

MASTER THESIS

Standardizing the requirements engineering process for a software product vendor.

Regie Mocking
Student Business Information Technology

June 27, 2017

**UNIVERSITY
OF TWENTE.**



"All models are wrong, but some are useful"
— George E.P. Box

"Models are to be used, not believed."
— H. Theil

Regina Adriana (R.A.) Mocking: *Standardizing the requirements engineering process for a software product vendor.*
Final Project, June 2017, Enschede

Supervisors:

N. Sikkel (*Faculty of EEMCS, University of Twente*)

J.M. Moonen (*Faculty of BMS, University of Twente*)

I. van Gisbergen (*Teamlead Information Analysts and Business Consultants at Sqills*)

M. Lanting (*Project Manager at Sqills*)

Summary

Purpose There is a gap in the scientific literature on the requirements processes for a software product vendor. The implementation process for customers has been researched and this gives insight in the vendor's perspective, but there is a gap on the specific vendor's processes. This may be because vendors don't feel the need to share their processes and customers need more documentation because for them the implementation is a one time experience.

This research aims to fill the gap in the literature by creating a model which can be used for the requirements processes for a software product vendor.

Results The model starts with a taxonomy of the requirements. The requirements are divided and further elicited in themes, which are groups of requirements, based on a topic, scenario or business process. After the initial elicitation phase, the requirements will be further processed per theme. This is done in four phases: elicitation, analyzing, modeling and evaluation. These phases will repeat every time until the customer and vendor agree in an evaluation that all requirements from that theme are processed correctly. In the model, this is shown as a spiral where the first round is a high-level analysis, the succeeding rounds are used to analyze the requirements in more detail and at the end of the final round, there is a consensus between the vendor and the customer. Of course, this process has to be fulfilled for every theme and they should be worked on at the same time to ensure that knowledge is shared between consultants.

Application Applying the model in a company means to create a protocol, using the model as a basis. Many elements can be included in this protocol, such as:

- standards for the themes, which are different for every product,
- elicitation techniques,
- documentation style and standards and
- task and role division.

Every company has to create their own protocol on the subjects that are relevant for their product, market and organization.

Validation and recommendation The validations show that it is a good generic model which can be applied to many situations. Various experts have indicated that they can apply it to their own company. It is however needed to create a protocol from it, because the model is too generic and everybody can use it in a different way if no specification has been made.

Preface

On page ii a quote is given by George Box: "*All models are wrong, but some are useful.*" Well, it is a bold statement to say that all models are wrong, but I can relate to the idea that no model is ever complete. Luckily, they can still be useful. This was also my goal, to create a model, a protocol, for Sqills which helps them process their requirements. In the end, the model may not be perfect, because the world is not perfect and it seems impossible to create something that suits every situation. However, as the second quote by H. Theil states: "Models are to be used, not believed" And I hope that this model can and will be used at Sqills.

I'd like to say thanks to everybody who helped me in the process of this final project. Extra thanks goes to my supervisors from the university and Sqills. Hans Moonen, my sort-of-neighbor, thank you for your positivity and the many good ideas that helped improve my thesis! Klaas sikkel, whom I worked for since my second year in college. Thank you for the dedication you have towards your students! The speed with which you answered e-mails and the fact that you could always make time for me was a huge help!

Of course the help within Sqills was also greatly appreciated! Marcel, thank you for the many meetings and so much insight in Sqills that you have provided. Inge, I really enjoyed the many conversations, the useful feedback, but also the non thesis related moments (e.g. about arts and crafts and the need for cappuccino!).

I would like to thank everybody else at Sqills. Especially the people who have helped me with the interviews, but also everybody whom I've shared an office with, enjoyed walks with during lunch, or who I've just had fun with: Thank you all!

And of course, last but certainly not least, Yme, thank you for putting up with me these last months! ;)

-Regie

Contents

1	Introduction	1
1.1	Background	1
1.2	Research Design	3
1.3	Methodology	3
1.4	Report contents	6
I	Background	7
2	Software Products	9
2.1	Implementation process	10
2.2	Critical success factors	11
2.3	Customization	14
2.4	Software product organizations	17
2.5	Conclusion	20
3	Requirements Engineering	21
3.1	Phases/activities	21
3.2	Types of Requirements	26
3.3	Requirements for software products	28
3.4	Conclusion	31
4	Current Situation	33
4.1	S3 passenger	33
4.2	Implementations	35
4.3	Conclusion	38
II	Model development	41
5	Proposed Model	43
5.1	General requirements process	43
5.2	Requirements types	46
5.3	Using the model	48
5.4	Summary	49

6	Validation with employees	51
6.1	Validation approach	51
6.2	Validations	51
6.3	Improvements	54
6.4	Model	55
7	Validation of second version	59
7.1	Validation approach	59
7.2	Validations	59
7.3	Improvements	62
8	Final version of model	65
8.1	Requirements taxonomy	65
8.2	Requirements phases	66
8.3	Full requirements process	68
9	Application of the model	73
9.1	Expert validation	73
9.2	General considerations	75
9.3	Application of the model at a company	77
III	Evaluation	79
10	Results	81
10.1	Answers to research questions	81
10.2	Goal of the research	84
11	Discussion	85
11.1	Abstraction level	85
11.2	Scientific implications	85
11.3	Recommendations for Sqills	86
11.4	Future work	87
	References	89
A	Validation questions employees/experts	93
A.1	English	93
A.2	Dutch	94
B	Handout for second validation	96
C	Example application model	99

Introduction

"Who has to adjust?"

This question has risen multiple times during the work on this thesis. It is a viable question when implementing a software product. The customer wants to have a system that fits their needs exactly and the product vendor wants to sell a standardized product. So should the customer adjust their processes or should the vendor adjust the system? This is a continuous trade-off between vendor and customer.

When a company decides to procure a new information system for their organization, various steps need to be taken and important choices need to be made. A lot of research and documentation on the processes for these purchasing companies is available. However the amount of information on the process of the software vendor is difficult to find. It seems that when a vendor has a good working process they are not likely to share this with the world. Either because they have no reason to, or to make sure they stay ahead of the competition.

For bespoke software, where software is created entirely for one specific customer, various strategies for the requirements process are known. The other extreme is commercial-of-the-shelf software. In this case software is bought as a whole and customization is not possible (i.e. Microsoft Office). The software is already there and when there are differences between the requirements of the customer and the features of the system, the customer will simply have to adjust. This implies there is no requirements process for the customer implementations.

Often, the software as offered by the vendor is somewhere in the middle of these two extremes. There is software available which has been created for a specific purpose and a specific market. However, the various customers in the market have different requirements for their specific situation. For every mismatch with the system, the decision has to be made to customize the software or to let the customer change their business processes: "Who has to adjust and to what price?"

1.1 Background

The research will be performed at Sqills. A software company who offers many solutions for various public transport companies and for other customers, such as the dutch "Postcode loterij". They also offer a seat reservation software product for public transport companies all over the world and this product is where the research focuses on.

1.1.1 About Sqills

Sqills is a software development company. During the foundation of the company, the owners had the idea of building software for the public transport sector because of their previous experience in that sector. When the opportunity came around to build seat reservation software for a German railway company, they used this project to create S3 Passenger: a product which could in a later stadium be sold to other customers as well.

The first version was created for this customer, but basic configuration options were directly implemented in the core of the system. For every new customer with new requirements, a choice was made whether to add new features to the generic part of the software, to add modules or to implement these requirements specifically for one customer.

At this moment, more than five years later, the system has evolved to a product which is ready to be configured and used by a new customer if their requirements are in line with the system. This means that the process of implementing the system at new customers is different from the earlier customers where there was more need for additions to the system. A standard procedure may be helpful for future implementations to ensure that all analysts have the same methods and that all requirements will be elicited. This way the procedure should help ensuring that the system fits to what the customer needs.

1.1.2 Problem statement

Considering the growth of Sqills and the continuous development of the software product they sell, it is wise to think ahead and look for possible improvements for the future. One of the questions that have risen is how to standardize the procedures which are used for the requirements process in the implementation of S3 Passenger software at the customer. This may help to optimize the implementation process to ensure that the system is in line with what the customers needs.

The problem in improving this process is that to the best of our knowledge there are no standards known in scientific publications. It is likely that companies that implement software packages have their own procedures. Unfortunately these are not publicly documented and scientific publications can barely be found on the vendor's perspective of software product implementations. This aspect is a gap in the literature and this paper aims to fill the gap.

1.2 Research Design

The research design consists of the goal that is set for this research, the research questions and the methodology that will be used.

1.2.1 Goal

The goal as written in the style of Wieringa (2014) is as follows:

The goal of this research is to **improve** the requirements engineering process for software product vendors **by developing** a model **which covers** the most important aspects of the requirements engineering process for customer implementations **in order to** help Sqills expand their protocol for S3 Passenger implementations.

1.2.2 Research Questions

The following research questions are meant to provide guidance in reaching the goal of the research:

1. How do existing software product vendors handle the early phases of the growth of their company and product and what are the possible pitfalls in this period?
2. What is the state-of-the-art in literature regarding the implementation process of software products and specifically regarding the associated requirements engineering process?
3. How is the requirements engineering process of S3 Passenger software at Sqills at this moment?
4. What is a procedure that improves the current requirements engineering process at Sqills?
5. Is the proposed procedure working properly according to the involved employees or can it be optimized?
6. What recommendations can be given when starting to use this new requirements procedure?

With this combination of questions, a protocol is developed which will guide the S3 consultants through the requirements engineering process during an S3 implementation. This protocol will then be validated and improved where necessary. The last questions look at possible consequences of the new protocol and how that may influence the organization.

1.3 Methodology

Figure 1.1 visualizes how the answers to the research questions contribute to the research goal in the style of Verschuren & Doorewaard (2007).

The research consists of four basic steps: literature study, case study, model development and validation. Some of these steps may be repeated and others consist of multiple phases, these will be explained below.

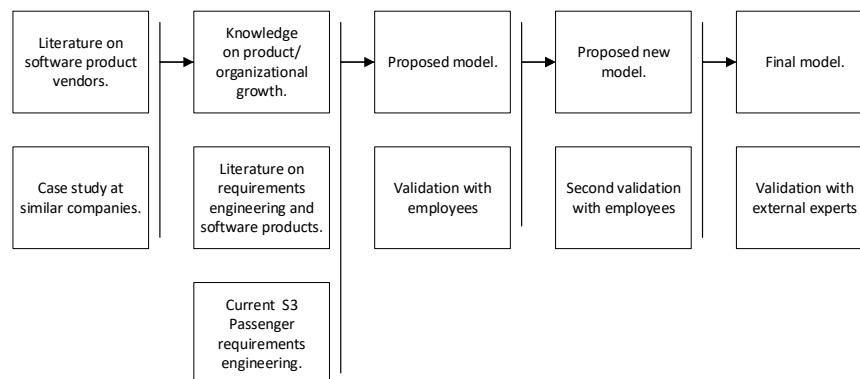


Figure 1.1: Research methodology in the style of Verschuren & Doorewaard (2007)

1.3.1 Literature study

In order to perform the case study and to develop a model, a literature study needs to be done to find the state of the art of the literature on the subject.

Goal The goal is to find information which can be used for designing a procedure for the requirements engineering process at Sqills. This information can be about every related subject, such as requirements engineering in general, software products and their implementation processes and other subjects that may come up during the literature search.

Method The first approach is to decide on the terminology to search for in the literature. When the found literature showed more terminology which seems useful, this was also used for a basic literature search.

The method which is used from this point is by Wolfswinkel et al. (2013) as shown in figure 1.2. This method starts when the researcher already has a set of X articles. From these articles the relevant ones are selected and with these remaining articles, forward and backward citations are searched for.

In the chapters which describe the literature findings, the used terminology is described also. The search engines used for the research are Google Scholar and Scopus.

1.3.2 Case study

A case study is performed at six software product companies

Goal The case study is meant to gain knowledge on the important aspects of developing a software product and having a growing product.

Method six Companies have participated by means of an interview during which is spoken about their product, their company and how they handled various aspects of the growth of both the product and the company. The approach is explained in more detail in chapter 2.4.

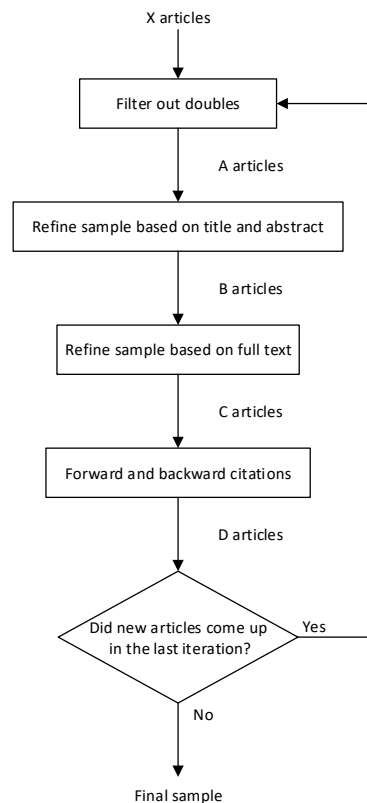


Figure 1.2: Literature review method by Wolfswinkel et al. (2013)

1.3.3 Model development

In line with research question 4, a procedure is developed for the requirements process for S3 Passenger product at Sqills.

Goal The goal is to create a procedure which is useful for employees who are entrusted with the implementation of S3 Passenger

Method The model is created based on literature and design choices of the researcher. With the validation, as explained next, the model is improved and if needed changed to be sure it is applicable to S3 Passenger implementations.

1.3.4 Validation

The developed model needs to be validated and improved where this seems necessary. This needs to be done after the initial development of the model and a second validation round needs to take place after the model has been improved/further developed.

Goal To check whether the developed model satisfies the needs of the users within Sqills.

Method here will be multiple validation rounds. After the creation of the first model, this will be shown to three employees with different responsibilities who can comment on it.

The second validation round is also with employees, but with a different set. to them, the new model and the handout will be shown. them will be asked what they expect to see in a handout and to state whether it would suit their needs.

The final model will be validated by external experts from different companies who have not been involved in the creation of the model and have experience in implementing various software products.

1.4 Report contents

The first part of the report describes the results of the literature study. This forms background information which will be used in the design of the procedure. This background information will be split into two subjects, namely software packages (implementations and organizations) in chapter 2 and requirements engineering in chapter 3. The last information that is needed to create a model is information on the current situation at Sqills, which is explained in chapter 4.

The second part of the report describes how the model is created. The first model will be described in chapter 5. The validation of the model will be described in chapter 6. In this and the succeeding chapters (7 and 8), the successive models are described and validated. The last validation by experts and some considerations are given in chapter 9.

The last part of the report describes the results, in chapter 10, the achievement of the goal will also be described here. After the results, in chapter 11, the practical and scientific implications of the results will be described, along with some ideas for future work. Table 1.1 gives an overview on the research questions, how they are processed and where the answers can be found.

Research Question	Type	Methodology	Chapter/section
RQ 1 Similar companies	knowledge	case study	2.4
RQ 2 Literature	knowledge	literature study	2 and 3
RQ 3 Current process	knowledge	Interviews and documentation	4
RQ 4 Create protocol	design	n.a.	5 and 8
RQ 5 Optimizing protocol	design	interviews and focus group	6 and 7
RQ 6 Recommendations	design	evaluations	9

Table 1.1: Overview on the research questions, their correlating methods and the chapters where the answers are described.

Background

In this part of the report background information is given on the subject of this thesis. The first subject that is covered here is software products, their implementations and organizations in chapter 2. Next, the focus will be put on a specific part of software production and implementation, namely the requirements process in chapter 3. The final chapter (chapter 4) describes the current situation at Sqills with regards to their product, their current processes and the history of the company.

2

Software Products

This chapter describes the state of the art on several aspects of software products, namely the types of software products, the implementation process and corresponding success factors and the customization of software products. Sections 2.1 through 2.3 are based on literature and section 2.4 is based on a case study which is described at the section itself. The literature is found using the method of Wolfswinkel et al. (2013) (See figure 1.2 on page 5) and various combinations of the following terminology: "Software product", "ERP", "ES", "Enterprise System", "SAAS", "IS", "Information System", "CotS", "Implementation Process", "CSF", "Critical Success Factor", "Software customization", "Software product management" and "Software product development"

Nowadays companies use computer programs for everything, ranging from customer relationship management systems to financial and revenue management systems. Many of these systems have to share data to perform well. It is also possible to use one system for a combination of these functionalities. A well known example is SAP, a company who offers many different modules in their product and delivers to various industries. (SAP SE, 2016)

In research and practice, various terminology is used for these type of systems. For example ERP, ES and IS. As shown in table 2.1a, the difference between IS, ES and ERP systems it not distinctive. Some differences as mentioned by Napier (25-03-2011) are: 1) ERP is aimed at improving the functions of the organization whereas the ES helps to improve the overall maintenance and accuracy, 2) ES is mostly used by big companies whereas for **SMEs** it is more common to use an ERP and 3) ERP is more restricted than an ES. Subramanian (26-1-2016) adds to this that ERP systems are more industry specific than ES. Globalteckz.com (9-11-2013) adds that the focus of ES can widen to external functions, such as suppliers.

SME: Small/Medium Enterprise

The software can be sold in multiple ways, the types which is mentioned most in this report are CotS, as SaaS or as bespoke software. Table 2.1b explains these terms and shows the main differences. Where CotS was the common way of selling software, SaaS is more up-and-coming in the past few years. Bespoke software is shown as a separate type in the table. However it can be part of the implementation of an ERP system when extra functionality is needed which is not yet included. Software product is an overarching term for non-custom software. In this report the term ERP will be used since it is more in line with the situation of Sqills because of the specific nature of the product.

When a company starts using an ERP system, they may need to change their way of working. (Pereira, 1999) Firstly they have to use another computer program, but it has more influence than that because business processes may need to be adjusted, it is also necessary to show the employees that the new way of working makes sense

IS	Information System	IS is a broad term used for systems which process or contain information
ES	Enterprise System	System which integrates many processes in a company
ERP	Enterprise Resource Planning	System which integrates many processes related to resources in a company

(a) Naming of package software

CotS	Commercial-of-the-Shelf	Software which is sold as a package and can sometimes be configured to suit the customers wishes
SaaS	Software-as-a-Service	Software which is offered with a license and where updates and other services are included
Bespoke Product	Bespoke software Software product	Software which is custom made for a customer Overarching term for non-custom software

(b) Forms of selling package software

Table 2.1: Types of software

and to convince them to be open to the new way of working. There are more aspects which can go wrong or can cause problems when using a new ERP system. In the next sections the process of implementing software, success factors for the implementation and customization are explained and described.

2.1 Implementation process

When a company decides to start using a new ERP system, many steps need to be taken. It starts with selecting the perfect ERP package and ends with maintaining the software after it has been fully implemented. The phases are defined differently by every practitioner or researcher. The following five stages are derived from Fui-Hoon Nah et al. (2001) and from the implementation roadmap from SAP SE (n.d.):

- **Orientation/sales/preparation**

The first phase starts at the customer with internally acquiring their requirements. Then they need to understand the available packages and assess the compatibility to their requirements. From that point the vendors are involved with the process and the selected vendor will be involved for the rest of the implementation process. (Finkelstein et al., 1996)

- **Analysis/design**

When the system has been selected, a fit-gap analysis needs to be done to understand where the system or the company needs to be adjusted. After a requirement analysis process, a design needs to be made for the changes to the system, the configuration of the system and the changes at the customer. An approach and planning for the rest of the implementation will also be made.

- **Project**

The implementation of the ERP system, which includes change management and business process remodeling for the customer and customizing and configuring the system.

- **Shakedown**

Shakedown is the final stage of the project where the system is tested, the final transition is made and it can be checked whether the employees know how to work with and operate the system.

- **Afterwards/onward**

After the transition to the system, it needs to be maintained and issues may arise on aspects that were not foreseen.

During each phase different actions will be performed, however, most actions take place during multiple phases. For example Requirements Engineering starts in the beginning with high level requirements, will proceed to functional requirements in the analysis and design phase and will be more detailed during the project phase where the system will be implemented or configured. It is also possible that business requirements change or are extended during the process. Another example is the training of the customer where a start will be made with training key-users and more training will be done for all users later in the process.

2.2 Critical success factors

Many papers have been written on what can go wrong with the implementation of ERP systems at companies and also many literature reviews have been done to combine these papers. The literature reviews give a better understanding of the overall issues and critical success factors which are most likely to arise when implementing an ERP system. The literature reviews used in this section are by Ahmad & Cuenca (2013); Fui-Hoon Nah et al. (2001); Tarhini et al. (2015); Finney & Corbett (2007) and Momoh et al. (2010). When a paper has given a top 10 of issues, only these issues are used in the derivation of the most popular **CSFs**.

There is a lot of overlap in the CSFs from the five papers. This may be because there are CSFs which won't change over time because some things are not easy to prevent. Besides this, some reviews use papers from the same period of time which results in using the same papers and and overlap in results. Combining the results from the five literature reviews is meant as an exploratory research. The result will most likely not be conclusive, but gives an overview on the CSFs which arise most often.

Combining the reviews and joining similar CSFs created a list of 22 issues. The researcher has divided the CSFs in three groups, namely "project", "business" and "technical". These are shown in table 2.2.

CSF: A CSF is a factor, like an action or an element in the implementation process which is seen as a critical factor for the success of the implementation.

2.2.1 Project

The implementation team is a team which consists of employees of the customer and the vendor. Their responsibilities lie mostly on the alignment of the customer's business and the new ERP system. The implementation team is supposed to keep an overview on the complete project and notice when aspects are going wrong, out of time or elements are missing.

Project	Business	Technical
1. top management support/commitment 2. project champion 3. ERP teamwork and composition, motivation and cooperation 4. business plan and vision+ timeframe 5. empowered decision makers 6. appropriate use of consultants 7. manage cost and time 8. adequate resources	9. selection of ERP 10. change management 11. business process re-engineering 12. effective communication 13. interdepartmental communication and cooperation 14. training	15. IT infrastructure 16. software requirements engineering, development, testing 17. monitoring and evaluation of performance 18. appropriate business and IT legacy systems 19. data quality 20. troubleshooting/crisis management 21. excessive customization 22. vendor's tool

Table 2.2: CSFs as derived from five papers and grouped in three themes

The critical success factors which are most important for the implementation team are mentioned in table 2.2. Top management support of the customer's company is the CSF which seems to be the most important one. If the top management is not committed to the implementation of a new system, the project is most likely to fail because they are not willing to spend time and resources and employees will also be less committed.

The second CSF is the presence of a project champion. This is to be an employee of the customer's company who can be a motivator and teacher of the new system. The teamwork within the implementation team is a CFS which means that the implementation team should work together in a honest way. They need to cooperate as one team. Their composition is also important because it is optimal with employees from different disciplines and with technical, consulting or business background. The fourth CSF is on empowered decision makers and that means that the people involved with the project should be able to make decisions without everything having to take the bureaucratic route. This fastens up the implementation process a lot. The appropriate use of consultants is also a CSF for some companies. They realize that they cannot do everything themselves, but some things are possible to do themselves. Knowing where to use consultants and where to do things yourself helps prevent issues later on or higher costs than needed. The last CSF is manage cost and time. Many projects are considered unsuccessful because they took too long or the costs were too high. It is up to the implementation team to keep watching the costs and planning.

2.2.2 Business aspects

The business aspects contain the CSFs which are directly related to the customer. Some of the previous CSFs are also related to the business, the difference is not strict, but it can be stated that these CSFs are more directly related to the business

instead of controlled by the implementation team.

The selection of the ERP system is one of the first mayor decisions to be made by the company and it is an important one because it will influence everything thereafter. The most important aspect is that it should fit with the business' requirements. This is what makes it an CSF. The next CSF is change management and is closely related to business process re-engineering (BPR). Change management means that the employees should be willing to change their way of working and their other customs to fit in with the new system. This change can include some BPR because actions may require that other actions have already been taken whilst that was not so with the previous system.

Communication needs to be effective, it needs to be timely and to the point so that all employees know what is happening and what is going to change in the future. This is does not only include communication towards the employees, but also communication within the implementation team(s). This is especially important when the ERP implementation is so big that there is an implementation team for every department. Communication could go wrong when one department changes a requirement without realizing that it influences the other departments. The last CSF on this list is training, which is partly the responsibility of the implementation team, they need to make a plan for it. However the business must provide the possibility for employees to take time to train with the new system and to ask questions without having to dive in too fast.

2.2.3 Technical aspects

The third category from table 2.2 is technical aspects, these are the aspects which are more closely related to the product and the software vendor. These aspects are most of the time the responsibility of the vendor and partly for the implementation team to keep an eye on.

The first CSF in this group is the IT infrastructure. If the new system is implemented, but there is not enough attention to the rest of the IT infrastructure of the company. Or if the system does not align with the companies infrastructure, it can have a lot of influence on the performance. This is seen as a critical factor for success. The phases of requirements engineering, development of the system and testing of the system are also CSFs because it is critical that a system matches the customer's wishes and works as promised. After the system is implemented, monitoring and evaluation of performance of the system is important to ensure that the system works and keeps working after more usage. Data quality refers to the data format and the amount of data that is present in the old system and how it is transformed to the new system. Perhaps data is missing or is saved in another format. If it is not looked at carefully, all data could be interpreted wrong. Troubleshooting and crisis management is important during the implementation because if there is no good crisis management, issues may not be handled well. This can ensure that sudden changes are not taken into account until it is too late and it will cost more money. Excessive customization will be explained further in section 2.3, in short it is so that more customization may seem better for the business processes, but it lacks the easiness of maintenance. By vendors tool is meant that the tools of the vendor should be used, also because maintenance is difficult if different tools are used.

2.3 Customization

When a customer has chosen an ERP system, the requirements of the customer are compared to the features of the system and they will most likely not be a perfect fit. Examples of this fit are explained in section 2.3.1. To make the system fit the customer, it needs to be configured and sometimes also customized. There is a lot of reasoning on customizing software because it has positive effects, but also negative ones. This will be explained in section 2.3.4.

2.3.1 Fit of the System

Parthasarathy & Daneva (2014) states that: "Any successful ERP implementation requires a complete fit between the ERP system and the business processes it supports (Rothenberger & Srite, 2009; Parthasarathy & Anbazhagan, 2007; Luo & Strong, 2004)"

This means that when a customer has chosen a vendor and a system (or even before making the decision), one of the first actions that need to be taken is a fit/gap analysis. In this analysis, the functionalities of the system are compared with the requirements from the customer, which is not an easy task. (Rolland & Prakash, 2000)

2.3.2 Customization types

When the gap between requirements and features needs to be bridged, it is important to ensure that the system can be used by the customer. This can be done in multiple ways as can be seen in figure 2.1. In this figure, the options which are darker have more negative effects on the system maintenance in a later stadium than the other options for customization. The option with the least negative effects on the longer term is to adjust the business processes to fit the new ERP system. This would mean that nothing about the ERP system has to be adjusted so that the vendor does not have to consider every variation of the implementation when adding new features to the system. It has the downside for the customer that they may have to change their way of working.

One aspect that is not mentioned in this figure is to adjust the basis of the system and add the customer's requirements to the core of the system for all new customers. This should only be done for requirements from which more (potential) customers can benefit. This approach does imply a change to the core of the system, but because it is changed for every customer, it is not likely to result in difficulties for specific implementations.

When the customer cannot change its way of working or is not willing to do so, it is often possible to adjust the ERP system with selection options and configurations. This means that features are already implemented in the ERP system. An example of something that often can be configured is to use Euro instead of Dollars or to add more users with certain authorizations. With bigger ERP systems, it may be possible to select modules which are needed or not needed for the customer.

If customization and configuration are not enough, it may be needed to create extra pieces of software to use on top of the ERP system. These can be bolt-ons, specifically made by the vendor of the ERP system. That would mean that the bolt-ons fit the system, but they will not automatically be maintained together with basic part of the system. Another option is to let an external party create the

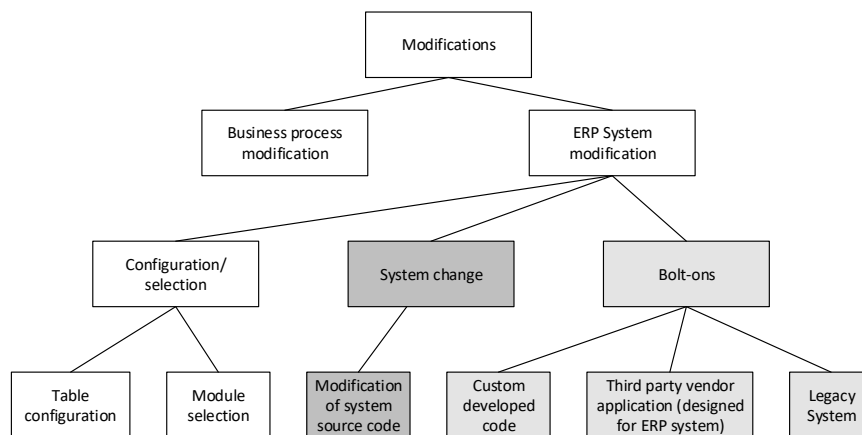


Figure 2.1: Options for customization by Rothenberger & Srite (2009)

bolt-on. In that case there is an even bigger chance of aspects not being explicitly suitable for the ERP system and the chance of it not being able to handle changes to the basic parts of the system is bigger. A last option is to connect the new ERP system to a legacy system that may have been in place before. In this case the same issues may arise as when changes to the ERP system are made which don't align with the legacy system.

The last option is to change system code, this has the greatest chance of issues arising in a later stage. To adjust parts of the ERP system means that every customer may have a different version of the system. For the vendor, this is likely to cause problems with version maintenance. Systems need updates, for example on security. If these updates need to be rolled out, it needs to be checked for each customers' version whether the changes influence the basic ERP system, the configuration and the environment. This is highly time consuming and it may be difficult to keep the knowledge of these various systems.

2.3.3 Reasons for customization

The simplified reason for customizing ERP software for a customer is that the software does not align with the business. Many concrete reasons are mentioned by Harris (2000) in Sharma et al. (2012). For example: The need for different formats of reports, needing different data fields to the database and lack of ERP domain knowledge by the developers. While these reasons are valid and are real concerns for the customers, they are the consequence of more general reasons.

As Light (2005) states: It is important to know the underlying reasons as these can lie in other aspects than simply the alignment of the system to the business.

In section 2.3.2 is described that on the longer term, the best way to make the system fit to the business is by adjusting the business processes. In this case no customization to the system is needed. However, often this does not happen. Either because the customer is not willing to change their processes or because it is not possible for them to adjust these. A reason for this can be that simply using the

best practices of the software package does not work because it does not meet the specific strategic priorities of a firm (Sharma et al., 2012).

Sharma et al. (2012) give the following underlying reasons for the previous statement: 1) The customer has unique manufacturing problems, 2) there is a need of flexible information system, 3) there is a difference in interest of the system vendor and the customer and 4) users do not have the time to fully understand the software package. These reasons are different for example in the last case it may be enough to explain the software whilst with the first reason, a more extensive solution may be needed.

Rothenberger & Srite (2009) divide the factors influencing customization of ERP systems in three groups, namely 1) Pre-project characteristics, which is mainly the organization structure and ERP knowledge. 2) Project characteristics, such as the management involvement, fear of personal disadvantage, the reliance on and the experience of the consultants and 3) Direct influences such as resistance to change and determination to avoid customization, which are influenced by the previous factors.

Zach & Munkvold (2012) tried to identify reasons adding up to the results of Rothenberger & Srite (2009). The ownership type and stage of growth of an SME were found as an addition to organizational structure. They have found that a lot of customization takes place after go-live of the system, which they link to the influence of the growth stage of a SME.

2.3.4 Effect of customization

In section 2.3.2 some types of customization are mentioned and what they would mean for the system. Figure 2.1 shows in which ways the gap between the customer requirements and the product features can be solved and which of these options will have the most influence later on, according to Rothenberger & Srite (2009).

Brehm et al. (2001) have proposed a similar result in which for each type of tailoring a system is shown how likely it is that difficulties will be experienced when the customer wants to upgrade to a new version of the system. Except for changing the business processes similar options are used as in figure 2.1. The result can be seen in table 2.3.

Tailoring type	Effort required for post implementation maintenance
Configuration	None/slight
Bolt-ons	Depends on coordination between ERP vendor and bolt-on vendor
Screen masks	Slight/moderate
Extended reporting	Moderate/heavy
Workflow programming	Moderate/heavy
User exits	Heavy
ERP programming	Heavy, if data from the ERP application is used
Interface development	Heavy/very heavy
Package code modification	Very heavy

Table 2.3: Maintenance effort required post implementation by Brehm et al. (2001)

As can be seen in figure 2.1, the results for various types of tailoring vary from almost 'none' to 'very heavy'. Of course these implications vary per software/ERP package and per customer and vendor. For example the screen masks may in some cases cause more issues when updating to a new version than in other cases where there is a standard for this. For that reason every vendor should have their own rules and guidelines which state how much tailoring they are willing to do.

As for the types of effects; the main effects from customization of software packages are the issues that arise when the software requires updates. The vendor usually sends out a big update once every few years to give a new version with added features and perhaps security updates. Besides these big updates it happens regularly that small updates are rolled out in between. When every customer has their own customized aspects of the software it is possible that functions won't work in the specific customization and the customer will need special attention. If this is the case for every customer, a lot of time is needed. Of course this costs money too, for that reason customers may decide not to get the new version. (Huang et al., 2012; Harris, 2000)

In Sharma et al. (2012), Harris (2000) mentions more pitfalls from customization, namely that a skilled developer is needed for correct rewriting, the fact that the balance of the software package may be disturbed which could hamper its working and that customization can consume resources which are vital for other projects.

Huang et al. (2012) also mentions more pitfalls, namely that the timeline of the implementation process will change when customization is needed and that the project costs will be higher. Besides the difficulties in updates, simple maintenance is also difficult because of the needed knowledge of the specific customization aspects.

2.4 Software product organizations

The organizations who develop and offer software products have a different approach to the development process than vendors who only do projects for single customers. Although this may seem obvious, there is not much attention in literature to the differences between the two types of vendors.

To discover challenges for software product companies, a case study is performed as a preliminary study on the subject. The full results of this study can be found in Mocking (2017). In this case study, six software product vendors are interviewed on their experience with the development of their product and the growth of their company.

This case study resulted in 1) a overview on choices that were made, which had an influence on how their organization and their product developed and 2) the challenges and opportunities for the companies, which are described below.

2.4.1 Choices in product development

With choices in product development is meant which choices the product developers and management have made during the development of the software product. Some of these choices may be made unconsciously and other are made intentional.

Original development

The original development of the product can have a lot of influence on the flexibility of the software. Sometimes a development company creates software for a customer and at a later point it appears that this type of software is also useful for other customers. They may adjust it to suit the other customer's wishes and create a software product this way. The opposite is when software is build create-to-order, where the software is created first and sold to customers later. This way all customization options are already in the software and few changes are needed in a later stage. This means that the developer has to finance the whole development themselves, whilst in another situation the customer already finances during the development. Somewhere in the middle there is software which is created for one customer with the immediate intention to sell it to more customers. This way the development is partly financed and the customer has to invest more themselves to include more customization options.

Organization structure

The organization structure of a company has quite some influence on the focus they have on the product. For example a company may have partner companies who help implementing the product at the customer and ensures that the developing company can focus more or even fully on the development of the product. Another structure is that companies have a parent company, which can mean that the company does not have full say in the strategic choices of the product. When a company is larger, they may need to divide in departments and how they do this can have a lot of influence. Examples is to have the company's developers work in either feature teams or module teams.

Product properties

A software product can be different in many ways and all these differences have influence in the maintainability of the software. For example whether the software is multitenant or not can have a lot of influence on the ease of maintenance and possibilities to push updates. The way of installing the software, whether it is installed locally at the customer or web-based/accessible via the cloud influences maintenance, but may also have legal consequences.

The amount of custom software in a product also has mostly effect on the maintenance and when there is a lot of custom software, the information on these aspects must be extensive in order to have good maintenance. Finally the financial model. It is also related to the organization, but it is set per product and it influences the product. Because when the customer only pays for the purchase of the software, there is few money to add new features because that will only happen when a customer pays for it. Whilst when customers pay a maintenance or usage fee, the developer have more money to make changes to the product.

Customer relations

The correct approach for customer relations differs per company, the type of relation they want to have with the customers and the amount of customers they have.

An option is to have all customer relations handled by partner companies. The relations with customers are less direct, but with many customers this may be an easy solution. There is a choice whether the support department will receive all questions from customers or whether the partner companies handle most questions and only forward the ones they can't handle. The same goes for feature requests.

With a lot of customers it may be difficult to have an overview on feature requests and how many customers want the feature. One of the interviewed companies in the case uses a system for this while others have regular contact with all customers and don't need a system.

2.4.2 Challenges and opportunities

The challenges and opportunities as derived from the case study are shown here. The first two sections, legacy software and version management are risks and vision is an opportunity

Legacy software

Legacy software is software which is in place for a long time of which the owners may not have complete knowledge any more.

There are positive aspects to legacy software, such as the fact that it is often stable because it has worked for a long time with the same functions. However, if something goes wrong, it is difficult to understand the problems and to address them because there is no or few knowledge on the subject.

The only way to avoid legacy software is to make sure there is documentation on all aspects of the product and to always have somebody responsible for every part of the code. When a part of the code is outdated, for example because the language is not supported any more, the code could be replaced. However, as stated before there are also positive aspects to having legacy code, so it is the question whether it is necessary to avoid it.

Version management

Version management is soon an issue when the software product is sold to more than one customer and they have different versions of the software. This can mean that one of the customers has more functionalities in their software. When for example a bug needs to be fixed, this has to be checked in both versions of the software because the versions might react differently to the bug fix.

When there are many more customers on the software product with all different versions, version management becomes almost impossible and every customer has to be maintained separately. A solution would be to have good documentation and to make sure that customers don't have solutions just for themselves. When there are only two or three versions at the customers, it is easier to test on all versions than when there are many more versions.

Vision

Vision is not a risk, but an opportunity to make the company better. In the case study it was clear that some companies have a clear vision for the future of their company and their product. At these companies, the product and organization seemed more structured and organized. They are able to consider every action to make sure it is in line with the vision.

2.4.3 Marketing value disciplines

Mocking (2017) found that the companies had chosen to focus on (a combination of) either the customer relations, the product or the organization structure. Which is similar to a theory by Treacy & Wiersema (1993) about market value disciplines

where the choice is between operational excellence, product leadership and customer intimacy. The idea retrieved from the case study is that every company has to start with a different value discipline than the one they end with. For example operational excellence is only interesting for a software company when they have a lot of customers whilst customer intimacy is much more interesting in the beginning stages of the software product. Product leadership should always be important for software companies as this decides their company and it is not possible to easily change their product along the way.

2.5 Conclusion

This chapter gave an insight in companies who offer software products instead of custom software projects. The many phases they go through in a full implementation process and the factors which have an influence on the success of this implementation can be important driving forces for employees to perform their tasks in a certain way and have specific future visions. The same reasoning goes for the effects that customization can have for the customer and for the vendor. The last section shows how different companies can behave and how their way of working affects the product they sell.

All these aspects can be seen as underlying reasons for ways of working which have to be reflected in the model which will be described later.

3

Requirements Engineering

This chapter describes the state of the art on Requirements Engineering. Similar as the previous chapter, this is based on literature. The literature is found using the method of Wolfswinkel et al. (2013) (See figure 1.2 on page 5) and various combinations of the following terminology: "Requirements Engineering", "Requirements Process", "Requirements", "Phases", "Process", "activities", "Elicitation", "Prioritizing", "Prioritizing", "Grouping", "Ranking", "Analysis", "Validation", "Verification", "Typology" and "Software product"

Requirements engineering is a broad subject in the computer science sector which is defined in ISO 29148:2011 as the following:

"Interdisciplinary function that mediates between the domains of the acquirer and supplier to establish and maintain the requirements to be met by the system, software or service of interest" (ISO/IEC 29148:2011, 2011)

This statement from the ISO standard 29148 shows the main aspects of requirements engineering. Namely the mediation between the customer and the supplier and maintaining the requirements for the system (to be). As the standard also states, it is an interdisciplinary function. This means that often people with different backgrounds need to work together to achieve the desired results. There are people needed who work in the business and understand the business needs, but also people who know about programming and know what the possibilities for the system are.

This chapter is divided in three sections. The first section describes the phases and activities that are involved with requirements engineering. The second section describes the contents and types of requirements that exist. The last section focuses on software products and how the requirements engineering process may be different for software products compared to bespoke software.

3.1 Phases/activities

The requirements engineering process consists of multiple phases, which altogether build up to a complete requirements document. In ISO 29148, the following phases are included in the definition of Requirements Engineering:

"Requirements engineering is concerned with discovering, eliciting, developing, analyzing, determining verification methods, validating, communicating, documenting, and managing requirements." (ISO/IEC 29148:2011, 2011)

Although this is the standard given by ISO/IEC, it is not a strict standard and it is normal to deviate from it. Many researchers use their own standards. Some examples are: Paetsch et al. (2003), who use five phases: 1) Elicitation, 2) analysis, 3) Documentation, 4) Validation and 5) Management. Escalona & Koch (2004) only uses three phases: 1) Elicitation, 2) specification and 3) validation. Zowghi & Paryani (2003) also uses five phases, although they are named differently: 1) Elicit requirements, 2) Analyze and model, 3) Specify, 4) Validate and 5) Manage. In another paper by Zowghi & Gervasi (2003), only four phases are used: 1) Elicitation, 2) completeness, 3) validation and 4) verification.

The paragraph above shows that there are many ways to shape a requirements process. The most common phases are 1) elicitation, 2) analysis and 3) validation. Within each of these three phases many activities need to take place. these activities are derived from various publications. (Van Lamsweerde, 2001; Nuseibeh & Easterbrook, 2000; Lauesen, 2002; Scacchi, 2002)

The phases and the activities which can take place within these phases are explained next.

3.1.1 Elicitation

The elicitation phase is the first phase in requirements engineering. In general this phase is meant to elicit all requirements and for the software vendor the intention is to have a good overview on the wishes of the customer for the system. This means eliciting the requirements, but also specifying and prioritizing them.

The most important activity is the gathering of requirements and this can be done in various ways. Lauesen (2002) has given an overview on many techniques from the customer's perspective. These are shown in table 3.1.

Even with all these techniques, it may be difficult to retrieve all requirements from a customer. According to Nuseibeh & Easterbrook (2000) the researcher must also have knowledge of cognitive psychology, anthropology, sociology and linguistics. Because the way the customer thinks and relates to colleagues, how the politics in a company work and what meaning can be given to what the customer states is important in really understanding the customer, the company and their way of working. It is also questionable whether it is needed to retrieve all requirements because a requirements document may become cluttered with unnecessary requirements. A combination of these techniques will most often ensure that the most important requirements are elicited.

When the customer has sent out an RFP, they have already elicited requirements within the organization. Sometimes this is done with the help of an external party who has knowledge on how to elicit requirements and on how to notate them. In that case it may still be needed to have elicitation sessions with the vendor because they have another vision on the needed requirements for their system.

However, eliciting the requirements may not be enough. An example for this is that not everybody uses the same terminology and people might interpret requirements in a different way. Some of the techniques above, such as brainstorm and interview could take away this miscommunication, still it is important to have an overview on the used terminology and what that means for the customer and for the vendor. It means that sometimes more elaboration on requirements is needed or meta-information needs to be given to be sure that the meaning is clear.

Elicitation technique	Explanation	Result of activity
Stakeholder analysis	Analyzing who the stakeholders are.	Overview of stakeholders
Interviewing	Retrieving issues from stakeholders	Information on chosen subject
Observation	Watching how employees perform their work at the moment	Retrieving details which may not be told instinctively
Task demonstration	Letting employees demonstrate their tasks	Learning about their current work and related issues.
document studies	Studying documents of the customer	Details on specific requirements and current situation
questionnaires	Useful when many employees have to be asked about the same subject	Information about chosen subject
brainstorming	Being creative with a group and create ideas	Random ideas on chosen subject
focus group	More structured discussion	More realistic ideas on chosen subject
domain workshops	Discussing with a more specific goal	e.g. user interface design or overview of business processes
design workshops	Discussing about designing something	Design of e.g. the user interface
prototyping	Showing a simplified version of the system	Feedback on satisfaction with the prototype
pilot experiments	Try out of the system with the customer	Knowing whether the customer can adapt or has to change their organization
study similar companies	Look at similar companies to see how they resolved certain issues	Ideas on solutions
negotiation	A discussion focused on resolving conflicts, e.g. about costs or risk	Resolved conflicts
risk analysis	Analyzing risks of requirements	Knowing how (not) to model certain requirements
cost/benefit analysis	Analyzing costs of implementing a requirement	Knowing whether it is worth the cost
goal-domain analysis	Analyzing whether you covered all the goals from the domain	Knowing the completeness
domain-requirements analysis	Analyzing whether you covered all requirements from the goal	Knowing completeness

Table 3.1: Elicitation techniques by Lauesen (2002).

Prioritizing

Another element that is important in the elicitation phase is to know what requirements are most important for the customer. This is called prioritizing. There are many ways to prioritize requirements. In many situations the customer will address in their RFP what the most important requirements are that have to be met.

When the customer has not addressed what the most important requirements are in advance, there are multiple ways to make these priorities and the best way will differ per customer. In the EasyWinWin methodology by Gruenbacher (2000) prioritizing the requirements is one of the main elements that help give a clear idea on the relevance of the requirement for the customer. There, the prioritizing also helps in later phases, such as the negotiation. In Berander & Andrews (2005) various methods are described. The distinction is made between two types of prioritizing. Namely: Ranking or grouping of requirements. With ranking, all requirements are ranked on importance, with grouping, all requirements are placed in groups and every group has an importance.

Grouping: A well-known method of grouping is MoSCoW. (Clegg & Barker, 1994) With this method, all requirements have to be described as "Must have", "Should have", "Could have" or "Would like to have". Another similar technique is to let the customer give a number to every requirement, ranging from 1 to 10 or from 1 to 5 where a low number means important and a high means 'nice to have'.

Downsides to these type of prioritizing is that the customer may give everything a high priority because they are afraid that the requirements with a lower priority will not be handled at all. This can mean that 50% of the requirements have to be treated as must-haves, which is probably not possible for the developers.

Ranking: Ranking the requirements means that all requirements have to be prioritized relative to each other. This can be done by giving every requirement a number, where the numbers would range from 1 to N where N is the amount of requirements. A downside of this type of prioritizing is that the relative distance between requirements is not clear. Another possibility is to let the customer divide a fictional amount of 100 euros over the requirements. This way it is clear which requirements are most important, but also how much more important they are for the customer than other requirements. For example, if two requirements get 20 euro and all others only get 1 euro, it is clear for the developers what they should focus on.

A downside of this type of prioritizing is that it costs a lot of time. Especially when hundreds of requirements are given for a system, this may take too much time.

3.1.2 Analyze

The person who elicits the requirements is often not the same person as who develops the system. For that reason the requirements have to be analyzed and a decision needs to be made as to whether the requirement can be integrated in the system and in what way this is possible.

This analysis can be done in many ways. Activities that may take place in this phase are the following:

Documenting: This is a continuous element of the requirements process where

all choices and options are documented so it is possible to understand them later on.

Specifying: When the stated requirement is a more general requirement, it may be needed to specify it in order to know if it influences multiple aspects of the system (to be)

Fit/gap analysis: If the requirements are for additions to an existing system or on the configuration of a software product, it is necessary to find out whether the requirements are in line with the current state of the system.

Risk analysis: In the case of additions to an existing system this analysis is meant to show what effects implementing these requirements can have.

Modeling: This means showing how it would be implemented in the system. This can be done by modeling languages (e.g. UML) or by building a prototype. It is meant to show the customer how the requirement will be implemented in the system.

Options: When a requirement can be implemented in more than one way, it is good to show the various options and let the customer decide. This can be done together with a risk analysis and an advise from the vendor.

At the end of the analysis phase, all requirements have been analyzed. If no possible misunderstandings have been found, it should be possible to create the system right from this phase. However, the validation phase is also there to ensure that the requirements document is in line with the customer's wishes.

3.1.3 Validation

Validation is the last of the derived phases. This does not mean that it is less important than the other phases in any way. In summary, this phase is to check whether the requirements which have been elicited and analyzed are in line with the opinion and ideas the customer has for the system.

The activities which need to take place are the following: The modeled options need to be discussed with the customer. For each aspect where it is in any way doubtful whether the solution is in line with the customer's wishes, the proposed solution needs to be discussed. In many cases, there will only be one solution because the system is designed that way.

When there are multiple options to bridge a gap or multiple ways to cover one of the requirements, these need to be discussed and explained. Negotiation may be needed if the customer had another vision than is delivered by the consultants.

So the most frequent activities are:

Controlling/verifying: Where is verified whether the created solution is in line with what the customer meant.

Negotiation: If there are multiple options or when it is not possible to realize the requirements, negotiation with the customer may be needed after explaining why choices are made.

Agreeing: The goal is to get to an agreement between the vendor and the customer on the full requirements document.

Evolving: If there is no agreement yet, the requirements may need to change and the requirements document will evolve into a realizable document.

Completeness

Completeness of requirements means that it must be ensured that all requirements are elicited and analyzed and all the customers expectations are covered. As Lauesen

(2002) states it is a factor that is stressed by researchers, although many practitioners realize it is unrealistic. Lauesen (2003) also states that no matter how complete the requirements document is, the developers will often use their own intuition alongside with the document when creating the system.

Davis (1990) states in Zowghi & Gervasi (2002) that "completeness is the most difficult of the specification attributes to define and incompleteness of specification is the most difficult violation to detect". Some ways to create a complete document are the following: Zowghi & Gervasi (2002) states that one way to ensure all subjects of requirements are in the document is to do a domain analysis. Carson (1998) agrees with the fact that domain analysis is needed, but adds that it is useful to define all interfaces and their behaviour in order to find subjects on which more information is needed. Firesmith (2005) adds to this that it is useful to add metadata to every requirement in order to find missing information. The needed metadata could be the following: 1) Project-Unique Identifier (PUID), 2) Prioritizing, 3) Rationale, 4) Source, 5) Status (may include more than one kind of status) and 6) Verification Method

3.1.4 Changing requirements

During the requirements elicitation phase, the requirements are retrieved from the customer and in the end there may be an agreement between the customer and the vendor on this list of requirements. However, in every project there will be requirements that change during the development phase, the implementation phase or even when the system is in use. (Segal, 2005) Nurmuliani et al. (2004) gives a list with reasons for changing requirements:

- Defect Fixing (e.g. bugs from earlier releases)
- Missing requirements
- Functionality Enhancement (e.g. extended wishes from customer)
- Product Strategy (decision from vendor)
- Design Improvement
- Scope Reduction
- Redundant Functionality (when function was already in product)
- Scope reduction
- Obsolete Functionality (functions which are not needed any more, especially in long-term projects)
- Erroneous Requirements (wrong requirements)
- Resolving Conflicts (e.g. between requirements or between users)
- Clarifying Requirements

These reasons have various origins. These can be in because of passing time or changing environments and they can be raised by any (combination) of the stakeholders in the project.

3.2 Types of Requirements

There are various ways to classify requirements. A common distinction is between functional and non-functional requirements. Functional requirements describe aspects such as logic in the system, data requirements and interfaces. Non-functional requirements describe quality and performance of the software.

In the book by Lauesen (2002) the requirements are categorized as the following: 1) Data requirements; 2) Functional requirements; 3) Quality requirements; 4) Managerial requirements; In this case data requirements consists of the system state, the databases and the input/output format for communication and functional requirements describe tasks and user interfaces. Lauesen agrees that most requirements engineers describe data and functional requirements together because the functional requirements are often based on the data, however, from a practical view the separation seems more appropriate. Quality requirements are based on performance, usability and maintenance and managerial requirements are based on the project, Lauesen describes them as the gray line between contractual issues and requirements.

Another distinction has been made by Bahill & Madni (2017) and it separates mandatory requirements from trade-off requirements. In this case the mandatory requirements are requirements that cannot be torn with and they have to be complied with exactly as described. The trade-off requirements often have a sense of variability in them. They may state that that a high-level function must be supported, but how it happens is not been made explicit or it could be a requirement for availability between certain percentages of time.

Requirements origin

Bahill & Madni (2017) also gives an extensive list with possible origins of requirements as can be seen in table 3.2. Bahill & Madni also show that not every researcher agrees with the taxonomy of requirements as he refers to Grady (1993), Wymore (1993) and Kerzner (2013) who respectively state that there should only be five (functional, performance, constraints, verification, and programmatic), six (input-output, technology, performance, cost, trade-off, and system test) and three (functional, nonfunctional performance, and supplemental) types of requirements. They state that every other source fits in with one of their sources.

Requirements can originate from:	
Functions	Input-Output
Technology	Performance
Cost	Trade-off
System Test	Built-in Self-Test
Company Policy	Business Practices
Systems Engineering	Project Management
Marketing	Manufacturing Processes
Design Engineers	Reliability
Safety	The environment
Ethics	Intangibles
Common Sense	Laws or Standards
The Customer	Legacy Requirements
Existing Data Collection Activities	Human Abuse
Political Correctness	Material Acceptance
Other Sources	

Table 3.2: A taxonomy of requirements (Bahill & Madni, 2017)

Level of the requirement

Lauesen (2002) gives a distinction between the level of the requirements. The "highest level" is a goal requirement which is mainly a requirement set by the management of the customer of the general goal of the system. The second level is "domain level" which describes the user tasks for the system. The third level is "product level" where functions are described as they should be in the system. The lowest level is "design level" where the requirements are understandable and usable by the programmers. Every new level is an elaboration of the previous level where the final system will be specified more concretely.

3.3 Requirements for software products

In this section, the focus is on the requirements process for off-the-shelf software and how it is different from bespoke software.

With bespoke software, the customer can decide on every aspect of the software he is purchasing. The vendor will most likely have some restraints considering the possibilities with the programming language, higher costs or the capabilities of the employees with regard to coding and maintaining the operations. However, everything is negotiable and customers have a lot of say in their product.

When the customer decides to purchase a software product, there is less choice on the exact contents of the product as the product already exists. An example is MS Office, a system which cannot be customized at all. A different example is SAP where there are more configuration options, which is necessary because the customers are different. With some products it may even be more extreme and parts of the system can be adjusted for the customer. This means that the requirements of the customer cannot always be just as detailed and will not always be granted just as easily.

Besides the customer having requirements for their own implementation of the system. There is also the matter of requirements for the system in general. As Finkelstein et al. (1996) states, there is a big difference between the peripheral requirements which are only meant for this customer and the requirements that apply to the standard system and are meant for every customer.

3.3.1 Proces

The requirements process in creating a software product and implementing a software product is different from the requirements process in a 'regular' bespoke product. Van De Weerd et al. (2006) describes the process for gathering requirements for the main product, which means the product's elements which are or can be used by every customer.

"Requirements management starts with gathering all requirements from within the company and from external stakeholders. These are translated to product requirements (i.e. requirements that will be implemented in the product) by removing the duplicates, connecting requirements that describe a similar functionality, and by rewriting them. Then, the requirements are organized per product and core asset. " (Van De Weerd et al., 2006);

As Van De Weerd et al. states, the requirements for the product are a combination of the requirements of all the customers and the requirements of the vendor themselves. Together they decide what the product will look like. The rest of this section focuses on the requirements that customers have for their new system or the implementation at their company.

In section 3.1 the general phases of requirements engineering are shown. In the following sections the phases are explained again with an application to software products

Elicitation

In the previous section about elicitation in general, many techniques are shown for eliciting requirements from a customer. There is not much difference between eliciting requirements for bespoke software compared with requirements for a software product. Therefore the techniques from table 3.1 are in general just as applicable. The only difference is that the product vendor already has knowledge on the market, this eliminates the study of similar companies as an elicitation technique.

Prioritizing the requirements is also useful for software product implementations, perhaps even more important than with bespoke implementations. This is because with software products it is often not possible to implement all wishes from the customer because they are not in line with the system as it is.

By prioritizing the requirements, the consultants know how important it is for the customer to have certain requirements implemented. If an important requirement is not possible with regular configuration, they need to discuss whether to custom build the feature or to stop with the implementation process.

Analyzing

There are many aspects related to analyzing of requirements. Where in the previous section fit/gap analysis is shown as a possible part of the process, in software products, this is the most important part of the process. This is because it is always about using an existing system and checking whether the requirements from the customer can be mapped to that system. So the fit/gap analysis has to be performed on every requirement that was stated by the customer.

Documenting and specifying requirements also needs to be done to have a good overview on the requirements and to be able to check whether there are related issues.

Modeling has to be done for every requirement, but it is mostly interesting when the fit/gap analysis shows a partial fit or no fit for the requirement. In this case the consultant has to consult with information analysts and the product owner about possible solutions or options for the gap. For every one of the possible solutions, a risk analysis should be performed to know the implications of adding functionalities on the performance on other functionalities.

Validation

Validation is the phase in which is checked whether a specification is correct, complete, unambiguous, consistent, modifiable, verifiable and traceable. (Lauesen, 2002) There are multiple ways to deal with these criteria. The requirements can be checked in isolation or against their surroundings, which can be other requirements or for example users.

In order to check the requirements with the users, it must be made sure that the users understand the requirements and know why certain solutions are or are not possible. If they don't agree, this can be discussed and this may give additional information so that a new analysis may need to take place.

3.3.2 Types of requirements

We can distinguish some types of requirements and see how they are handled in various variations of software packages. The four types from Lauesen (2002): Data, Functional, Quality and Managerial, are used here.

Data

A data requirement is a requirement that states what data must be kept and how it may be changed in the system. In a fully off-the-shelf product, such as MS Office, it is not possible to change the data used in the system and every customer will use the same.

In a more specialized system, the customer may be able to customize what data they would like to let the system handle and who has the rights to change what data. Also reporting can often be changed to suit the data needs of the customer.

In a more extensive system, more configuration for data is often possible. For example a system for a store where for some customers it must be possible to add additional information about products, such as expiration date or safety measures. If the customer has these requirements, it is likely that a system will be chosen which can handle these data requirements. If it appears that the requirements can't be met in the system, it is more likely that the vendor will look for a work-around when the customer is big enough. When there are more customers with the same wish for data possibilities, it may be added to the standard system.

Whilst at bespoke systems all data requirements can be met in the system under development, because it is custom made for the customer.

Functional

For functional requirements the same reasoning can be used as for data requirements. Functional requirements are all actions that can be done with the system and the logic of what the system does with all data. For an off-the-shelf system such as MS Office, all logic is decided and is the same for all users. In a more specialized system the users are often able to customize the system so that it has different logic where that is needed for them.

In a extensive system, there may be invisible configurations where the vendor configures the system so that the needed logic is possible for the customer.

Quality

SLA: service level agreement
Quality requirements are non-functional requirements and are most of the time covered in a **SLA**. In an off-the shelf system such as MS Office, the customer has almost no choice with regards to performance and usability requirements. Service requirements are not applicable because the software is purchased and installed locally, so the customer is not dependent on up-time by the vendor.

With more specialized systems, these quality requirements may become more important. Especially with SaaS systems, where the software is delivered as a service including cloud access, these requirements are important. They state how accessible the service is (in percentages of time) and for example on which platforms these

can take place. Often it is possible to negotiate these requirements. The vendor has to be honest about what they can actually offer and the customer will most likely want to have the highest requirements.

With bespoke software, there is often not much more negotiation possible than with specialized SaaS systems.

Managerial

Managerial requirements describe the process of implementing software, this can include a timeplan and the types of communication during the process.

With small off-the-shelf systems, this is often not applicable because purchasing it does not entail any communication with the vendor.

In bigger projects, this becomes important, especially in projects where implementation costs multiple months or years. Some companies will have standard processes and are not willing to ease into the requirements of the customer, but often these requirements are easily negotiable and are just meant to let the process run smoothly.

Level of requirements

When a customer requires a new software system, they create a list of requirements. Depending on the company and the consultant they may use, the list will look different for every customer.

When deciding on a software product, the high-level requirements are used to filter out the software products which will most likely not work for the company. With a select group of products left over, the lower level requirements will be used to decide on the product. It is possible that vendors offer a prototype of their system based on these requirements. This means that the goal and process requirements are supposed to be covered and the customer can state whether the system implements these requirements good enough.

When a decision has been made for a vendor and system, the domain level and product level requirements are analyzed and have to be implemented.

Another approach for a vendor is to go through all the requirements at the same time and don't pay attention to high versus low level requirements. A reason for this may be that the high level requirements can easily be met, but the solution does not align with the low-level requirement the customer had in mind. In that case the customer can decide not to choose the system in a later phase because of the details which are not as they want them.

A change in the implementation of low-level requirements often means that the customer has to change their business processes or way of working, which they don't want. But the vendor will also not want to change too much to their system, which gives the question of who has to adapt, or is there another alternative?

3.4 Conclusion

This chapter covers many aspects of requirements. Not all these aspects will be used in the final model, but they all add to the basic understanding of what a requirement is and how a requirement should be processed. The basic phases, namely elicitation, analyzing/modeling and validation will be used in the later model. Also aspects such as prioritizing and the fact that requirements can easily change during a project, will be taken into account.

Current Situation

Sqills has many products and many customers. The main product is S3 Passenger, which is sold to multiple customers in public transport industry. Besides S3, Sqills has many other customers for whom websites and webshops are built. The research focuses on the product S3 and this chapter gives more background on the system and its implementations.

At first the system is described and in the second section the implementation process at the customer is explained.

4.1 S3 passenger

The S3 Passenger system is a system made for railway and bus companies. The system was originally created for one customer with the intention to sell it to more potential customers and that worked out. At this moment there are seven customers using the system or working on implementing the system.

The main functionalities are inventory management with seat reservation logic and ticket sales/distribution across various sales channels. This means that all routes and stations are logged and that possibilities for combining routes in one ticket are made. Besides that there are many other possibilities with the system, such as a module "revenue management" and a mobile device for conductors.

4.1.1 Original development

The first development of S3 started with the request of HKX, a railway company in Germany. Sqills was already looking for an opportunity to create a software product for railway companies as they had experience in the sector and felt that there would be a market for a product like that. So they took this opportunity and not only built the system for HKX, but also spent some of their own hours into generalizing the system so that it would be possible to use it for other customers too.

After this first implementation, indeed more customers came who were interested in the system. The generalization of the system during the first implementation was useful, but not every functionality the next customer wanted from the system was implemented yet. Over the years as new customers were interested the system was also extended to suit their wishes. Extra modules were created and new functionalities were added. At this moment the system seems complete and it is useful for many customers in the current state. However, the system will never be fully finished as it is not possible to know what every next customer wants with it.

4.1.2 Functionalities

A definition that Sqills uses for S3 Passenger is the following:

S3 Passenger is:

- a modular software suite with a Software as a Service model for
- orientation, booking, reservation and multi-channel ticket distribution of
- multi-segment and multi-stopover passenger transport services
- with or without seat reservation
- including sales of additional on-board products and services and
- after-sales operations of previously created bookings

The whole system has a modular set-up, is SaaS oriented and web-based. A standard implementation includes all the basic modules which are needed to set-up the customer's environment. This includes the routes and network where the customer provides transport, but also the timetables and CRM processes that are used in the company. A typical environment for an S3 Passenger implementation can be found in figure 4.1.

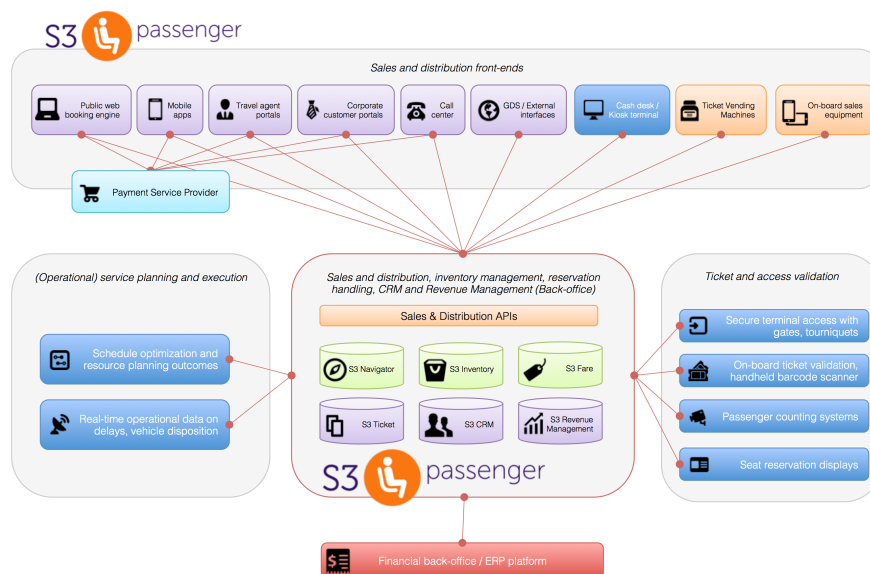


Figure 4.1: A typical implementation environment

The main modules of S3 are:

1. Navigator: timetables and routes, stops and connections and correlated management.
2. Fare: products, tariffs, discounts and prices. Also additional products such as bike and food. Many rules to use for combinations of options.
3. Inventory: Keeps definitions of the inventory material (trains etc). Also has the possibility to adjust materials and configurate seats and group bookings. Seat reservations happens in this module.
4. Ticket: The basis of all transactions. Keeps client data and property of tickets. Also creates booking flow logic and possibilities of aftersales.

5. CRM: Keeps customer details, is separate from the other front-ends. Technically it is optional, most of the time it is used for maintaining user profiles.
6. RM: Revenue management: Is also optional. It contains logic that decides how expensive tickets should be to get the highest profit in the end. Has historical analysis and forecasts.
7. Auth: Responsible for administrative tasks regarding users and their authorizations.
8. Config: Technical and functional configuration management, so contains many parameters.
9. Orientation: A supporting module for the other subsystems
10. Communication: all external communication to customers, passengers and other users, is related to ticket module for communication about fulfilling and delivering tickets.
11. Payment: IS between the back-office and front-end so that the outside world is not directly communicating with the back-office. Ensures validation of payments.
12. Validation: Inspecting of tickets and validation by the conductor. Also supports tickets of other distributors.

The first six modules are clearly visible in figure 4.1. The other modules are more in the background and not directly accessible for customers. The figure also shows the possibilities for sales and distribution (on the top of the figure), for validation of the tickets (right side of the figure) and for planning and distribution (on the left side of the figure). These are all interfaces which can be created by Sqills and are just a layer on top so the customers can use it. The financial system or ERP system which is shown on the bottom of the figure is an external system of the customer for which APIs are made to connect. There is no intention to include the functionalities of these type of systems into S3 Passenger.

4.2 Implementations

The first customer implementation of S3 Passenger was at the company for whom the original system was built, namely HKX in Germany. Many functionalities were decided on based on this customer and the knowledge of the project managers.

After the first implementation of S3 Passenger at HKX was finished, the system was requested by a customer Sqills was already familiar with, namely Syntus. They used the S3 Passenger software in a slightly different way from the original meaning. What Syntus needed was a system that helped them decide whether buses had to drive or not. So seats needed to be reserved in order to let the bus drive at all. Whilst the intention of the system is that the bus is certainly driving.

After this implementation a contract was signed with Irish Rail. The system was not complete enough for this customer. For that reason the implementation process is taking more than two years. Several modules are added or extended and also custom implementations are done for the customer.

For customer Ouibus, the implementation was done within half a year and additions were mainly made on the Revenue Management module. In 2016 two other implementations were done for IZY (Thalys) and Locomore.

Every implementation process so far has been adjusted to the needs of the customers and the customers were able to steer the process a lot. An overview on

the process so far is shown in section 4.2.1. Section 4.2.2 will give more details on the requirements engineering process of the previous implementations.

4.2.1 Implementation process

Most implementation processes start the same way. The (potential) customer sends out a request for proposal. If Sqills is selected, the high-level requirements must be met and the next step is to get the rest of the requirements elicited. This is mostly done with brainstorms performed by Sqills or a partner company. Sqills assesses the requirements and decides whether or not to implement them.

The technical implementation has three main elements, namely the implementation of the functional requirements, the creation of the required interfaces and the configuration for the IT landscape. After technical implementation the system has to be tested extensively with the data and settings from the customer. When all the testing is positive, it is possible to go live. After going live, support is given and if any changes need to be made, this is possible.

With the experience gained from the mentioned implementations, a general roadmap has been set-up for the implementation process. All phases and corresponding deliverables are shown in 4.1. This roadmap is still subject to change as new projects may give new insights for improvement.

RFP: request for proposal

- **Sales:** The first phase starts when a potential customer addresses Sqills with a **RFP**. Often this RFP contains the high level (non-)functional specifications of the customers wishes. Sqills can analyze whether S3 Passenger fits the requirements and if not, whether they want to extend the system so that it fits the requirements.

When these decisions are made, a response is send and if the (potential) customer is interested, an implementation and communication plan can be made.

- **Project initiation:** This is the first phase after a contract with the customer is signed. In the initiation phase many aspects have to be set-up as shown in 4.1. All requirements have to be elicited, made concrete and agreed on by the customer and Sqills.
- **Configuration:** The configuration phase means that all functional requirements which can be configured have to be realized.
- **Custom development:** Custom development is only necessary when it is agreed that Sqills will build functionalities which they do not want to build into the general system. Then functionalities will be custom developed for the customer.
- **S3 Product development:** This phase is only necessary when the customer wants functionalities of which Sqills belief that are also necessary for other customers. These functionalities will be built into the standard modules of S3.
- **Testing:** Testing has to be done for each of the previous three phases and for the system combined. There is a master plan for testing which includes all elements that need testing and combined testing. The system also needs to be tested with the real data as the customer is going to use it.
- **Migration/Hand over:** During the migration, the customer goes live. This means that all previous steps must be finished and all data must be correct in the system.
- **Closure:** In the closure phase, the project is evaluated and if needed, all documentation is updated.

- **Support:** Support is a continuous process after the go live where the system is monitored and any problems are addressed.

In the latest implementations (Irish Rail) a partner company is used. The partner company assisted mostly in acquiring the requirements for S3, but before that the company also assisted the customer in choosing the vendor (Sqills).

In most situations, a potential customer sends out a request for proposal. This request consists of a list of high-level requirements for the system and Sqills (and other vendors) create a proposal for the implementation of their system. The customer will then choose one of the vendors who offers the most desirable solution.

Sqills has a overview on the deliverables as they are In table 4.1 every deliverable is shown

4.2.2 Requirements process

For every project the requirements process has gone differently. In most cases the requirements are already known at the beginning of the project because the customer has send out a RFP. In these cases, the salesperson from Sqills firstly checks the requirements for compatibility and the customer decides whether to continue with the project together with Sqills.

When a project is continued, the business consultants from Sqills get to work and review all requirements again. Often they see that the salesperson has handled the requirements more lightly than they do and they see more flaws and issues which are not as easy to implement as the salesperson has stated before. The business consultants analyze every requirement and if needed they ask for additional information, either from the customer or from the product owners or information analysts at Sqills.

All requirements and questions are noted in the system Sqills uses for this. The system is also accessible by the customer, so they can read questions and reply to them. In some implementation the requirements are analyzed by multiple consultants. In this case the requirements are divided over groups and these groups are divided over the consultants. When a requirement is fully analyzed and the solution had been approved by the customer, it is marked as closed in the system.

All requirements have to be analyzed this way and when there is an agreement on all of them, the project goes to the next phase of configuration and custom development. Of course there will always be more requirements which pop up after the original analysis had been done and then these requirements have to be analyzed again and checked what the possibilities are.

There is also an example of an implementation where the customer did not have requirements set up in advance. During the requirements process for this customer, the consultants worked even more closely together with the customer and together they worked through various user stories and derived requirements from that. for the further analysis of the requirements, the requirements were mainly grouped in the same way as the requirements were retrieved

4.2.3 Partner companies

In some cases partner companies have been used. The partner companies have different roles in different companies. The customer can have asked the partner company to help them in the search for a software vendor or it can be a partner

company who provides parts of the solution for Sqills. Various situations will be described below.

Customer contact

In general there is a lot of contact with the customer needed during the implementation. This is to make sure the solution fits their needs and they understand each other. Some partner companies expect that all communication goes via them instead of directly to the customer. Sqills tries to avoid this, as an extra step in the communication process often causes miscommunication.

Eliciting requirements

When the customer requests a partner company to assist them in the implementation of a new software product, often this partner is involved in the whole process. This starts with eliciting the requirements to use for a RFP. They will also assist when more information on the requirements is needed until the whole implementation process is finished.

Langer (2016) states that for the customer there are many advantages to outsourcing the elicitation part to a third party: 1. Existing IT staff does not have the expertise or time. 2. An outside view can be advantageous. 3. Existing list of preferred solution providers. 4. Overall savings. 5. Faster solution. 6. Less politics.

Implementation partner

Every partner is an implementation partner as they assist in the implementation process. However, there are partners who don't assist in the eliciting of requirements. These partners may be helping in the development of certain aspects of the product or they may help with delivering hardware which is needed for the product.

4.3 Conclusion

This chapter was meant to give insight in Sqills as a company, their product and the way they work at the moment. This chapter gives information for the first version of the model to be based on. All this information will be used in the model because the model has to be in line with the current situation. The later validation of the model will give even more insight in the way of working and will give more direction to the applicability.

Sales

- Answers to a Request for proposal
- Contract: SLA, DAP
- Implementation plan
- Communication plan
- List of potential new product features
- List of high-level requirements
- Git/gap analysis
- Architecture landscape (high-level)

Project initiation

- Set-up of project team
- Set-up of project governance
- Set-up of confluence page
- Kick-off
- Agreed list of detailed functional and non-functional requirements
- Business processes (re)design
- Interfaces specification agreed
- Specification for SaaS environment (otap+migration environment)
- Migration plan

Configuration

- Functional specifications agreed
- Agreed configuration of S3

Custom development

- Detailed specifications for front-end and interface development
- Custom made software developed and unit tested
- Interfaces developed and unit tested
- Unit test scripts developed
- Migration scripts developed

S3 Product Development

- S3 product development
 - New product release
 - Test report
- S3 implementation
 - Updated user guides
 - Updated training material

Testing

- S3 implementation
 - Master test plan
 - Detailed test plans for system test, end2end tests and uat
 - System test S3
 - End2endtest S3 and external systems
 - Performance test
 - Security test
 - Implementation is ready to be handed over to support
- Customer
 - Accepted solution (S3, S3 and external systems)

Migration/ Hand over

- Switch over playbook
- Accepted migration scripts
- Go live

Closure

- Discharge
- Project evaluation document
- Updated knowledge base

Support (standby project)

Table 4.1: Current implementation roadmap for S3 (January 2017)

Model development

In this part of the report the proposed model for Sqills is described. Using the information from the previous part, firstly a basic model is designed (chapter 5) which is validated with employees within Sqills (chapter 6). The results from this validation are used to create the second version of the model as shown in the last section of 6. This version of the model is also validated with a (partly) different group of employees (chapter 7). Altogether, these chapters work towards the final model as described in chapter 8. The final model has been shown to experts outside of Sqills, their opinion, together with the opinions of the Sqills employees, form the basis of the considerations which are shown in chapter 9

5

Proposed Model

The goal of this research is to propose a protocol for Sqills which they can follow in their requirements engineering process. As explained in section 3, there are no general protocols for RE for software packages. For that reason, a design has been made, using the process from SAP as retrieved from Daneva (2004), in combination with literature from i.a. Lauesen (2002) and the design choices of the author.

Firstly, the main elements of the model will be explained. After that, some key indicators on how to use the model will be explained and the chapter concludes with a summary.

5.1 General requirements process

In this section, the requirements process is described. This process consists of the various steps/phases that users of the model need to go through.

5.1.1 Prioritizing requirements

In this model, the aim is to tackle the most important requirements first, just as can be seen in the model of SAP in Daneva (2004). In the conversations with the employees, they all mention a different vision on what the most important requirements for S3 software are. One employee mentioned that the seating process is the most important together with the aftersales possibilities. Which means, the order in which seats are filled when groups want to get seats together: do they first fill up a row or are they placed in two by two behind each other? There are many options for this and it is important that this is possible in the system. The second mentioned aspect is aftersales, which was mentioned by both employees. Aftersales means the process and possibilities after a seat is booked. The questions in this case are: is it possible for reservations to be canceled? Is it possible to change the reservation to another seat, to another day or even to another route? And who can make these changes? Can the customer do this or is it only possible for booking agents to make changes?

The other opinion on the most important requirement is that it is up to the customer what they find important. When the customer can think of a hundred requirements, only a part of these are as important that the project could be canceled because of it. It is most important to prioritize the requirements of the customer and see if the most important ones can be met before starting with an implementation project. (See section 3.1.1 for information on prioritizing)

5.1.2 Processing requirements

Processing requirements is a comprehensive process which requires a lot of attention by Sqills. As mentioned in section 3.1 there are many phases in a requirements process. For this model the decision has been made to use the phases elicitation, analyzing, modeling and negotiation. This decision has been made based on the fact that processing requirements for a software product is a bit different from a bespoke product. Elicitation is a mandatory step. Analyzing and modeling have purposely been pulled apart to be aware of the fact that a lot is already possible in the product or in previous implementations of the product. The analyzing phase can analyze this and the modeling phase will focus on fitting the solution to the specific customers wishes. The negotiation phase is where the team from Sqills can discuss the modeled solution with the customer.

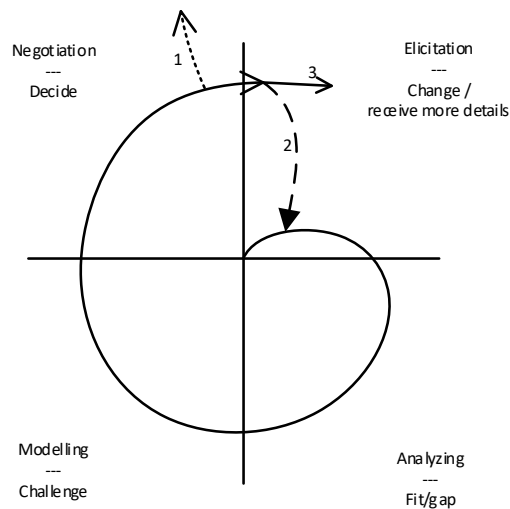


Figure 5.1: Phases every requirement has to go through

Elicitation

Elicitation is the first step in the process. Often the customer will have requirements beforehand, especially in the beginning. In this phase, not only all requirements need to be elicited, they also need to be prioritized to know which ones are most important and need to be looked at first. Elicitation can happen in multiple ways and for every subject a different approach may be most useful. For example for the regular modules of the system, a discussion or brainstorm session with the customer is the best way. For the IT infrastructure the optimal way may be to look at documentation on the requirements from systems which need an interface.

Analyzing

When the most important requirements have been elicited, they must be analyzed to see whether they fit in with the system. This fit/gap analysis sorts the requirements

in a few categories. Which are roughly: 1) requirements which are possible without any configuration or adaption of the system, 2) requirements which are possible with some standard configuration 3) requirements which can not be realized within the standard configuration.

When requirements fall in the third category, a deeper analysis is needed. It is welcome to know if other previous customers have had the same wishes and to look how it was solved in their case. If not, it must be decided how to handle the requirement. The options are as described in section 2.3.2 and for each of the possibilities, a risk analysis is needed to see the impact of this requirement/feature on the functionality of the system.

Modeling

In the modeling phase, all requirements and their analysis are considered again and for each requirement, even for the ones which fit in the first category, a view is given on how it will be realized in the system. For the requirements in the last category, the ones which are not standard possible, it is possible that no solution is given if it seems impossible. The more likely situation is that alternatives are given for these specific requirements. The contents of those alternatives depend on whether Sqills wants to put the functionalities in the system and whether Sqills is willing to build custom functions for this customer.

Negotiation

In the negotiation phase, all modeling of the requirements need to be discussed with the customer and the customer can state whether the solution will work for them. It can happen that in the elicitation phase, a requirements has been described in such a way that multiple interpretations are possible. If the customer and Sqills had a different interpretation, this will become clear in the negotiation phase only if all requirements, including the ones of which the functionality is already in the system, are discussed.

The negotiation phase can have multiple results. Especially in the beginning, if must-have requirements from the customer cannot be implemented, this phase can mean the end of the implementation process because the system does not meet the wishes of the customer. (1 in figure 5.1) The phase can also result in the need to go back to the requirements which were set and to analyze and model them again. (2 in figure 5.1) If everything went perfectly, the next step is to move on to the next set of requirements. (3 in figure 5.1) These requirements have a lower priority than the requirements in this first process or to have a finished document.

5.1.3 Full requirements cycle

The implementation team will have to go through these four phases multiple times, starting with the requirements with the highest priority and ending with the requirements with the lowest priority. For this reason the phases are combined in a spiral as shown in figure 5.2.

The spiral shows the described phases and how at the end of the negotiation phase there are three possible ways to proceed. Figure 5.2 shows a spiral with four cycles through the four phases. This amount can be different for every process and depends on the amount of requirements and the ability of the implementation team to prioritize them.

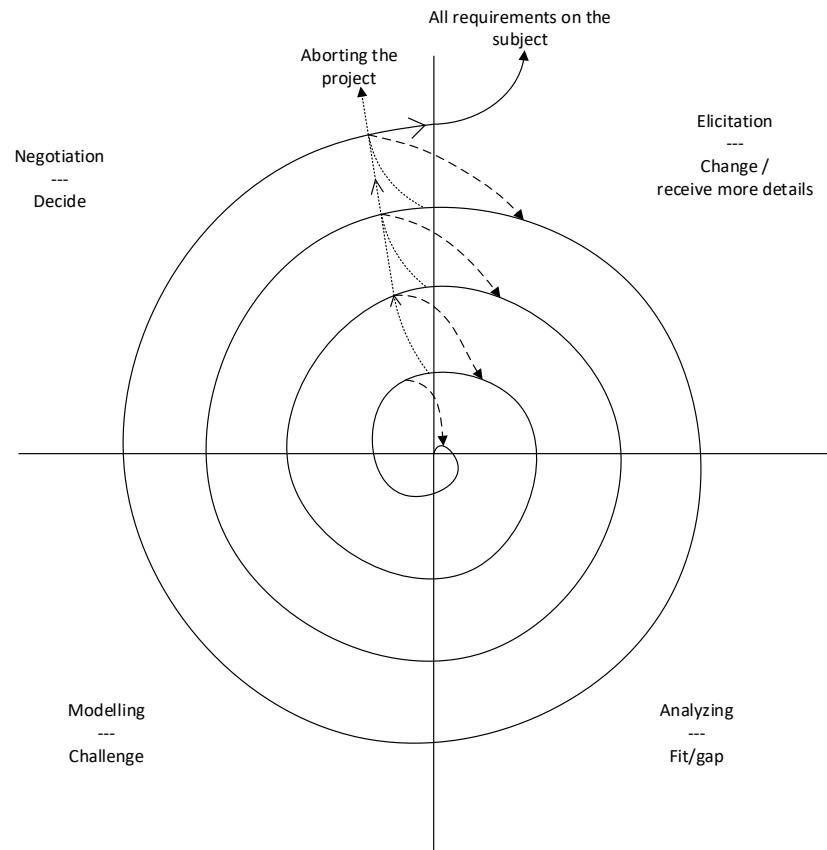


Figure 5.2: Spiral representation of the full requirements process

5.2 Requirements types

Above, the general requirements process is described. However, one of the main challenges is to elicit all requirements and have a complete requirements document that covers all aspects of the system that needs to be implemented.

To reach this completeness, it can be helpful to have a guideline as to which subjects there are which need elicitation. For that reason, a distinction has been made to group the type of requirements. On each group, the general process (the spiral) has to be applied to make sure that the most important requirements for every requirement type will be analyzed firstly.

The elements that have been derived from the literature and the conversations at Sqills are 1) the non-functional requirements, 2) the functional requirements per module and 3) the requirements for interfaces.

Another distinction of requirements types as mentioned by one of the employees is whether they come from the customer directly or whether they are derived via sessions with Sqills. The requirements which come directly from the customer, for example in a RFP, are often the most important requirements which could be a deal breaker for the implementation.

The types of requirements are visualized in figure 5.3. In this figure it can be seen that the top 10% of the requirements can be treated as the most important requirements which must be met in order to make the implementation successful. These are the most important requirements by the customer, but they can be supplemented by Sqills with features of which they know are important or even dealbreakers for many customers.

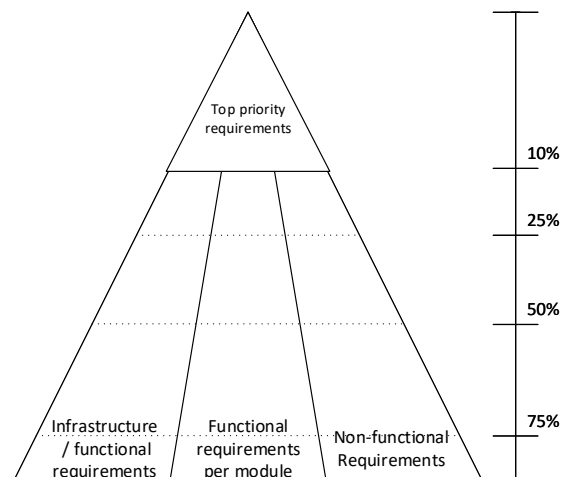


Figure 5.3: Elements to work on in a requirements process

5.2.1 Customer requirements

The customer requirements is a collection of the requirements which are stated by the customer before the implementation or even before the **RFP** has been sent. **RFP: Request for proposal** These requirements can be extensive in which case they need to be prioritized, if needed together with the business analysts. In the case that they have few requirements, it is almost certain that these requirements are the most important requirements which decide whether or not to buy the system.

5.2.2 Functional requirements per module

The functional requirements are the biggest group of requirements. In this group the main functionality of the system must be recorded. When the customer has stated requirements, many of these will be functional. The fit/gap analysis is important for this group because a gap could mean that the customer is not able to use the software the way they need. For example in the S3 software, a customer may need to be able to distinct female from male (overnight) train carriages while that is not possible yet in the software. This could mean that the software is not usable and for that reason the requirement must be known in time.

Because there are so many functional requirements, another distinction must be made within functional requirements. For that reason it is split up per module. Another aspect which is closely related to this group is data requirements. Per module can be sought for the data requirements and the needed functionality related

to these requirements. Within the modules, guidelines must be created to ensure that all needed requirements are elicited. These criteria are not yet visible in this model because of the specific nature of the system it needs to be applied to.

5.2.3 Non-functional requirements

Non functional requirements are requirements which don't describe functionality of the system, but aspects like performance, usability and maintenance. Aspects such as maximal down-time (reliability) and response time are covered in the non-functional requirements. **SLA**: service-level agreement. Sqills has a standard **SLA** which contains the standard agreements with current customers and the goal is to have the same SLA for every customer.

In the elicitation and negotiation phases of the non-functional requirements, the focus will be on having the customer to accept the standard SLA and the analysis/modeling phase will mainly focus on risk analysis for modifications of the SLA.

5.2.4 Infrastructure requirements

The infrastructure requirements consist of multiple types of requirements. It is a type of domain analysis which shows to which other systems the software must be linked and how and where the software is installed and used. It is closely related to non-functional requirements with regards to availability and speed of the software. Besides that, also the requirements of interfaces (data definitions) with other systems have to be stated.

5.3 Using the model

The current model is created based on known information from literature and common sense. That means that it has not yet been discussed with employees and the usability is yet unknown. However, there are some ideas on how to use the model, these are described in this section.

5.3.1 Placement in the implementation process

A standard implementation process starts with sales. In this phase the sales employees look to see whether the system fits the high-level needs of the potential customer. After this phase, an exploration phase starts, for which a contract has been signed between the vendor and the customer. From this point on the model can be used.

The model covers the exploration phase in which is researched whether the system fits the needs of the customer and how all important requirements can be implemented/configured. After the exploration phase, a design of the system is ready and an agreement is made for the customer to definitively start using the product. The model will still be used after this phase because more requirements may be elicited and the cycle can still be used for that. However the negotiation phase may be different because it is more explicitly focused on finding solutions and aborting the project is unlikely at this point.

5.3.2 Users of the model

In the first phase of the implementation process, the salespersons discuss the possibilities of the system with the potential customers. When that phase has been finished and the exploration phase starts, the business consultants start working along. Together with the sales person, they are the main users of the model. Together with the customers they work on prioritizing requirements. They perform the fit/gap analysis for every requirement and model how it can be implemented in the system.

With some implementations, Sqills uses partner companies to assist in the implementation and the requirements process. In this case they will also have to use the model. In practice this means that the partner companies would have to work with the model in the same way as the regular employees do. It may depend on the partner company how willing they are to work in this way, but it would bring advantages if everybody working on the project addresses issues in the same way.

5.3.3 Practical use of the model

The requirements are often given in advance by the customer. The pyramid is used to derive whether all requirements are elicited by using the categories. These categories need to be filled in by the users of the model and are based on experience with the system itself, so it cannot be in the standard model.

Every requirement that is set by the customer is added to the tasklist of the business analysts. In this list, the requirement will have a priority and it will be possible to write down the fit/gap and how it will be resolved. When the requirements have been analyzed and modeled, this is noted down with the requirement.

During the negotiation phase in the cycle, this list can be used to explain the solutions to the customer and to discuss the validity of the solutions.

5.4 Summary

Figure 5.4 shows how the Spiral from figure 5.2 and the elements from figure 5.3 will be combined. The top 10% of the requirements are the first elements which need to be considered in the spiral. In the figure it can be seen that the requirements which fit into these 10% are the darkest color, just as the inner circle of the spiral. The requirements which fit into the later stages/lighter colors in the pyramid of requirements will be treated in a later cycle of the spiral.

The business analysts and salespersons who use the model will have to fill in the exact contents of the categories in the pyramid. When they have placed all requirements in their tasklist/system, they can describe their analysis and model there. This ensures that there is a good overview on the customer's requirements for the negotiation phase.

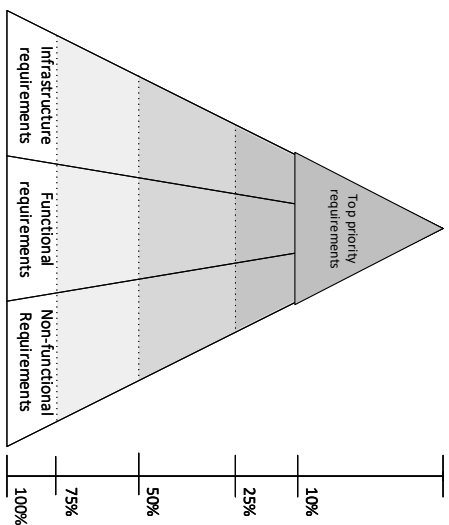
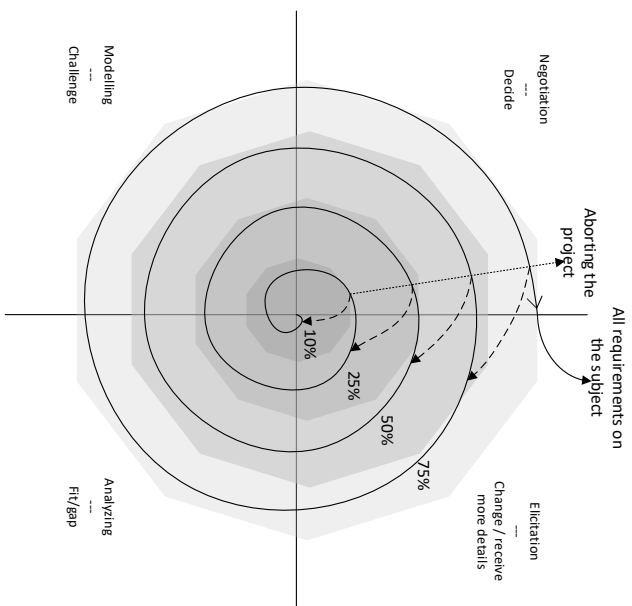


Figure 5.4: Mapping of pyramid on spiral with coloring

6

Validation with employees

The framework as described in chapter 5 will firstly be validated by discussing it with employees/experts who work at Sqills and are involved in the S3 Passenger implementations. There will be an individual meeting with every employee during which the framework will be described and their opinion is asked. the goal is to find out whether the framework is applicable to Sqills and what adjustments may need to be made to make it better suitable.

6.1 Validation approach

In every meeting the researcher will firstly explain the goal of the research and the way the framework has come to be. The employee will be asked whether he agrees with the meeting being recorded for the ease of the researcher. All employees agreed with the recording of the meeting. The employees are encouraged to ask as many questions as they want to and to doubt every aspect of the model to help the researcher improve the model.

During the meeting firstly the model will be explained to the employee and it will be made sure that the employee understands the various elements. During the explanation, the employee can give remarks or ask questions. After the explanation, questions about the completeness and usability of the model are asked where is referred to previous projects of the employee. If there is enough time, the remarks given during previous meetings will be discussed with the employee.

The interviews will all take place in Dutch as this is the main language of the employees. A list with detailed questions in Dutch and English that can be asked are shown in appendix A.

After each meeting, the recording is transcribed and a report is created in the language of the meeting. This report is then sent to the interviewee and he or she will have the chance to provide feedback when they think it is useful. Every report will be analyzed directly after the employee has had a chance to look at it. This way the advise from an employee can be used in the next meeting.

6.2 Validations

Four employees with different backgrounds in the company have been asked to participate. For every employee a short description of their background and experience is given with a summary of their opinion. Then a more extensive version of the meeting is given.

6.2.1 Employee 1 - senior business consultant

The first interviewee is a business consultant with many years of experience with the implementations of S3 Passenger. In general the interviewee likes the model and sees that it can be applied to many different requirements processes. The most important improvement would be to make it more easily applicable by adding explanations, tips and tricks to the model.

Often customers already have an extensive list of requirements. So the elicitation needs to be focused more on getting additional information than on retrieving basic requirements. For this way it may be more useful to have the division of requirements on low-level process flows instead on the current division on system modules because the customer is more likely to understand these business flows and give more insight in the requirement this way.

The rest of the spiral is applied most of the time. However, with partner companies, some part are done mostly by them or together with the partner. The goal is to have the partner company do as much as possible. The main aspects the partner company should be able to do is the elicitation and the fit/gap analysis. For the modeling part the partner should be able to do standard configuration. Sqills will need to do specific implementations or new feature implementations. The negotiation part should be done by the partner and Sqills, depending on the outcomes of the analysis and modeling parts.

The sales phase cannot be included in the model at the moment. Sqills and the product will need to be more mature in order for that to be possible. At the moment Sqills is focused on selling the product and in order to do that, they tend to say that certain features are in the product (out-of-the-box) whilst they are not. This means that later on the feature needs to be added or it has to be noticed by the business consultant that this needs to be worked on. The interviewee also states that the product needs to mature because sometimes features are not yet part of the product whilst that seems like a feature that should already have been in the product.

Main additions: 1) division of requirements on low-level process flows, 2) Elaborate on possibilities for partner companies and 3) Maturity organizational (sales) and product.

6.2.2 Employee 2 - trainee business consultant

The second interviewee is a trainee business consultant who works at sqills for 6 months. He has worked on few projects and has a vision on various aspects of the requirements process. In general the interviewee likes the model, there are however some aspects he does not fully agree with or thinks should be explained more clearly.

The interviewee believes that the negotiation phase should be a combination of evaluation and negotiation. When Sqills has modeled a requirement, they evaluate it by explaining it to the customer. When the customer understands the modeled solution he can describe whether it is in line with what he meant with the requirement and if not so, the next step is negotiation. Where multiple solutions may be discussed and if no decision can be made, the consultant will need more time to analyze and model the requirement again.

The problem that arises is that the customer (or the involved partner company) may not fully understand the modeled solution. So it is important for all parties to

really understand each other. Especially in this phase of the process. In an ideal situation, all information would be clearly known in advance, however quite often a requirement is reviewed and revised multiple times. Spending more time on the elicitation phases and understanding all parties would help this situation. The many changes to requirements is also a reason not to be bound to requirements lists and work more agile.

Prioritizing the requirements can be a good idea because it will show whether the system will suit the customer, however it can also have negative effects. For example when the top 10 requirements are barely met by the system, the customer may hold back whilst all other requirements are covered. Commercially that would not be a good thing. However it could be commercially be positive when we have an out-of-the-box solution for the top 10 requirements and not much of the other requirements.

At this moment not much is done with the priorities. It could be handy to do so in order to have a good idea on the likeliness for success. What happens now is that the requirements are grouped per theme and checked per group. Because many requirements are linked to each other, more information on the requirements may be in the related requirements. This information would be missed when the top prioritized requirements are checked by themselves.

The division of requirements in themes is good for analyzing, but also for because every theme will likely have a specific function and with that a specific contact person at the customer who understands the subject and knows whether the modeled solutions are correct and sufficient. For Sqills it will be handy to have the requirements per module when implementing the system, maybe there is a point in time when the requirements have to be reshuffled for this.

At this time the sales phase does not fit in with the spiral model because Sqills is still too much focused on selling and a requirement may be marked as possible a bit too soon. So the analysts have to check all requirements again to be sure that they have a solution in the system.

Looking at the high-level requirements firstly and then the low-level requirements is not a good idea according to the interviewee because a small aspect of a requirement can have a lot of influence on the question whether the system is in line with the requirements.

Main additions: 1) elaborating negotiation/evaluation phase, 2) grouping per theme and 3) importance of details.

6.2.3 Employee 3 - information analyst

The fourth interviewee is an information analyst who works closely to the developers for S3 implementations. His tasks are mainly to translate the requirements to tasks for the developers and for that reason is not as involved with the model as the previous employees.

In his opinion his tasks should not be seen as separate steps, but as one general procedure. For him it is also not needed to retrieve requirements, but only sometimes to elaborate on them and get more information. Negotiation is a wrong term in his opinion because it will scare the customer. Although it may be the correct term, it should not be used towards the customers.

Prioritizing the requirements seems like an impossible task for customers because they want all requirements in the system and often they find them all important.

It may be possible to state that the requirements will be released in three separate releases. The customer can probably state which requirement has to be in the first release and which requirements can wait until a second release perhaps a month later. It is important to be honest to the customer so that he does not think that the low priority requirements won't be implemented at all.

The interviewee believes that the most important aspect when having contact with the customer is the type of contracts and the culture of the customer and partner companies. For example a contract or partner company may prohibit that a requirement can be changed, even when both Sqills and the customer want to change it. As it happens often that requirements change, especially with long-term projects, this may seem like something that should easily be done, but it is not always the case.

The power of Sqills is its flexibility to adjust to the customer. Sqills may need to mature a bit now that there are more customers at the same time and building new functionalities depends on more customers at the same time. However, the flexibility is important and should be preserved.

Main addition is: 1) the influence of the contracts and customer's culture and 2) customers can not easily prioritize the requirements.

6.3 Improvements

The employees mentioned a lot of improvements. In this section these improvements will be summarized.

6.3.1 Mentioned by the interviews

The following additions are mentioned by the interviewees:

1. Division of requirements should be on low-level process flows.
2. The role of partner companies should be clearer.
3. The maturity of the organization (mainly with regards to sales).
4. Elaboration of the negotiation/evaluation phase.
5. Grouping of the requirements per theme instead of module.
6. Working from high-level to low-level won't work because details can have a lot of influence on the result.
7. The culture of the customer and the contracts/agreements are important for the complete process.
8. Customers can not easily prioritize the requirements.

6.3.2 Mentioned by regular conversations

The researcher has also spoken with employees outside of the interviews, so these conversations are not recorded and written down entirely. However, they did add information and opinions which should be taken into account. The employees who offered these opinions are an operational director and a implementation teamlead. Their additions are the following:

9. Perhaps a third dimension could be added to the spiral, namely to work from high-level to low level.

10. Categorizing the requirements per module does not work for consultants and customers.
11. The percentages are difficult to maintain, better would be to use MoSCoW.
12. It should always be possible to go to elicitation, not only from negotiation.
13. Negotiation has a too negative sound to it.
14. Non-functional requirements are few and actually standard.
15. The stakeholders are not mentioned in the model.
16. There should be a better guideline to describe how the analyst should work in practice.
17. It is possible for a analyst to work through the spiral many more times than just one round per group of requirements.
18. It is not clear when you can go to the next phase.

As can be seen, some of the items are mentioned in both the lists (e.g. 5 and 10 or 8 and 11) but also there seems to be an opposite of requirements, e.g. 6 and 9.

The researcher has looked into the best way the comments can be fitted into a new model and quite some changes are made. To ensure that the model will not be a combination of many small elements, the new model will be described as a whole in the next chapter.

6.4 Model

The improvements given by the employees, are transformed to changes which lead to second version of the model. This model is not described in as much detail as the original version, but the most important changes per section are described here.

6.4.1 Taxonomy

The division of requirements over three groups (non-functional, functional and architecture) has been adjusted to groups based on themes. For that reason they will be called **themes** from now onward. These themes can be based on business processes, function groups or other themes on which the consultant and the customer agree. the prioritizing of the requirements will also take place, but it will not be noted in percentages as this seems to be too difficult. It will be prioritized using MoSCoW or by letting the customer decide what they think needs to be in the first release and what can wait until later releases. The taxonomy is visualized in figure 6.1.

6.4.2 Processing requirements

The validations show that the word negotiation in the last phase has a negative sound to it which is not widely appreciated. The evaluations also show that the last phase seems to be the most important because it ensures that the customer and the vendor understand each other. There is often a lot of miscommunication on this aspect, and for that reason, the past phase is elaborated.

The whole phase is now called evaluation, which stands for evaluation of this round through the spiral. The subphases are 1) explanation, where the modeled solutions are described, 2) discussion, where the vendor and the customer can discuss

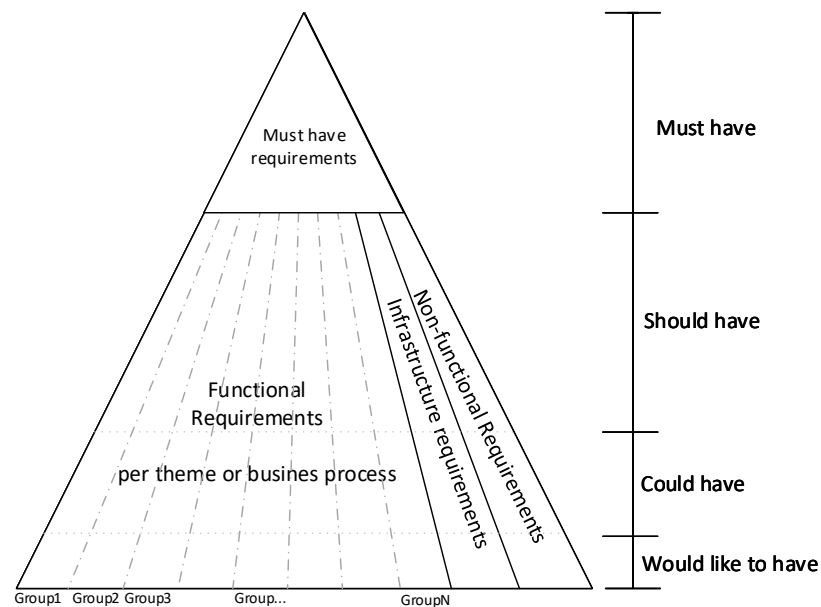


Figure 6.1: taxonomy of requirements

the modeled solution and whether they are applicable and 3) confirmation, where the vendor and the customer have to agree on the path to follow next.

In the first version of the model, it was described that the first elicitation also contained the initial retrieving of requirements, the dividing in groups and the prioritizing. This is now an activity that takes place before the spiral starts. The elicitation phase consists of getting to know the requirements better and if needed, getting more information on them from the customer.

The analyzing and modeling phases have not changed.

6.4.3 Full requirements cycle

There is one big difference with the first version of the model. Where in the first model, all requirements were supposed to be processed in the same spiral and in the last phase the decision was made which requirements to work on in the next round. In the new version, every theme (group of requirements, defined in taxonomy) has its own spiral and within that spiral, all requirements from the theme will be processed. In the first round, the requirements should be processed high-level to give an indication of the likeliness of a good implementation. This should be done for all themes simultaneously.

In the succeeding rounds, all requirements from the theme have to be analyzed in more detail to obtain a full requirements document after the last round. A practical explanation of the use of the model in this way is given in appendix C

To illustrate that every theme has its own spiral, figure 6.3 is given.

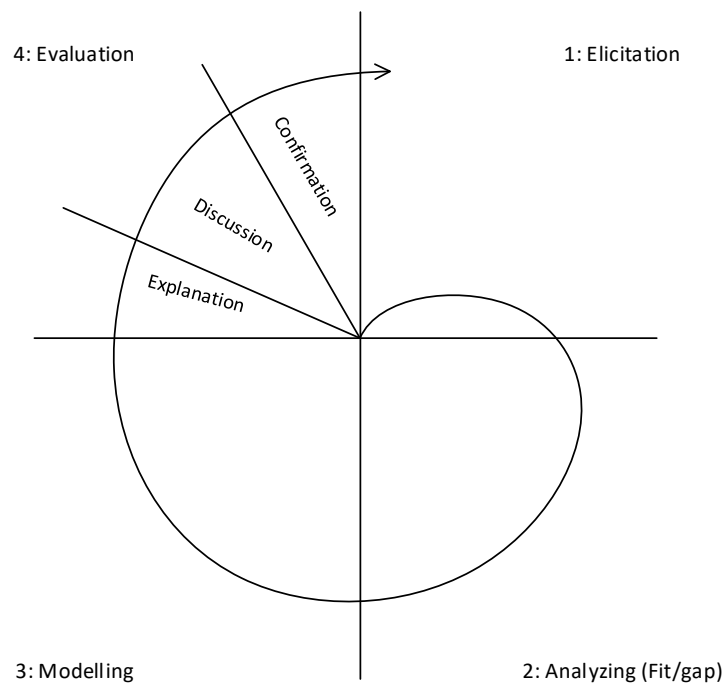


Figure 6.2: Basic requirements process

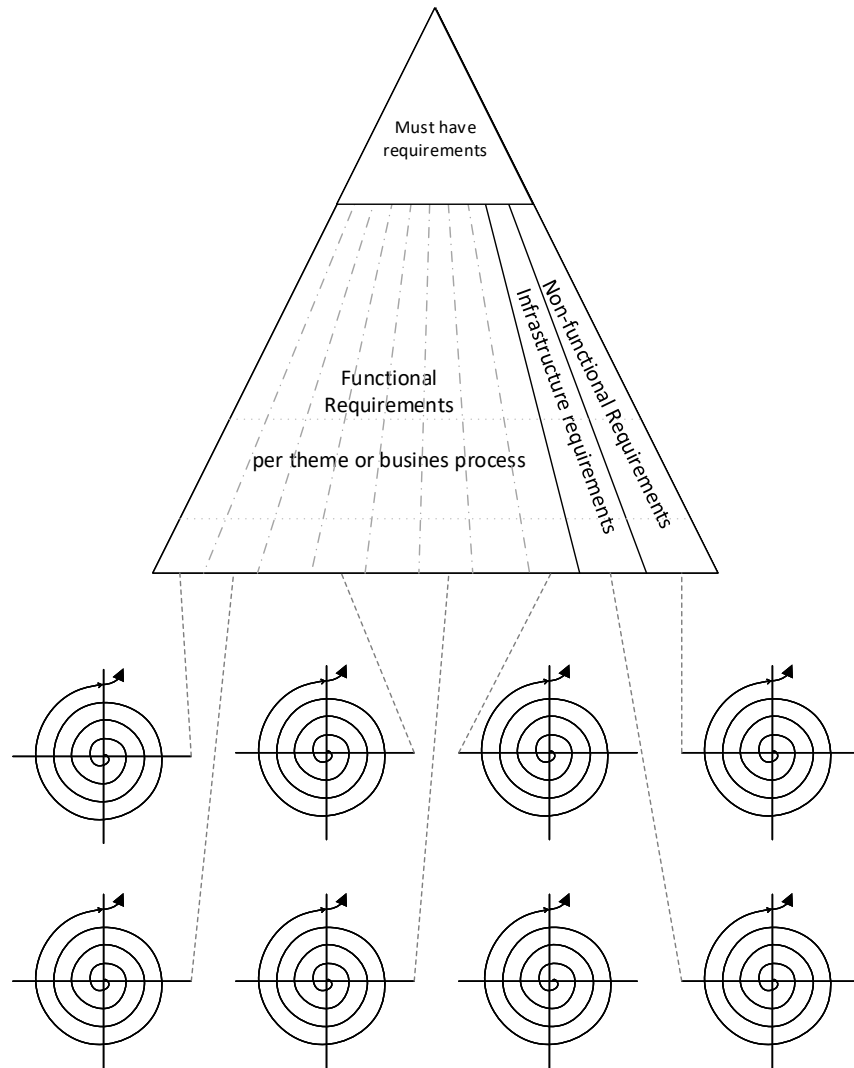


Figure 6.3: A spiral for every requirements group.



Validation of second version

In this chapter the validation of the second version of the model is described. first an outline of the setting of the validation is given. Then the results of this validation are shown.

7.1 Validation approach

This validation is approached in a similar way as the validation of the first model. Four employees are asked to help with the validation. Using a presentation, the model will be shown to them and they are asked to provide feedback. The presentation also includes an example of an application of the model, which is shown in appendix C. This is used as a clarification for the interviewee.

The interviewees are asked: 1) whether they can relate the model to the current way of working, 2) to see whether aspects could improve the current way of working and 2) whether aspects of the model would be impossible to work with. After the evaluation of the model, the interviewee is asked whether he or she would like to have a handout for the model. If requested, an example handout will be shown. (see appendix B The correlated questions with this subject are: 1) What they would see as the goal for such a handout, 2) in which situation it would be used and by who and 3) what should be on such a handout.

7.2 Validations

Four employees have been interviewed. Of whom two have seen the first version of the model an two have not yet seen the model in any form. All employees have regular contact with customers, although their experience within Sqills ranges from more than 8 years to half a year.

7.2.1 Interview 1 - trainee business consultant

The first interviewee is a trainee business consultancy and was also interviewed as a validation for the first version of the model. Because of that, the model could be compared to the first version by the interviewee.

The interviewee mentions that some phases do overlap. For example there are projects where the fit/gap analysis overlaps with explanation and elicitation because more information is needed for the fit/gap and this information can only be gathered by explaining to the customer why more information is needed. Also it may be possible to do a quick fit/gap analysis by head during the elicitation. With

regards to the change of the negotiation phase to elicitation, the interviewee thinks the change is good, and mentions that negotiation is actually a part of discussion, so it is still there.

In the themes, it may be possible that they are treated too separately and it is important that they are looked at all together because a requirement from one theme may add information to another theme. Especially in later rounds of the spiral it may become more difficult to understand the work of the other business consultants.

A handout for the basic model does not seem interesting because the model is pretty basic and easy to understand. A handout could be useful when going to a customer to explain the situation and the role and tasks that they and the consultants have. This might also help steering the customer into having a good (diverse?) project team.

A handout for the business consultant would be useful when the consultant has had a presentation or otherwise an explanation of the model. In that situation the handout could offer additional information about techniques and possible tools that can help during their work.

7.2.2 Interview 2 - project manager

The second interviewee is a project manager who has a lot of experience in implementing software products and started working as a project manager for Sqills not too long ago. The interviewee has also seen the first version of the model, and thus was able to compare with the first version.

One of the aspects the interviewee did explicitly not like, was the fact that negotiation was removed and replaced by evaluation. The interviewee believes that the model as it is now, is based on Sqills as a company as it works at the moment and believes that the company is too nice to their customers. The company should be more firm and strict in stating what their product is and that they should not want to change the product for a customer. In this way perhaps the word clarification would have been better because it implies explanation and does leave less room for input and requests from the customer.

In the model, the interviewee states that it is strange that only one of the four phases is elaborated while for example discussion is a vital part of most of the analyzing and modeling phase too. The interviewee also states that there are many more must haves than is shown in this figure and that would like to have should have been won't have. The taxonomy also implies that the must haves are a separate group from the themes while it should be a part of the themes.

One of the aspects that the interviewee misses in the document is time, because the amount of time that is given for the project, determines to what extent the model can be used.

The interviewee does not like the idea of a hand out for the model. The model is mostly on the pre-sales side of the process. Since the model implies our way of working and could be shown to the customer in the orienting phase, it could be combined into the presentation that is standard given in this process.

The business consultant will most likely not use a hand out because if it has to add more value, it should be more than a handout, namely a whole document including techniques.

7.2.3 Interview 3 - business consultant

The third interviewee is a business consultant who works at Sqills on the S3 software. This interview is the first encounter with the model and the interviewee believes that it roughly represents the current way of working.

The four phases are similar to the current way of working, however the interview mentions that they often go back to elicitation sooner because more information can be needed during analyzing and modeling. In that case the evaluation and modeling phase are not always worked on and it can be considered a step back to elicitation during analyzing.

Prioritizing the requirements is useful. At this moment customers often use MoSCoW. The Sqills consultant does have to work through the priorities also because what the customer sees as a must have, is not always a requirement that the consultant should be working on firstly. Besides, the consultant checks whether the must haves are really necessary by looking at the current situation and asking about the effects of having this requirement implemented.

The themes are useful and there are many standard themes which could be defined. In the last project, the interviewee ran into a problem that was not accounted for, but after asking around, it appeared that it was a problem in earlier implementations too. With an overview, this theme could be added so that it is not forgotten in later implementations.

On a handout, the interviewee would like that aspects, such as the theme from above, would be shown. A handout with only the general model would not be useful enough, but an addition of these themes or techniques at the analysis would make it useful. It would also be handy to use it to explain to others how we work, and how the division of tasks is. However, it is important to be careful not to tell others exactly what to do. For example with implementation partners, they already know how to do evaluations and would not need or use our hints on that subject.

7.2.4 Interview 4 - project leader

The fourth interviewee is a project leader who works at Sqills for more than 8 years in various functions, mainly as information analyst and as project leader. The interviewee thinks the model in general is applicable, but has constraints and remarks about the way the model can be applied.

For example on the evaluation phase, the interview stresses that you should not give the customer too much reason to believe that they can get everything that they want. In the interviewee's opinion it would be better to just use explain and don't go in a discussion with the customer. A similar situation is at the prioritizing of requirements, where the model does not have a "won't have" section and this way the customer may believe that every requirement can be met.

The underlying reason for this opinion is that the product from Sqills is still often sold as a project instead of a product. The interviewee believes that the goal is that the product is sold on a turn-key base and the customer does not have any influence on customization of the product. The only thing the customer is allowed to do, is the implemented configuration which is explained in a manual. However, the product is still growing and the company still has a customer-intimacy oriented strategy, although they want to reach an operational excellence strategy. Sqills is

still growing and may get to a point where the product is 'complete' and can be delivered without customization. However, it is doubtful whether it is possible to reach such a point because the market is not as big as most other turn-key software products.

The second aspect which the interviewee mentions as an important element of the model is the combination of circumstances for the specific implementation process. An example is the contract that the customer has with the vendor. Elements of the contract that have big influence are the possible involvement of implementation partners and whether the requirements are fixed in the contract or that they are flexible.

Time restrictions can also have much influence on the implementation process because there is more or less time to get to an agreement on the requirements before the contract is signed or before a start needs to be made on the actual software implementation and perhaps development.

Another aspect is the culture and way of working at the customer. Their culture and internal politics can decide who gets to speak with the vendor and whether these people have knowledge of the processes can easily differ. Also the consultants of Sqills may have different approaches which will make that the model will be used differently. For example by easier giving in to customers wishes or to keep strictly to the product roadmap.

The interviewee does not believe that a handout is useful for the model. If a handout were made, this could not contain all information needed by business consultants. It would be better to have a full document about the model and its application at Sqills. This could contain the various themes that can be used and how they should be used. It can also contain various techniques all four phases and the circumstances where they are useful. The last important aspect mentioned by the interviewees is to state clearly in the document for which circumstances (as mentioned above) the model is useful and how it could be used in various circumstances.

7.3 Improvements

On some aspects, the interviewees have similar opinions, on other aspects they are not in line at all. Firstly the remarks will be summarized and shown below.

1. The phases are too restricted, sometimes there is overlap between phases.
2. It appears a strangely that only the evaluation phase is elaborated on.
3. The word negotiation has a negative sound to it (too strongly).
4. The word negotiation gives too many options to the customer.
5. Prioritizing should be done together by the consultant and the customer.
6. Themes are a good idea.
7. Culture and time are not included in the model.
8. A handout of two pages cannot give enough information .
9. A handout can be useful to show customers how the company works.
10. A handout has to truly provide additional information for it to be used.

There are some remarks that are not in line with each other (e.g. remarks 3 and 4) and many remarks are on similar themes. In the next sections, firstly the remarks which are based on the model are described and this is succeeded by the remarks and possibilities for the handout.

7.3.1 Model

All of the interviewees mention that the phases (elicitation, analyzing, modeling, evaluating) seem correct, but there are some remarks. The first is that it is often needed to go back to elicitation during analyzing, so they can not exactly be seen as separate phases. The same goes for analyzing and modeling, which are overlapping phases. The model shows all these phases separately, but in practice, it should be possible to go back and forth more and combine activities.

Another remark is that only the fourth phase (evaluation) is elaborated while there are also subphases in the other phases. At this moment it seems like this phase is emphasized and the rest may be less important, which was not the goal of the researcher.

Some discussion took place about the word negotiation (as is was in the first version) versus evaluation (as is is in the second version). This change and the discussion shows that the problem here is deeper than just in the model. It shows that the company does not know how firm they want to be to their customers and how willing they are to implement unique requirements. Whilst some see evaluating as a moment to discuss the possibilities with the customer, others are more strict and think that even negotiation implies that Sqills is giving in to the customer's requests to easily.

All interviewees agree that prioritizing is important, however, they also state that the consultant should have more say in this. The consultant can help guiding the customer by showing what actually can not be done (won't, but would like to have) and the consultant knows better what requirements should have priority because they are not enough in line with the current software.

The themes are seen as a good idea because they can be filled in practical and for every customer the themes that are needed can be chosen to work with. There is some discussion about the themes with non-functional and architectural requirements. Some employees think these are good separate themes whilst others think they should be incorporated into the rest of the themes. Luckily the themes can be decided on per implementation and the consultant can use the themes that he/she likes.

Two elements that were not taken into account during the creation of the model which are mentioned by some of the interviewees are culture and time. Culture can have lots of influence on the techniques that can be used, especially at the elicitation and evaluation phase. Time is also important because with this model, it is possible to be working on the spirals for an enormous amount of time. If an implementation has a timelimit, it can be stated that only 5 rounds through the spiral are possible, or an actual timelimit, for example two months can be decided on. This also influences the schedule of the consultant because they need to make sure that all requirements (from all themes) are worked on within that time limit.

7.3.2 Handout

During the interviews, the interviewees have been asked how they would feel about a hand out. In what situation they would like to see it used and what information it should contain in that case.

In general, they stated three situations in which information about the model

should be used. These are: 1) to explain the way of working to customers, 2) to explain the model and way of working to new business consultants and 3) to get information from on how to perform their tasks. These options are discussed below.

Customer:

the interviews state that it could be interesting to show the model to customer to make sure they know and understand how we work. In this case they would need to get information about the division of tasks and what their and the consultant's role is during the project. This should be explained in the beginning of the requirements analysis phase.

Some interviewees agree that a handout can help in this situation. Other interviewees state that it would be better to incorporate it within a presentation about the whole project, of which the requirements analysis phase is a part. There is already a presentation given to customers with whom Sqills is looking into implementation possibilities, so this may be a good option.

New business consultant:

For a starting business consultant it is important to understand how Sqills works. So if the model will be used, it has to be explained. However, all interviewees agree that a handout with general information does not provide enough information about the way of working or is useless within a few days because the model is easily comprehensible.

The conclusion can be made that a simple handout with general information on the model is not something that would be used often.

More information on tasks:

When the handout is meant to be used by business consultants who have been working with the model for a longer time, it should contain more detailed information about the application of the model at Sqills. Two of the interviewees mentioned that this was not possible on a handout, but a whole document would be needed to make it useful. The two other interviewees did not mention their concerns about this.

The mentioned aspects that could be on the handout or document are the following:

- Themes (For example: aftersales, booking flow, agent flow, revenue management, seat allocation, tickets, architecture, disabilities and non-functional)
- Elicitation techniques
- Discussion techniques
- Effects of circumstances

8

Final version of model

The second version of the model was validated and some remarks for improvement were given as described in the previous chapter. With these improvements, the final version is created. In this chapter this final version of the model is described and some considerations for usage are described, as well as the researched possibilities for a handout.

8.1 Requirements taxonomy

Often the customer already has a list of requirements. In other situations these requirements have to be elicited together with the customer. The parts below describe how a taxonomy can help eliciting and organizing the requirements.

8.1.1 Taxonomy

In most situations, the customer already has requirements. The issue that needs to be tackled here is how to orderly process these. The evaluations have showed that it is useful to process the requirements per **theme** because many requirements are related to each other and information from one may be needed to correctly process the others. Depending on the situation it may also be useful to have a separate group of requirements for the IT landscape and the non-functional requirements. It needs to be mentioned that these groups can be different per customer and especially the last two are optional. The requirements may in some cases be placed with the regular themes rather than in a separate group.

A **theme** is defined as a group of requirements which are on the same subject or business process.

The taxonomy can also be used for eliciting requirements from the beginning when the customer does not have requirements yet. It would be useful to have a standard list of themes. The business consultant can look at these themes (together with the customer) and decide which are needed in the specific implementation. The themes will then give guidance in retrieving all requirements. Examples of themes are: aftersales, booking flow, agent flow, revenue management, seat allocation, tickets, architecture, disabilities and non-functional.

8.1.2 Prioritizing

Prioritizing the requirements was an issue during the evaluation. Most of the interviewees mentioned that it is not easy for customers to prioritize their requirements and others even say it is impossible for customers to do so because they may feel that the requirements with a lower priority won't be implemented. For that reason the customers should not be asked what their top 10% requirements is, but they may be able to sort the requirements using the MoSCoW approach. (explained in

section 3.1.1). An option which is provided by the validation interviews is to let the customer prioritize the requirements by deciding which requirements have to be in the first release and which requirements could (if necessary) wait until a second or third release.

After letting the customer prioritize the requirements to their wishes, the business consultant should also look at the given priorities and add his/her opinion. The consultant has more knowledge about the software and may state which elements are an issue and for that reason need to be addressed first. Also, the consultant may state that some requirements are simply not feasible and should be in the last group of MoSCoW (Won't but would like to have).

8.1.3 Visualization

In figure 6.1 the requirements are visually divided over the themes. In the second version, the group of must-have requirements was also shown in the triangle. The validations showed that this is not necessary and the prioritizing is only shown aside from the pyramid.

There are many themes possible, as shown before. Because the division will be different for every customer, the specific themes are not shown.

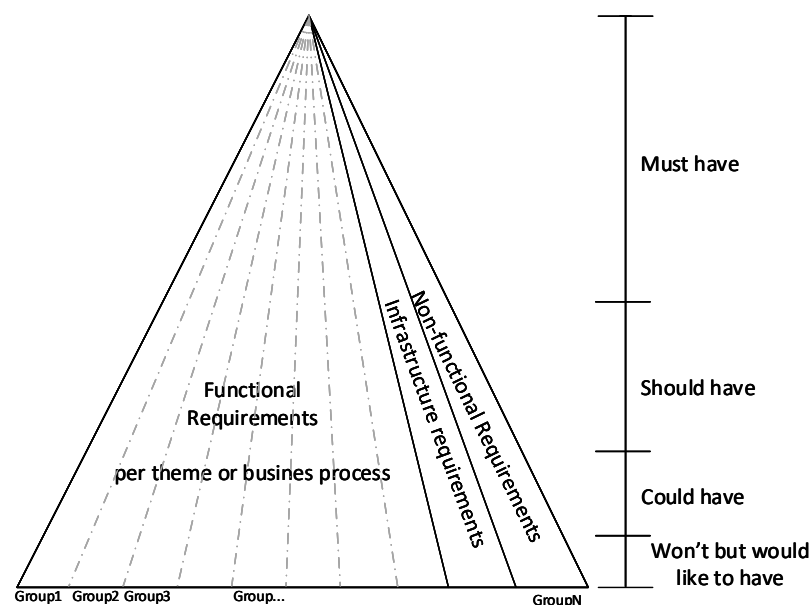


Figure 8.1: taxonomy of requirements

8.2 Requirements phases

There are four main phases when processing requirements

The general requirements process stays mainly the same, which means that there are four **phases**, as shown in figure 8.2:

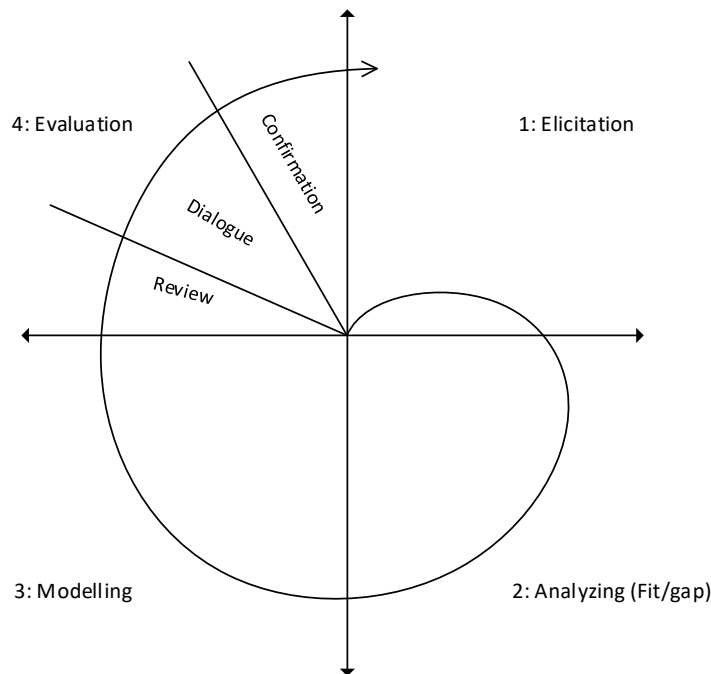


Figure 8.2: Basic requirements process

1. Elicitation: The phase in which information is retrieved from the customer. When the customer already had a list of requirements, this phase can be used to retrieve additional information. The two main actors in this phase are the customer and the business consultant from the vendor.
2. Analyzing: In the analyzing phase, the elicited information is analyzed and compared to the system that is offered. A fit/gap analysis will be performed. The main actors in this phase are the business consultant and an information analyst from the vendor. If a large gap is found, the product owner can also be involved in deciding whether functionalities need to be added to the system.
3. Modeling: The modeling phase takes the information from the analyzing phase and for every fit, it needs to be shown how this fit is realized in the system. For every gap, it is needed to show why it is not possible and what other possible solutions are. The most important actors are again the business consultant and the analyst. If needed, the product owner can be involved also.
4. Evaluating: In this last phase, the business consultant goes back to the customer to discuss the findings from the analyzing and modeling phases. The consultant needs to explain the modeled solutions and make sure that the customer understands what the results are. If needed, there is a discussion about the results. The contents of this discussion depends on the growth of

the product and organization and the main difference may be in how willing the organization is to give in to the customer's wishes. After this discussion, it is important that the customer and the consultant agree with each other on how to proceed with the project. For example they can decide to cancel the whole project or to continue another round through the four phases.

Only the last phase is elaborated further in figure 8.2 because the validations showed that this phase was deemed the most important one because it ensures that the vendor and the customer understand each other.

The phases as mentioned and described, have different contents. However, it is likely that for example in the analyzing phase, the consultant realizes they need more information than they have retrieved in elicitation. For this reason it should be clear that the order of the four phases is a guideline and it is possible to quickly ask the customer for extra information. So the phases are meant as separate phases, but some overlap is not prohibited, just as it is possible that some phases take more time than others. The exact contents and length of each phase will differ per implementation.

8.3 Full requirements process

The main phases as explained in the previous section need to be gone through multiple times if all requirements from a theme should be processed correctly. Reasons could be that the information was incomplete, or because the choice was made to focus on other requirements first.

A **round** is a combination of the four known phases in the spiral

For that reason the model is visually shown as a spiral which should be read from the inside out. Where every **round** has the four known phases and when ending the last round, all requirements are processed correctly.

8.3.1 Full spiral

Every theme has its own spiral in which the consultants go through all requirements. This is visually shown in figure 8.4 .

In the sections below, the most important aspects of the different rounds in the spiral are explained. It starts with pre-spiral, this shows what has to be done before a start with the spiral can be made.

Pre-spiral

Before the consultant can start using the spiral, the requirements must be known and it must be clear which requirements belong to the theme from this spiral. In this process the consultant has received the requirements from the customer or has elicited them with the customer. The requirements are divided over themes or are elicited in themes which are then appointed to a consultant. A glossary or agreement on terminology is also made during the pre-spiral period.

First round

During the first round of the spiral, the elicitation phase is meant to ensure that all current requirements of the group are known and can be processed. During the analysis phase the requirements will be checked with a quick scan. This way for every requirement is known whether it is directly in line with the system (out-of-the-box), whether it is impossible in the system (would certainly require customization)

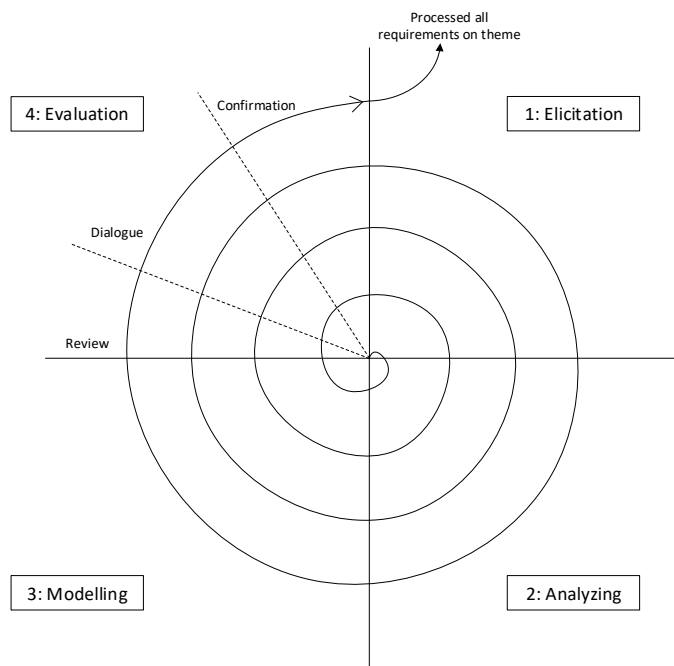


Figure 8.3: Full Spiral

or whether it may be possible with just some configuration. In the modeling phase the analyst will create an overview on the fit of the requirements to the system and may think of general solutions to the requirements which don't have a full fit on the system.

During the first evaluation phase, the fit/gap will be discussed and there should be a discussion on the requirements that don't have a fit on the system. When the gap is too big, the vendor and the customer will have to get to an agreement whether the customer is willing to adjust their requirements or whether the vendor is willing to customize aspects for the customer. In the end of this first round, a decision can be made whether to continue the collaboration or to stop if the gap is too big. When continuing, the consultant will take information from the discussion and use it to know what information is needed to analyze the requirements in more detail during the next round.

Successive rounds

During the rounds after the first one, the requirements are analyzed in more detail and more detailed options will be modeled. If the negotiation part shows that more information is needed to correctly model the requirement, this will happen in the elicitation part of the next round. It is possible that there are many rounds when the theme is about a complex business process, in that case often it will appear that more information is needed and the explanation may need a lot of time. In the end, every requirement should be understood and modeled into something that fits into the system and the customer accepts as a solution.

Last round

During the last round of the spiral, the last evaluation phase will show that the customer and the vendor agree on the requirements and the modeled solutions. At this point no more information seems to be missing. All the modeled solutions together form the requirements document on the requirements group that belong to the spiral.

However, requirements can always keep on changing. The customer may later realize that some of his wishes are not in the document or the vendor may realize that a certain implementation appears to be impossible. For that reason, the consultant must realize that there could always be another last round.

8.3.2 Combining spirals

Every group of requirements has their own spiral (see figure 8.4). However they are not fully individual processes. Often it is useful to discuss more groups of requirements at the same time with the customer. This means that the evaluation and perhaps the elicitation phases of some spirals will overlap.

The most important overlap is the first round in the spiral. As explained, in this round the requirements will be checked generally and broad solutions are given. This is meant as an exploratory phase and it has to be done for every requirement group at the same time. This way the discussion after the first round can be based on all requirements and an informed decision on whether to continue the project can be made.

Another aspect is that sometimes requirements will overlap and have to do with requirements from another requirements group. They may influence the same module configuration and for that reason should be considered at the same time. For that reason it is important that consultants know talk to each other when they work on the requirements at the same time and that the modeled ideas are shared between the requirements groups.

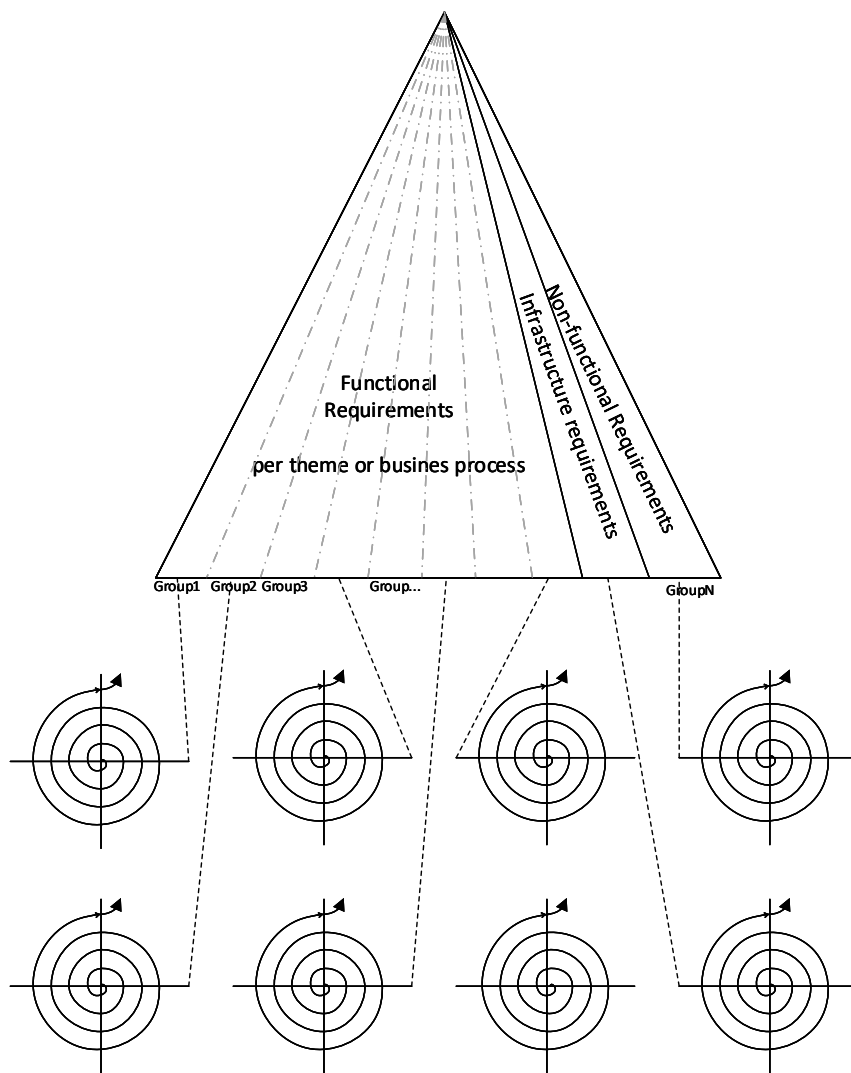


Figure 8.4: A spiral for every theme.

Application of the model

9.1 Expert validation

After completion of the model, some external experts were approached to discuss the model and see whether the model could be applied in their situation. These experts are employed at various companies who offer various software products or projects. In the meetings, the goal of the research and the model was described to them. Followed by a description of the model, using a presentation. During and after the presentation, the various elements were discussed.

9.1.1 First expert

The expert firstly focused on understanding the company for which the model has been made. this was done by getting a global view of the product, the market and the background of the company's organisation. Afterwards the model was explained to the expert and some subjects were spoken about, such as the correctness of the model and the applicability of the model on other companies.

In general the expert thinks the model is applicable to many situations, which is mainly caused by the fact that the model is generic. It would fit on many companies and different types of software production. Even on projects, the model works partly. The fit/gap analysis is less useful, but the themes and way of further processing the themes could work.

The first aspect of interest of the expert is how the themes are defined. It is a domain driven aspect and it may be different in every company. After the researcher explains that it may be defined using the company's experience to create a standard list. The expert agrees that this may work, but warns for the possibility that the consultant gets 'lost' in his experience and is less open for unique aspects of specific customers. The expert himself would work from scenarios and workflows and combine these towards themes.

The expert focuses strongly on the degree of orthogonality between the themes. His conclusion is that probably no theme will be fully separate from others. As every theme may be based on scenario's, there will be output from every theme that can be considered a hand-over to other themes. Often this hand-over is exactly what the customer sees. It would be good if that and other dependencies were shown in the model to ensure a good handling of the different themes.

The expert sees that the model is close to sales instead of the actual implementation. Also it is important for a company's employees to have a similar view on the the approach to customers and customization of the product. They should apply the method in the same way, which requires a protocol or other kind of document, explaining the application of the model for the company.

9.1.2 Second expert

The second expert gave insight in the application of the model in a SaaS and Agile oriented organization.

The current description of the model is working towards a full requirements document, which implies a good usage for waterfall projects. In that case the requirements document can be completed before the developers start with the next phase of the project.

In the company of the interviewee, a more agile approach is used, which he finds clearly represented in the model. The iterative way of working in the spiral shows how the customer is involved in the process and how a new subject can be addressed with every new iteration.

The company develops a SaaS solution. In their case the front-end is custom built for every customer and this is done by other companies or by the customer themselves. The company does assist in configuring the system. Most of the attention of the company goes to developing the back-end of the system and developing new features. The goal is not to work towards one customer implementation, but to work towards a cooperation with a customer and gradually receive income after finishing the initial implementation. Their challenge is to find the features that customers want to have implemented in the system afterwards. For the analysis of these features, the spiral model can be used because of the resemblance with agile working.

9.1.3 Third expert

The third expert works at an IT company who delivers an integration Platform as a service (iPaaS) to various companies. The interviewed expert works on developing the back-end of the system.

One of the things of the model he was wondering about, is why the taxonomy is pyramid shaped. While the given answer was that it was a smart shape in the first version and stuck along, he also stated that it is now a nice characteristic of the model.

The company works agile and the interviewee sees this reflected in the model by using the spiral shape. In his opinion the spiral may be a better way to represent the iterations because it gives more perspective to the process by showing that there is an endpoint to work towards. In their implementations the modeling is more than just looking how the implementation can be done, but it is directly implementing the solution in their modeling software. A minimal viable product is created in the first iteration and in the later iterations, this product is extended to the final product. So this model is not just used for the requirements, but also for the implementation of the product itself.

The last topic the interviewee proposed, is the applicability to other situations. For example on software projects instead of products. In which case most of the model is applicable, only the themes are harder to predefine, which is more important for products.

9.2 General considerations

The interviews with external experts and employees from Sqills have given insight in considerations for using the model. The subjects below will differ per implementation project, per company and perhaps even per consultant who works on the implementation.

9.2.1 Theme definition

How the themes are defined depends strongly on how the model is used. For example when the company works agile, the themes should be defined in such a way that they can be used for sprints. When the company works more by the waterfall methodology, this is not needed as much.

One remark on theme definition as given by an expert is that when themes are based on scenario's, it is likely that one theme has an output which is to be used as input for another theme. This stresses how important it is for consultants to communicate a lot and know when their decisions may influence another theme.

9.2.2 Agile working

The contents of a protocol also strongly depend on the way of working within the company. Especially the contents of the 'modeling' phase may differ between those companies. With Agile working this phase can be used to develop the system. With the waterfall method, this phase can be used to create the functional design of the system. The deliverables from this phase should be stated in a protocol.

9.2.3 Time

In some implementation projects, the time seems to be of the essence and the customer needs the software implemented as soon as possible. While in other implementation projects, the customer has lots of time and doesn't want to compromise anything for a speedy result. Most of the implementations will be somewhere in the middle.

The factor time can have a lot of influence on how the model is used. For every time constraint, a compromise has to be made. It is possible that the team does not spend time on prioritizing the requirements. In the end this can result in some must-haves that are not implemented or a lot of time that has been spent on requirements which are not as important for the customer. Another option is not to divide the requirements into themes, however, the consultants still need to divide the requirements into groups for themselves to work on, so it does not make sense to skip this step.

It is likely to spend less time per theme on the spiral. For example it can be agreed that the processing of the requirement may take a specific amount of weeks or rounds through the spiral. The downside of this agreement is that it is possible that mistakes are discovered in the last week or round and that they cannot be processed correctly any more because of time constraints.

Time constraints will be handled differently in every project and these need to be discussed between the customer and the consultant.

9.2.4 Culture

When using the model, it is important to keep in mind that it suits the culture of the company. For example when the company focuses on customer intimacy (section 2.4.3) the model should be implemented to reflect that contact with the customer is more important and more time needs to be spend on solutions for possible gaps.

Another aspect is the culture within the customer's organization. For example they may have the opinion that certain employees or departments should be involved in the process at another point in time than other companies for various reasons. In that case the evaluation phase has to be suited to the situation where other employees of the customer are involved or specific themes cannot be evaluated at the same time with the same employees.

A special case is when the customers are from another country. In that case the cultural aspects may be entirely different. By using the theory of Hofstede et al. (2010), an indication can be made on the culture of the customer and the approach can be changed for it.

9.2.5 Maturity

With product maturity is meant that a product can be relatively new and therefore less complete than a product that has been around for years. A product that is less mature will need more work and it is likely that customers have requests or requirements of which the vendor believes should be, but aren't yet in the product. With a more mature product the vendor will say that the product is complete as it is and will not be adjusted for one or a small group of customers.

The practical difference is that consultants can be more strict when the product is mature because all the most common requirements are already in the system. In the model, this means that the evaluation phase is approached differently. The modeled solution will be shown, but the solutions given for the gap will be less extensive and it is more likely that the customer has to adjust instead of the vendor. For less mature products, the evaluation phase is meant to get to an agreement between customer and vendor and the customer has more influence on the possible adjustments to the system.

A company has to grow along with the product and with a more mature product, the organization has to become more mature too in order to maintain a vision for the product.

9.2.6 Partner companies

In some implementations, vendors work together with implementation partners. When such a collaboration is started, it needs to be clear how the responsibilities are divided. This model gives a guideline for the general processes and can be used to create a task division. For example in figure 8.3 the two phases on the top of the spiral can be handled by the implementation partner, whilst the two lower phases can be handled internally at the vendor. When the product is mature and there is enough documentation on the system, it may be possible to let the implementation partner do most of the work. When there is a gap, is it useful to get the vendor involved in deciding how the gap can be covered, for example on the decision of placing functionalities on the product roadmap.

9.3 Application of the model at a company

As this model has been created specifically for Sqills, there are some recommendations for usage that can be made. Firstly the practical use of the model will be described as it is applicable to Sqills. Next the option for creating a hand-out will be described. This has been asked for in the employee validations and the conclusion on it will be given here. Finally some considerations will be given on the usage of the model and how the model can be used in different types of companies. This can also be understood as ways the model can still be used when the internal organization of Sqills changes.

9.3.1 Practical use of the model

In most cases, customers will send out a request for proposal and Sqills has to check the requirements they have given. When this is an extensive list, the requirements will first have to be placed in the taxonomy triangle to have a division over groups. This way the consultant can check the requirements on a high level using the groups and he can give an overview on the likeliness that the system suits the needs of the customer. After all requirements have gone through the inner circle of their own spiral, a decision needs to be made based on these first findings.

After this decision, the consultant can continue and check all requirements in more detail and model the requirements more specifically to the system. By having the groups, it is possible for the consultant to work through the requirements and have the needed context of the requirements. After a consultant is done with the group of requirements he can decide to discuss it with the customer, he can also decide to work through another group of requirements first and then talk about the results from both groups with the customer.

In the case that the customer does not send out a request for proposal, but contacts Sqills directly with questions about the system without having defined requirements, the situation is different. When the customer does not have predefined requirements, Sqills has to work together with them to find the requirements. To do this, the same model can be used, however some aspects will be different.

Together with the customer, the consultant from Sqills has to define all processes or themes which should be in the system. This is the starting point from which to elicit the main requirements. For the first round of the circle, the consultant has to spend much more time eliciting the most important and unique requirements. For every defined group, these requirements have to be elicited and analyzed/modelled. After that a decision can be made whether to continue with the product and elicit more detailed and specific requirements. From this stage on, the process is generally the same as in the first situation. However, much more attention must be paid to eliciting requirements and details of the customer's wishes.

In appendix C an example is shown on how the model can be used in practice.

9.3.2 Handout

One of the possible practical implications was the request for a handout of the model. The original ideas for this handout were to be a guideline for business consultants or to show to customers as an explanation of the method Sqills uses.

The validations show that there are three possible goals for such a handout.

Customer:

The interviews state that it could be interesting to show the model to customer to make sure they know and understand how we work. In this case they would need to get information about the division of tasks and what their and the consultant's role is during the project. This should be explained in the beginning of the requirements analysis phase.

Some interviewees agree that a handout can help in this situation. Other interviewees state that it would be better to incorporate it within a presentation about the whole project, of which the requirements analysis phase is a part. There is already a presentation given to customers with whom Sqills is looking into implementation possibilities, so this may be a good option.

New business consultant:

For a starting business consultant it is important to understand how Sqills works. So if the model will be used, it has to be explained. However, all interviewees agree that a handout with general information does not provide enough information about the way of working or is useless within a few days because the model is easily comprehensible.

The conclusion can be made that a simple handout with general information on the model is not something that would be used often.

More information on tasks:

When the handout is meant to be used by business consultants who have been working with the model for a longer time, it should contain more detailed information about the application of the model at Sqills. Two of the interviewees mentioned that this was not possible on a handout, but a whole document would be needed to make it useful. The two other interviewees did not mention their concerns about this.

The mentioned aspects that could be on the handout or document are the following:

- Themes (For example: aftersales, booking flow, agent flow, revenue management, seat allocation, tickets, architecture, disabilities and non-functional)
- Elicitation techniques
- Discussion techniques
- Effects of circumstances

Evaluation

In this part of the report the research will be validated. Firstly, in chapter 10 the answers to the research questions will be given and the goal of the research will be evaluated. In chapter 11, the research will be discussed on the subjects of abstraction, scientific and practical implications and ideas on future work will be given. The document will close with the references and the appendices.

10

Results

This chapter will give an overview on the results of this study, starting with the research questions and finishing with the original goal of the research.

10.1 Answers to research questions

Seven research questions were created to give guidance in achieving the research goal. All research questions will be answered in this section. Most of the research questions have an extensive answer in another chapter. In that case a reference will be made to the chapter that contains the answer to the research question. A summary of the answer will be given in this section.

1. How do software product vendors handle the early phases of the growth of their company and product and what are the possible pitfalls in this period?

This research question has been answered using a case study at six companies who all have a software product. From this case study can be concluded that multiple aspects have an influence on these companies. For example the way the first development of the product has went, the way the organization is structured, the properties of the product and the way customer relations are handled. The companies have faced similar challenges, for example the matter of legacy software and not knowing what the best way is to handle version management. The case study shows that having a vision for the company and the product is helpful. The case study has also given a theory on the applicability of the Marketing value disciplines of Treacy & Wiersema on software product companies.

A more extensive answer can be found in section 2.4

2. What is the state-of-the-art in literature regarding the implementation process of software products and specifically regarding the associated requirements engineering process?

Much information can be found on software implementation processes. However, it is difficult to find information that focuses on the role of the vendor. A lot of research has been done on the role of the customer and how they can get the best out of the software. In chapter 2 information is given on types of software products and the phases that are regular during an implementation process. There is much information on the critical success factors for an implementation and information is given on software customization. This includes different types of customization, the possible effects of customization and why it occurs.

Chapter 3 gives information on requirements engineering. A lot of this information is later used when creating the model in later chapters. The phases that every requirement process goes through are described with the information from various researchers. Also an overview is given on types of requirements that can occur and how requirements can change during a project. The application of the topics on software products is shown in section 3.3

There is a lot of information to be found on the subjects and while here only the topics are mentioned, a summary on the useful state-of-the-art has been given in chapters chapter 2 and 3.

3. How is the requirements engineering process of S3 Passenger software at Sqills at this moment?

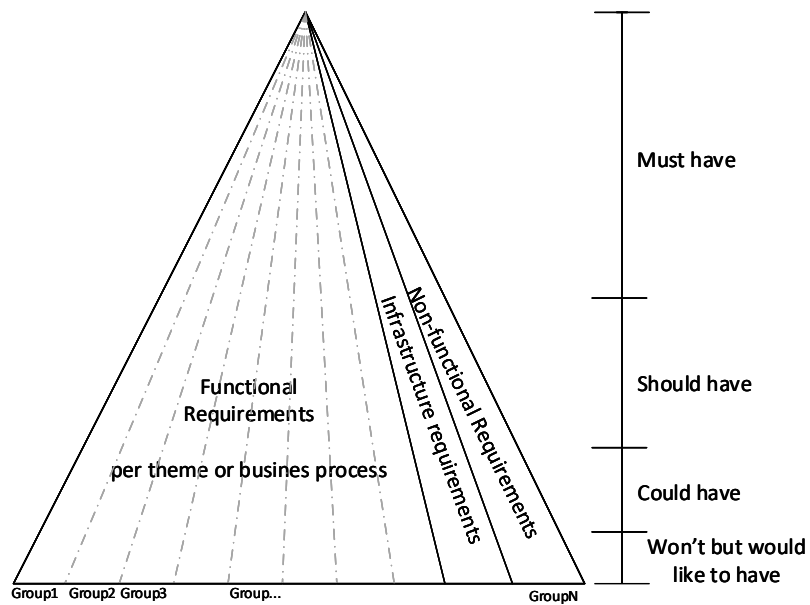
Less than 10 full implementations of the S3 Passenger software have taken place yet. In some of these implementations the requirements were handed by the customer or the implementation partner. In other situations the requirements were retrieved together with the customer. Every requirement was checked for the possibilities in the system. For some of the implementations a lot of work was done on expanding the core of the system or custom functionality was created. In other situations the customer only needed a part of the system implemented. More information on the system and the variety of implementations can be found in chapter 4

4. What is a procedure that improves the current requirements engineering process at Sqills?

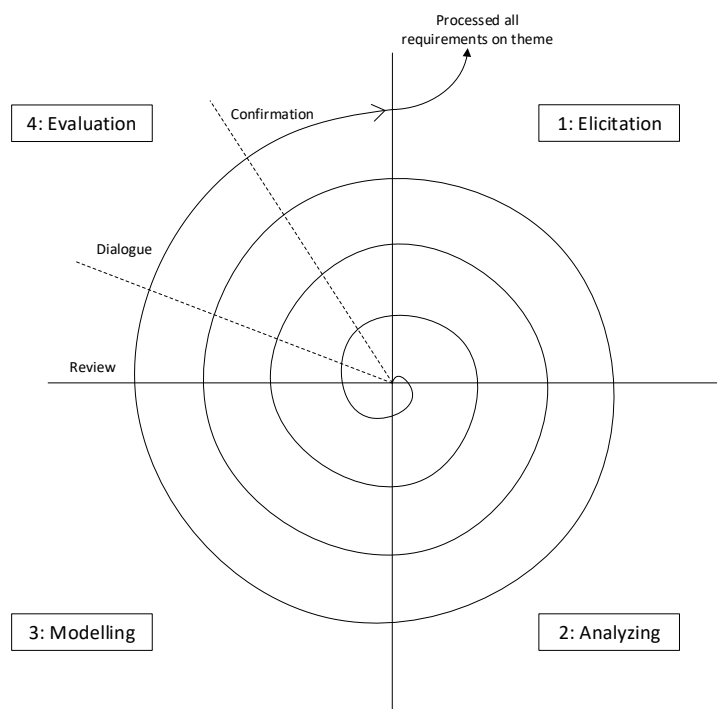
The model starts with a taxonomy of the requirements. (see figure 10.1a) The requirements are divided and further elicited in themes, which are groups of requirements, based on a topic, scenario or business process. After the initial elicitation phase, the requirements will be further processed per theme. This is done in four phases: elicitation, analyzing, modeling and evaluation. These phases will repeat every time until the customer and vendor agree in an evaluation that all requirements from that theme are processed correctly. In the model, this is shown as a spiral (see figure 10.1b) where the first round is a high-level analysis, the succeeding rounds are used to analyze the requirements in more detail and at the end of the final round, there is a consensus between the vendor and the customer. Of course, this process has to be fulfilled for every theme and they should be worked on at the same time to ensure that knowledge is shared between consultants. More detail can be found in chapter 8.

5. Is the proposed procedure working properly according to the involved employees or can it be optimized?

Before creating the final version of the model as referred to at question 4, some early versions were created. These versions have been evaluated with the involved employees to find what their ideas and wishes are. These evaluations give the confidence that the final model can be applied and well used at Sqills according to the involved employees. The early model and validations can be found in chapters 5 through 7



(a) Taxonomy of requirements



(b) Spiral for processing requirements

Figure 10.1: The developed model.

6. What recommendations can be given when starting to use this new requirements procedure?

When a company starts using the model, they should create a protocol, based on their own organization. This means they have to take into account their own way of working with regards to agility, the software's properties, the type of customers, the usage of partner companies and many other aspects. For that reason it is important that they know themselves well and use input from multiple employees to create a protocol.

10.2 Goal of the research

The goal of the research as stated in chapter 1 is the following:

The goal of this research is to **improve** the requirements engineering process for software product vendors **by developing** a model **which covers** the most important aspects of the requirements engineering process for customer implementations **in order to** help Sqills expand their protocol for S3 Passenger implementations.

The thesis aimed to develop a model that can be applied to more companies. However, the original intent of the research is to let this model be a guideline for Sqills to expand their protocol.

The goal has been reached, as a model has been created. Expanding the protocol at Sqills is not included in this thesis and needs to be done by employees of Sqills. A practical guideline on the approach Sqills should take on this, will be given in section 11.3.

Discussion

11.1 Abstraction level

The original intention of the researcher was to provide Sqills with a protocol for their requirements process. This was supposed to be explicitly for Sqills and directly applicable.

Half way through the project the evaluation results showed that the employees had different opinions on the current way of working. They had similar differences on the future of the product and the way their work and attitude would change. This resulted in many different visions of employees which should all be implemented in the model.

Taking into account the many different visions and the fact that Sqills and the product S3 Passenger is continuously evolving, the decision has been made to keep the model general. This way it is not only applicable to Sqills, but also in many other situations. Besides that, it is useful for starting companies, but also for more full-grown companies. The considerations of section 9.2 give an indication of the various ways the model can be used.

11.2 Scientific implications

This section on scientific implications is threefold. The first question is whether the topic of this paper is useful for the scientific community. The second question is whether this specific research adds scientific value and the third topic is the limitations of this research.

11.2.1 Gap in the literature

In the beginning of this research, the researcher found that there is a gap in the literature on requirements processes specifically for software products. There is research on software products, but this is mostly on management issues and on the general roadmap for a product. Literature on requirements analysis for software products is not available.

The question rises whether this is a topic that needs scientific research, or should be handled by practitioners? It shows that at least Sqills could have used the information. However, the question is not easy to answer and will be left for the scientific community to decide on.

In this research a start is made to close the gap. A beginning had been made on finding related literature and a model has been made that can be used by software product vendors. To further close the gap, it is important to have more research on software products and the companies who offer them. Questions can be what the differences are between the companies and how that relates to the lifespan of the product, the quality of the product and the satisfaction of the customer. By knowing the differences between these companies, it may be possible to work towards an optimal situation for vendors of software products.

11.2.2 Scientific value

As this research is the first attempt (to the best of the researchers knowledge) there were no existing theories on requirements for software products to use. For that reason a standard iterative design process was used with a practical validation approach. The first version which has been created by the researcher is not scientifically valuable, but the final model is based on concrete findings and therefore usable.

The opinions of Sqills' employees were all different and gave many different insights which are all described. This ensures that the model is not only for Sqills, but more widely acceptable and could therefore be a basis for scientific research.

11.2.3 Limitations

As there was no previous research to be found on the subject, there was no information on the pitfalls for similar projects. This lack of hints and tips made that the whole research is based on ideas on the researcher, supervisors and direct peers. Some aspects, such as the difficulty to create a concrete model for the current situation of Sqills, was found later on. If the researcher realized this earlier, more time could have been spent in finding experts from outside Sqills.

The validation of the model is the most limited part of the research. The employees from Sqills did give good insights in working processes. However, it would have been good to have a more varied group of people or techniques for validation. Firstly it would be good to validate the model with employees from various companies, for example ones who are around for longer, or exactly the opposite, companies who are relatively young.

Another validation option that would have been nice, but is more difficult to realize, is to have a practical validation. In that case, it would be needed for companies to be willing to work with this model. A researcher would have to observe and describe whether the model works and which techniques would be best to use in their situation.

11.3 Recommendations for Sqills

This research has been performed on behalf of Sqills and their goal is to be able to create a protocol for the S3 Passenger implementations. This is possible using the model that has been created. There are two main recommendations for Sqills when creating this protocol.

1. Decide how strict the protocol needs to be.
Do the consultants need a lot of freedom to adjust their way of working if it suits them or their customer or is the goal to have one way of working that everybody has to align with. And on which subjects is strictness needed?
2. Make good use of different people when creating a protocol.
When creating a protocol, it is important that everybody agrees on it and can work with it. The evaluations in this research have shown that different employees have different opinions on their way of working and what the 'correct' way is to treat customer's unique requirements. For this reason the a well thought of combination of employees should be involved in creating a protocol.

These two recommendations are the basis steps that should be taken when starting to create a protocol. Next there are some recommendations for the contents of the protocol.

3. Communication with the customer.
The protocol should include information on how to communicate with the customer. During most of the implementations, the information is shared via the information system of Sqills, however, when this is not possible for any reason, an alternative standard needs to be set.
4. Documentation of S3 Passenger.
The consultants need to look up information on the system regularly and when using partner companies they also need easy access to documentation. There needs to be a clear standard on the documentation and a plan should be made on how to keep this documentation up-to-date.
5. Task division with partner companies.
In the previous implementations, the partner companies have had different roles. As partner companies become a standard approach, it is useful to set a standard with regards to the influence they may have and the amount of information they should have access to.

When the protocol is created and in use, there is one important recommendation for maintenance.

6. Keeping the protocol up to date.
It is easy to spend some time on creating a protocol and then let it be used by the employees. However, the protocol must stay up to date to make sure it stays used. For example adding themes when they are encountered in projects or adding employee roles or information on the system documentation is something that should always be kept up to date.

11.4 Future work

This research is aimed to provide a model for the requirements process for the software product implementations at Sqills. While working on the project many ideas came up on how to extend the research and topics were found on which more information would have been nice. All these ideas and topics have been gathered and form the basis of the ideas for future research. Most of these ideas are useful for Sqills internally, other may need a more scientific approach.

Correctness and applicability of the model: The model has only been validated with three experts and has not been put to a practical test yet. So the most important future work is to apply the model to various companies, see how they use it and how their protocol will look like. The differences between organizations can be compared to the difference in the protocol they created.

Another aspect that can be tested is to what extent the consultants follow the protocol and to what extent they feel comfortable with using such a protocol.

Software prioritizing: There are many theories on how to prioritize software requirements as explained in section 3.1.1. Which of these techniques are used at Sqills at the moment and which techniques give the best result for the S3 Passenger software at Sqills? What role should a business consultant have in this prioritizing process?

Grouping of requirements: In the model, the requirements are placed in themes where every theme is about one business process or a topic that makes sense for the customer. It is interesting to find which themes are most common to use and which themes work the best for different types of customers. Are these themes similar for other vendors who have a different product?

Fit/Gap analysis: One of the aspects that was not elaborated in detail in the thesis, is the fit/gap analysis. This is an analysis, specific for S3 Passenger and it may cost a lot of time to elaborate on the subject. However, it may be possible to standardize this process to. Examples are to check for requirements (e.g. based on keywords) whether they appeared at other customers too and relate to how they were modeled in those cases. Another example is to use these keywords to search documentation and use this to find out whether there is a fit. Rolland & Prakash (2000) may be a useful reference.

Version release management: There is already quite some literature available on release management as has been found in this research. The companies from the case study mentioned this as an issue. Sqills has not yet mentioned this as an issue. However, it is always good to look at these aspects before they go wrong. How is release management handled at Sqills at the moment and how can it be improved?

Feature teams: Feature teams is a type of organizational structure where the development teams are not based on functionality or modules of the system, but on feature groups, which often transcend the existing modules. What are the advantages and disadvantages of these type of themes? Does this work well with the described model?

Amount of Requirements: How many requirements are needed for a good implementation, how many of these requirements are necessary and how many of these seem to be unimportant after all. Perhaps: how many of the requirements are normally implemented or discarded and how does this relate to the characteristics of the customer?

References

- Ahmad, M. M., & Cuenca, R. P. (2013). Critical success factors for erp implementation in smes. *Robotics and Computer-Integrated Manufacturing*, 29(3), 104-111.
- Bahill, A. T., & Madni, A. M. (2017). Discovering system requirements. In *Tradeoff decisions in system design* (pp. 373–457). Springer.
- Berander, P., & Andrews, A. (2005). Requirements prioritization. In *Engineering and managing software requirements* (pp. 69–94). Springer.
- Brehm, L., Heinzl, A., & Markus, M. L. (2001). Tailoring erp systems: a spectrum of choices and their implications. In *System sciences, 2001. proceedings of the 34th annual hawaii international conference on* (pp. 9–pp).
- Carson, R. S. (1998). 1.6. 4 requirements completeness: A deterministic approach. In *IncoSE international symposium* (Vol. 8, pp. 738–746).
- Clegg, D., & Barker, R. (1994). *Case method fast-track: a rad approach*. Addison-Wesley Longman Publishing Co., Inc.
- Daneva, M. (2004). Erp requirements engineering practice: lessons learned. *IEEE software*, 21(2), 26–33.
- Davis, A. M. (1990). *Software requirements: analysis and specification*. Prentice Hall Press.
- Escalona, M. J., & Koch, N. (2004). Requirements engineering for web applications-a comparative study. *J. Web Eng.*, 2(3), 193–212.
- Finkelstein, A., Spanoudakis, G., & Ryan, M. (1996). Software package requirements and procurement. In *Software specification and design, 1996., proceedings of the 8th international workshop on* (pp. 141–145).
- Finney, S., & Corbett, M. (2007). Erp implementation: A compilation and analysis of critical success factors. *Business Process Management Journal*, 13(3), 329–347.
- Firesmith, D. (2005). Are your requirements complete? *Journal of Object Technology*, 4(1), 27–44.
- Fui-Hoon Nah, F., Lee-Shang Lau, J., & Kuang, J. (2001). Critical factors for successful implementation of enterprise systems. *Business process management journal*, 7(3), 285–296.

- Globalteckz.com. (9-11-2013). *Differene between erp and enterprise systems*. Retrieved from <http://blogs.globalteckz.com/difference-between-erp-enterprise-systems/> (Accessed: 2017-1-19)
- Grady, J. O. (1993). *System requirements analysis*. New York: McGraw-Hill.
- Gruenbacher, P. (2000). Collaborative requirements negotiation with easywinwin. In *Database and expert systems applications, 2000. proceedings. 11th international workshop on* (pp. 954–958).
- Harris, R. (2000). Customization versus standardization: striking a balance in erp software. *Machine Design*, 72(14), 64–69.
- Hofstede, G., Hofstede, G. J., & Minkov, M. (2010). *Cultures and organizations: Software of the mind*. revised and expanded. McGraw-Hill, New York.
- Huang, S. Y., Chen, H.-J., Chiu, A.-A., & Hsieh, S.-L. (2012). The factors of erp customization from consulting company's perspective. *MIS REVIEW: An International Journal*, 17, 1–30.
- ISO/IEC 29148:2011. (2011). *Systems and software engineering - life cycle processes - requirements engineering*. International Standardization Organization.
- Kerzner, H. (2013). *Project management: a systems approach to planning, scheduling, and controlling*. John Wiley & Sons.
- Langer, A. M. (2016). Establishing requirements using a request for proposal (rfp) and a request for information (rfi). In *Guide to software development* (pp. 49–70). Springer.
- Lauesen, S. (2002). *Software requirements: styles and techniques*. Pearson Education.
- Lauesen, S. (2003). Task descriptions as functional requirements. *IEEE software*, 20(2), 58–65.
- Light, B. (2005). Going beyond 'misfit' as a reason for erp package customisation. *Computers in Industry*, 56(6), 606–619.
- Luo, W., & Strong, D. M. (2004). A framework for evaluating erp implementation choices. *IEEE transactions on Engineering Management*, 51(3), 322–333.
- Mocking, R. (2017). *Growth at software product vendors: A multiple case study*. (Master's research project)
- Momoh, A., Roy, R., & Shehab, E. (2010). Challenges in enterprise resource planning implementation: state-of-the-art. *Business Process Management Journal*, 16(4), 537–565.
- Napier, A. (25-03-2011). *What is the difference between erp and enterprise system?* Retrieved from <http://ezinearticles.com/?What-Is-The-Difference-Between-ERP-And-Enterprise-System?&id=6114179> (Accessed: 2017-1-19)

- Nurmuliani, N., Zowghi, D., & Powell, S. (2004). Analysis of requirements volatility during software development life cycle. In *Software engineering conference, 2004. proceedings. 2004 australian* (pp. 28–37).
- Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering: a roadmap. In *Proceedings of the conference on the future of software engineering* (pp. 35–46).
- Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements engineering and agile software development. In *Enabling technologies: Infrastructure for collaborative enterprises, 2003. wet ice 2003. proceedings. twelfth ieee international workshops on* (pp. 308–313).
- Parthasarathy, S., & Anbazhagan, N. (2007). Evaluating erp implementation choices using ahp. *International Journal of Enterprise Information Systems*, 3(3), 52.
- Parthasarathy, S., & Daneva, M. (2014). Customer requirements based erp customization using ahp technique. *Business process management journal*, 20(5), 730–751.
- Pereira, R. E. (1999). Resource view theory analysis of sap as a source of competitive advantage for firms. *ACM SIGMIS Database*, 30(1), 38–46.
- Rolland, C., & Prakash, N. (2000). Bridging the gap between organisational needs and erp functionality. *Requirements Engineering*, 5(3), 180–193.
- Rothemberger, M. A., & Srite, M. (2009). An investigation of customization in erp system implementations. *IEEE Transactions on Engineering Management*, 56(4), 663–676.
- SAP SE. (n.d.). *Asap 8 methodology for implementation*. Retrieved from https://roadmapviewer-supportportal.dispatcher.hana.ondemand.com/#/group/3DAE6BF3-610C-4FC5-83E9-D7595854F5F8/roadmap/SUITEONPREMAGL/phase/SUITEONPREMAGL_1/node/SUITEONPREMAGL_1 (Accessed: 2017-01-17)
- SAP SE. (2016). *Explore the sap portfolio*. Retrieved from <http://www.sap.com/solution.html> (Accessed: 2017-01-19)
- Scacchi, W. (2002). Understanding the requirements for developing open source software systems. *IEE Proceedings-Software*, 149(1), 24–39.
- Segal, J. (2005). When software engineers met research scientists: A case study. *Empirical Software Engineering*, 10(4), 517–536.
- Sharma, R., Patil, S., & Tandon, A. (2012). Customization and best practices model for adopting erp system: an analysis. *International journal of business strategy*, 12(1), 1–9.
- Subramanian, B. (26-1-2016). *Difference between erp software system and enterprise system*. Retrieved from <https://www.linkedin.com/pulse/difference-between-erp-software-system-enterprise-balasubramanian-g> (Accessed: 2017-1-19)

- Tarhini, A., Ammar, H., Tarhini, T., et al. (2015). Analysis of the critical success factors for enterprise resource planning implementation from stakeholders' perspective: A systematic review. *International Business Research*, 8(4), 25.
- Treacy, M., & Wiersema, F. (1993). Customer intimacy and other value disciplines. *Harvard business review*, 71(1), 84–93.
- Van De Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006). Towards a reference framework for software product management. In *Requirements engineering, 14th ieee international conference* (pp. 319–322).
- Van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. In *Requirements engineering, 2001. proceedings. fifth ieee international symposium on* (pp. 249–262).
- Verschuren, P. J. M., & Doorewaard, J. (2007). *Het ontwerpen van een onderzoek*. Den Haag: Lemma.
- Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering*. Springer.
- Wolfswinkel, J. F., Furtmueller, E., & Wilderom, C. P. (2013). Using grounded theory as a method for rigorously reviewing literature. *European journal of information systems*, 22(1), 45–55.
- Wymore, A. W. (1993). *Model-based systems engineering* (Vol. 3). CRC press.
- Zach, O., & Munkvold, B. E. (2012). Identifying reasons for erp system customization in smes: a multiple case study. *Journal of Enterprise Information Management*, 25(5), 462–478.
- Zowghi, D., & Gervasi, V. (2002). The three cs of requirements: consistency, completeness, and correctness. In *International workshop on requirements engineering: Foundations for software quality, essen, germany: Essener informatik beitiage* (pp. 155–164).
- Zowghi, D., & Gervasi, V. (2003). On the interplay between consistency, completeness, and correctness in requirements evolution. *Information and Software technology*, 45(14), 993–1009.
- Zowghi, D., & Paryani, S. (2003). Teaching requirements engineering through role playing: Lessons learnt. In *Requirements engineering conference, 2003. proceedings. 11th ieee international* (pp. 233–241).



Validation questions employees/experts

Below, the full version of the preparation of the meetings is shown as an addition to the information in section 6. This is a basic outline in English and in Dutch because all interviewees were Dutch and using this language makes the interview go easier. The questions were not stated literally, but they were used as a guideline throughout the interview and to make sure that the important aspects were spoken about. The introduction is a collection of information that needs to be told in the beginning of each interview.

A.1 English

Introduction

- Reason for meeting
- Format of the meeting
- Status of the graduation project
- voice recording of the meeting
- Goal of the meeting
 - To get as much feedback as possible

Explanation of the model and questions

- Firstly the model needs to be explained by the researcher
 - A presentation will be used to show the model and its aspects on a big screen
 - The same order for explaining the model will be used as is used in chapter 5
 - During the explanation, the employee has every possibility to ask questions
When the employee asks a question about the interpretation of the model, he will be asked to explain his vision on the possible interpretations and the possible effects of these.
 - The employee will be asked whether he fully understands the model.
- IS the model complete?
 - Does the model help in eliciting all requirements?
 - Is there an activity missing in the model? Or is there an activity that needs more attention?
- What are the implementations the employees has worked on earlier?
 - What implementations have you worked on before?
 - Are there elements of the model which you have applied (unknowingly?) on these projects?

- Are there elements of the model which would have been nice to use on these projects?
- Are there elements of the model which would not have worked on these projects?
- Were there moments during the projects when you were frustrated, you were angry or you were unable to sleep because of the issues that arose.
- The researcher shows remarks that were given by previous meetings (if there is time left)
 - This remark has been given at an earlier meeting.
 - Do you agree with this remark?
 - Why do you or don't you agree on this remark?
 - Would that be applicable to every project?

A.2 Dutch

Introductie

- Reden van het gesprek
- Vormgeving van het gesprek
- Status van het afstudeerproject
- Geluidsopname van het gesprek
- Doel van het gesprek
 - Graag zo veel mogelijk feedback

Uitleg model en vragen

- allereerst wordt het model uitgelegd door de onderzoeker
 - Een presentatie wordt gebruikt om het model en zijn aspecten op een groot scherm te laten zien.
 - Dezelfde volgorde voor het uitleggen van het model wordt aangehouden zoals hij beschreven wordt in hoofdstuk 5.
 - De medewerker heeft op elk moment de kans om vragen te stellen. Wanneer de medewerker een vraag stelt over de interpretatie van het model, dan zal de onderzoeker vragen wat de mening van de medewerker is over die interpretatie en de mogelijke gevolgen daarvan.
 - De medewerker wordt gevraagd of hij/zij het model begrijpt.
- Is het model compleet?
 - Helpt het model met het ophalen van alle requirements?
 - Mist er een activiteit in het model? Of is er een activiteit die meer aandacht zou moeten krijgen?
- Wat zijn de eerdere implementaties waar de medewerker aan gewerkt heeft?
 - Welke implementaties heb je eerder aan gewerkt?
 - Zijn er elementen van het model die je (onbewust) toegepast hebt tijdens die projecten?
 - Zijn er elementen van het model die een toevoeging hadden kunnen zijn op de projecten?
 - Zijn er elementen van het model die niet gewerkt hadden bij de projecten?
 - Zijn er momenten geweest bij de projecten waardoor je heel gefrustreerd was/boos was/ervan wakker hebt gelegen? Wat is er gebeurd op die momenten?

- De onderzoeker oppert de wijzigingen die bij eerdere gesprekken naar voren zijn gekomen
 - Deze opmerking is gegeven door bij een eerdere bespreking (naam van betreffende medewerker niet noemen)
 - Kan je je vinden in deze opmerking?
 - Waarom ben je het er wel of niet mee eens?
 - Zou dat op verschillende projecten van toepassing zijn?

B

Handout for second validation

See next page

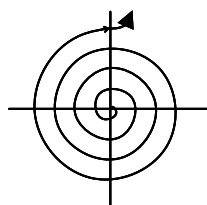
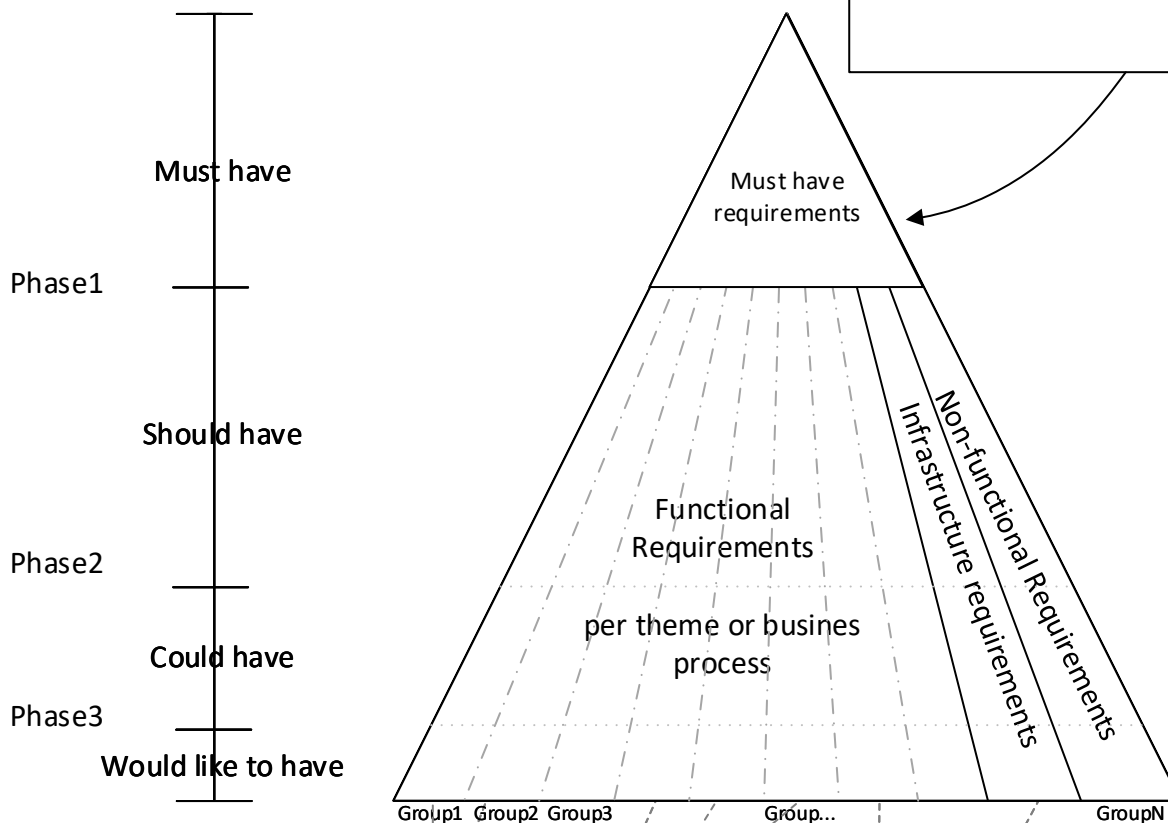
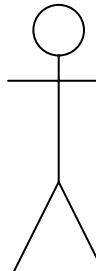


New Customer

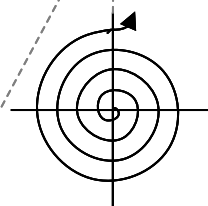
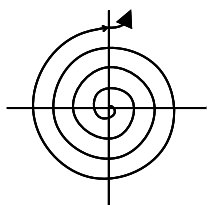
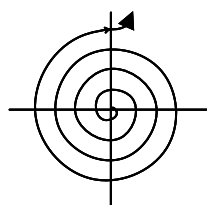
Actions:

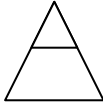
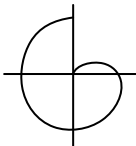
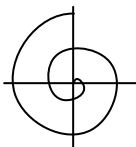
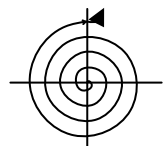
- Reveicing or Eliciting Requirements
- Agreement on terminology
- Division over Themes (business process or subject)
- Prioritization of requirements

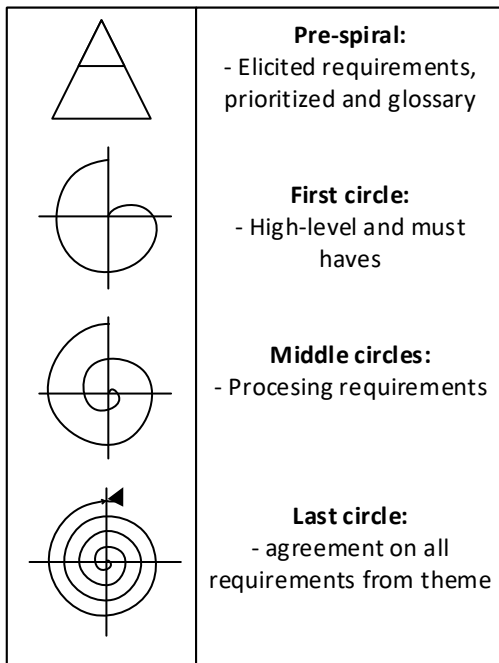
MoSCoW
delivery phases



One spiral for each requirements group (Theme)



	Pre-spiral: - Elicited requirements, prioritized and glossary
	First circle: - High-level and must haves
	Middle circles: - Processing requirements
	Last circle: - agreement on all requirements from theme



	Elicitation	
stakeholder analysis	organisational	pre-spiral
brainstorming	creativity	pre-spiral/first circle
focus group	creativity	pre-spiral/first circle
Interviewing	specific topic	all circles
Observation	details	all circles
Task demonstration	obstacles	all circles
document studies	specifics	all circles
questionnaires	multiple people	all circles
domain workshops	design	middle circles
design workshops	design	middle circles

Discussion
 Priorities
 Negotiation
 Tit-for-tat etc?

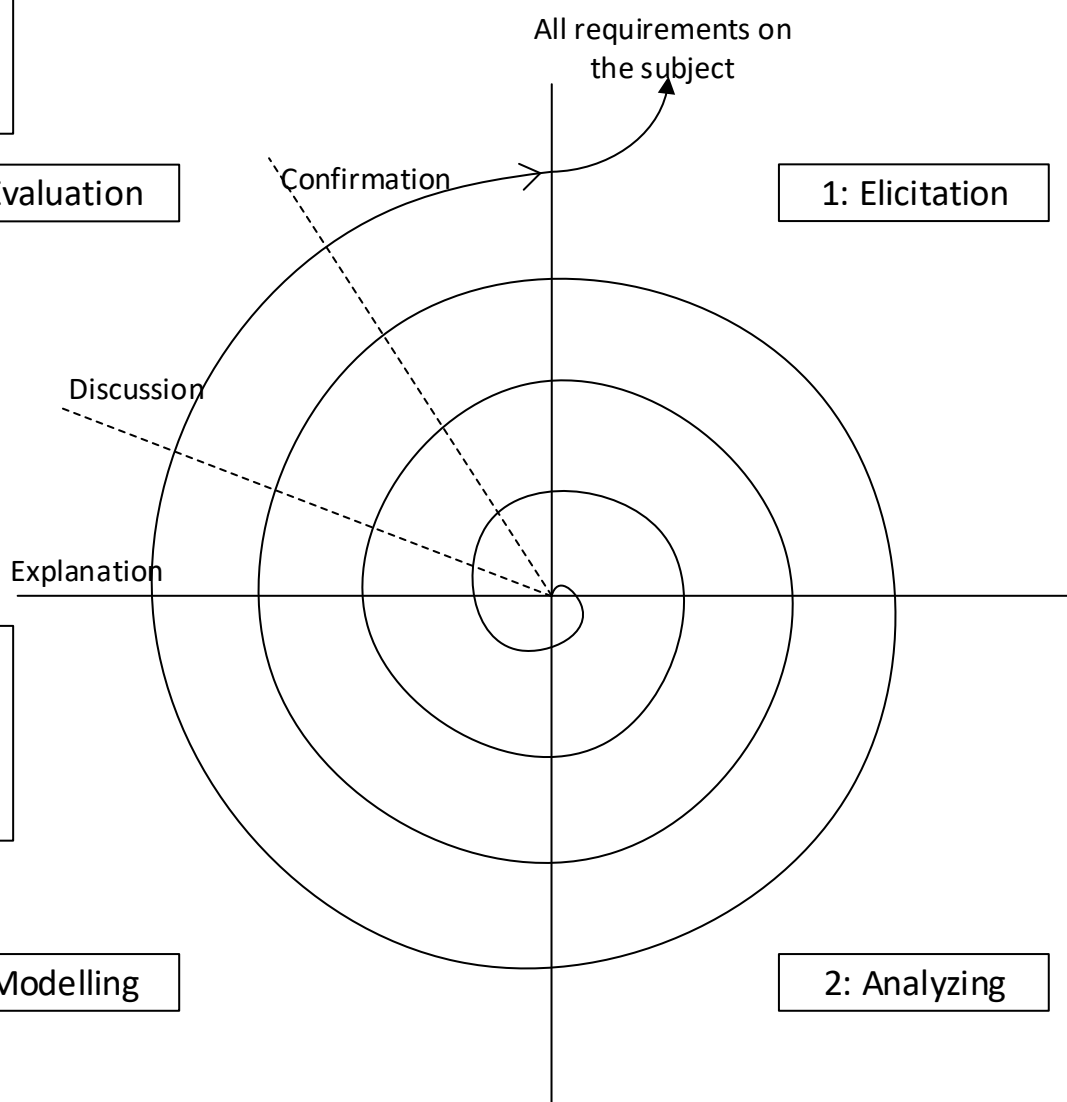
Explanation
 prototyping
 pilot experiment

4: Evaluation

1: Elicitation

3: Modelling

2: Analyzing





Example application model

In this section a practical description is given on a fictional project that has been implemented using the developed model. The fictional customer is called MiniBus. The name makes sense as the customer is a small company who only wishes to implement minimal functionalities of the S3 Passenger software. MiniBus has Very few requirements, namely 34. In collaboration with the consultant from Sqills these requirements have been divided over three themes and there are no separate groups for non-functional or infrastructure requirements. (see figure C.1a)

The customer was also able to show which requirements were most important by stating what requirements were absolutely needed for a first release. With the amount of requirements being so small, this group contained about 80% of the requirements. Nonetheless, a general idea on Minibus' priorities of the requirements was created this way.

The business consultant started three pages on the Sqills confluence page, each of them containing one theme and the correlating requirements. Together with a colleague he decided to work on one of the themes and the other colleague would start working on the other two themes. Within a week they finished the high-level check of all requirements. This can be seen in figure C.1c as for all three the spirals, the first round was completed. In the discussion with the customer both parties were hopeful that the implementation would be successful, so they continued the project.

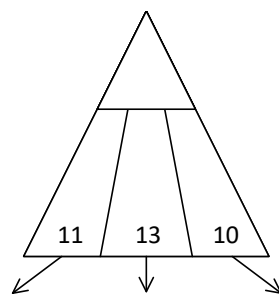
After two weeks, the consultants have worked through all requirements from all three spirals for another time and have analyzed and modeled solutions for the requirements. The end of this phase can be seen in figure C.1d. However, after the evaluation phase, some changes in the planning had to be made as can be seen in figure C.1e. For the first group, Minibus agreed with most of the modeled solutions, however, some of the models were not entirely in line with what they wanted. The consultant already anticipated this and decided to look at these requirements for some more time. There were not that many requirements in the third group and the requirements that were given were noted down very clearly. This ensured that the consultant modeled solutions that were fully in line with the wishes of the customer. In the discussion, the consultant and Minibus decided that everything from this group was in order. The second group gave much more problems. It appeared that one of the requirements meant much more than was stated on the original requirements. The consultant and Minibus had a fierce discussion and they decided that the functionality Minibus wanted in the system was realizable, but it was quite unique for the systems features. For this reason the consultant decided to add more cycles to the spiral so that he keeps in mind that more time must be spent on this theme. The final result is to be seen in figure C.1e

After the consultant has realized that the second theme costs more time, they spent much time on that and a few rounds later they are almost finished. However

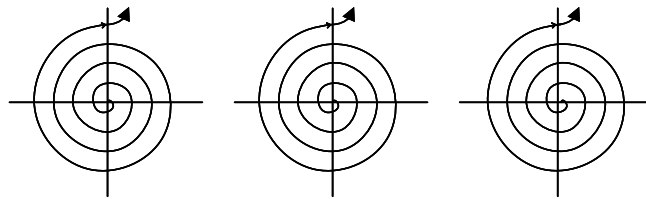
they discovered that one of the requirements from theme 2 also had some influence on the requirements from theme 3. For that reason they picked up theme 3 again. At this point theme 1 has been fully analyzed and MiniBus and the consultant agree on the modeled solutions of the requirements. The status can be seen in figure C.1f.

Finally, after looking into theme 3 again and finalizing both theme 3 and 2, all requirements are covered and the main requirements process is done as can be seen in figure C.1g. However, even when all three the spirals seem to be done, it is always possible that another issue arises. When that happens, it is still important to follow the correct process of eliciting, analyzing, modeling and evaluating.

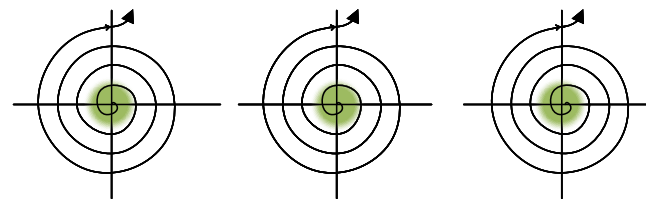
Figure C.1: example of an implementation using the proposed model.



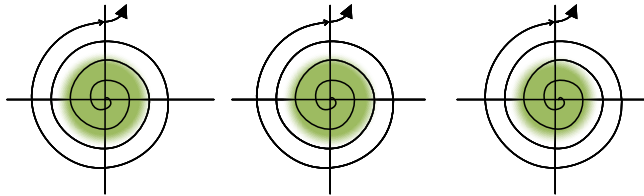
(a) Requirements taxonomy



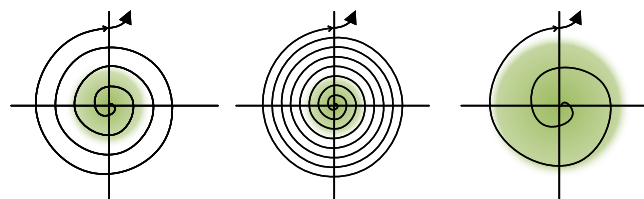
(b) Basic spirals



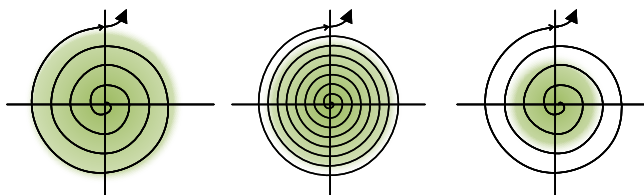
(c) High-level analysis done



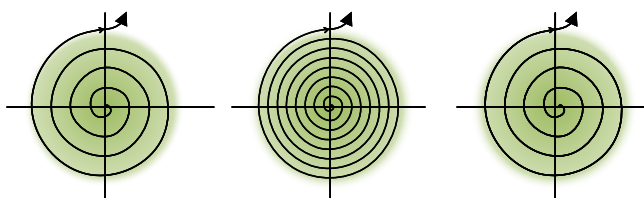
(d) First detailed analysis done



(e) First theme needs some more attention, second theme appears to have more difficult requirements than known before, third theme is already done



(f) First theme is done, second theme is almost done, but has a requirement which influences the third theme



(g) All three themes are fully processed (for now)