, ranging from better representation models (pictures) to some more detailed semantics (e.g. incorporate AND/OR realization relationships).
- Some requirements can be derived directly from the architecture. Currently we do not map these requirements to the originating architectural elements. This can be desirable in the future.
- Detailed analysis of the role of non-functional requirements could be desired. We have already selected the NFR framework as an analysis technique. Detailed study could lead to aspects we can use.

References

- [1] I. Alexander and S. Robertson. Understanding project sociology by modeling stakeholders. *Software, IEEE*, 21(1):23–27, 2004.
- [2] IF Alexander. A Taxonomy of Stakeholders: Human Roles in System Development. *International Journal of Technology and Human Interaction*, 1(1):23–59, 2005.
- [3] C. Alves. COTS-Based Requirements Engineering. *Component-Based Software Quality: Methods and Techniques*, 2003.
- [4] C. Alves, J. Castro, and F. Alencar. Requirements Engineering for COTS Selection. *Third Workshop on Requirements Engineering*, 2000.
- [5] C. Alves and A. Finkelstein. Challenges in COTS decision-making: a goal-driven requirements engineering perspective. *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 789–794, 2002.
- [6] C. Alves and A. Finkelstein. Investigating conflicts in cots decision-making. *International Journal of Software Engineering and Knowledge Engineering*, 13(5):473–493, 2003.
- [7] C. Alves, X. Franch, J.P. Carvalho, and A. Finkelstein. Using Goals and Quality Models to Support the Matching Analysis During COTS Selection. *Proc. of ICCBSS 2005*, 2005.
- [8] J. Amsden. Modeling soa: Part 1. service identification. 2007.
- [9] BA Andrews and WC Goeddel Jr. Using rapid prototypes for early requirements validation. *Digital Avionics Systems Conference, 1994. 13th DASC., AIAA/IEEE*, pages 70–75.
- [10] AI Anton. Goal-based requirements analysis. *Requirements Engineering, 1996., Proceedings of the Second International Conference on*, pages 136–144, 1996.
- [11] A.I. Antón, W.M. McCracken, and C. Potts. Goal decomposition and scenario analysis in business process reengineering. *Proceedings of the 6th international conference on Advanced information systems engineering table of contents*, pages 94–104, 1994.
- [12] A. Arsanjani. Service-oriented modeling and architecture. *IBM Developer Works*, 2004.
- [13] C.M. Ashworth. Structured systems analysis and design method (SSADM). *Information and Software Technology*, 30(3):153–163, 1988.
- [14] A. Aurum and C. Wohlin. *Engineering And Managing Software Requirements*. Springer, 2005.
- [15] M.N. Aydin and F. Harmsen. Making a method work for a project situation in the context of CMM. *Product Focused Software Process Improvement (Profes 2002), Rovaniemi, Finland*, 2002.
- [16] P. Berander and A. Andrews. Requirements Prioritization. *Engineering and Managing Software Requirements*, 2005.
- [17] R. Biddle, J. Noble, and E. Tempero. Reflections on CRC cards and OO design. *Proceedings of the Fortieth International Conference on Tools Pacific: Objects for internet, mobile and embedded applications-Volume 10*, pages 201–205, 2002.

- [18] R. S. Bittler and G. Kreizman. Enterprise architecture process: Evolution 2005. *www.gartner.com*, 2005.
- [19] Bizzdesign. History of bizzdesign. Available on: *www.bizzdesign.nl*, 2007. visited on:10-08-2007.
- [20] D. Björner and C.B. Jones. *The Vienna Development Method: The Meta-Language*. Springer-Verlag London, UK, 1978.
- [21] F. A. Blaauboer. Requirements in functional it management. Technical Report TR-CTIT-06-07, Enschede, March 2006.
- [22] S. Bleistein, K. Cox, J. Verner, and K. Phalp. B-SCP: a requirements analysis framework for validating strategic alignment of organisational IT based on strategy, context and process. *Information and Software Technology*, 48(9):846–868, 2006.
- [23] S.J. Bleistein, K. Cox, J. Verner, and K.T. Phalp. Requirements engineering for e-business advantage. *Requirements Engineering*, 11(1):4–16, 2006.
- [24] B. Boehm and P. Bose. A collaborative spiral software process model based on theory w. *Software Process, 1994. 'Applying the Software Process', Proceedings, Third International Conference on the*, pages 59–68, 1994.
- [25] B. Boehm, A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy. Using the WinWin spiral model: a case study. *Computer*, 31(7):33–44, 1998.
- [26] J. Bouman, J. Trienekens, and M. van der Zwan. Specification of Service Level Agreements, clarifying concepts on the basis of practical research. *Proceedings of the Software Technology and Engineering Practice conference*.
- [27] I.K. Bray. *An Introduction to Requirements Engineering*. Addison-Wesley, 2002.
- [28] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
- [29] S. Brinkkemper. Method engineering: Engineering of information systems development methods and tools. *Information and Software Technology*, 38(4):275–280, 1996.
- [30] R. Broekstra. Implementation environments for semantic web services. Master's thesis, University of Twente, 2007.
- [31] J.M. Carroll, S.R. Alpert, J. Karat, M. Van Deusen, and M.B. Rosson. *Raison d'Etre: capturing design history and rationale in multimedia narratives*. ACM Press New York, NY, USA, 1994.
- [32] J. Castro, M. Kolp, and J. Mylopoulos. Towards requirements-driven information systems engineering. *Information Systems*, 27(6):365–389, 2002.
- [33] P.P.I.N.S. CHEN. The Entity-Relationship Model-Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.
- [34] Betty H. C. Cheng and Joanne M. Atlee. Research directions in requirements engineering. In *FOSE '07: 2007 Future of Software Engineering*, pages 285–303, Washington, DC, USA, 2007. IEEE Computer Society.
- [35] L. Cherbakov, G. Galambos, R. Harishankar, S. Kalyana, and G. Rackham. Impact of service orientation at the business level. *IBM Systems Journal*, 44(4):653–667, 2005.
- [36] L. Chung. *Non-Functional Requirements in Software Engineering*. Springer, 1999.

- [37] L. Chung and K. Cooper. COTS-Aware Requirements Engineering and Software Architecting. *Proc. of Software Engineering Research and Practice Conference (SERP)*, 2004.
- [38] K. Cox and K. Phalp. From process model to problem frame—a position paper. *9th International Workshop on Requirements Engineering—Foundations for Software Quality (REFSQ'03)*, pages 93–96.
- [39] K. Cox, K.T. Phalp, S.J. Bleistein, and J.M. Verner. Deriving requirements from process models via the problem frames approach star, open. *Information and Software Technology*, 47(5):319–337, 2005.
- [40] G Cuevas, A Serrano, and A Serrano. Assessment of the requirements management process using a two-stage questionnaire. *Proceedings of Fourth International Conference on Quality Software*, pages 110–116, 2004.
- [41] A. Dardenne, S. Fickas, and A. van Lamsweerde. Goal-directed concept acquisition in requirements elicitation. In *IWSSD '91: Proceedings of the 6th international workshop on Software specification and design*, pages 14–21, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.
- [42] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993.
- [43] JE Dobson, AJC Blyth, J. Chudge, and MR Strens. The ORDIT approach to requirements identification. *Computer Software and Applications Conference, 1992. COMPSAC'92. Proceedings., Sixteenth Annual International*, pages 356–361, 1992.
- [44] SM Easterbrook. *Handling Conflict Between Domain Descriptions with Computer-Supported Negotiation*. University of Sussex, School of Cognitive and Computing Sciences, 1991.
- [45] M.E. Fagan. Design and Code Inspections to Reduce Errors in Program Development. *IBM Systems Journal*, 38(2/3):258–287, 1999.
- [46] M. Febowitz, S. Greenspan, H. Reubenstein, R. Walfordt, and G.T.E.L. Incorporated. ACME/PRIME: Requirements Acquisition for Process Driven Systems. *Proceedings of IWSSD*, 8:36.
- [47] P. Fenkam, H. Gall, and M. Jazayeri. Visual requirements validation: case study in a CORBA-supported environment. *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pages 81–88, 2002.
- [48] A. Finkelstein and W. Emmerich. The future of requirements management tools. *Information Systems in Public Administration and Law*, 2000.
- [49] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. *International Journal of Software Engineering and Knowledge Engineering*, 2(1):31–58, 1992.
- [50] D. Firesmith. Prioritizing Requirements. *Journal of Object Technology*, 3(8):35–47, 2004.
- [51] A. Fuxman, L. Liu, M. Pistore, M. Roveri, and J. Mylopoulos. Specifying and analyzing early requirements: some experimental results. *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International*, pages 105–114, 2003.
- [52] K. Go and J.M. Carroll. The blind men and the elephant: views of scenario-based system design. *interactions*, 11(6):44–53, 2004.
- [53] J.A. Goguen and C. Linde. Techniques for Requirements Elicitation. *Requirements Engineering*, 93:152–164, 1993.

- [54] J. Gordijn and JM Akkermans. Value-based requirements engineering: exploring innovative e-commerce ideas. *Requirements Engineering*, 8(2):114–134, 2003.
- [55] J Gordijn, Pl van Eck, and R Wieringa. Requirements engineering techniques for e-services. 2008. Accepted.
- [56] J. Gordijn, E. Yu, and B. van der Raadt. E-service design using i* and e (3) value modeling. *IEEE Software*, 23(3):26–33, 2006.
- [57] O. Gotel. *Contribution Structures for Requirements Traceability*. PhD thesis, London, England: Imperial College, 1995.
- [58] O. Gotel and A. Finkelstein. An analysis of the requirements traceability problem. *Proc. of the second IEEE International Conference on Requirements Engineering*, pages 100–107, 1995.
- [59] O. Gotel and A. Finkelstein. Contribution structures [requirements artifacts]. In *RE '95: Proceedings of the Second IEEE International Symposium on Requirements Engineering*, page 100, Washington, DC, USA, 1995. IEEE Computer Society.
- [60] O. Gotel and A. Finkelstein. Extended requirements traceability: results of an industrial casestudy. *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 169–178, 1997.
- [61] Q. Gu and P. Lago. A stakeholder-driven service life cycle model for SOA. *Foundations of Software Engineering*, pages 1–7, 2007.
- [62] Ron Hakvoort. Method for developing product architectures. Master's thesis, University of Twente, 2005.
- [63] C.L. Heitmeyer, R.D. Jeffords, and B.G. Labaw. Automated consistency checking of requirements specifications. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 5(3):231–261, 1996.
- [64] P. Heymans and E. Dubois. Scenario-Based Techniques for Supporting the Elaboration and the Validation of Formal Requirements. *Requirements Engineering*, 3(3):202–218, 1998.
- [65] H. Holbrook. A scenario-based methodology for conducting requirements elicitation. *SIGSOFT Softw. Eng. Notes*, 15(1):95–104, 1990.
- [66] Sebastiaan Hoogeveen. Traceability in architectures. Internal BiZZDesign research, 2007.
- [67] P. Hsia, D. Kung, and C. Sell. Software requirements and acceptance testing. *Annals of Software Engineering*, 3:291–317, 1997.
- [68] P. Hsia, J. Samuel, J. Gao, D. Kung, Y. Toyoshima, and C. Chen. Formal approach to scenario analysis. *Software, IEEE*, 11(2):33–41, 1994.
- [69] E. Hull, K. Jackson, and J. Dick. *Requirements Engineering*. Springer, 2005.
- [70] A. Hunter and B. Nuseibeh. Analysing inconsistent specifications. *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 78–86, 1997.
- [71] M.E Jacob, H Franken, and H Van den Berg. *Enterprise Architecture Handbook*. Bizzdesign academy publishers, 2007.
- [72] M. Jackson. Software requirements & specifications: a lexicon of practice, principles and prejudices. 1995.
- [73] I. Jacobson. The use-case construct in object-oriented software engineering. *Scenario-based Design: Envisioning Work and Technology in System Development*, pages 309–338, 1995.

- [74] M. Jirotko and J.A. Goguen. *Requirements engineering: social and technical issues*. Academic Press Professional, Inc. San Diego, CA, USA, 1994.
- [75] S.V. Jones and N.A.M. Maiden. Rescue process manual version 4.1. Technical report, Centre for HCI Design, City University, London., 2004.
- [76] SV Jones, NAM Maiden, K. Zachos, and X. Zhu. How service centric systems change the requirements process. *Proceedings of 11th International Workshop on Requirements Engineering: Foundation for Software Quality*, 2005.
- [77] H. Jonkers, M. Steen, and P. Strating. Een service-georiënteerde ontwerpaanpak. 2007.
- [78] R.S. Kaabi and C. Rolland. Eliciting service composition in a goal driven manner. *Proceedings of the 2nd international conference on Service oriented computing*, pages 308–315, 2004.
- [79] E. Kavakli. Goal-Oriented Requirements Engineering: A Unifying Framework. *Requirements Engineering*, 6(4):237–251, 2002.
- [80] L. Kolos-Mazuryk. Re for pervasive services, 2005.
- [81] J. Kontio et al. A COTS Selection Method and Experiences of Its Use. *Proceedings of the 20 th Annual Software Engineering Workshop, Maryland, November*, 1995.
- [82] G. Kotonya and I. Sommerville. Requirements engineering with viewpoints. *Software Engineering Journal*, 11(1):5–18, 1996.
- [83] G. Kotonya and I. Sommerville. *Requirements engineering*. Wiley, 1998.
- [84] B.L. Kovitz. *Practical software requirements: a manual of content and style*. Manning Publications Co. Greenwich, CT, USA, 1998.
- [85] R.J.H.M. Kroese. Ontwerp en implementatie van een bpm governance framework. Master's thesis, Universiteit Twente, 2007.
- [86] M. Lankhorst. *Enterprise Architecture at Work: Modeling, Communication and Analysis*. Springer, 2005.
- [87] A. Lapouchnian. Goal-oriented requirements engineering: An overview of the current research. *University of Toronto*, 2005.
- [88] D. Lau and J. Mylopoulos. Designing Web services with Tropos. *Web Services, 2004. Proceedings. IEEE International Conference on*, pages 306–313, 2004.
- [89] J. Lee. Extending the pots and bruns model for recording design rationale. *Proceedings of the 13th international conference on Software engineering*, pages 114–125, 1991.
- [90] J. Leite and PA Freeman. Requirements validation through viewpoint resolution. *IEEE Transactions on Software Engineering*, 17(12):1253–1269, 1991.
- [91] J. Leite, G. Rossi, F. Balaguer, V. Maiorana, G. Kaplan, G. Hadad, and A. Oliveros. Enhancing a requirements baseline with scenarios. *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 44–53, 1997.
- [92] J. C. S. P. Leite. Viewpoint analysis: a case study. In *IWSSD '89: Proceedings of the 5th international workshop on Software specification and design*, pages 111–119, New York, NY, USA, 1989. ACM Press.
- [93] K.R.P.H. Leung and WL Yeung. Generating User Acceptance Test Plans from Test Cases. *Computer Software and Applications Conference, 2007. COMPSAC 2007- Vol. 2. 31st Annual International*, 2, 2007.

- [94] S. Lichtenstein, I.i Nguyen, and A. Hunter. Issues in it service-oriented requirements engineering. *Australasian Journal of Information Systems*, 13, 2006.
- [95] J. Luijpers. White paper project start architectuur.
- [96] N. Maiden and S. Robertson. Developing use cases and scenarios in the requirements process. *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, pages 561–570, 2005.
- [97] NA Maiden and C. Ncube. Acquiring COTS software selection requirements. *Software, IEEE*, 15(2):46–56, 1998.
- [98] NAM Maiden, H. Kim, and C. Ncube. Rethinking Process Guidance for Selecting Software Components. *Cots-Based Software Systems: First International Conference, ICCBSS 2002, Orlando, FL, USA, February 4-6, 2002: Proceedings*, 2002.
- [99] NAM Maiden and G. Rugg. ACRE: selecting methods for requirements acquisition. *Software Engineering Journal*, 11(3):183–192, 1996.
- [100] NAM Maiden and A Sutcliffe. Requirements critiquing using domain abstractions. *Requirements Engineering, 1994., Proceedings of the First International Conference on*, pages 184–193, 1994.
- [101] J. Mostow. Toward better models of the design process. *AI Magazine*, 6(1):44–57, 1985.
- [102] A Mulholland and A.L Macaulay. Architecture and the integrated architecture framework. *Whitepaper, Capgemini*, http://www.capgemini.com/services/soa/ent_architecture/iaf/.
- [103] L. Murray, A. Griffiths, P. Lindsay, and P. Strooper. Requirements traceability for embedded software — an industry experience report.
- [104] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: a process-oriented approach. *IEEE Transactions on Software Engineering*, 18(6):483–497, 1992.
- [105] S. Nanduri and S. Rugaber. Requirements validation via automated natural language parsing. *Journal of Management Information Systems*, 12(3):9–19, 1995.
- [106] C. Ncube and N. Maiden. PORE: Procurement-Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm. *International Workshop on Component-Based Software Engineering*, 1999.
- [107] CJ Neill and PA Laplante. Requirements engineering: the state of the practice. *Software, IEEE*, 20(6):40–45, 2003.
- [108] B. Nuseibeh and S. Easterbrook. Requirements engineering: a roadmap. *Proceedings of the Conference on The Future of Software Engineering*, pages 35–46, 2000.
- [109] MP Papazoglou and D. Georgakopoulos. Service-Oriented Computing. *Communications of the ACM*, 46(10):25–28, 2003.
- [110] M.P. Papazoglou and P. Ribbers. *E-business: organizational and technical foundations*. John Wiley & Sons, 2006.
- [111] M.P. Papazoglou and W.J. Van Den Heuvel. Service-oriented design and development methodology. *International Journal of Web Engineering and Technology*, 2(4):412–442, 2006.
- [112] D.L. Parnas and D.M. Weiss. Active design reviews: principles and practices. *Proceedings of the 8th international conference on Software engineering*, pages 132–136, 1985.

- [113] A. Payne. *The essence of services marketing*. Prentice Hall New York, 1993.
- [114] L. Penserini, A. Perini, A. Susi, and J. Mylopoulos. From Stakeholder Needs to Service Requirements. *Proceeding of International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements*, 2006.
- [115] F.A.C. Pinheiro. *Requirements Traceability*. 2003.
- [116] K. Pohl, R. Dömges, and M. Jarke. PRO-ART: PROcess based Approach to Requirements Traceability. *Poster Outlines: 6th Conference on Advanced Information Systems Engineering, Utrecht, Netherlands, June, 1994*, 1994.
- [117] K. Pohl and P. Haumer. Modeling contextual information about scenarios. *Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ*, 97:187–204, 1997.
- [118] M.E. Porter and V.E. Millar. How information gives you competitive advantage. *Harvard Business Review*, 63(4):149–160, 1985.
- [119] B. Potter, J. Sinclair, and D. Till. An introduction to formal specification and Z, 1992.
- [120] C. Potts. Using schematic scenarios to understand user needs. *Proceedings of the conference on Designing interactive systems: processes, practices, methods, & techniques*, pages 247–256, 1995.
- [121] C. Potts. Fitness for use: the system quality that matters most. *Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ*, 97:15–28, 1997.
- [122] C. Potts. ScenIC: a strategy for inquiry-driven requirements determination. *Requirements Engineering, 1999. Proceedings. IEEE International Symposium on*, pages 58–65, 1999.
- [123] C. Potts, K. Takahashi, and AI Anton. Inquiry-based requirements analysis. *Software, IEEE*, 11(2):21–32, 1994.
- [124] B. Ramesh. Factors influencing requirements traceability practice. *Communications of the ACM*, 41(12):37–44, 1998.
- [125] B. Ramesh and M. Jarke. Toward reference models for requirements traceability. *IEE transactions on Software Engineering*, 27:58–93, 2001.
- [126] B. Ramesh, T. Powers, C. Stubbs, and M. Edwards. Implementing Requirements Traceability: A Case Study. *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, pages 89–95, 1995.
- [127] Requirements engineering using the SOS paradigm. Requirements engineering using the SOS paradigm. *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*, pages 260–263, 1993.
- [128] M. Rietmann. Dsdm in a bird's eye view. 2001.
- [129] S. Robertson. Requirements trawling: techniques for discovering requirements. *International Journal of Human-Computer Studies*, 55:405–421, 2001.
- [130] S. Robertson and J. Robertson. *Mastering the requirements process*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1999.
- [131] W.H. Roetzheim. *Developing Software to Government Standards*. Prentice Hall, 1991.
- [132] C. Rolland and R.S. Kaabi. An Intentional Perspective to Service Modeling and Discovery. *Relation*, 1:1.
- [133] C. Rolland, C. Souveyet, and CB Achour. Guiding goal modelling using scenarios. *Software Engineering, IEEE Transactions on*, 24(12):1055–1071, 1998.

- [134] K.S. Rubin and A. Goldberg. Object behaviour analysis. *Communications of the ACM*, 35(9):48–62, 1992.
- [135] R. Schick. Scenarios and scenario-based methods for requirements engineering.
- [136] K. Schoman and DT Ross. Structured Analysis for requirements definition. *IEEE Trans. on Software Engineering*, 3(1), 1977.
- [137] H. Sharp, A. Finkelstein, and G. Galal. Stakeholder identification in the requirements engineering process. *Database and Expert Systems Applications, 1999. Proceedings. Tenth International Workshop on*, pages 387–391, 1999.
- [138] S.S. Some. Use Cases based Requirements Validation with Scenarios. *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05)-Volume 00*, pages 465–466, 2005.
- [139] I. Sommerville, P. Sawyer, and S. Viller. Viewpoints for requirements elicitation: a practical approach. *Proc. Third IEEE International Conference on Requirements Engineering (ICRE 98)*, 1998.
- [140] G. Spanoudakis. Plausible and adaptive requirement traceability structures. In *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 135–142, New York, NY, USA, 2002. ACM Press.
- [141] J. Spencer et al. TOGAF "Enterprise Edition" Version 8.1. 2004.
- [142] A. Sutcliffe and A. Gregoriades. Validating functional system requirements with scenarios. In *RE '02: Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*, pages 181–190, Washington, DC, USA, 2002. IEEE Computer Society.
- [143] A. Sutcliffe. A conceptual framework for requirements engineering. *Springer*, 1(3):170–189, 1996.
- [144] A. Sutcliffe. A technique combination approach to requirements engineering. *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 65–74, 1997.
- [145] A. Sutcliffe. Scenario-based requirements engineering. In *RE '03: Proceedings of the 11th IEEE International Conference on Requirements Engineering*, page 320, Washington, DC, USA, 2003. IEEE Computer Society.
- [146] A. Sutcliffe, J. Galliers, and S. Minocha. Human errors and system requirements. *Requirements Engineering, 1999. Proceedings. IEEE International Symposium on*, pages 23–30, 1999.
- [147] A. Tang and J. Han. Architecture rationalization: A methodology for architecture verifiability, traceability and completeness. In *ECBS '05: Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, pages 135–144, Washington, DC, USA, 2005. IEEE Computer Society.
- [148] C.P. Team. CMMI for Development, Version 1.2. *Preface, Software Engineering Institute, Carnegie Mellon University, August*, 2006.
- [149] The Standish The Open Group. Togaf™ version 8.1.1 "enterprise edition. <https://www.opengroup.org/architecture/togaf8-doc/arch/>, 2006.
- [150] T. Thiadens. *Manage IT!: Organizing IT Demand and IT Supply*. 2005.
- [151] Harmen van den Berg and Henry M. Franken. *Handboek Business Process Engineering*. Bizzdesign B.V., 2003.

- [152] WJ van den Heuvel and HA Proper. De pragmatiek van architectuur. *Informatie*, (11), pages 12–17, 2002.
- [153] P. van Eck and R. Wieringa. Requirements Engineering for Service-Oriented Computing: A Position Paper. *Proceedings of the first international E-services workshop*, 1:23–29, 2003.
- [154] A. van Lamsweerde. Elaborating security requirements by construction of intentional anti-models. *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 148–157, 2004.
- [155] A. van Lamsweerde. Goal-oriented requirements engineering: a roundtrip from research to practice. *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*, pages 4–7, 2004.
- [156] A. van Lamsweerde et al. Goal-Oriented Requirements Engineering: A Guided Tour. *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, page 249, 2001.
- [157] A. van Lamsweerde and E. Letier. From object orientation to goal orientation: A paradigm shift for requirements engineering. *Proc. Radical Innovations of Software and Systems Engineering, LNCS*, 2003.
- [158] A. van Lamsweerde and L. Willemet. Inferring declarative requirements specifications from operational scenarios. *Software Engineering, IEEE Transactions on*, 24(12):1089–1114, 1998.
- [159] Axel van Lamsweerde. Requirements engineering in the year 00: a research perspective. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, pages 5–19, New York, NY, USA, 2000. ACM Press.
- [160] P Verschuren and H Doorewaard. *Het ontwerpen van een onderzoek*. Lemma Utrecht, 2000.
- [161] B. Vrijkorte. Semantics in service-oriented-architectures. Master's thesis, University of Twente, 2006.
- [162] DR Wallace and RU Fujii. Software verification and validation: an overview. *Software, IEEE*, 6(3):10–17, 1989.
- [163] K. Weidenhaupt, K. Pohl, M. Jarke, and P. Haumer. Scenario Usage in System Development: A Report on Current Practice. *IEEE Software*, 15(2):34–45, 1998.
- [164] D. Werth, K. Leyking, F. Dreifus, J. Ziemann, and A. Martin. Managing SOA Through Business Services—A Business-Oriented Approach to Service-Oriented Architectures. 2007.
- [165] K.E. Wiegers. Writing quality requirements. *Appeared in Software Development*, May, 1999.
- [166] K.E. Wiegers. When telepathy won't do: Requirements engineering key practices. *Cutter IT Journal*, May, 2000.
- [167] R. Wieringa. *Requirements Engineering: Frameworks for Understanding*. Wiley, 1996.
- [168] R. Wieringa. Traceability and modularity in software design. *Proceedings Ninth International Workshop on Software Specification and Design*, pages 87–95, 1998.
- [169] R. Wieringa. Requirements engineering: Problem analysis and solution specification (extended abstract). *Lecture Notes in Computer Science*, 3140:13–16, 2004.
- [170] R. Wieringa. Lecture notes pasr. 2006.

- [171] R. Wieringa, J. Gordijn, and P. van Eck. Value framing: A prelude to software problem framing. *1st International Workshop on Advances and Applications of Problem Frames (IWAAPF)*, 2004.
- [172] R. Wieringa, N. Maiden, N. Mead, and C. Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering*, 11(1):102–107, 2006.
- [173] R. Wieringa and I. NetLibrary. *Design Methods for Reactive Systems: Yourdan, StateMate, and the UML*. Morgan Kaufmann Publishers, 2003.
- [174] RJ Wieringa. Postmodern Software Design with NYAM: Not Yet Another Method. *Requirements Targeting Software and Systems Engineering*, pages 69–94, 1998.
- [175] G.M. Wijers. *Modeling support in information systems development*. Thesis Publishers Amsterdam, The Netherlands, The Netherlands, 1991.
- [176] SP Wilson, TP Kelly, and JA McDermid. Safety Case Development: Current Practice, Future Prospects. *Safety and Reliability of Software Based Systems-Twelfth Annual CSR Workshop, Bruges, Belgium*, 1997.
- [177] J. Xiang, L. Liu, W. Qiao, and J. Yang. SREM: A Service Requirements Elicitation Mechanism based on Ontology. *Computer Software and Applications Conference, 2007. COMPSAC 2007-Vol. 1. 31st Annual International*, 1, 2007.
- [178] S. YAMAMOTO, H. KADYA, K. COX, and S. BLEISTEIN. Goal Oriented Requirements Engineering: Trends and Issues. *IEICE TRANSACTIONS on Information and Systems*, 89(11):2701–2711, 2006.
- [179] Eric S. K. Yu and John Mylopoulos. Understanding why in software process modeling, analysis, and design. pages 159–168, 1994.
- [180] ESK Yu. Towards modeling and reasoning support for early-phase requirements engineering. *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 226–235, 1997.
- [181] K. Yue. What does it mean to say that a specification is complete? *Proc. IWSSD-4, Fourth International Workshop on Software Specification and Design*.
- [182] J.A. Zachman. A Framework for Information Systems Architecture. *IBM Systems Journal*, 38(2/3):454–470, 1999.
- [183] V.A. Zeithaml and M.J. Bitner. *Services marketing: integrating customer focus across the firm*. McGraw-Hill, 2006.
- [184] O. Zimmerman, P. Krogdahl, and C. Gee. Elements of service-oriented analysis and design. *IBM Developer Works*, 2004.
- [185] Z. Zlatev, P. van Eck, R. Wieringa, and J. Gordijn. Goal-Oriented RE for e-services. *International Workshop on Service-oriented Requirements Engineering workshop at RE*, 4, 2004.

Appendix I. GOAL ORIENTED RE

Introduction

A popular RE technique within problem-oriented RE is Goal Oriented RE (GORE), GORE has received a large amount of research efforts over the past years and its popularity has increased ever since. The main reason for this is the inadequacy of traditional system approaches (e.g. structured or object analysis), at the requirements level, these approaches treat requirements as consisting only of data and processes and do not capture the rationale for the systems, making it difficult to understand high-level concerns in the problem domain [87].

According to van Lamswearde [157] GORE ends where most traditional specification techniques would start, it focuses on the activities that precede the formulation of system requirements.

Goals are either an objective that the system should achieve through cooperation of agents in the software-to-be and in the environment [159] or, as Anton states it, goals are high-level objectives of the business, organization or system, they capture the reasons why a system is needed and guide decisions at various levels within the enterprise [11].

Goals may be formulated at different levels of abstraction ranging from high-level strategic concerns to low-level technical concerns. Goals also cover different types of concerns, functional and quality concerns.

GORE views the system-to-be and its environment as a collection of active components called agents, or as Anton describes them, agents are the entities or process that seek to achieve goals within an organization or system based on the responsibility that they must assume for the achievement of certain goals (e.g. in an electronic meeting system an meeting initiator is the agent responsible for calling, or initiating a meeting) [10][11]. A requirement is a goal whose achievement is the responsibility of a single software agent, while an assumption is a goal whose achievement is delegated to a single agent in the environment. Requirements implement goals the same way as programs implement design specifications [87] [157].

A basic tree structure is usually not enough to display goals. Firstly the distinction between soft goals and hard goals needs to be made. Soft-goals are goals that cannot be easily satisfied. Abstract, high-level and NFR are usually represented as soft-goals. Hard-goals are goals that are directly measurable and usually equate to concrete requirements that have a KPI attached. Hard-goals usually achieve soft-goals. Another extension is to include the actor / agent in the model, the notation of agent is a simple way to represent someone who achieves goals.

Van Lamswearde [159] also mentions why goals are needed:

- goals provide an exact criterion for sufficient completeness of a requirements specification, the specification is complete with respect to a set of goals if all the goals can be proved to be achieved from the specification and the properties known about the domain considered [181] [159]
- Explaining requirements to stakeholders is an important issue; goals provide the rationale for requirements. A requirement appears because of some underlying goal which provides a base for it. A goal refinement tree provides

traceability links from high-level strategic objectives to low-level technical requirements [159] [101] [89].

- Goal refinement provides a natural mechanism for structuring complex requirements documents for increased readability.
- Goal refinements provide the right level of abstraction at which decisions makers can be involved for validating choices being made or suggesting other alternatives overlooked so far.
- Goals have been recognized to provide the roots for detecting conflicts among requirements and for resolving them eventually;
- Goals drive the identification of requirements to support them, together with scenarios they are the driving forces for a systematic requirements elaboration process;
- A goal refinement tree provides traceability links from high-level strategic objectives to low-level technical requirements;
- Goal models provide a way to communicate requirements to customers. Goal refinements offer the right level of abstraction to involve decision makers for validating choices being made among alternatives and for suggesting other alternatives

Goal Identification

Goal identification is not an easy task. Goals could be explicitly stated by the stakeholders or in the various sources of information available to requirements engineers. However, most frequently goals are implicit and therefore the elicitation process must take place. A preliminary analysis of the current system/organization is an import source of goal identification. This analysis can result in a list of problems and deficiencies that can be formulated.

Goals can also be elicited from available documents, interview transcripts, process descriptions, etc, the suggestion is to find intentional keywords in the documents [159].

Anton [10] mentions that using process descriptions is very useful to identify an initial set of goals, although it is not wise to only use process descriptions. Other sources of information, such as interviews, diagrams should be used to supplement the goal identification process.

Stakeholders express their requirements in terms of operations or actions, rather than goals. It makes sense to look for action words such as schedule or reserve when gathering requirements for a meeting scheduler system.

Van Lamsweerde mentions [159]: although goal based reasoning is appropriate for RE, goals are hard to understand for some stakeholders. Therefore scenario-based elicitation is a good alternative for stakeholders to discuss the system. Therefore some techniques have been developed to elicit goals using scenarios. Potts also suggests [] that it is unwise to apply goal based requirements methods in isolation and says that they should be complemented with scenarios [121] [133] . Some techniques interpret scenarios as containing information on how goals can be achieved (operationalize goals) [11] [65], other techniques suggest that a goal is

only a contextual property of a use case [73] [91](i.e. it is a property that relates the specific scenario to its organizational context).

In [133] Rolland describes that it is wise to couple goals and scenarios together in a bidirectional manner, scenarios are used to elicit goals and goals are used to elicit scenarios. Another important characteristic of this approach is the distinction between the refinement relationship and the AND/OR relationship among goals. Thirdly they suggest methodological guidelines on how to associate goals with scenarios.

Goal Analysis

One of the steps in analyzing goals is to refine the initial goals, goal refinement is a core activity within any GORE approach and therefore every technique has its own ways to refine the initial set of goals. We will try to discuss the essence behind goal refinement without discussing every GORE technique in detail.

After identifying the goals, goals can be refined to simpler goals until it is no longer possible to obtain simpler goals. This is usually done by asking HOW questions and refining goals through AND/OR refinements. Many GORE approaches mention when determining how a high-level goal can be refined, you need to consider alternative ways of refining it to make sure that as many options are explored [28].

When eliciting more abstract goals (e.g. going up in the tree), you can ask yourself WHY questions. An important reason to find more abstract goals is that once a more abstract goal is found, it is possible to refine it and find important sub goals that were originally left undetected. The identification and further refinement of higher-level goals lead to more complete goal models and thus to more complete requirements specifications.

Within goal based reasoning, satisfying goals is an important topic. Goal satisfaction can either be qualitative or quantitative. Letier and Van Lamsweerde prefer a qualitative approach to a quantitative approach, because qualitative techniques are much richer in detail.

When using a quantitative approach goals, goal satisfaction can either be absolute (goals need to be satisfied completely) or partial degrees in which goals can be satisfied.

Goal based methods

Some popular GORE methods are

- KAOS [41]
- I*/ tropos [180] [32]
- GBRAM [11] [10]
- ORDIT [43]
- NFR framework [36] [104]
- B-scp

In the section below we will discuss some of the available GORE methods.

KAOS

KAOS is a GORE approach, with a large amount of formal analysis techniques; it stands for Knowledge Acquisition in auTOMated Specification. It is a multi paradigm

framework that allows combining different levels of reasoning. KAOS is focussed on goal satisfaction and the systematic building of complete, conflict-free goal based requirements modeling [155].

KAOS is semi-formal for structuring and modeling goals, qualitative for selection among the alternatives and formal for accurate reasoning. In KAOS a goal is a prescriptive statement of intent about some system whose satisfaction in general requires the cooperation of some of the agents forming that system [154]. Goals in KAOS can be both functional and non functional (soft goals). Goal refinement ends when every sub-goal can be realized by some agent. These end goals (terminal goals) become the requirements of the system. Overall a KAOS specification is a collection of the following models:

- goal model (where goals are represented and assigned to agents)
- object model (a UML model that can be derived from formal specification of goals since they refer to objects or their properties)
- operation model, which defines various services to be provided by software agents

More information about KAOS can be found here [41] [42] [157] [156].

NFR

The NFR framework deals with Non-Functional requirements [104] [36]. It concentrates on modeling and analyzing non functional requirements. The main goal of this framework is to get the developer think of non functional requirements in system development. It comprises the following activities: capturing NFRs for the domain of interest, decomposing NFRs, identify possible NFRs operationalizations, dealing with ambiguities, tradeoffs, priorities and interdependencies among NFRs, selecting operationalizations, supporting decisions with design rationale, and evaluating impact of decisions. The main idea of this approach is to model and refine non-functional requirements and to expose positive and negative influences of different alternatives on these requirements.

GBRAM

GBRAM (Goal Based Requirements Analysis Method) [11] [10] is a GORE method that emphasizes on the initial identification and abstraction of goals from various sources of information. This method is useful to identify, elaborate, refine and organize goals for requirements specification. The method discusses goals from two views, goal analysis and goal evolution. The method assumes that no goals have been documented or elicited from stakeholders and thus can use existing diagrams, textual statements, interview transcripts, etc. Goal analysis is about the exploration of information sources for goal identification followed by organization and classification of goals.

Goals refinement concerns the evolution of goals from the moment they first identified to the moment they are translated into operational requirements for the system specification.

GBRAM helps in goal elicitation and refinement by providing requirements engineers with standard questions. In GBRAM, goals, agents, stakeholders are specified in textual form only, called goal schemas, the method does not provide a graphical representation.

i*

i* is a technique that focuses on modeling and reasoning support for early phase requirements engineering. It tries to capture the understanding of the organizational context and rationales that lead up to systems requirements. It consists of two main modeling components. The Strategic Dependency (SD) model is used to describe the dependency relationships among various actors in an organizational context. The Strategic Rationale (SR) model is used to describe stakeholder interests and concerns, and how they might be addressed by various configurations of systems and environments [179] [180].

i* can be used for both early and late phases of RE. During the early requirements phase i* is used to model the environment of the system to be, it facilitates the analysis of the domain by allowing the modeller to diagrammatically represent the stakeholders of the system, their objectives and their relationships [87]. During the late phases the i* models are used to propose the new system and the new processes and evaluate them on how well they meet the functional and non-functional needs of the users.

Tropos is an extension framework for i*, it is a requirements driven agent oriented development methodology [28] [87]. Tropos guides the development of agent-based systems from the early requirements analysis through architectural design and detailed design to implementation [87].

ORDIT

ORDIT [43] describes an approach to RE within the context of organizational change, it focuses on the representation of organizational requirements in the design of socio-technical systems which are intended to emphasize the relationships between organizational structure IT systems. The model contains a process model, an enterprise modeling language, an information modeling language, a role reference model and supporting tools, with a focus on the process and enterprise model.

B-SCP

B-SCP provides an integrated framework for i* goal modeling, Jackson context diagrams and Role Activity Diagrams (RADs), the goal model is integrated with context diagrams via a problem diagram framework, in a progression of problems from the strategic-business level problem to the system level problem. A critical step in B-SCP is to scope the strategic-level business problem diagram using the Weill and Vitale business model framework. RADs are connected to the goal model and context diagrams by cross-referencing according to mapping rules [22].

In evaluation the technique they found it promising but in its current state not without its fair share of difficulties.

In Table 20 we show the different GORE methods with their main goals, advantages and disadvantages.

Method	RE phases	Main goal	Advantages	Disadvantages
KAOS	Specification	Relating business goals to functional and non-functional system components	Well accepted, modeling techniques, thorough	Too formal, expects knowledge of discrete mathematics

GBRAM	Specification	Relating business goals to functional and non-functional system components	Provides insight in techniques to use to elicit goals	No modeling technique
I*/TROPOS	Elicitation/Analysis	Elicit early requirements	Focuses on rationale behind requirements	Impractical modeling
NFR	Specification	Analyze solutions based on non-functional requirements	Modeling technique, gives attention to NFRs	Does not focus on elicitation, analysis. Other techniques can be used to acquire requirements
ORDIT	Elicitation/Analysis	Understanding the current organization situation	Complete process model, focuses on early requirements	Little empirical validation (that we were able to find)
GQM	Validation	Validate specification to stakeholder goals	Easy to use well known validation technique	
B-SCP	Elicitation/Analysis	Provide practical goal oriented method	Complete, tries to combine multiple techniques	Not practical to use according to the writers

Table 20: summary of analyzed goal methods

Appendix II. RE FOR BESPOKE SYSTEMS

Introduction

Requirements engineering for tailor made systems is the traditional RE. This is where the requirements engineer analyzes the wants of the organization for a tailor made software solution. The most important facet of RE for bespoke systems is that you get what you specify. It is therefore of utmost importance to get the requirements as detailed and correctly as possible. A complete requirements specification ensures that the delivered system meets those wants.

Activities

The main activities within RE for bespoke systems are elicitation, analysis, negotiation, validation and verification. They all have the common goal to get a clear and correct RS [84] [83] [21] [27].

Pre-elicitation activities

Requirements Engineering starts when somewhere in the organization the need for change is identified and the decision is taken to create an (IT) solution for it [21].

Requirements frequently start with a vague statement of intent; the first problem is to establish the boundary of investigation and the scope of the intended system [143].

In [108] the author describes that some assessment of a project's feasibility and associated risks needs to be undertaken, and RE plays a crucial role in making such an assessment. Groundwork also includes the identification of a suitable process for RE, and the selection of methods and techniques for the various RE activities. A technique prescribes how to perform one particular activity and a method provides a prescription for how to perform a collection of activities, focusing on how a related set of techniques can be integrated and providing guidance on their use.

Stakeholders are an important source of information during the elicitation phase, but for some reason stakeholders, and particularly, stakeholder identification have been somewhat neglected in the RE literature. A few stakeholder identification techniques can be found here [137] and [1]. Adequate, timely and effective consultation of relevant stakeholders is of great importance in the requirements engineering process, missing stakeholders result in missing requirements and escalating project costs [137].

We will define a stakeholder as [1] [137]:

Stakeholders are individuals or organizations who stand to gain or lose from the success or failure of a system.

Stakeholders can be categorized as follows, end-users, managers, engineers, customers and external bodies such as regulators, domain experts and so on. They all have different goals and will try to satisfy their own without recourse to others [137].

A stakeholder can be someone who finances the project; someone whose skill is needed to build a product, such as a network specialist or a usability expert, an organization whose rules developers must obey, such as a tax lawyer firm or a standards institute or an external organization that can influence project success [1].

Elicitation

The first phase in RE [108] is eliciting requirements and has received a great amount of attention in the scientific literature, because the activities in this phase contain

aspects of several different fields, amongst which sociology, computer science and anthropology.

In the past this phase was considered as common sense and just a part of the analysis process. The term elicitation is preferred to capture to avoid the suggestion that requirements are out there to be collected simply by asking the right questions [108], this phase is important because users hardly know what they want and stakeholders have conflicting interests [27].

During elicitation there are three main concerns, what information to elicit, from whom and what techniques can be used to elicit the information.

One of the most important goals of elicitation is to find what problem needs to be solved, and hence identify system boundaries. These boundaries define where the final delivered system will fit into the current operational environment. Identifying and agreeing a system's boundaries affects all subsequent elicitation efforts. The identification of stakeholders and user classes of goals and tasks, and of scenarios and use cases all depend on how the boundaries are chosen. A second goal in elicitation is to gather a list of problems requiring solution and thirdly to find out what the client imposed constraints are [27].

In reviewing the literature we have found two definitions that describe requirements elicitation very well, and both definitions supplement each other rather well.

Kotonya and Sommerville provide the following definition for requirements elicitation [83]:

The activity that encompasses learning about the problem to be solved, understanding the need of potential users, trying to find out who the user really is and understanding all the constraints on the solution.

Cheng and Atlee provide another definition [34]:

Requirements elicitation comprises activities that enable the understanding of the goals, objectives, and motives for building a proposed system. Elicitation also involves identifying the requirements that the resulting system must satisfy in order to achieve these goals.

Information gathered in this phase often needs to be interpreted, analyzed, modeled and validated before the requirements engineer can feel confident that a complete enough set of requirements has been collected.

A good requirement should be traceable to business objectives and should be related to system lifecycle components. It should be consistent with the scope and constraint of the product, incorporate stakeholder expectations, should be measurable against acceptance criteria, and should be maintainable over the project's lifecycle [40].

Techniques

In requirements elicitation are a number of techniques available where the requirements engineer can choose from. This choice depends largely on the time and resources available to the requirements engineer. Easterbrook, Nuseibeh [108] and Goguen, Linde [53] distinguish a number of available elicitation techniques:

- Traditional techniques include a broad class of generic data gather techniques. One can think of the use of questionnaires and surveys, interviews and analysis of existing documentation such as organizational charts, process models or standards and user of other manuals of existing systems.
- Group elicitation techniques aim to foster stakeholder agreement and buy-in, while exploiting team dynamics to elicit a richer understanding of needs. A focus group is a group interview, groups are brought together to discuss some topic of interest to the researcher. Focus groups have the advantage of allowed more natural interactions between people than questionnaire interviews, or even open ended interviews.
- Prototyping is used for elicitation where there is a great deal of uncertainty about the requirements or where early feedback from stakeholders is needed. This can be combined with other techniques (e.g. scenario's).
- Model-driven techniques provide a specific model of the type of information to be gathered and use this model to drive the elicitation process (e.g. goal-based methods and scenario based methods).
- Cognitive techniques include techniques such as protocol analysis, laddering and repertory grids. Protocol analysis asks a subject to engage in some task and concurrently talk aloud, explaining his/her thought process
- Contextual techniques, these include the use of ethnographic techniques (e.g. participant observation)
- Introspection amounts to imagining what kind of system the designer wants if he was doing the stakeholders job. This method can be useful, but it has the problem that the introspection of an expert in a different is unlikely to reflect the experience of actual users. They conclude with that introspecting should be carefully checked with methods like interviewing or protocol analysis.
- Viewpoints in RE have been recognized as a practical tool to elicit and validate requirements [82] [139] [49] [92]. Viewpoints take the perspective of a stakeholder, actor or role to discover or validate system requirements.

In addition to the techniques mentioned above, [129] identifies an additional number of elicitation techniques, which she calls trawling because like fishing she runs a net through the organization and traps as many requirements as she can (We will only discuss the techniques not mentioned above).

- Abstraction, this technique is based on making useful generalisations. The purpose of the generalisations is to understand and articulate the rules that apply to a specific domain of knowledge and even to discover rules that are shared between domains.
- Business events, this involves the investigation of some activity, work or business in the world. This can be used to identify the boundaries of the project.
- Neurolinguistic programming, this is a technique to learn how to get a better understanding of what people say.
- Reusing requirements, if requirements are consistently documented, then requirements can be reused.

Analysis

After the elicitation phase it is time to analyse the gathered information. At this point the requirements engineer has a description of the problem domain, a list of problems requiring a solution and the client imposed constraints on the solution. In [27] Bray provides us with the following definition for Requirements Analysis:

Through study of a problem domain, the achievement of understanding of and the documentation of the characteristics of that domain and the problems (requiring solution) that exist within that domain.

This phase discovers the, sometimes abstract, system requirements and the activities comprise both modeling and analyzing.

There are many possible ways to analyze the initial set of requirements, the traditional techniques (e.g. structured analysis, modern structured analysis, object oriented analysis, NYAM) or the more newer techniques that focus on problem analysis (e.g. problem framing, theory building, goal based analysis).

The traditional techniques focus more on modeling the proposed solution, instead of analyzing the problem to be solved. Most techniques even take a readily available requirements document for granted.

The traditional methods have been around such a long time, that We will describe them only briefly. For more information We will refer to their originating sources.

Techniques

The literature recognizes a number of techniques that can be used [27] in requirements analysis. We will summarize them below.

Structured Analysis

Before Structured Analysis (SA) requirements documents were usually written in poorly maintainable textual specification documents, SA changed all this with a graphical notation for systems analysis [136]. Ross and Schoman argue in their seminal paper about structured analysis that design methods prior to structured analysis were ad hoc at best and there was a need for a structured design method in requirements definition. They discuss the need of problem analysis, although compared to current knowledge SA is more solution specification, before actually building the system. They try to answer this by asking three questions, why are we going to build the system? To answer this question they introduce the context analysis phase. This phase describes the boundary conditions of the system by describing certain economic, operational and technical feasibilities.

The second question focuses on: What is to be build? This introduces the concept of functional specification. Lastly they argue that you need to answer how a system is build and introduce design constraints as the answer.

SA mainly tries to model and understand the current solution system, in the time that SA was introduced these were mainly non-automated work procedures, that operates within the problem domain and has a similar function to the new system required [27].

To model the data, SA uses data flow diagrams (DFD) and later down the road Chen [33] introduced a data structure diagram, usually in the form of an entity relation diagram (ERD).

After a while researchers and practitioners started to notice there was something wrong with SA, this ultimately led to modern structured analysis (proposed by

Yourdon), the main differences between SA and modern structured analysis are that modern structured analysis ignores the modeling of the current pre-existing system, it adds an preliminary phase to develop an "essential" model, it introduces top-down functional decomposition, places more emphasis on information modeling (ERD) and behavior modeling (state transition diagrams); and finally prototyping is given a more important role.

SSADM

SSADM is a method based around the principles of SA. We will discuss this method briefly. For more information about SSADM read [13]. SSADM introduced entity life histories (ELH) and explicit stated requirements to SA.

SSADM is based around a number of basic principles, these are:

- data-driven
- differentiation between logical and physical
- different views of the system
- both top-down and bottom-up
- user involvement
- quality assurance
- self documenting

SSADM contains 6 stages, stage 1 is the analysis of systems operations and current problems, stage 2 contains the specification of requirements, stage 3 describes the selection of technical options, stage 4 is all about data design, in stage 5 processes are designed and finally in stage 6 the physical design of the system is made.

Object oriented analysis

Object oriented analysis (OOA) adopted the ideas behind OO programming, instead of modeling objects from the solution domain (solution system), OOA focuses on modeling objects from the problem domain, although according to Michael Jackson the problem domain is much too rich to be captured in terms of objects [72].

Another problem with this kind of analysis is that both modeling techniques in OOA and OOD are very similar and objects appear to be in both the problem and solution domain, the line between problem and solution is hard to distinguish.

But when an OOA approach is chosen, the outline of the method is usually like this [27]:

- identify objects within the problem domain
- define attributes and methods of those classes
- define the behavior of those classes
- model the relationships between the classes

The main problem with OOA is that isn't really analysis, it still assumes the existence of a requirements document. This means that OOA assumes that problem domain analysis and specification already has been accomplished. OOA is suitable for high-level systems design.

Problem framing

Problem framing is a technique that Michael Jackson introduced in RE [72], the idea behind problem framing is to use a generalization class of common problems. A problem is structured according to principle parts and a solution part. If other problems fall into the same frame, it is possible to reuse the techniques and methods used to solve that problem. A problem frame describes a general problem, so it is possible that the names of the principle parts are not the same, you can solve this problem by looking for the characteristics of the principle parts of the problem frame and the problem under investigation. A rewarding advantage for finding the right problem frame is that it comes with an efficient method to solve the problem. Jackson mentions the following example frames:

- Simple IS frame
- Simple control frame
- JSP frame
- Connection frames
- Work piece frames

Every frame mentioned above comes with a set of techniques and methods to analyze the problem, for example the content for an information problem comprises [27] the data model for the sub-domains, the characteristics of each problem sub-domain, how the system can access the state of the relevant sub-domains and events in the problem domain, distortions and delays introduced by any connection domain, where initialization data are to be extracted from existing files and the queries to be supported and the relevant responses.

A more advanced problem frame is the multi-frame problem, a number of several simple problems behave as one realistic problem.

According to [171] this technique is hard to use in multi-business information systems, because multiple businesses are involved each with their own distinct understanding of a certain problem frame. The authors propose a technique to use e3-value networks to develop a common understanding of the problem frame, although value oriented thinking is not yet common in requirements engineering. They conclude with that it is still uncertain if value modeling is suited for framing software problems.

In [39, 38] an approach is proposed and analyzed to use business models as a basis to create problem frames. They tested this approach in a real-world setting and concluded that there was much resistance from the company to use the problem-frame approach. Also, they encountered a number of problems using this approach. It was often impossible to identify a suitable problem-frame. They question the validity to use problem-frames in an (e)-business context, because problem-frames emerged from more traditional Software Engineering (SE) problems, these problems are not always the same problems encountered in businesses.

In [23] the ideas behind the approach above have been revamped, a new approach is proposed containing a business strategy dimension. The approach tries to achieve this by using Problem Diagrams (problem diagrams are used to create problem

frames), goal modeling and business process modeling. The organization's business strategy is represented as a problem diagram, where the context part of the diagram represents business model context and the requirement part of the problem diagram represents the objectives of the business strategy and business processes, modeled using goal-based RE techniques. In order to derive at the system requirements, the model goes through a number decomposition phases. Goals are used to increase traceability from high-level strategic objectives to low-level system requirements. Business process modeling is used to gain more understanding about the problem where more detail is needed to understand the needed requirements; the goals that the business processes achieve are represented in the goal model.

The above two techniques led to a problem frame based requirements analysis framework, incorporating business strategy, business processes and GORE techniques. This approach (B-SCP) is focused to be applicable in industry, but still lacks empirical testing [22] [178].

Negotiation

After the initial specification and analysis of requirements an initial set of requirements emerges. This set of requirements still needs further analysis to be of any use, requirements from different stakeholders can conflict with each other and some are more important than others. In requirements negotiation these issues are to be resolved. Easterbrook provides the follow definition for requirements negotiation [44]:

Requirements negotiation is a collaborative approach to resolving conflict by exploration of the range of possibilities, it is characterize by the participants attempting to find a settlement which satisfies all parties as much as possible

The best way to resolve these issues is to discuss these requirements with the different stakeholders. This can be done in group meetings or in single meetings. In the end the requirements have to be prioritized and an agreement on them has to be made [21].

Negotiation itself is all about structuring issues between stakeholders and to develop alternatives to solve problems, after negotiation stakeholders eventually agree on the solution that has the most support between the stakeholders. Negotiation is performed around three dimensions: conflict resolution strategy, collaboration situation and negotiation support tools [14].

Requirements negotiation has the following benefits [14]:

- Understanding project constraints, negotiating about the requirements makes the stakeholders aware of budget and time constraints;
- Adapting to changes, negotiating makes stakeholders aware of alternatives and in times where alternatives are needed stakeholders are already aware of them;
- Better team learning, because different stakeholders with different experiences participate in the negotiating process they transfer their knowledge to the other participants;

- Discussing requirements helps bringing unspoken thoughts and hidden issues to the negotiation table;
- In large requirements projects there are a large number of stakeholder goals, discussing these goals helps to reduce complexity;
- Negotiating about requirements helps to create a shared vision between stakeholders;
- Negotiation helps to get a better solution system, because of a shared understanding of project goals and the solution is based on a subset of requirements that is beneficial to all involved parties.

In [50] Firesmith discusses the need for requirements prioritization, he mentions the following benefits for requirements prioritization:

- it is possible to construct a schedule that focuses on the important requirements first
- improved customer satisfaction
- stakeholders discuss all requirements and not just their own
- lower risk of project cancellation
- it is possible to estimate project returns better
- investments get prioritized

Berander and Andrews [16] performed a literature review about requirements prioritization and mention that using the right requirements is fundamental to the system's success. It doesn't matter how solid the end-product is, if you've build the wrong system. The challenge is to select the right requirements from the given set of requirements and to realize this one has to prioritize the requirements. Some organizations experience prioritization as easy, some as very hard, but both types of organizations agree that it is a **fundamental** activity. The most common aspects of prioritization are importance, penalty, cost, time, risk and volatility.

There are numerous challenges and associated risks that must be addressed when prioritizing requirements, including [50]:

- **Mandatory nature of requirements**, a requirement is by its definition mandatory so it is hard to agree upon what is needed the most.
- **Large number of requirements**, in large projects there can be hundreds of different requirements, prioritizing these can take up a large amount of time.
- **Limited resources**, implementing all requirements is costly so a set of requirements has to be chosen that captures the essential functionality of the system and is affordable.
- **Quality requirements are under used**, quality requirements are often under-specified and therefore will not be prioritized correctly.
- **Changing requirements are difficult to prioritize**
- **Incompatible requirements**, some requirements negate or make implementing other requirements very hard (i.e. performance versus maintainability)

Techniques

There are many ideas and methods within RE that try to solve the problem of negotiating a set of requirements. Some methods use goal based reasoning, viewpoints, qualitative reasoning and many formal methods.

REMAP [79] captures the history of design decisions in the early stages of systems development in a structured manner. From the requirements analysis phase it records the essence of discussions/deliberations of requirements, or the rationale behind the modeling techniques used.

SIBYL [79] is a system designed to help users represent and manage the qualitative aspects of the decision making process, such as the alternatives under discussion. SIBYL is organized around decision graphs which record the positive and negative aspects of choosing from a set of alternatives to satisfy a goal.

The following requirements prioritization techniques are available [50, 16]:

- ROI calculation (determine which set of requirements is more favorable in financial terms)
- Pair wise comparisons
- Prioritization workgroups
- 100 dollar test (give every stakeholder "100 dollars" and let him divide it among the requirements)
- Voting schemes (each stakeholder is allowed to cast his vote on a number of requirements)
- Weightings(weight each requirement from a specific stakeholder)
- Boehms Win-Win technique [25] (an iterative conflict-goal management technique for stakeholders)
- Quality Function Deployment (QFD) (a group decision making technique often used in product or service development.)

Specification

After analyzing and negotiating the requirements it is time to document the requirements into a single specification document. Bray [27] distinguishes two requirements documents, firstly the requirements document that describes the problem domain and the problems to be solved, secondly a requirements specification that describes the behavior of the solution system that solves the initial problems (requirements), this document is the intended output of the RE process. The first document can be used as an input document for the second.

Where Bray clearly uses the problem based requirements document as an input to create the requirements specification, Wieringa [169] takes a more separatist view (as explained in the introduction). He clearly distinguishes that a requirements specification either describes the problematic phenomena or, when RE is seen as solution specification, the solution system behavior.

A first step in specifying external behavior is transforming the requirements into functions of the system, usually there isn't much difference between a requirement and a function of the system, although requirements shouldn't mention the system under development (SuD), functions can mention the SuD. Mapping requirements into functions is a very abstract way of describing functional behavior. A less abstract way in describing system behavior can be done using formal methods and models (e.g. data-flow diagrams and state-transition diagrams) [27].

Requirement documents/specifications are usually build around standard templates, a few templates found in the literature are the Volere template, IEEE 830 standard and the specification template by Bray [27].

Formal specification

A requirements specification is formal when it uses a mathematical notation to specify the requirements. The advantages of using a formal specification are that it is easier to check the model for inconsistencies and implementations can be derived that could be shown to have the correct behavior. Although formal specification sounds good in theory, it hasn't been adopted in practice because it requires the practitioner to possess a high degree of understanding of predicate calculus. A few formal specification methods are the Vienna development method [20], Albert II [64] and Z [119].

Validation

After the requirements are documented they need to be validated. Validation means that you need the stakeholders to agree that the specified requirements meet their needs and find any errors that occurred during the RE phase. Bray [27] distinguishes again between the requirements document and specification. The requirements document describes something that exists (thus can be wrong that it describes the wrong thing), the requirements specification describes something that does not exist yet, it cannot be wrong in the same sense (it can describe the wrong behavior). The literature describes a number of specific validation methods, e.g. rapid prototyping [9], validate requirements through the use of viewpoints [90], validation methods based on scenarios [142] [138], safety cases (proof that an application is safe to use in a safety critical situation) [176], visualization of requirements (e.g. using multi-media techniques) [31] [100] [47], validation through natural language parsing [105], using GQM for requirements validation [79] and even role playing use cases [17].

But as Sutcliffe [143] describes it, the state of the art in requirements validation is still discussing the requirements with the relevant stakeholders in a semi-formal way.

Techniques

Most methods described in the literature are somewhat a variation of general validation techniques, which we will describe below [27, 21]:

- simple checks
- document reviews
- logical analysis
- prototyping, use of scenarios
- functional test planning
- writing the user manual
- goal tracing

Simple checks

This is a straight forward technique where the requirements engineer checks if the requirements document contains all requirements and if the requirements specification contains all the requirements mentioned in the requirements document.

Document reviews

Document reviewing is one of the oldest techniques for validating requirements; a review is an examination of the requirements document. In software-engineering this technique is known as a design review [45, 112].

The goal of a requirements document review is to get constructive criticism, so errors are found.

The following roles can be identified in requirements reviews [27]:

- the presenter
- the scribe
- referee
- domain experts
- requirements experts
- client representatives

Formal validation

When a requirements specification is formal enough, logic can be used to determine the validity of the specification, one method to check formal specifications is the QC-logic approach [70], or using logic to check formal TROPOS specifications [51], or validation using Albert II [64] or the SCR method [63]. When behavior is specified using finite state machines (FSM) (e.g. mealy diagrams, STDs) formal reason can be used to check for:

- every event is recognized in at least one state
- every transition has only one trigger
- every state is reachable by one transition at least

Prototyping (and scenarios)

In [130] the authors describe that using a prototype is a good way to find more requirements and visualize requirements that are hard to understand. The prototype constructed in this phase is not meant to evolve into the final product, it is only used for elicitation and validating. A prototype can be seen as a simulation of the product to be, it describes in a clear visual way the behavior of the system, prototypes are particularly useful when combined with scenarios (use cases). The prototype then enacts the use case.

Ideally speaking, these prototypes are created from the requirements specification automatically, this eliminates the possibility that mistakes are made in creating the prototype. This is possible when sufficient formal languages have been used (e.g. state chart diagrams).

Scenarios

Scenarios by themselves are also an excellent way to validate the requirements. The scenario describes a certain business event where the system needs to respond to and describes a set of tasks that needs to be executed by the users of the system.

The literature describes many variations on scenario validation (e.g. scenarios using Bayesian Belief Nets [142], a iterative method of assessing the implications of human error on system requirements [146] and scenic (a method that uses the inquiry cycle for validation) [122]).

Functional tests

After the specification process it is possible to start designing the functional tests of the system. It is not yet possible to execute those tests, but designing them helps the engineers find specification errors [27].

Writing the user manual

The rationale in writing the user manual after the specification is the same as above with the functional tests. Writing the user manual forces a close examination of the specification. This extra examination will make any omission and inconsistencies more apparent.

Requirements verification

Requirements verification is the activity that makes sure the product has been built right. After the product has been built, checks are done to make sure the product meets the requirements. The easiest way to perform requirements verification is to organize an acceptance test. An acceptance test is used to obtain confirmation by the client that the delivered product meets all his demands (requirements), the general way of doing this is to use the requirements specification. Specification based test case generation is a kind of black box testing because the software is viewed as a black box that transforms inputs into outputs based on the specification of what the software is supposed to do. This approach considers the external views of the software and hence is a testing of the product, but not the design decisions or program code [93]. For information about software verification and acceptance testing the following articles describe the process and the state of the practice [162] [67].

Techniques

- Acceptance tests

Existing methods

Within RE a number of methods emerged to guide the requirements engineer during the RE process. In the section below we will discuss a number of these methods.

To discuss goal based RE (GORE), we will adopt the work of Lapouchnian, who did a review of available GORE techniques [87].

ACRE

In [99] Maiden presents a framework that assists requirements engineers in choosing techniques for requirements elicitation (ACRE). Practitioners are often unaware of the range of methods available. They argue in the paper that more than one acquisition method is needed to capture the full range of complex requirements for most complex systems. ACRE provides methods for acquiring requirements from stakeholders, rather than for mining requirements out of documents. ACRE acquires both requirements for the system and knowledge about that system's domain and environment. The framework offers 12 acquisition methods (roughly the above mentioned techniques).

VORE

In [82] [139] Kontonya and Sommerville describe a way for eliciting requirements using viewpoints. They argue that it is widely accepted that a multi-perspective approach to requirements engineering can, potentially, lead to requirements specifications which are more likely to satisfy the needs of a diverse set of system

stakeholders. Good requirements engineers will always consider these different perspectives but viewpoint oriented requirements engineering methods which explicitly identify and separate different system viewpoints are not widely used in practice.

They designed the viewpoint method to be as flexible as possible, their rationale behind this is that although a method may be conceptually elegant, stakeholders and end-users are reluctant to adapt to the structures imposed by methods.

The problem with more structured approaches is that they are too rigid. They are based around the idea of a single type of viewpoint and require the specification to be fitted around this concept. They have their own ideas and do not have time to think about how to present them in what is (to them) an artificial framework.

Viewpoints may be used during the early stages of a requirements engineering process as a structuring mechanism for requirements elicitation and analysis.

NYAM

A method that contains all these elements is NYAM (Not Yet Another Method) [Wieringa1998a, [173]. NYAM combines the essence of both SA and OOA methods thus combining the best of both techniques. SA is very strong in explaining the external functions of a system, while OOA techniques are suitable to describe the internal decomposition of a system. NYAM contains functional, behavior, communication and decomposition techniques. Functions can be specified using an initial mission statement that describes the goal of the system and what it will and will not do. This mission statement can be refined with a function decomposition tree. The root node is the initial mission statement, while every leaf of the tree contains an external systems function. An event-response specification describes the reactions (responses) of the system to certain events. Behavior is specified using mealy machine diagrams and sequence diagrams, communication is specified with a communication diagram and sequence diagrams and finally the decomposition of the system is specified with class diagrams and a function decomposition table.

Scenario based methods

Scenario based techniques are popular techniques [107], partially because they are easy to use for practitioners and non-technical stakeholders [34].

A scenario is a sequence of interaction events between the system to be and its environment in the restricted context of achieving some implicit purpose. There are a number of scenario-based techniques proposed to help assist the RE process (e.g. elicit requirements in hypothetical situations [123], to help identify exceptional cases [120], to make abstract models more complete [134], to validate requirements in combination with prototyping [144]).

Use of scenarios is usually wide spread, scenarios can be used to describe the environment of the system, without talking about the system under development, but usually they describe user-machine interactions [135]. Scenarios can be used to elicit new requirements, but they are mostly used in the later phases of requirements engineering as they describe the intended use of the system.

Usually users start by identifying large scope scenarios that cover their current organization and processes and refine them to more detailed scenarios in response to inquiries from designers.

Stakeholder involvement is important because it helps designers avoid interpreting system requirements too early. Scenarios are also used in different phases of

requirements engineering, they can start as a tool for elicitation, secondly they can be used as test cases to validate a current model the of system during refinement and finally they can become test data to verify that the final system meets the requirements expressed in the scenarios.

In [163] the authors conducted a survey of available scenario usage in practice. They concluded that scenarios where used in the following situations:

- Use Scenarios when abstract modeling Fails.
- Use scenarios to explain prototypes.
- Scenarios as means for complexity reduction.
- Scenarios as means for reaching partial agreement and consistency.
- Reflection on static models.

Van Lamsweerde mentions the positive aspects and negative ones of scenarios in [158].

On the positive side, scenarios provide an informal, descriptive and concrete way of describing the dynamic aspects of environment interactions. They can therefore be used easily by multiple stakeholders with different backgrounds to build a shared view of the "visible part" of the system. Scenarios are also applicable to use in a variety of purposes in the RE lifecycle (e.g. elicitation, analysis, validation) .

On the downside, scenarios are naturally incomplete, in the case of interaction sequences they are very procedural (thus introducing risks of over-specification), not entirely suitable for the early stages of RE and they leave required properties about the envisioned system implicit (these need to be made explicit in order to support analysis, negotiation, etc)

There are a number of popular scenario based techniques and models available such as the formal scenario analysis [68], the inquiry-based requirement analysis [123], SCRAM [144], RESCUE [96] and the CREWS project [117].

These methods are a general framework for RE and use scenarios for elicitation, analysing, validating and verification. A description of these methods can be found in the chapter below (mostly adapted from [52]).

Formal scenario analysis

Hsia and associates proposed a formal approach to scenario analysis, a requirement analysis model in the early phases in software development. Their approach defines systematic development stages from the initial semi-formal scenarios to the final formal scenarios. Each of the stages, except the first, use scenario's in formal notation, which enable the system analysts to derive part of the system requirement specification. More information about this model can be found here [68].

Inquiry Cycle

Potts *et al*/developed the Inquiry Cycle Model of requirement analysis, which is "a structure for describing and supporting discussions about system requirement." The model consists of three phases of requirements: documentation, discussion, and evolution. These three phases make a cycle for acquiring and modeling the knowledge of problem domain. Their scenarios are part of the requirements documentation, which are in hypertext form. The scenarios are intended to be

semiformal; in fact, they are expressed in tabular notation. "In the broad sense, a scenario is simply a proposed specific use of the system". More specifically, a scenario is a description of one or more end-to-end transactions involving the required system and its environment." More information can be found here [123]

CREWS

The CREWS project [117] (Cooperative Requirements Engineering with Scenarios) is a large European project on scenario use in requirements engineering. Based on existing techniques and tools, the project is trying to make scenario usage a systematic engineering discipline; hence, its approach to scenario usage is tool- and/or method-driven. The project has specific goals. With the multimedia recorded system use, it provides a common medium for discussing system requirements among multiple stakeholders. With natural language understanding, it provides a device to elicit requirements in a graphical representation. By cooperative requirement animation and by comparison between a specification and its test scenarios, it expedites the validation process of a system. In the CREWS project there is a framework to classify scenarios in scenario-based approaches. They give four dimensions: the form view, the contents view, the purpose view, and the lifecycle view. The form view represents the format of scenarios; for example, narrative texts, graphics, images, videos, and prototypes are in the form view. In addition, formality (e.g., formal, semi-formal, and informal) is discussed from this viewpoint. The contents view expresses what knowledge scenarios express. It consists of four elements: abstraction, context, argumentation, and coverage. In the purpose view, scenarios are categorized from the reasons of usage. The lifecycle view deals with how scenarios are handled (e.g., creation, refinement, and deletion). Applying the framework to eleven scenario-based approaches, they found out that scenarios are used to describe system behaviours interacting with its environment and most of them are in textual representation. They pointed out the importance of the formalization of scenario-based approaches, the variety of the application fields, and the need for practical scenario evaluation.

SCRAM

In [144] Sutcliffe describes a scenario based RE framework called SCRAM. Requirements are elicited by presenting user with a prototype-simulation of a design, combined with rationale based techniques for structuring probe questions. Sutcliffe proposed this technique based on the hypothesis that technique integration provides the best avenue for improving RE. SCRAM uses three techniques:

- use of prototypes
- scenarios
- design rational

These techniques are place inside a method to provide proves guidance for the requirements engineer. The method has the following phases:

- Initial requirements capture and domain familiarisation. This is conducted using conventional interviewing and fact finding techniques to gain sufficient information to develop a first concept demonstrator.
- Specification and development of a concept demonstrator. A more or less scripted early prototype of the system

- Requirements analysis-validation session. The users involved in the initial requirements capture are invited to analyze the concept demonstrator. This is a recorded session for further analysis
- Session Analysis. Data collection during the previous phase is analyzed and conclusions are reported back to the users. This leads to the iteration phase of further refinement of the product demonstrator.

RESCUE

In [96] Suzanne Robertson and Neil Maiden describe their experiences with RESCUE, a scenario based requirements elicitation approach.

RESCUE is a framework to guide a requirements engineering project to deliver a complete, correct and testable specification of requirements for a future system [75].

It recognises real-world constraints on the process, and also supports the analysis of current work practices to inform the change that will arise from the introduction of the new system. In addition, it uses creative design processes to generate additional requirements and to underpin these requirements with high-level design alternatives. The RESCUE process consists of a number of sub-processes, organised into 4 ongoing streams. These streams run in parallel throughout the requirements specification stage of a project, and are mutually supportive.

Method	RE phases	Main goal	Advantages	Disadvantages
Formal scenario method	Elicitation/ specification Analysis/	Use semi-formal techniques to derive a formal specification		Too formal, not very practical to use
Inquiry cycle	Elicitation/ analysis	Provide a method for discussion, documentation and evolution based on scenarios	Provide standard questions to facilitate scenario discussion	
SCRAM	Elicitation/ Validation Analysis/	Combine a number of techniques to create a RE method	Combination of well known techniques	
RESCUE	Elicitation		Well defined process	
CREWS	Elicitation/analysis/validation	Provide a complete scenario based framework	Provided results that scenarios are useful in describing system behavior	

Table 21: overview scenario based methods

Appendix III. REQUIREMENTS ENGINEERING AND COTS SELECTION

Introduction

In order to discuss the differences between classical RE and RE for COTS we will use the work of [3], she summarized existing work about COTS based requirements engineering. Classical requirements engineering (RE) is aimed at specifying the requirements for a custom made system. Classical RE starts with identifying the user needs, analyzing these needs, documenting these needs and the validation of these needs. It is important that the requirements specification for a custom made system is as precise as possible, since it is used to develop the new solution system.

Where custom build systems only addresses the needs of a single customer, commercial-off-the-shelf (COTS) are build to address the needs of the market. Multiple customers have comparable requirements for the solution system, the main task of requirements engineering here is to evaluate the available COTS products against the needs of the customer, and this means that COTS alternatives need to be assessed against the customer requirements. The problem becomes even worse when different COTS based components have to work together. The products from different vendors need to be adapted to be able to work together. It is possible that many conflicts will arise in this process therefore an *intensive evaluation* is a fundamental activity within COTS based development. The evaluation process is used to identify possible conflicts and inconsistencies.

Another major difference between classical RE and RE for COTS is in the way how detailed the requirements specification actually is. As mentioned before, it is of utmost importance that the requirements specification for a custom made system is as complete as possible. But for a COTS based solution the requirements specification can be much less complete and detailed. Requirements should be specified with more flexibility and should only discuss the desirable needs. Requirements prioritization is an important activity within RE for COTS, it is not always possible for COTS based solutions to address all needed functionality, therefore it is important to determine what functionality is good enough. Another major difference is that requirements elicitation loses much of its importance. It is not needed to elicit all requirements; COTS based products usually provide the requirements for the system. During the elicitation phase high level goals are identified using typical elicitation techniques (such as interviews, use cases). From these goals it is possible to already select an initial set of COTS based products.

The evaluation of COTS demands some inexact matching between customer requirements and products features. E.g. there may be requirements that are not met all by the product, requirements that are partially met and requirements that are fully met by the product. If critical features cannot be met by the solution system an intensive negotiating process is needed to balance conflicting interests between what is wanted and what is possible to meet.

An important distinction that needs be made is that requirements engineering for commercial products comprise of two RE phases for both the developer and the customer. The provider tries to develop a generic component or system and sell it to as many customers as possible. The requirements engineering phase for the customers is more about comparing the different solutions to the stated needs and identifying the most suitable solution.

Existing methods

We can find a number of different methods in the literature that describe the process of RE for COTS. We will discuss these methods briefly to try and understand the essence of these different methods and use their core components as techniques in our own method.

OTSO

One of the firsts attempts in the literature to develop a method for COTS selection is OTSO [81]. OTSO covers the entire COTS selection process. The main pillars of OTSO are that it describes a well-defined, documented process, it provides hierarchical and detailed evaluation criteria, it provides a model for making alternatives comparable in terms of costs and added value they produce and it uses appropriate techniques for consolidating evaluation data. The phases of OTSO comprise of:

- Search;
- Screening;
- Evaluation;
- Analysis;
- Deployment;
- Assessment;

The most important phases in OTSO are the evaluation and the analysis phases. Evaluation criteria are gradually refined as the selection process progresses. The evaluation criteria are formally defined so that the evaluation of alternatives can be conducted efficiently and consistently. There is a template available for this. The evaluation attribute definitions should include a detailed description of the attribute, its rationale as well as the scale and measurement unit used. The evaluation definition process decomposes the requirements for COTS software into hierarchical criteria set and the different options are analyzed using multiple criteria decision making. The advantages of this technique are that it is a thorough and logical method that has proven itself. The disadvantage is that it takes requirements specification for granted. It does not discuss how to elicit the requirements and hardly mentions what the effects are of non-required features.

PORE / BANKSEC

A second contribution to the COTS requirements literature is PORE. PORE is a template based approach to support selection that is based on iterative requirements acquisition and product evaluation [106] [97]. When PORE starts there are very few requirements and a large number of possible solutions. Using the templates in an iterative manner it is possible to refine the list to more requirements and lesser solutions. PORE uses a number of techniques, like knowledge engineering techniques, multi-criteria decision making and general requirements elicitation techniques. PORE also provides the evaluation experts with a few guidelines how to derive test cases to select and reject potential products. The method suggests to use criteria to determine if the requirements are met by the possible solution systems. One of the advantages of PORE is that it is based on empirical research. All the templates have been derived from practical use. Another advantage is that PORE builds on existing requirements elicitation techniques to derive the initial set of user requirements. But as the authors mention in the follow-up research project

(BANKSEC) PORE is still a research prototype, and the process of requirements product compliance is ill-defined [3].

CRE

CRE is a COTS selection method that is based upon the NFR framework and emphasizes the importance of non functional requirements [4]. Non functional requirements are used as the important factors in deciding what possible solution should be selected. They argue that it is possible to determine a number of solutions that are functionally the same, but they differ in how well they deliver that functionality. The advantages of CRE are that it uses the well-known NFR framework to model the non-functional requirements and that it provides guidelines on how to acquire and specify non-functional requirements. The draw-backs however is that analyzing the non-functional requirements take up a large amount of time and the method does not address issues of quality testing. The method also lacks support for cases where non functional requirements are not properly satisfied.

CARE

CARE is a goal-oriented COTS requirements engineering method [37]. CARE argues that requirements for COTS products need to be flexible, because they are constrained by the capabilities of available COTS products. Requirements are classified as native (requirements elicited from consumers) and foreign (requirements of the COTDS components). The main driving force behind this method is that it argues that it is required to bridge the gap between the sets of native and foreign requirements and that it is required to explore different alternatives of matching. CARE is based upon i* modeling concepts. The disadvantages of this method are that it does not provide any systematic solution to support the possible mismatching between both specifications

Conflict Management method

In [6] [3] the authors propose a conflict management method for COTS-based Systems.

The main idea behind this method is to define goals the system should meet and operationalize these goals using features from existing COTS packages. This method is not suited for the entire selection process but only for conflict management. The framework handles and deals with conflict instead of trying to eliminate it. The method extends the selection process, as conflicts will still arise even after selecting a COTS product.

The method starts with describing goals and candidate products that should be obtained from the organization and the marketplace. The two specifications should be compared to find similarities and differences. That is, how goals are met or not by the features of the COTS product. After mismatches are identified the conflict resolution process begins.

In order to specify the goals, the authors use KAOS as a specification language. An important feature is to identify conflicts between goals (e.g. MaintainBuyerHistory conflicts with MaintainBuyerPrivacy). They also argue that it is needed to differentiate the goals into core goals and peripheral goals. Core goals are supported by all products and peripheral goals help to differentiate between the products. Another way to analyze goals is to add two distinct attributes to the goals, namely desirability and modifiability. Desirability prioritizes the goals by assigning numbers to make a relative distinction of importance between the different goals. Modifiability

concerns with the feasibility to change a critical goal or feature whenever a conflict arises. This attribute is considered of utmost importance in this framework. As soon as possible products are identified after analyzing the core goals they lose their importance to peripheral ones. They do point out that comparing packages to goals is a manual operation, because package specifications are usually written in natural language.

For the matching process they define a number of patterns, namely:

- ♦ fulfill, feature satisfy a goal
- ♦ differ, product provides a feature that partially satisfies a goal
- ♦ extend, products provide an extra feature
- ♦ fail, product does not provide a feature that was requested by a customer goal

The fulfill pattern is the only pattern that doesn't generate a conflict, but the other patterns do. After conflicts have been identified they have to be analyzed, they propose risk analysis. In order to resolve a conflict the reasons of its occurrence should be explored. Also it is very important to record all rationale in the analysis and decision making process, because it is possible conflicts change in the near future and information about these conflicts should be recorded to aid conflict resolution in the future. In [7] the authors extend the framework with quality models and scenarios. Quality models are used to define metrics which can be measured to determine the suitability of the product. Scenarios are used to determine and analyzing possible conflicts in the matching process. The benefits of using scenarios to deal with conflicts are as follows:

- To explore resolution alternatives and highlight products limitations;
- To identify associated risks with each COTS;
- To explicit evaluate the impact of decisions.

The advantages of this method are that it provides explicit reasoning how to link product features to requirements using the well know goal-driven orientation in requirements engineering. The method also provides a detailed strategy how to measure the compatibility between product features and the requirements and provides a scenario driven way to explore resolution alternatives, making it a very complete method for COTS selection. One of the drawbacks of this method is that it requires some formal understanding of utility theory. Another disadvantage is that the method has only been applied to a literature case study instead of intensive empirically testing.

Appendix IV. RE AND SERVICES

Introduction

The shift to service oriented computing probably has a number of effects for RE. Van Eck and Wieringa have the following to say about these effects [153]. They argue that RE will split into two separate kinds of requirements engineering, RE for service consumers and RE for service providers. Service consumers are the user groups responsible for some business function and need IT support for this. Service providers are the user groups that are responsible for designing, implementing and providing IT services.

The distinction between RE for service providers and service consumers comes from the separation of concerns. Service consumers are only interested in performing a certain task as well and efficient as possible and need support for this task. Service providers are interested in providing a service as effectively and efficiently as possible. In general service providers deliver a highly standardized service and as a consequence they cannot meet each specific business function demanded by the consumers. Service consumers need to align their tasks and processes around these generic services.

Lichtenstein *et al* [94] position RE for services between IT service management and the traditional socio-technical issues in requirements engineering, the goal of the RE process here is to produce the service specification requirements, but in practice this would most likely directly lead to an SLA. After an initial case study they concluded that there are three main implications for service-oriented RE.

- service providers clearly need to recognize the customer's needs, but consumers also need to address the service providers needs;
- service performance is very important, both consumer and provider play an important role here (e.g. the consumer needs to provide performance metrics, etc);
- Social issues are still important, stakeholder groups must be able to communicate and negotiate their conflicting IT service needs and this affects team composition on the provider side (i.e. personnel with strong negotiating skills).

Requirements engineering for service consumers

Service consumers [153] can be found in all departments except the IT department. Their goal is derived from the business goals of the department in which they are located. Service consumers try to support this mission as effectively and efficiently as possible. The primary task of the requirements engineer is to align business processes and user tasks with services offered. The requirements engineer determines the need for a new service and suggests some changes to certain tasks and business processes to adapt to the offered services [153] [46]. Greenspan [127] also mentions that in service specification there are two main tasks, analyze the business processes to comply with the service and describe the systems that need to comply with the service.

Gu and Lago mention [61] that the requirements engineer is in general responsible for checking if the functionality matches the requirements and Greenspan mentions when a new service is introduced the specification should contain the *service work tasks, responsible business units* and their realization in the capabilities of the enterprise's system.

Requirements engineering for service providers

Requirements engineering for service providers takes place in the IT department of an organization or even a separate organization when this is outsourced. The goal of the requirements engineer here is to ensure the availability of a service infrastructure for current and future use. Service providers need to develop this as efficiently and effectively as possible. This has as a result that they try to focus on a generic standardized infrastructures.

The requirements engineer is not only responsible for specifying the service functionality, but also for infrastructure design [153].

Gu and Lago [61] in general agree with the view described above. They too argue that requirements engineering for service providers is about developing a reusable infrastructure to gain investments returns as soon as possible. Service providers do not only need to analyze the objective, functionality interface and the quality of the services, but also need to consider the issues relating to by which means their clients are able to access these services (e.g. service level agreements, policies, etc). They also argue that the requirements gathered in this phase should be used to (re)model the business processes [61].

Existing methods

In the literature we can find a number of initial attempts for service oriented RE. In this section we will briefly describe these methods and provide an analysis table to establish the core elements behind these methods.

The literature describes a number of techniques for RE for services, most of them adopt a goal driven elicitation technique. We have not only looked at explicit RE methods, but also at service-oriented design methods to identify what those methods describe about RE for services.

An Intentional Perspective to Service Modeling and Discovery

In [132] [78] the authors propose a technique to elicit requirements for services using a goal driven manner. They argue that organizations find it hard to describe services in software terms; services for service consumers should be described in terms the organization can understand, (i.e.) using tasks and goals. They propose a technique that describes the *intentional* services of an organization, using goal based approaches. It identifies services that should be provided by each organization and distribute and orchestrate the coordination between those services to achieve the purpose of the cooperative process

An intentional service is a business-oriented service described in an *intentional perspective*, focusing on the *goal* it allows to achieve rather than on the functionality it performs. An intentional service is executable by composing it of a number of software services. There are three different aspects in the description of an intentional service, namely the *service interface*, the *service behavior* and the *service composition*.

The service interface is specified through the fulfillment of a goal, given an initial situation and terminating in a final situation. *Service behavior* is specified through pre and post conditions that are the initial and the final sets of states. Thirdly services are classified as an aggregate or an atomic service. Atomic services relate to operationalized goals (goals that are fulfilled by a functionality provided by the

service). Aggregate services correspond to high level goals that need to be decomposed in lower level goals that can be operationalized. They argue that it can be understood that aggregate services lies on a goal-driven composition to bridge the gap between the actual functionality (captured in atomic services) and the high level perception of strategic/ tactical goals. Their technique comprises 5 steps.

- 1) Identify services from process maps; a process map is basically a directed graph with intentions as nodes and strategies as edges between the intentions. A map visualizes the intentions and how they are accomplished. From this model it is possible to derive a number of services.
- 2) Determine key playing actors, this step of the methodological approach focuses on relating each section of the map, i.e. each service, to the actor who will provide the service.
- 3) Identify legacy services, the purpose of this step is to identify parts of existing legacy information systems of the partners' organizations that can be packaged as e-services and reused as such in the new cooperative application. UML activity diagrams can be used to visualize the message passing and identify available legacy services.
- 4) Distribute services among actors, in this step services are distributed among the actors, assuming that all the services under the control of a provider actor should be gathered into one single composite service. The distribution leads to identify as many distributed services as provider actors.
- 5) Orchestrate services, to get a full specification of the service. This is done through dependencies between the different maps. These dependencies show the different actions between the services and therefore determine the behavior of the service.

The end result of this method is an identified service and it provides a behavior design for the service. The advantages are that it provides a complete service elicitation method, but there are also a number of disadvantages. The method is untested in practice. It is only suitable for service providers and its value for service consumers is unclear.

Designing for services using TROPOS

Another attempt to service design is through the use of Tropos [88] [114]. Tropos is a GORE development method that supports both early and late requirements as well as architecture design. Tropos was not heavily modified to support the different nature of services.

The authors used Tropos to create a goal and actor diagram of the intended service. The actor diagram is in fact an architectural design in terms of subsystems interconnected through data and control flows.

TROPOS is a well known goal-oriented RE technique, therefore the approach is reasonably tested in practice. The disadvantages however is that TROPOS was not designed specifically for a service orientation. TROPOS models usually have the tendency to grow beyond a point that they are practicable to use.

SREM: A service requirements Elicitation Mechanism based on Ontology

A third explicit requirements engineering approach is proposed here [177]. SREM uses a generic list of questions to narrow service requirements down to specific

expression of user preferences. After this phase, a service requirements and capability ontology is adopted to capture services requirements. This ontology is based upon *i** modeling concepts. SREM integrates the different requirements models by different requestors. SREM takes the viewpoint of a systematic approach in requirements engineering that can address the process of transforming legacy systems into easily reusable and customizable services.

There are three major components within SREM, service requirements modeling ontology, a service requirements elicitation process and a service requirements reconciliation process.

Requirements are elicited through determining the intended need of a system. The requirements engineer needs to ask questions, based upon said ontology, to determine the intended need. The approach provides the requirements engineer with predefined questions. The end result of the elicitation approach is a hierarchical requirement model. Not every service consumer has the same needs; SREM therefore provides a mechanism that assigns scores to the models. If certain parts of models overlap, they receive a higher score. Thus creating a common service definition.

SREM is a requirements engineering approach that focuses on service identification from legacy systems. The end-result is yet again an identified service and its advantage is that it is a complete elicitation method that uses practical techniques like stakeholder questions to identify the services. The major disadvantage is that it is yet again only a theoretical approach.

RE for e-services

In [55] the authors describe a few RE techniques for e-services. The authors argue that before a web-service is designed it is important to understand the service the web-service implements. This understanding is about the inter-organizational business processes, but also on the value the service creates. And that a requirements specification for services functions as a bridge between business considerations and web-services. This approach is only applicable for the early phases of e-service development. Because e-service development is of a multi-disciplinary nature, they argue that multiple viewpoints should be taken into account. The task of a requirements engineer is to match business processes of a set of business actors to consumers needs in a market, in a way that results in a specification that is detailed enough to design the service-based specification. The viewpoints are:

- Business value viewpoints, in this viewpoint, three descriptions of the service are produced using e3value. These descriptions describe the consumer need and allocated this need to services, identify the activities for these services and quantify the value exchanges for each business actor.
- Process viewpoint, this describes the inter-organizational business process and intra business tasks.
- Information system viewpoint, this viewpoint describes a workflow-like specification to be used by some enactment engine. The service is specified using WDSL and BPEL, through automatic UML conversion.

Other variations in using e3value as a RE technique for e-services can be found here [56] and here [26]. In [56] an e-service is designed using i* and e3 value and in [185] e3 value is used in combination with a goal-oriented method to recognize service provisioning patterns.

The advantages of this approach are that it uses different viewpoints to describe the service. These viewpoints not only provide information to the technical designers, but also abstract from implementation technology and provide a description service consumers can understand. The disadvantage is that the method is not tested in practice (like most reviewed methods).

Blueprinting

In [153] the authors propose to use a general service design technique. This technique is called service blueprinting. Service blueprinting describes all activities that are carried out by a service provider and its customer to deliver a service. There are four kinds of activities involved in service blueprinting:

- service customer activities
- onstage contact employee activities
- backstage contact employee activities
- support activities

Blueprinting is a very popular technique in services marketing. It allows to structurally designing the service and provides as a document that the technical designers can use to design the service.

A service design methodology

In [111] the authors propose a methodology for service development. This methodology focuses mainly on service providers. Not the entire methodology is relevant for the requirements engineering viewpoint, but they do describe a number of interesting techniques.

They argue that it is a key requirement to understand the business environment and to make sure all necessary controls are incorporated into the design of a service-oriented solution. Activities in this phase are, analyzing business needs in measurable goals, reviewing the current technology landscape, conceptualizing the requirements of the new environment and mapping those to new or available implementations. Planning also includes financial analysis of the costs and benefits, etc.

In the analysis phase they propose where the requirements of the service are investigated, more specifically reviewing business goals and objectives that drive the development of business processes. Business analysts create an as-is process model to allow the stakeholders to understand the portfolio of available services and business processes. The end result of this analysis phase is the to-be process model. This analysis phase encourages business process (re) design.

- process identification
- process scoping
- business gap analysis

- process realization

IBM service identification method

According to Amsden [8] SOA-based infrastructures need to be business relevant, driven by the business requirements the SOA supports. Business requirements need to be formalized and the level of abstraction needs to be raised. He proposes a method based on separation of concerns and loose coupling. Therefore he separates the tasks of both the business analyst from those of IT staff members. This method is partly UML based. Firstly they model the business goals, and later in the method they model the business processes that achieve these goals. According to this method the business executive is responsible for conveying business goals and objectives and the business analyst is responsible for analyzing the business requirements. This method starts with describing the business motivation that establishes the business goals and objectives that are to be achieved, followed by high-level business-process design. This enables to express the business organizational and operational requirements to meet the goals and objectives. These motivations and process models establish the context for identifying the services that are connected to the business requirements.

The business-process models are used as a service requirements contract. The services need to comply to the business processes. Performance is measured through the use of KPIs. After that, service specifications and service providers are designed and connected. The process of deriving candidate services from process models is called *domain composition*. Another important step in their approach is to link the business process to their respective use cases.

In the case that the business processes are not elaborate enough to be run directly they are seen as a service specification. These requirements contain the *tasks*, *roles* and the *rules*. This specification does not contain any implementation or IT architectural concerns.

The IBM service identification method provides an introduction to service identification. It is a less detailed version of SOAM/SOAD to provide practitioners and introduction to service oriented RE. It is therefore an easy to use method, which uses known concepts like goal identification and goal modeling and formalized business requirements. Business process models are used as context models for the services.

SOAM/SOAD

Another method proposed by IBM is SOAM/SOAD (service oriented architecture modelling and service oriented analysis and design) [184] [12]. This first part of this technique has some resemblance with RE, the technique is used to identify the requirements for services through analyzing legacy systems, business goals and business processes. They also explicitly argue that there are distinct requirements for either the consumer role or the service provider role. In the consumer role the activities are service identification, categorization, exposure decisions, composition and to determine the quality of service (non-functional requirements).

Services are identified through a combination of a top-down, bottom-up and middle-out techniques. In the top-down view a blueprint of business use cases provides the specification for business services. This composes the business domain into its functional areas and subsystems. In the bottom-up view in identifying services existing legacy systems are analyzed to identify their possible role in service

provisioning. The middle-out view is a goal-service modeling technique to validate and find other services not captured by the other views.

They also argue that object-oriented analysis is not sufficient for service oriented way of thinking; therefore they propose the service-oriented analysis and design technique. SOAD combines elements from OOA, BPM and EA to create a technique that can be used to identify and specify services. They use BPM and stakeholder interviews (direct requirements analysis) to identify candidate services and the individual services are specified with the operations (activities from business processes) they need to accomplish.

This method by IBM also encourages the use of multiple viewpoints, or techniques, for service oriented RE. The end results are a service specification, service requirements and a global service design. The method also uses known RE principles and techniques, like goal identification, scenarios and stakeholder elicitation. The disadvantages are that the method is the design philosophy from a commercial company and therefore less scientific based.

BPM as RE for SOA

In [164] the authors propose to use business process management (BPM) as a requirements engineering technique for (business) services. Business services act as an abstraction layer between the business and the IT. Business services are specified in this technique through their business relevant inputs and outputs. Functions of a business service are business activities that are hierarchically structured, that can be supported by IT. Business services both execute processes and functions. The usage of a business service is subject to a SLA that determines the quality of its execution.

They argue that the only way to deploy a SOA is that it has to meet its organizational requirements. They define services as "the layer between rapidly changing business requirements represented by business processes and steadily evolving system landscape that ought to meet these requirements."

They also argue that a SOA component should resemble business functions instead of, for example, a database connection layer. Web-services are often used to implement these SOA components (or business services).

BPM as RE is not really a specific RE method. But more a technique that can be used to analyze and design the business processes that support the services. Its advantages are therefore that it is a well known accepted technique, but not a complete RE method.

Appendix V. REQUIREMENTS TRACEABILITY

Introduction

Requirements traceability, [58] has been the focus of ongoing research for the past decade, it is recognized as a concern in an increasing number of standards and guidelines for RE. Roetzheim identified RT as a quality factor and non-functional requirement [131].

Ramesh and Jarke [125] give the following definition of RT: RT is a characteristic of a system in which the requirements are clearly linked to their sources and to the artifacts created during the system development life cycle based on these requirements.

Spanoudakis [140] defines traceability as the ability to relate requirements specifications with other artifacts created in the development life-cycle of a software system.

Murray [103] defines traceability as the ability to identify requirements at different levels of abstraction and to show that they have been implemented and tested.

However, we believe the definition provided by Gotel and Finkelstein [57, 59] is more suitable for this research, because it also makes a clear distinction where the requirements originate from and to what artifacts they lead to.

RT refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases.

Types of RT

Requirements traceability is a broad concept, within RT a number of different types can be recognized.

Backwards and forwards traceability

Within the RT literature the concepts of backward and forwards traceability have standard definitions [115]:

Backwards traceability is the ability to trace a requirement back to its source (e.g. stakeholder, law, argument, etc)

Forwards traceability is the ability to trace a requirement to components of a design or implementation.

Pre RS and post RS traceability:

Gotel and Finkelstein did a thorough analysis of the requirements traceability problem [58]. They mention that forwards and backwards RT is clearly an important feature of traceability, they do point out that the separation between pre-RS and post-RS should also be made. Most of the problems in RT occur because of a lack of distinction between those two subjects. The main differences are the information they deal with and the problems they can assist. Post-RS traceability depends on the

ability to trace requirements from, and back to, a baseline (RS) through a succession of artifacts in which they are distributed. Pre-RS depends on the ability to trace requirements from, and back to, their originating statements.

Pre-RS traceability refers to those aspects of a requirement's life *prior* to inclusion in the RS (e.g. from stakeholder to requirement).

Post-RS traceability refers to those aspects of a requirement's life that result from inclusion in the RS (e.g. from requirement to design artifacts, to a property of the system).

Pre-RS traceability is useful when we want to know what happens when requirements changes and we need to validate that new requirement with its original sources. Post-RS is useful to get the design module to which a requirement was allocated to when we want to check if the design satisfies all requirements.

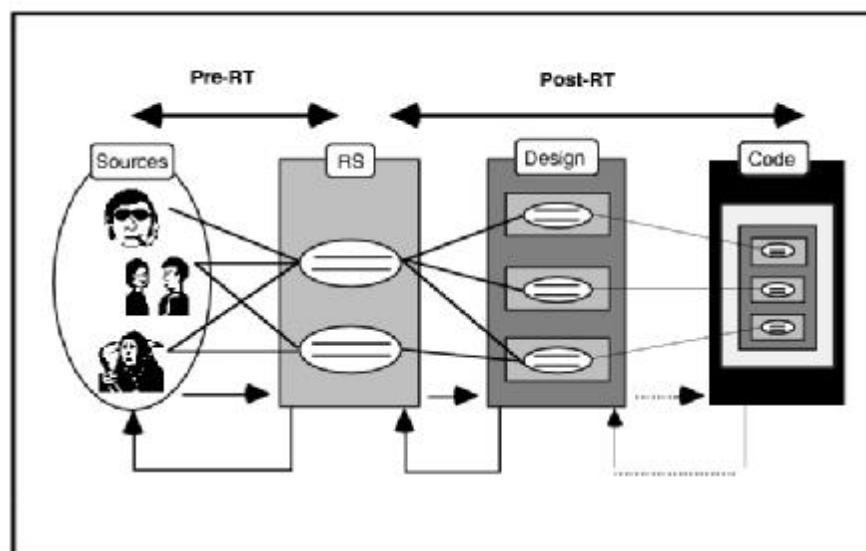


Figure 44: pre and post re traceability [58]

Inter and extra-requirements traceability

Pinheiro [115] also distinguishes inter and extra-requirements traceability. When links between requirements themselves are traced we speak of inter-requirements traceability. The links between requirements and other artifacts is called extra-requirements traceability.

Inter-requirements traceability refers to the relationships between requirements

Extra-requirements traceability refers to the relationships between requirements and other artifacts

Inter-requirements traceability is important during the requirements analysis phase, e.g. when we want to know what requirements derived from a certain requirement.

The Need for RT

In [126] Ramesh *et al* discuss the need and why requirements traceability should be implemented, they distinguish different levels of needs grouped for a different audience.

(Project) Management

- Requirements traceability is a must for survival, because customers want it
- It minimizes the possibility of missing requirements
- Easier to track the progress of a project
- Conflicts between requirements are detected easier, thus minimizing project costs
- Improved validation
- Future projects last shorter, due to the availability of previous design decisions

System Designer/ Engineer

- It enables the engineer to record their decisions and results
- It provides a justification for these results
- It provides design alternatives
- Design can be verified against the requirements easier
- He can re-use certain components because he stored the different design decisions

Customer

- The quality of the product can be compared with the user requirements
- Acceptance testing and requirements can be linked directly
- User requirements are linked to design requirements, so that development personnel can keep their focus on the user requirements

Different usages of traceability

Traceability in practice can be divided into two distinct traceability practices, one for high-end users and one for low-end users. We will summarize the work of Ramesh [124] to explain the differences between these groups.

A first obvious distinction between low-level and high-level use is the complexity of the systems. When systems get more complex, the need for RT increases. Low end users of traceability usually work with less complex systems.

A second distinction is that low-end users view RT simply as a mandate from project sponsors, whereas high-end traceability users view RT as part of a quality engineering process. High-end users have a better understanding of the advantages of adding traceability to the engineering process.

Low-end users use simple traceability techniques, such as traceability schemes to model dependencies among the requirements, allocation of requirements to system components and links from requirements to compliance verification procedures.

High-end users demand more from RT, they capture design rationale, the evolution of design artifacts, and they see RT as an ongoing process improvement program.

In the context of systems development low-end user have no systematic method for RT, whereas the high-end users developed well-defined system development policies (standardized methods), high-end users are trained to understand the importance of traceability and see it as part of the engineering job.

Another major distinction between low-end and high-end users is that low-end users see it as an expensive, overrated activity and when the first budget issues arise within a project, traceability is the first thing to go. High-end users see traceability as a mechanism for process improvement and a trace of the process of systems development.

Low-end users also focus more on post specification traceability; they ignore the question "Where do requirements come from", whereas high-end users focus on both pre RS and post-RS traceability. Low-end users use static documents to create a form of traceability, but when traceability is seen as an important activity tools are required. High-level users of traceability understand this and develop tools that are embedded in the development environment. Another major difference between low-end and high-end use is that low-end user's capture traceability information for all requirements, but high-end users understand that not all requirements are equal. Therefore they only capture traceability information for the most important requirements. Table 22 shows an overview of the text above.

	High end users	Low end users
Complexity	High	Low
Reasons	Required for quality engineering	Mandate from management
Techniques	Extensive tooling	Simple diagrams
Methods	Explicitly defined method	Ad hoc
Need	Process improvement	First thing to go
Focus	Pre-RS	Post-RS
Capture	Only for the important requirements	All

Table 22: overview traceability usage

Traceability in the RE process

When we look at RE we see a process that runs through three dimensions, these dimensions are [116]:

- Specification, from incomplete descriptions to complete requirements documents
- Representation, from informal textual statements to (more) formal representation models.
- Agreement, requirements engineering is all about agreeing on a set of requirements where all stakeholders can identify with.

In order to create requirements pre-RS traceability [116], traceability must be an integral part of the RE process. In the way of working section we will describe a RE process model and explain how traceability can be accomplished within this process model.

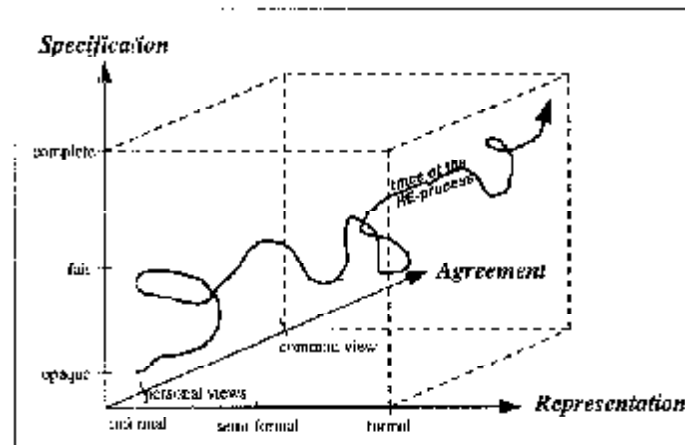


Figure 45: Framework for traceability [116]

When we analyze this picture further, we can discuss the kind of information will be recorded:

- information alongside the representation dimension (e.g. requirements statements, behavior models, etc)
- Information alongside the agreement dimension (rationale, decisions, alternative options, etc)
- Information alongside the specification dimension (the requirements)
- Relations between these elements (which decision led to a certain requirement and what where the alternative options, which requirement led to another requirement, etc)
- Relating the previous steps to the stakeholders that produced these requirements

Techniques

In this section we will discuss the different basic techniques to represent traceability links. The literature describes in essence two standard techniques in RT, matrices and cross references.

Matrices

One of the most common ways to represent traceability is through the use of traceability matrices. Requirements are associated to objects of software development (e.g. design documents, system components, test procedures, etc). For example, requirements are allocated to the vertical columns and design artifacts to horizontal columns. The cells can then used to show that a certain requirement links to that particular design artifact.

Cross references

Cross references are common to use to incorporate some form of traceability. Within requirements specification requirements can cross reference to each other or to different documents. The relevant links are embedded as points in a text (e.g. hyperlinks). Cross referencing is useful for written specifications but not for a concise representation of links, as can be done with matrices.

Appendix VI. RM AND EA

In this section we will discuss the topic of EA. The main goal of this section is to find out how existing EA frameworks deal with the topic of RM. We have chosen the selected frameworks after an initial analysis of DoDaf, IAF, Zachman, GEAM and TOGAF. These frameworks are discussed in [71]. We have evaluated these frameworks on how extensively they discuss requirements. We particularly looked at how the frameworks provide rationale for architectural elements (solution systems) and any method guidance. In this chapter we have chosen the three frameworks that discuss these issues the most. These are the Zachman framework, IAF and TOGAF. An evaluation of these frameworks is useful because it provides guidance to gathering requirements for the method. By also looking at highly accepted frameworks and select the most common elements the method will be based on "best practices" derived from the literature.

In section 2.1 we will discuss the general topic of EA. This is based on the work of [71]. Section 2.2 discusses popular EA frameworks and how they deal with requirements. In section 2.3 we will provide the results from our analysis.

What is Enterprise Architecture?

The word architecture is associated in the field literature with two interpretations. Very often one can guess from the context which of the two interpretations is intended. The first refers to the inherent architecture of a system. System theory states that a (complex) system can be divided in smaller less complex sub-systems that combined will form the original system. This combination of lower order systems and the manner in which they function together defines the inherent architecture of the original system and is referred by some authors to as the "architecture". In this line of thinking, the ANSI/IEEE 1471-2000 standard is proposing a definition that is considered to be the reference:

The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.

The second interpretation relates to the assertion that it is necessary to make explicit the inherent architecture of a system by means of architecture descriptions. Next to documenting the decomposition of the system, an architecture description may refer to many other aspects such as function, data, time, motivation, etc. The ANSI/IEEE 1471-2000 standard is very precise in this respect and makes a clear distinction between architecture (which is conceptual / intangible) and architecture descriptions (which are concrete/tangible), which are subsequently defined as:

A collection of products to document an architecture.

Architecture description is thus the second connotation of the word architecture. The relation between architecture as description and architecture as inherent architecture is that the architecture description represents and conveys the essence of the inherent architecture. In the theory and practice of EA, the second interpretation is the most commonly used, and therefore we will adopt it as well. Furthermore, the definition of EA we propose below fits very well in the "architecture as description" line of thinking.

In this thesis we adhere to the following definition of the concept of Enterprise Architecture that is based on the existing literature:

Enterprise Architecture (EA) is the complete, consistent and coherent set of methods, rules, models and tools which will guide the (re)design, migration, implementation and governance of business processes, organizational structures, information systems and the technical infrastructure of an organization according to a vision.

More specifically, this definition can be complemented with additional characteristics such as those mentioned below.

Enterprise Architecture comprises a collection of simplified representations of the organization, from different viewpoints and according to the needs of different stakeholders.

Enterprise Architecture is a design or a description that makes clear the relationships between products, processes, organization, information services and technological infrastructure; it is based on a vision and on certain assumptions, principles and preferences; consists of models and underlying principles; provides frameworks and guidelines for the design and realization of products, processes, organization, information services, and technological infrastructure.

EA frameworks

In this section we will briefly discuss some of the available EA frameworks. We will not study all available frameworks extensively, that would be a study by itself. We only want to understand the general idea behind RM applied in other frameworks. These frameworks have been extensively discussed in the handbook EA [71] and in [66]. Since this is a research done at BiZZdesign, we will summarize and reuse this material. After an initial analyse we concluded that the Zachman framework, TOGAF 8.1 and the IAF framework are suitable candidates to discuss.

Zachman framework

In 1987, Zachman introduced his "Framework for Enterprise Architecture" (see [182] and Figure 46) although back then it was called "Framework for Information Systems Architecture". The framework as it applies to Enterprises is simply a logical structure for classifying and organizing the descriptive representations of an Enterprise that are significant to the management of the Enterprise as well as to the

development of the Enterprise's systems. It was derived from analogous structures that are found in the older disciplines of Architecture/Construction and Engineering/Manufacturing that classify and organize the design deliverables created during the process of designing or producing complex physical products.



Figure 46: Zachman Framework

The figure above shows the "Framework for Enterprise Architecture", usually known as the Zachman Framework.

The two abstractions that could be relevant for requirements traceability are the "how" and "why" viewpoints. When we trace the different viewpoints from top to bottom we can see how an artifact is realized and what goal it realized.

TOGAF 8.1

TOGAF is a process oriented framework [141]. The steps within this framework are iterative and cyclic. The Open Group Architectural Framework (TOGAF) originated as a generic framework and methodology for development of technical architectures, but over the years became an enterprise architecture framework and method. The new version 8.1 of TOGAF is called the "Enterprise edition" and is dedicated to enterprise architectures.

TOGAF has three main components:

- A methodology called Architecture Development Method (ADM), considered to be the core of TOGAF. ADM is actually a step-wise iterative approach for the development and implementation of enterprise architecture (see Figure 47: TOGAF).
- The Enterprise Continuum, which comprises a virtual repository containing all of the architecture assets. This is initially populated by two reference models, the TOGAF foundation architecture and the integrated information infrastructure reference model.
- The TOGAF Resource Base - a set of tools and techniques available for use in combination with the TOGAF ADM (views, architecture board, architecture

compliance, architecture governance, architecture maturity models, patterns, principles, skill frameworks, building blocks examples, etc.).

TOGAF itself doesn't contain any modeling techniques, but does recommend what techniques can be used in every phase.

Each step within ADM interacts with the requirements management phase. Requirements here are being actively refined from high level organizational goals, to business requirements, information system requirements and infrastructure requirements. In the architecture change management phase the requirements are compared to the actual situation and a change project is executed to close the gap.

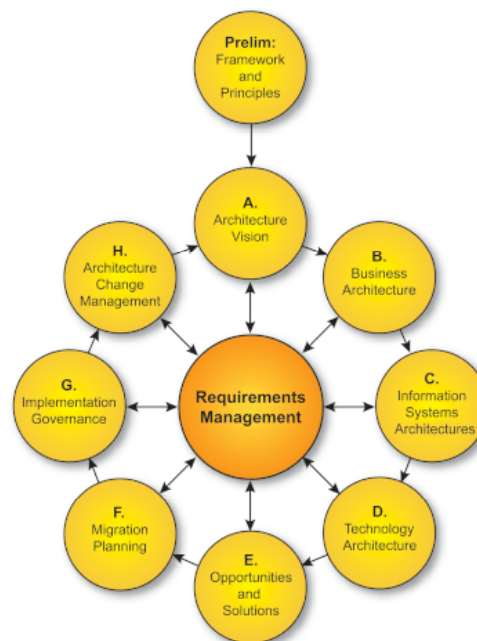


Figure 47: TOGAF ADM

IAF

Capgemini promotes for more than ten years its own framework, namely the Integrated Architecture Framework (IAF) (see [102]) that reflects a service oriented way of thinking. Figure 48 shows the basic structure of IAF. The model is broken down into Aspect Areas (business, information, information systems, technology infrastructure, security and governance) and Abstraction Levels (contextual, conceptual, logical and physical). Each "cell" in the IAF model has a defined set of Artifacts. Views then allow the architects to bring together and visualize the artifacts to help in modeling the architecture and to communicate the architecture with the various stakeholders.

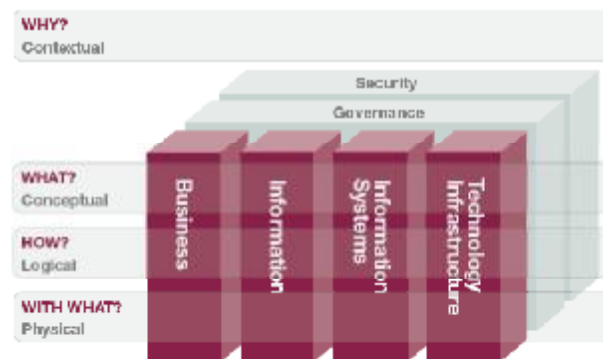


Figure 48: IAF framework

The IAF argues that architecture is actually a continuous improvement process, in which architecture responds to the overall business and IT environment. Consequently architecture becomes a part of the overall design and governance continuum for a business. Therefore, the life cycle of architecture is regarded, as an iterative process supplemented with learning from the experience of implementation.

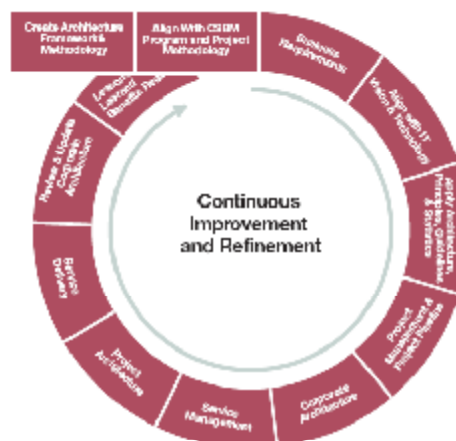


Figure 49: continuous improvement and refinement

When we look at the life-cycle model and the provided white paper [102] we can distinguish a high level requirements management method. At the contextual level we see the why question. This is where business requirements are defined. Business requirements are the principles behind the architecture and they recommend to record the priority, implications and rationale. At the conceptual level they decompose the requirements into more detail and yet again the recommend to record their priority, rationale, etc. At the logical level they use solution alternatives; these are technology independent solutions which provide more or less the same outcome. Different alternatives are tested here and in the end the best one is chosen.

Appendix VII. SOA

Introduction

One of the requirements imposed by BiZZdesign is that the method should be applicable to use in a service-oriented organization. Therefore we will discuss the concept of services briefly in this chapter. The main goal of this chapter is to get a global understanding of services, SOA and service level agreements.

In section 3.1 we will discuss the concept of SOA, section 3.2 discusses how layering realizes SOA. Section 3.3 discusses the concept of a service. Section 3.4 explains how web services are used to realize services. Section 3.5 discusses service level agreements and in section 3.6 we will provide analyses of the provided results.

The concept of Service-Oriented-Architecture

A common vision in the scientific literature is that software needs to support the agile and changing business needs [161]. Therefore software needs to be built up from discrete parts of functionality. These parts should be reusable and reconfigurable to keep adapting to the changing needs. This way of thinking led to the concept of service-oriented-architecture (SOA).

In traditional organizations the role of IT is more a enabler of business strategy, whereas in a service-oriented organization IT plays a strategic role in business transformation, including the creation of new sources for business revenue, whereas there is a logical separation of business need (consumption) from fulfillment (service provision). The need can be fulfilled by multiple providers.

In the software terms, a SOA is a logical way of designing software systems to provide services to either end-user applications or other services distributed in a network through published and discoverable interfaces [110].

Layering

In SOA the design philosophy of component based development is re-used. Functionality should be split up over components that are loosely coupled. The SOA philosophy is much broader than the component based philosophy. The SOA philosophy can be depicted as a pyramid [110] [161] (see Figure 50). The top layer describes how organizations should think about service level agreements, assurance and support for the information systems architecture. The composition layer encompasses necessary roles and functionality for the consolidation of multiple services into a single composite service. The resulting composite services may be used by service aggregators as components in further service compositions or may be utilized as applications by service clients.



Figure 50: layering in SOA [109] [161]

Services also require change in organization structure and roles. The transition to a service-oriented enterprise requires loosely coupled organizations that are based on the role definitions of *service providers* and *service consumers* [77]. Another important change has to do with the availability of services; services can be compared to the raw materials of traditional enterprises. Therefore negotiating with and changing services providers must occur much faster than in the traditional way. This requirement will lead to the emergence of service intermediaries. This will lead to an environment where consumers have the possibility to switch to a different service provider. Service intermediaries can help create trust by judging providers on past experiences with other consumers. The bottom layer is about basic services, their descriptions and basic operations (publication, discovery, selection and binding).

The concept of service in SOA

In the last three decades we have witnessed a strong migration of enterprises from a function-based organization (following a division in departments such as sales, production, logistics, etc.) to process-driven organizations. However, the last years have brought in the attention of the scientific and practitioner communities a new concept, namely the concept of service [71]. We define a service as [161]:

A service is a component that offers its functionality to external components through an interface that is described in a standardized way.

The consequence is that a new shift has begun, in which the enterprise aims to become service driven. This means that now internal business processes are no longer central in the organization. Instead they become just means to realize and deliver services to the client. Consequently, the client together with the services he requires forms the new organizing principle of the modern enterprise. Related to this

new phenomenon organizations must also consider the migration from process-driven to service-driven enterprise architectures.

Services can be distinguished on a number of different levels [77] [86]:

Business Service, this is a coherent unit of functionality that offers value to the environment, independent of internal realization.

Application Service, this is an externally visual able unit of functionality, offered by one or more components, available through univocal defined interfaces and offering value to the environment

Infrastructure Service, an externally viewable unit of functionality, offered by one of more infrastructural nodes, available through univocal defined interfaces and offering value to the environment. Although the literature does recognize this kind of service, in this book we will only focus on business services and application services.

These services communicate in a layered manner to each other, business services are the services organizational departments deliver to each other and their customers. In general business services are realized through business processes. These business processes are able to use automated application services. When it is possible to fully automate the business process, one application service will realize this process, thus creating a one-to-one relationship between an enterprise service and an application service.

Application services are the services automated systems deliver to each other and the organization, application services can be divided into a number of different variants [77].

Web services as an implementation of services

Web services can be seen as the implementation of the service-oriented philosophy [110]. A web service is a collection of business functions or capabilities that can be published to a network such as the internet that completes tasks, solves problems or conducts transactions. Each web-service is a building block that enables the sharing of software functionality with other applications residing outside the web services native environment [110]. Services are offered through components that can be created using traditional programming environments like JAVA or .NET. The idea behind web service technology is that services created through these components can always communicate with each other, no matter the implementation and development platform.

Service level agreements

According to [150] SLAs are clear descriptions of activities performed by a supply organization under orders of a demand organization. It provides in some detail when, how and where the activities are executed. It uses a language the demand organization understands and is created to provide a certain level of expectation. The SLA contains agreements on the quality, quantity and the costs of the IT facilities.

The process of constructing an SLA is in general is [150]: firstly the business situation needs to be explored. The size of the organization to be supported is looked at, types of operational processes to be supported and the users that are

going to use the facilities. The second step is to determine the profiles of the business process to be supported and the definition of the desired support and thirdly the SLA needs to be drawn up with an indication of the products to be delivered and their qualities.

Bouman *et al* [26] discuss the issues in specifying an SLA, they argue that an SLA is no longer seen as just a financial contract but an instrument for managing customers expectations. The major problems concerning SLAs are:

- most SLAs are specified in the effort the provider has to spent when a problem occurs, but no commitments are specified regarding the effectiveness of a service for a customer's business processes;
- unclear service specifications (e.g. availability of a service, does 98% uptime mean the service can be down an entire week after being up for 51 weeks);
- incomplete specifications;
- Description of SLAs is usually too much technology-oriented.

To solve these problems the authors suggest a few guidelines when drafting an SLA, these guidelines describe a few necessary tasks and required contents of an SLA:

- determine the user needs early;
- some aspects of a service are more important than others, focus on those;
- create a readable document and include decisions made
- define roles early (who is responsible for what)
- review the draft SLA
- the SLA is a document that brings two parties together, therefore the balance between commitments on results and efforts should be determined between both provider and consumer.

They also describe a process how to shape a service level agreement. They used interviews to shape a first draft and identify the main components of a SLA. A second step was to make the different components of the SLA measurable. They observed that it was required to define metrics to be able to measure the SLA. Another important step was to include the decisions made into the SLA, to make the document more understandable for the consumer of the service. The document was also divided into user friendly sections, concentration at one topic at the time. The process in which a SLA is developed is spiral like, constant reviews and learning and re-gathering data enabled to produce a complete specification that was understandable and acceptable by both parties.

The specification always refers to the IT objects and its surround business processes to enable the user to understand the business context of the IT object.