How to improve Requirements Engineering

Author:
Jasper van Duijnhoven

Studentnumber:
0824054

Date:
03-03-2014

Study:
Master Information Science
Radboud University Nijmegen

First examiner:
dr. Stijn Hoppenbrouwers (Radboud University)

Second examiner:
prof. dr. ir. Theo van der Weide (Radboud University)

Supervisor:
Hans ter Huurne (Heijmans N.V.)

heijmans

Radboud University Nijmegen

Abstract

Combing theory with practice, we have analyzed the Requirements Engineering practices at the IT department of construction company Heijmans. We have analyzed several topics, as found in the literature study, focusing on three areas: the process, the people and the products. By conducting interviews and case studies, we mapped the framework from literature to the Heijmans situation. By doing this we discovered seven topics where Heijmans can improve its Requirements Engineering practices. Therefore Heijmans should focus on: expectations management, requirements management, business rules, functional designing techniques, project methods, responsibilities and uniformity. On these seven topics we have constructed an advice for Heijmans, how to handle them.

Acknowledgements

By writing these last lines, my research period of six months comes to an end. The thesis which lies before you is not an product of the author alone. I have had help of numerous people. First of all I would like to thank Hendrik Jan Smaal of Heijmans for the opportunity to get an internship at Heijmans and to be able to do the research at Heijmans. I would also like to thank Hans ter Huurne for helping me formulate this interesting research topic and for all the constructive feedback I received from him. Of course I would also like to thank all employees of Heijmans who helped me come to this final result, not only those who I have had the pleasure to interview for the research. Stijn Hoppenbrouwers, my first examiner, also deserves a special thank you. During the whole research he provided feedback numerous times to keep the bar high for the result. I would like to thank Theo van der Weide for being the second examiner, making the circle complete, after being the lecturer of one of my first courses at the university and being the examiner for my Bachelor thesis. Last but not least I would like to thank all my friends and family for the support during the research, especially Sanne for keeping me motivated at all times.

Because my mother is not with us anymore and will not be attending my graduation, I would like to dedicate this Master thesis to her.

In loving memory, Olga van Duijnhoven-van Hal.

Index

1. Introduction

This document represents my Master thesis, finalizing my Master Information Science at the Radboud University in the Netherlands. Because I think it is very important to combine theory with practice, I chose to combine the thesis with an internship at a company. This company was construction company Heijmans. For me this was a great opportunity to see what role IT plays in companies where IT is not the core business, but still a very important tool to operate as one of the biggest competitors in its market. I have tried to analyze their Requirements Engineering practices from the perspective thought by the courses in the Information Science curriculum. Doing this I used the Master thesis of Chris Kleine Staarman, who graduated on the topic 'Agile Requirements Engineering', as a sound basis for my literature research. The goal of the thesis combined with the internship was not only to improve my knowledge about Requirements Engineering in practice, but also to help Heijmans improve their practices on this area.

1.1 The Heijmans IT department

Heijmans N.V. is one of the biggest construction companies in the Netherlands, employing more than 8000 people. They are active on the areas road construction, civil construction, utilities construction and real estate. The IT department, which controls the IT practices of all business units, consists of around 50 internal employees with different specialist fields. These employees are supported by several outsourced employees.

It is very important to know that Heijmans tries to keep the development or customization of applications to a minimum. Their strategy is to acquire the needed functionality by purchasing an existing product of a supplier. This does not mean the processes are very different from those of a company who develops its own applications, for instance the development of requirements is still present.

The department is divided in two sub departments: Operations (who maintain the current IT application landscape, IT services and IT infrastructure) and Business Information Alignment (BIA, who expand or improve the application landscape and other information needs). My internship takes place in the BIA department.

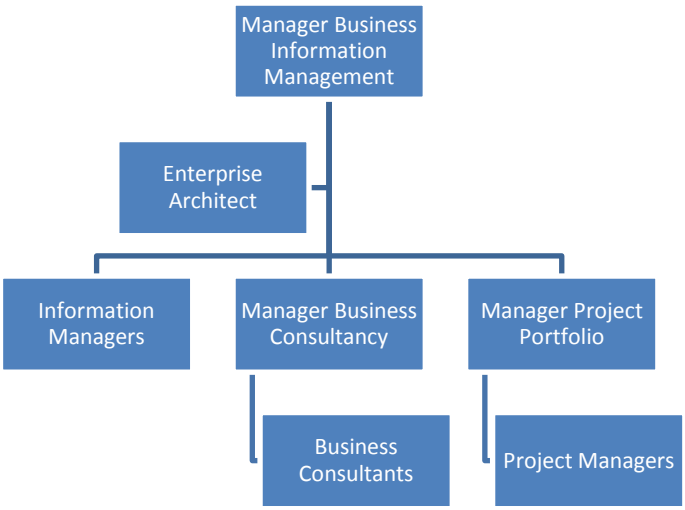

Figure 1: schematic representation of the Business Information Alignment department

1.1.1 Information Management

Information Management forms the bridge between business and IT. The Information Managers are each responsible for a 'business unit' (i.e. real estate, utilities, roads and civil) in the field of Business-IT. Their responsibilities are adjusting the information streams, the processes, the applications and the IT resources in their responsible business part. In collaboration with colleagues from the business units, the right Business-IT strategy is determined. This will result in the Business Information Plan.

1.1.2 Business Consultancy

The Business Consultants are active on the interface of business processes, IT applications and Telecom. They give advice and give solutions to problems (with and without IT). When colleagues from IT have related problems, the Consultants search for solutions together with them. The Consultants support the business with implementing the best suitable solution. They have an extended network of people, both internal as external, which gives them a lot of knowledge and expertise.

1.1.3 Project Portfolio Management

Project Portfolio Management is responsible for managing the projects determined in the Business Information Plan and managing the relations between these projects. They operate as project leaders in those projects. They control time, budget and results.

1.1.4 Enterprise Architecture

The Enterprise Architect defines and guards the enterprise architecture on the areas of IT applications and information processes. In this, he monitors that all changes in the IT landscape are compliant with the architecture, maintaining coherence, efficiency and stability on the landscape.

2. Problem statement

At the end of many IT projects, the delivered product appears to not satisfy the needs of the end user. In most of the cases the conclusion is drawn that the requirements do not meet the need of the user or that the requirements are not translated properly into features. We can state that in order for an IT project to be successful, the requirements have to describe a proper solution for the problems or the needs, and that those requirements have to be of good quality. One of the most used pictures about failing Requirements Engineering is the one below, which is characteristic for what can go wrong in a project.

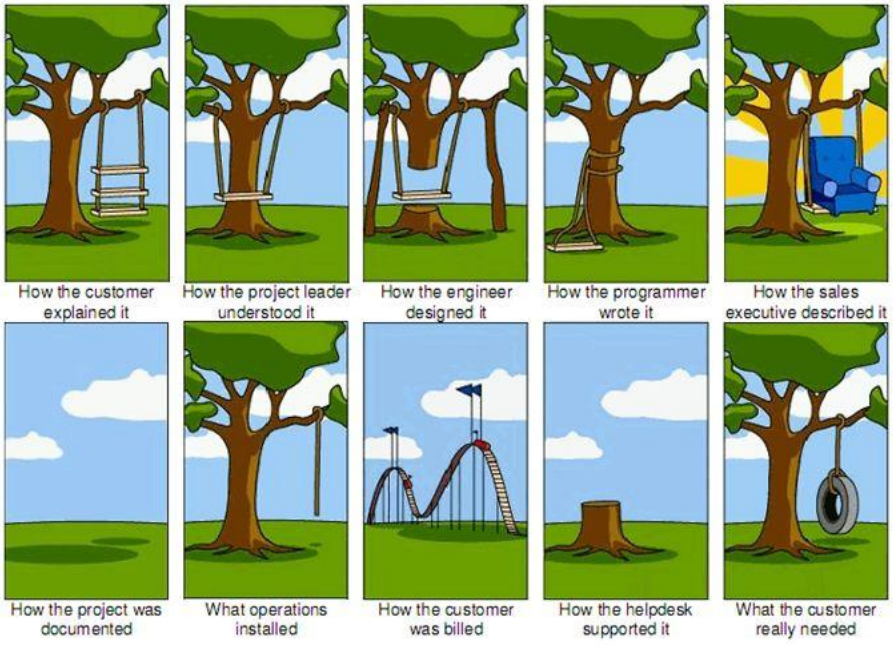

Figure 2: the importance of Requirements Engineering
(http://knovelblogs.com/wp-content/uploads/2012/08/Anforderungen_WAS_aus_Pj_wurde.jpg)

In the light of failing projects or projects that can be considered less successful, Heijmans is no different. The IT department of Heijmans executes around 30 projects a year to expand or adjust the application landscape of Heijmans. These projects apply different project methods per project. In most of the cases the traditional Waterfall method is used, but in recent projects the Agile method is applied sometimes. These two project methods require different methods of Requirements Engineering at first glance.

The purpose of this research is to examine the Requirements Engineering practices at Heijmans, and to give an advice how to improve them. By improving the requirements practices, Heijmans can improve the quality and validity of the requirements, resulting in more successful projects. This leads to the following main question that needs to be answered:

How can Heijmans improve its Requirements Engineering practices?

To be able to answer the main research question, two sub questions have been formulated. First we need to look at indicators for failing projects, that have a link with requirements:

What are indicators for possible problems in Requirements Engineering practices?

This will give us a sound basis for analyzing the requirements practices at Heijmans. Then we have to analyze the requirements practices at Heijmans using those indicators:

What do those indicators tell us about the Requirements Engineering practices at Heijmans?

3. Method

To answer the research questions we used several methods. We started with a literature study, to examine the most common bottlenecks in the requirements engineering process. We used a combination of literature about Requirements Engineering and IT project methods, especially about Requirements Engineering in Agile projects, due to the tension between Agile en requirements. In the literature we looked at why projects failed because of bad requirements and the 'do's and don'ts' in Requirements Engineering. The results of the literature acted as a framework of indicators for the second and third part of the research.

In the second part we conducted interviews with several employees of Heijmans, who are in any way involved in the requirements process. In these interviews, project managers, business consultants (as requirements engineers), developers and information managers (as representatives of the business) had their say. This provided us with varied viewpoints on the Requirements Engineering practices at Heijmans. The indicators found in the literature study, provided us with questions for the interviews, supplemented with questions about the gathered experiences of the Heijmans IT processes in the internship. We chose an informal format for the interviews, because we feel this suits the situation best (e.g. my internship at Heijmans). To let the interviewees share their experiences in a broader way than only answering the questions and with the use of examples, we got a more complete picture of the situation at Heijmans. This handed us valuable information, which would not have been discovered if the interviews were strictly formal. To get from question to results, we recorded the sound of the interviews. We used the recordings to get the information the interviewees handed us back on paper in 'story form', making sure the answers to the questions were present in the story. From these stories we abstracted the results.

In the third part of the research we conducted two case studies on IT projects at Heijmans, focusing on the requirements of course. We also looked at the templates for the design documents. As mentioned before, we will use the gathered data from the literature study again as a framework for looking at the requirements. The results of this part gave us a grip when drawing conclusions from the results of the interviews, like a validation. The two cases were chosen in consult with the project managers.

At the end of the research we were able to answer the research questions by combining the results of the different parts, and thus form an advice for Heijmans to improve their requirements process.

4. Theory

4.1 Requirements Engineering

In this research Requirements Engineering (RE) practices will be divided into three parts:

1. **The process**
   This includes all RE activities.
2. **The people**
   This includes all people involved in those RE activities.
3. **The products**
   The input and output that those RE activities and people involved deliver.

In these three areas lie all the possible bottlenecks when it comes to RE practices. Let us first look at what the available literature has to say about these three areas.

4.1.1 The process

Paetsch, Eberlein & Maurer (2003) describe the Requirements Engineering process as the following steps:

1. **Elicitation**
   In this step the requirements are discovered by communicating with the customer, for instance via interviews.
2. **Analysis**
   In this step the requirements are checked for necessity, consistency, completeness and feasibility.
3. **Documentation**
   In this step everything is documented to create an overview for later stages in the project and to be able to trace features back to requirements.
4. **Validation**
   In this step the documentation is reviewed by the stakeholders, to make sure that the proposed product is what they need.
5. **Management**
   In this step the implementation has begun. Activities that fall under management of requirements are change and version control, requirements tracing and requirements status tracking. The relation between requirements, design and implementation is maintained.

Because the management step is typically part of the implementation process and the other steps are not necessarily part of the implementation process (they are not most of the time), we can construct the following diagram.

Figure 3: the classic Requirements Engineering process

In the Requirements Development process requirements are, simply said, created. In the Requirements Management process requirements are maintained and if necessary edited. This last fact means that one or several Requirements Development processes could exist inside the Requirements Management process with some projects, in an iterative way.

But what makes RE such an important part of the software development process? As *Shams-Ul-Arif, Qadeem Khan & Gahyyur (2009)* and *Eberlein & do Prado Leite* point out, only 26% of software projects are considered successful. The keyword that is often mentioned as the cause for failure is bad requirements. Furthermore good requirements are often mentioned as the main reason for project success. Even half of the canceled projects were aborted due to a lack of requirements. Furthermore they state that 25% is the ideal percentage of effort that has to be put in RE practices in a project.

4.1.2 The people

Wupper (2012) demonstrates in his book, that people are an important factor in designing systems. When you want to build a system you need domain knowledge. Most of the time that specific domain knowledge can be found with domain experts. For instance, when you want to build an HRM system for a bank, you need people who have knowledge about HRM (systems) and people who have knowledge about banks, and maybe even people who have knowledge about the specific culture and processes of that particular bank. Most of the time these different knowledge areas are not invested in the same person. This has to be taken in account when determining who you want to get involved in a project.

When looking at the stakeholders in an IT project we have to look at four groups, according to *Wupper (2012):*

- The client
- The builder
- The user
- The environment

The first three groups appear obvious, but the last one is an odd one. One example that could be placed in the group 'environment', are partner companies. The builder can build an magnificent system that meets the requirements of the client, and every user uses it exactly as it was meant to be (and they love working with this system). But when the cooperation with partner companies is made impossible because they use other standards than the standards of the output of the system, we can say the system is a failure. This means all groups must be analyzed before designing a system. But it does not stop there. The differences between the groups must also be analyzed. Certain requirements from the client can contradict the wishes of the user, and this has to be managed.

The previous is needed to come up with a good design. But are the people irrelevant when the design is ready? No, definitely not. It should be very important to be able to hold people accountable and responsible for specific tasks or products. Let us introduce the term 'requirements owner'. Like the product owner is responsible for the product, the requirements owner is responsible for the requirements. Typically this person is also the one who developed the requirements. An especially important role for the requirements owner should be reserved in the requirements management phase of a project. This person should keep an overview on all requirements related objects, like updating requirements if necessary, or making sure the product is checked with the requirements.

4.1.3 The products

Documentation can be seen as the tangible input and output of the RE process. Input are document templates or workshops for instance, and output are documents like functional design documents or program demands documents. This output is not the end product (although it can be part of it), but it is the input for the implementation phase, technical design phase and the test phase. This does not make those documents less important, surely input is required to produce output. One could even say, good input is required for good output. This makes it important to analyze the input and output products of the RE process, to get a complete overview of the requirements phase.

4.2 Software Engineering methods

Because Heijmans uses different Software Engineering methods, we will first look at some different methods. Today there are loads of different Software Engineering methods, yet they all incorporate some kind of RE practices. The two most famous software engineering approaches are probably the traditional Waterfall method, also called the plan driven approach, and the revolutionary agile methods, which are iterative and incremental approaches with far less planning. A big misconception is however that in Waterfall methods everything is planned and in agile methods nothing is planned. According to *Boehm (2002)*, we should look at the different approaches as their location in a spectrum, with hackers who do not incorporate any planning on the far left, and the point where every little detail, every little pebble stone is planned on the far right.

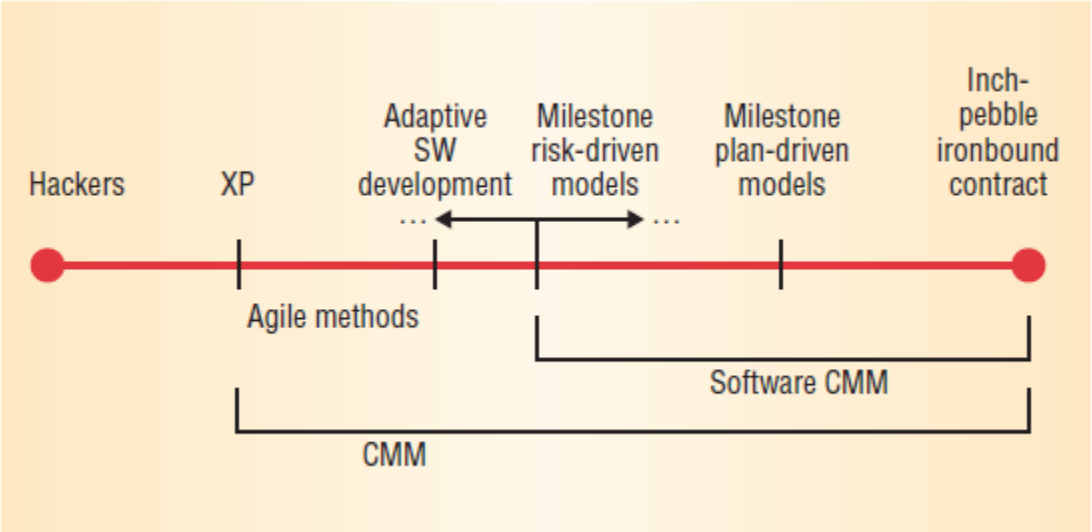

Figure 4: the planning spectrum (adopted from Boehm (2002))

As we can see the different agile methods are on a broad spectrum, indicating that the amount of planning differs a lot between them.

4.2.1 Waterfall method

The Waterfall method dates from 1970, when the first formal description was given in an article by Winston W. Royce. The model is described as a sequential method, divided in steps.

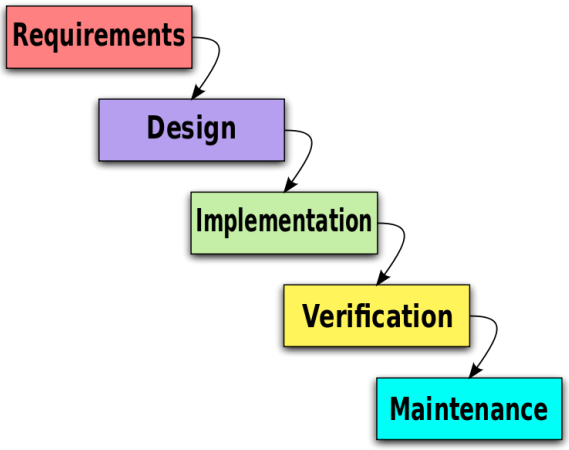

Figure 5: A graphical representation of the Waterfall principle (adopted from http://en.wikipedia.org/wiki/Waterfall_method)

As shown in shown in figure four, formally there is no way back when the project advances to the next step. This is also one of the biggest critics on the Waterfall method, next to the fact that it is impossible to plan everything from the start of the project. However, we have just established that the Waterfall method does not pretend to do so, as shown in figure 3.

4.2.2 Requirements Engineering in the Waterfall method

The RE process is quite clearly located in the first step of the Waterfall method. Because it is formally not possible to change the requirements in a later phase, they have to be developed very precisely to avoid problems or ambiguity in later stages. This is a bit problematic, because it is impossible to foresee problems that will arise in those later stages. The customer could want something different all of a sudden, or some technical aspects could pose problems. Nevertheless the accurate development of the requirements yields create strength in creating a good overview of the whole project, with good documentation for back tracking.

4.2.3 Agile methods

This approach was first mentioned in the *Manifesto for Agile Software Development* in 2001. Agile is not 'one' method, but more a collection of methods which all use an iterative and incremental approach. The basic thoughts behind Agile are that extensive planning before a project starts is impossible and unnecessary. The idea is to just begin with the most important features and deliver a working product and the end of the first cycle. At the end of the first cycle the customer gives feedback and the product is tested. After that comes the second cycle where the feedback of the customer is used to improve the product and more functionality is added. An Agile project has several cycles before the final product is delivered and key factors in this process are close customer relations (and thus feedback) and adding functionality.

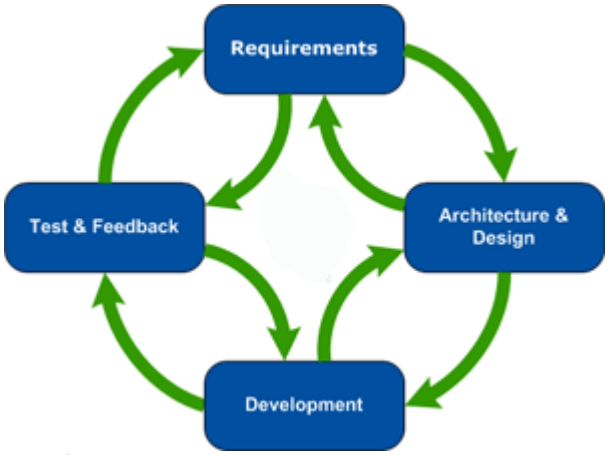

Figure 6: A graphical representation of the agile process (adopted from http://www.managedmayhem.com/wp-content/uploads/2009/05/agiledevelopmentprocess1.gif)

As shown in figure 5, the steps in the process do not have a strict order in which they are taken. This creates flexibility in the development process.

4.2.4 Extreme Programming (XP)

Let us first look at two examples of Agile methods, starting with XP. Kent Beck was the person who first introduced XP in 1999, when he wrote the book *Extreme Programming Explained: Embrace Change. Cohen, Lindvall & Costa (2004)* summarize the twelve rules of XP as following:

1. **The planning game**
   At the start of each iteration, customers, managers and developers meet to flesh out, estimate and prioritize requirements for the next release. The requirements are called "user stories" and are captured on "story cards" in a language understandable by all parties.
2. **Small releases**
   An initial version of the system is put into production after the first few iterations. Subsequently, working versions are put into production anywhere from every few days to every few weeks.
3. **Metaphor**
   Customers, managers and developers construct a metaphor, or set of metaphors after which to model the system.
4. **Simple design**
   Developers are urged to keep design as simple as possible, "say everything once and only once".
5. **Tests**
   Developers work test-first; that is, they write acceptance tests for their code before they write the code itself. Customers write functional tests for each iteration and at the end of each iteration, all tests should run.
6. **Refactoring**
   As developers work, the design should be evolved to keep it as simple as possible.
7. **Pair programming**
   Two developers sitting at the same machine write all code.
8. **Continuous integration**
   Developers integrate new code into the system as often as possible. All functional tests must still pass after integration or the new code is discarded.
9. **Collective ownership**
   The code is owned by all developers, and they may make changes anywhere in the code at any time they feel necessary.

10. **On-site customer**
    A customer works with the development team at all times to answer questions, perform acceptance tests and ensure that development is progressing as expected.
11. **40-hour weeks**
    Requirements should be selected for each iteration such that developers do not need to put in overtime.
12. **Open workspace**
    Developers work in a common workspace set up with individual workstations around the periphery and common development machines in the center.

Cohen et al. (2004) states that among the negative sides of XP are the team size, only small teams are supported by the method, and the possibilities for distributed teams, where XP only support 'one room' development.

4.2.5 Scrum

Ken Schwaber first described Scrum in 1996. An interesting fact is that the word scrum is borrowed from Rugby. In Ruby scrum means that players huddle together in an attempt to advance down the playing field. The project cycles in Scrum are called 'sprints'. What makes these sprints so special is that the team meets every day about the progress on the work and the planned work. Often a board with sticky notes is used where the board is divided into a backlog (work to be done), current work, and work that is done and ready for testing. This creates a good overview on the project and its tasks for all team members. It can also create a higher commitment, because you feel like a real team instead of sub teams, and your own performances are directly visible in the Scrum meetings.

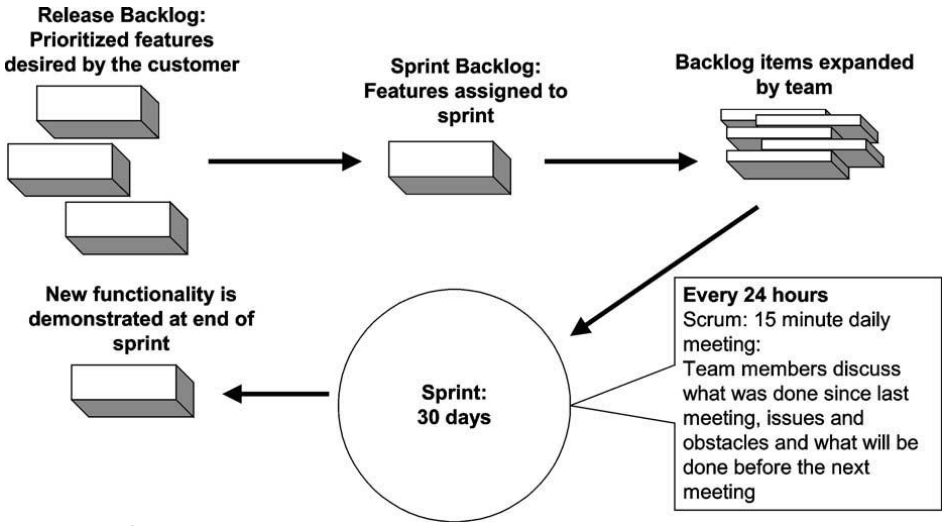

Figure 7: the Scrum life cycle (adapted from http://www.controlchaos.com)

Scrum has somewhat the same disadvantages as XP, having only support for small teams that work at the same location.

4.2.6 Requirements Engineering in Agile methods

Now here comes the tricky part. Formally RE is part of the Agile methods. However, the scale of RE is often very small. For instance documentation is often seen as a step which only slows the Agile process and is therefore skipped according to *Paetsch et al. (2003)*, which makes requirements tracing very hard. They also state that Requirements Management is not part of Agile development, and that the remaining steps are intertwined.

Shams-Ul-Arif et al. (2009) distinguishes five Agile RE models with respect to Requirements development:

1. **Input/output Requirements Engineering process**
   Combine input to create output.

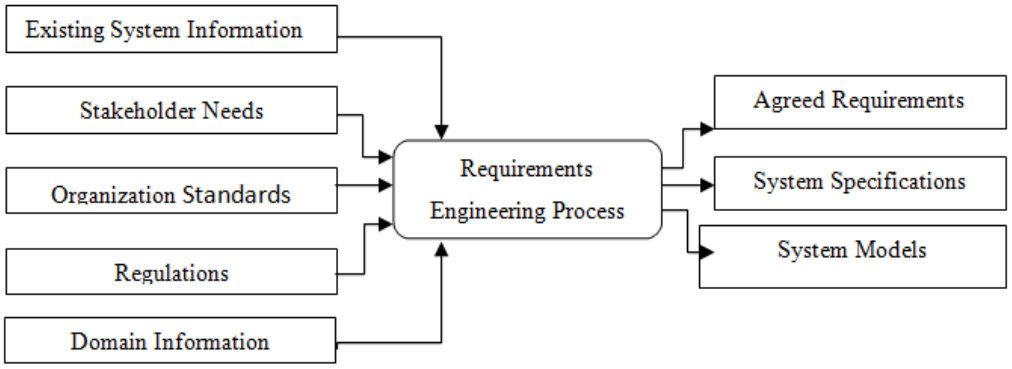

Figure 8: I/O RE process model (adapted from Shams-Ul-Arif et al. (2009))

2. **Linear Requirements Engineering process**
   Much like the Input/output RE process, but the 'black box' is now divided into different phases.

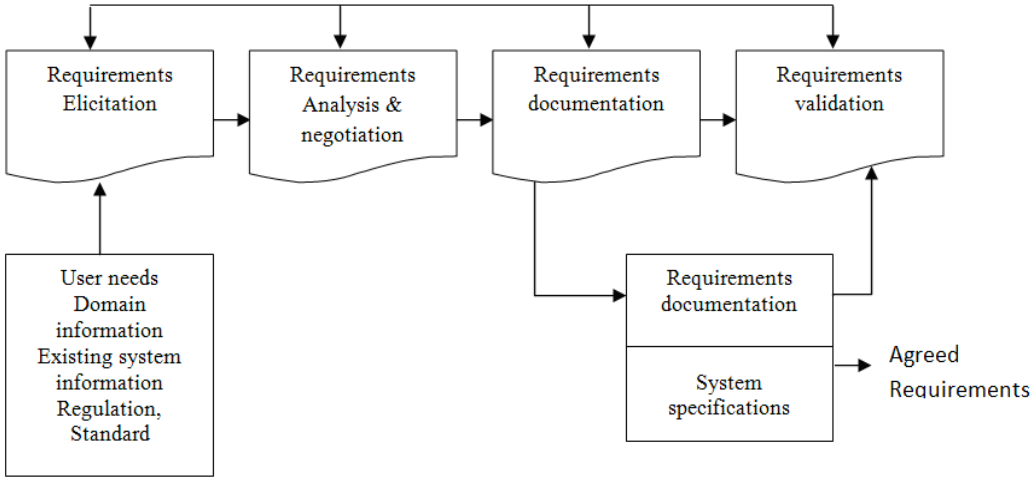

Figure 9: Linear RE process (adapted from Shams-Ul-Arif et al. (2009))

3. **Linear iterative Requirements Engineering process**
   The biggest difference with the previous models is the iterative approach. This model is best used when the requirements have to be pinpoint accurate, allowing requirements engineers to follow the process over and over again until the stakeholders are satisfied.

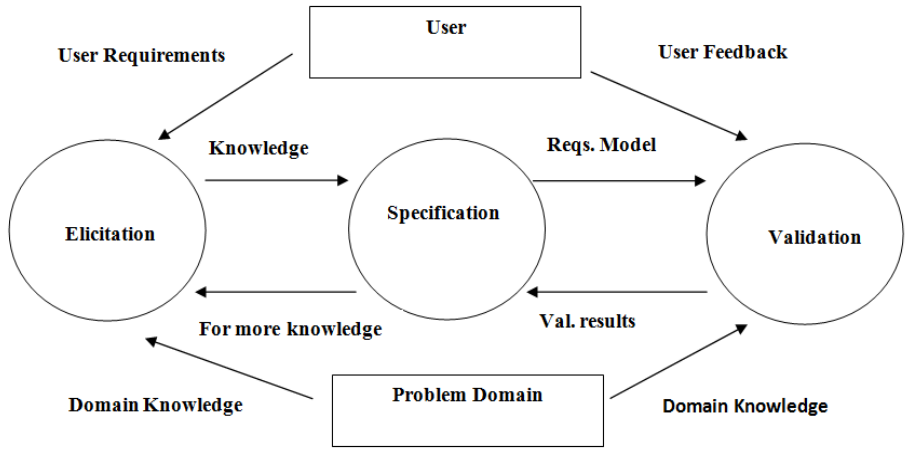

Figure 10: Linear iterative RE process (adapted from Shams-Ul-Arif et al. (2009))

4. **Iterative Requirements Engineering process**
   Compared to the linear RE process, this model has better version-by-version release support.
5. **Spiral model of Requirements Engineering**
   The process is performed in spirals, with each spiral representing a complete version of the product. The main focus of this model is handling risks, which the other models do not incorporate.

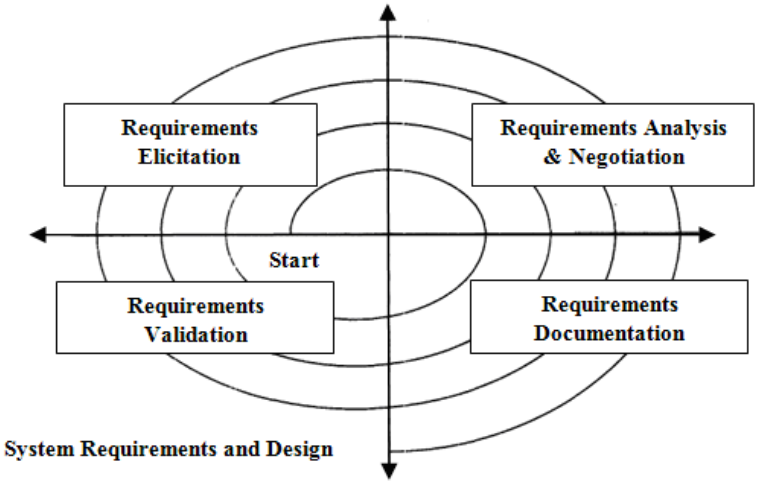

Figure 11: Spiral model of RE (adapted from Shams-Ul-Arif et al. (2009))

When we transform our own model of RE presented in paragraph 3.1.1 into an Agile process, it will look like the figure below. As you can see there is no strict order anymore and the differences in phases become blurry and are not present in each cycle per se. While the traditional one is typically seen once in a project, the Agile one can be seen many times.

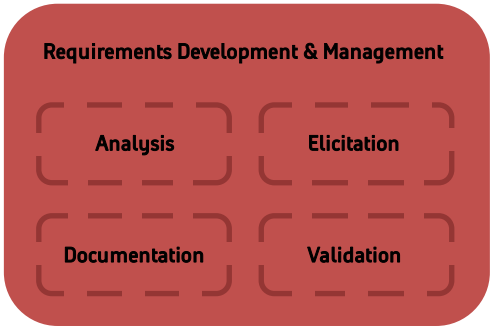

Figure 12: the Agile Requirements Engineering process

Cao & Ramesh (2008) conclude that pre-specified requirements are often not suitable for Agile software projects. Therefore they hand us seven 'best practices' for RE:

1. **Face-to-face communication over written specifications**
   Transfer requirements direct face-to-face, instead of documenting them first.
   Advantages: it saves time and the customer has direct input.
   Disadvantages: the customer has to have time available, and customers tend to distrust Agile methods when they are used to traditional methods.
2. **Iterative Requirements Engineering**
   Developing requirements during the implementation process (which is in line with the Agile philosophy).
   Advantages: good customer relations and requirements which are crystal clear because of direct contact with the customer.
   Disadvantages: bad cost/time predictions, bad overview due to a lack of documentation, and a bad focus on non-functional requirements.
3. **Requirement prioritization goes extreme**
   Prioritize requirements each cycle including other tasks like bug fixes, and the drive behind prioritizing is only added business value where costs and risks are more generally the focus in traditional methods.
   Advantages: the importance of each requirement is better visible and does not have to be frozen.
   Disadvantages: the system could become unstable because of constant re-prioritizing, and non-functional requirements are less important in the first phases (because the lack of added business value) which becomes a problem in the end phases.
4. **Managing requirements change through constant planning**
   Constantly re-adjusting the planning.
   Advantages: a dynamic project with respect to problems that arise.
   Disadvantages: adjusting the planning is not always enough, sometimes the whole process has to be redone.
5. **Prototyping**
   Building prototypes to get immediate feedback from the customer.
   Advantages: immediate feedback from all stakeholders.
   Disadvantages: bad scalability and robustness.
6. **Test-driven development**
   implementing tests before any features are implemented, which act as very precise requirements (with a high detail level).
   Advantages: documentation can be paired with existing test code, developers can try out different things because the tests give immediate feedback.
   Disadvantages: developers are not accustomed to writing tests first, a lot of details have to be known before developers can write useful tests.
7. **Use review meetings and acceptance tests**
   Create feedback moments after each phase using meetings and tests.
   Advantages: aside from the feedback, the customer will get a good overview of the current development situation which creates trust.
   Disadvantages: often it is hard to get every stakeholder in one room each cycle due to busy schedules.

Aside from acknowledging the fact that good requirements are of critical importance for software projects, and that Agile projects tend to throw some aspects of RE overboard, *Eberlein & do Prado Leite* also give four aspects to focus on to improve the use of RE in Agile projects:

1. **Customer interaction**
   Even inside end user groups there are often different viewpoints, which have to be identified.
2. **Analysis (validation and verification)**
   With Agile projects one can often find validation, but no verification.
3. **Non-functional requirements**
   Often underappreciated in projects because they do not add business value, but very important for the product as a whole.
4. **Managing change**
   Requirements Management has to be part of Agile projects.

4.2.7 Waterfall vs. Agile

The battle between traditional development methods and Agile development methods is an ongoing battle for many years. It will remain this way for many years to come. Both sides have their advantages and disadvantages.

For instance Agile methods can come in handy when the requirements cannot be developed beforehand according to *Rising & Janoff (2000)*. This situation makes the Waterfall method useless. Also because close customer relations is needed with Agile methods, the customer is more involved in the project as pointed out by *Paetsch et al. (2003).* This way the customer can steer the project more and is kept in de loop about the progress. *Cohen et al. (2004)* stated that Agile methods can improve the innovative process better than the traditional methods, because these have detailed planning with little room to be creative. If you look at it the other way around, this means that projects where a strict planning and control is needed, the Waterfall method is better suited than Agile methods. Examples can be projects in the military or at space agencies, projects where detailed requirements and architecture are very important, especially non-functional requirements like security.

Turk & Rumpe make six points where Agile does not have support for, making the Waterfall method the better one:

1. Limited support for distributed development environments
2. Limited support for subcontracting
3. Limited support for building reusable artifacts
4. Limited support for development involving large teams
5. Limited support for developing safety-critical software
6. Limited support for developing large, complex software

As mentioned before: there is no winner at this point, nor are there any signals that there will be in the future. So for now it is wise to follow *Boehm (2002)'s* vision, to find a good balance between the development methods. Companies have to consider for themselves which method to apply to which project, and maybe even apply a combination of both fields. There are no laws stating that we cannot go back one step in the Waterfall, or that we cannot do requirements before starting an Agile project.

4.3 Architecture in Requirements Engineering

Wupper (2012) states in his book, that clients nor builders are trained to make sure a system fits in its landscape. In the physical building world, the profession 'architect' is designed for this purpose. When looking at IT systems as digital buildings, we can say an IT architect (or information architect) is needed. In the times we live nowadays, companies have multiple IT systems. This makes it very important that the coherence between these systems is 'healthy'. Using the word healthy, one must think about things

like effective and efficient communication between systems, no systems that do more or less the same thing in different ways, or systems that fit in the organization with its people and processes. No proliferation of systems. If we look at architecture in RE, the demand could be that there must be at least some kind of sound architectural analysis when making the design, even if there is no architect available. Only then can companies keep a healthy IT landscape. This is of course also directly linked to the success of a project. A system that does not fit the existing architecture of an organization, cannot be seen as successful.

4.4 Business Rules in Requirements Engineering

Since the Heijmans IT department only develops and acquires applications for its own business, business rules should be a vital part of a functional design. Business rules can provide means for better communication between the people in the business, and the people at the IT department. Because often, those two 'kinds of people' speak different languages as it seems, or at least with different vocabularies. *Kapocius & Butleris (2006)* also stress that these miscommunications can deliver bad quality requirements. In their research they have developed a 'business rules driven information system requirements specification' method. This can support the requirements elicitation phase.

Business rules can also provide a knowledge database when developing. If we look at business rules as some sort of constraints, we can state that many software systems in companies like Heijmans have a great amount of constraints incorporated inside them. This makes those systems very complex. A developer cannot learn all those constraints by hart, so documenting business rules is a must for finishing IT projects successfully. According to *Gottesdiener (1999)* business rules should therefore be developed in synergy with Use Cases, to explain why a certain 'basic course of events' flows like that specific way the Use Case describes.

4.5 Lean

In this research the Lean philosophy will be used as a 'goggle' for looking at the RE practices. It will be used because Heijmans has recently adapted Lean for some specific building projects and it plays an important part in the future strategies of Heijmans.

Lean is derived from the Toyota Production System in the 1990's. The basic thought behind Lean is eliminating 'waste'. Everything in a process what does not add value for the customer is considered waste. Some examples of waste are stock (it just sits there waiting to be sold), transport and managers (they may not like it, but they do not directly create value, however they can enable others to create more value). *Poppendieck & Poppendieck (2003)* translated the seven manufacturing wastes, originally created by Shigeo Shingo, into seven wastes of Software Development. Below the list of the seven wastes of manufacturing with their corresponding Software Development wastes:

Manufacturing wastes	**Software Development wastes**
1. Inventory	partially done work
2. Extra processing	extra processes
3. Overproduction	extra features
4. Transportation	task switching
5. Waiting	waiting
6. Motion	motion
7. Defects	defects

Lean is a very interesting but dangerous philosophy, because while it is supposed to improve a process, it could also harm a process. We can compare applying Lean to improve a process with the euphemism of pruning a fruit tree. You have to cut some branches to enable it to produce bigger and better fruits. However, if you cut too many branches, you could achieve the opposite effect. The same could apply to the RE process.

Poppendieck & Poppendieck (2003) say everything except analysis (the first step of requirements in figure 4) and implementation is waste in the Waterfall method. But we all have to agree that, although documentation does not add value to the customer (there are exceptions), having no documentation whatsoever is impossible for most IT projects, if not all. So you have to be careful not to eliminate waste which will decrease the value added by the non-waste steps. Furthermore they show us one of the reasons why Agile development is that popular today. In Agile development the amount of time spend waiting on other people or partial products is far less than with Waterfall development, so the percentage of the process which adds value is much higher with Agile development. But read carefully, the percentage is higher, not necessarily the absolute value.

Donald Reinertsen is the author of three best-selling books on product development. In his first book he described how the principles of Just-in-Time manufacturing could be applied to product development, something that is now known as Lean development. So what Reinertsen tells us, is to focus on the 'why' instead of the 'what'. Why do we use this technique instead of that technique, or why should we document requirements? It is about using the toolbox delivered by theory, and applying what you need inside that toolbox on practice. In a nutshell he explains that something cannot be considered waist if you believe it adds value to the customer.

5. Analysis

5.1 Requirements Engineering Framework

Now let us examine the indicators found in theory. As mentioned before, these indicators will be our basis for the questions that will be asked in the interviews. Again we will look at the three areas: process, people and products.

5.1.1 The process

The traditional RE approach can be seen as a sound approach, but with the drawback of being a slow and inflexible process. The risks in this approach lie in the quality of the output of each step. The Agile RE approach cuts the time drawback from the traditional one. It also makes RE more flexible, however this can also result in a lot of changes. This is why *Rising & Janoff (2000)* and *Cohen et al. (2004)* conclude that Agile is the way to go when the requirements are vague in the beginning, small teams are available, and when innovation plays a large role in the project. When a strict approach is needed, the traditional Waterfall method is recommended. *Cao & Ramesh (2008)* add, that when there are quick changes in competition, stakeholder preferences, technological development and time pressure, pre-specified requirements are not suitable. In this situations the Agile approach is needed.
The last is of less importance in the Heijmans situation, because the changes in competition have less impact on the short term IT strategy, and the changes in technological development are captured by the suppliers. But in the interviews, we will try to find out what the criteria are for choosing a specific project method. Are the criteria based on the amount of requirements and how clear they are? Or are they based on the preferences of the supplier?

Paetsch et al. (2003) make a very important statement. In Agile projects, there is often very few documentation available. This makes a very important risk factor for project failure, which has to be considered beforehand. *Turk & Rumpe* learn us that Agile development in combination with large complex systems and distributed development teams (including suppliers), is not a particularly good idea.
In the interviews we will try to find out what the experiences are on this topic at Heijmans.

Eberlein & do Prado Leite emphasize two things. First, Requirements Management has to be part of the RE process, even in Agile projects. Second, not only validation (are the requirements what the user wants?) has to be a step in the RE process, but also verification (are the requirements of good quality?). This is a very valuable research topic for the interviews. With the ideal process blueprint in our mind, we will try to map the Heijmans situation on that blueprint, in order to see where the differences are and if these differences could be considered bottlenecks.

Indicators:

- The implementation of the RE process as described by *Paetsch et al. (2003)* focusing on requirements development and requirements management (looking at both Waterfall and Agile).
- Project planning and the place of RE in that planning with *Boehm (2002)'s* view on the planning spectrum in mind.
- Implementation of one of the five specific Agile RE processes described by *Shams-Ul-Arif et al. (2009).*
- The use of *Cao & Ramesh (2008)'s* and *Eberlein & do Prado Leite's* best practices for Agile RE.
- The Waterfall vs. Agile arguments with arguments by *Rising & Janoff (2000), Cohen et al. (2004), Cao & Ramesh (2008)* and *Turk & Rumpe*.
- The importance and position of architecture in RE.

5.1.2 The people

A factor which lies in all of the seven Agile RE best practices of *Cao & Ramesh (2008)* is the customer. Close contact and lots of feedback is a very important issue for Agile projects to be successful. Both *Paetsch et al. (2003)* and *Eberlein & do Prado Leite* agree to this. At Heijmans the customer is close by for the IT department, because they are co-workers. But does this make it easier?

Rising & Janoff (2000) think it is very wise to get the people who made the functional design involved in the implementation process. So the Requirements Development phase and the Requirements Management phase cannot have totally different people involved.
This research topic is closely related to the process part. Is there requirements management, and if so, who is responsible for this part? These will make crucial questions for the interviews.

Indicators:

- The importance of close communication and feedback as described by *Cao & Ramesh (2008), Paetsch et al. (2003)* and *Eberlein & do Prado Leite.*
- The interpretation of the different roles, as emphasized by *Rising & Janoff (2000)* and *Wupper (2012).*
- The importance of a stakeholder analysis according to *Wupper (2012).*

5.1.3 The products

Every requirement has to have a priority, according to *Rising & Janoff (2000)*. Priorities can change during the project, but you have to always classify requirements on importance. This makes sure the whole team has the same focus. *Daneva, van der Veen, Amrit, Ghaisas, Sikkel, Kumar, Ajmeri, Ramteerthkar & Wieringa (2013)* add that prioritizing is especially important when the project is a fixed price project. To improve quality of requirements, they should not only have a priority, but also be S.M.A.R.T. (Specific, Measurable, Attainable, Relevant and Time-bound) for instance. In the interviews and the case studies we will try to find out more about the overall quality of the requirements at Heijmans.

Eberlein & do Prado Leite state that non-functional requirements must not be forgotten. When forgotten, the consequence can be an instable or insecure system. Especially in Agile projects people tend to forget about non-functional requirements, as they do not deliver direct value to the customer.
This is an important potential factor for failing projects, so it will be interesting to find out how Heijmans handles non-functional requirements.
When we look at what has to be included in the products, *Gallardo-Valencia, Olivera & Sim (2007)* advise to make use of 'Use Cases'. These will reduce the ambiguity on the requirements and give a better overview on how the system will work in the end. *Daneva et al. (2013)* show us that delivery stories can be a valuable addition to user stories. Delivery stories can be seen as user stories on the implementation process, helping to assess the project's risks better. These are specific techniques. Does Heijmans use these techniques? And if not, why not?

Indicators:

- The availability of documentation in Agile projects as mentioned by *Paetsch et al. (2003).*
- The development of non-functional requirements as emphasized by *Eberlein & do Prado Leite.*
- The use of priorities for requirements as emphasized by *Rising & Janoff (2000)* and *Daneva et al. (2013).*
- The use of specific techniques as emphasized by *Gallardo-Valencia et al. (2007)* and *Daneva et al. (2013).*
- The use of business rules as recommended by *Kapocius & Butleris (2006)* and *Gottesdiener (1999).*

5.2 Interviews

To come to a complete objective view on the whole, interviews were conducted with people with different backgrounds, and different positions or tasks in the RE process. A total of eight interviews were conducted. For every interview there was a set of questions, that acted as guidelines. The questions can be found in the appendix. However, these guidelines were not followed strictly, to keep the informal character present in the interview. Depending on the answer follow-up questions were asked. These follow-up questions came forth from the answers the interviewees gave. Every interview was recorded to be able to analyze the answers at a later stage. Below is a list of the interviewed people and their roles in projects:

- Manager Business Information Alignment

  The manager Business Information Alignment is responsible for the whole BIA department. He reports directly to the Chief Information Officer of Heijmans. The base for the interview with him was that he has an excellent view on the overall success of the IT projects, and furthermore the possible causes for failure of those projects. We did this interview first to get a good impression of the situation and to get a better idea who to interview next.

- Project managers

  The project managers are in the lead in all the official IT projects at Heijmans. We interviewed several project managers for the same reasons as the manager of the BIA department, but this time for a more detailed view on the situation.

- Business consultants

  The business consultants are responsible for the requirements development phase in IT projects, together with people from the business. Interviewing them gave us some detailed information about what techniques are used in this phase and how the consultants grade their own work.

- Team Integration

  We also interviewed someone from team Integration. This team consists of programmers, for the projects where custom implementation is needed. As they get the requirements as input for their work, they delivered us meaningful information about the quality of the requirements.

- Information Manager

  We interviewed one information manager. The information manager is de the link between the business units and the IT department. They are responsible for ascertaining problems and needs from their business unit, that can be solved with IT. With this task they are also responsible for making project propositions. When a project has been giving a green light, they will take place in the control group of the project. The information manager provided us with some information such as how successful the product is for the business. Also the information manager provides the business consultants with a starting point, such as a problem description.

All the interviews are anonymized, to give the interviewees a chance to give their honest opinion about how well the Heijmans RE practices are designed and operating. Below a list of the meta data of the interviews:

Date	Interviewee	Location	Duration
21-10-2013	Manager BIA	Rosmalen	48 minutes
24-10-2013	Business Consultant	Rosmalen	54 minutes
25-10-2013	Project Manager	Rosmalen	42 minutes
27-11-2013	Project Manager	Rosmalen	33 minutes
29-11-2013	Information Manager	Rosmalen	38 minutes
29-11-2013	Project Manager	Rosmalen	54 minutes
11-12-2013	Team Integration	Rosmalen	37 minutes
23-12-2013	Business Consultant	Rosmalen	50 minutes

These interviews give us different viewpoints, different information, and above all different opinions on what could be improved at Heijmans. As mentioned in the method, the interviews were informal of nature. This had the effects we hoped to achieve with this approach. Interesting information was gathered that was not an answer to any of the questions, but does help us form a complete view on the situation at Heijmans. For instance, the expectations about what the input and output should be in the RE process, are not mentioned explicitly in any literature. By asking not predetermined follow-up questions, we found out that there are some mismatches in those expectations. Let us look at the most important topics that could indicate a source of problems or bottlenecks in the RE practices.

5.2.1 Expectations

The first thing we experienced in some interviews was the differences in expectations. One can expect a full blown functional and technical design, but if none has the capabilities to deliver such designs, one will be disappointed. Or when a full blown design was not possible because only vague requirements could be developed, one will also be disappointed. The following two quotes illustrate the mismatch in expectations best, the first one from someone in the implementation phase , the second one from someone in the design phase:

> *"In many cases there is no functional design and sometimes it is even constructed afterwards."*

> *"There is always a functional design, but not necessarily with the label functional design. The only exception is a project for an update."*

Then there is something we like to call 'expectation expectations'. One can expect that when the other person does not understand something, this person expects he or she has to ask questions to the first person to clarify things. But when the second person expects the first person to come to him or her to clarify things, there is a mismatch in expectation expectations. This results in two people of groups of people who just sit there waiting for each other to come to them for clarifications. Again from the perspective of the programmer:

> *"Often there is no or minimal documentation, due to lack of time in most cases. There have been differences in interpretation in past cases."*

And from the perspective of the requirements engineer:

> *"There is not a lot of ambiguity, I would have expected more questions in projects."*

Another interesting point made by one of the project managers, but not directly linked to RE practices:

> *"In most projects there is no real business case made beforehand and budgets appear to be incorrect sometimes."*

That said, there is also some kind of culture difference between the two parties, the two parties being the two sub departments of the IT department. One is really IT oriented in the sense of keeping systems up-and-running, while the other is more business oriented in the sense of new or improved systems to help the business progress to the next level, while they are both part of the same IT department.

5.2.2 Requirements management

From the interviews it appears requirements management is being neglected a bit at Heijmans. Of course there are examples that show us otherwise, but in most cases the business consultants have no role in the implementation process. The following quotes are characteristic for requirements management at Heijmans:

> *"In most projects there is no business consultant involved in the implementation process."*

> *"Requirements management is simply not present at Heijmans."*

> *"We have no role in the implementation process, there could be a bigger role for us there."*

> *"I would like to have a business consultant involved in the implementation process, because we would have the creator of the functional design present."*

Any changes made in the requirements are done in discussion with the control group, but they are not documented. Also the actual product is rarely checked with the functional design. As one of the project managers explained:

> *"Sometimes the product is different from the functional design or the project proposition. The documents are being edited in the end most of the time, to be in line with the actual product."*

We can conclude almost all interviewees also agreed requirements management should be given more attention, and business consultants should be involved more in the implementation process.

5.2.3 Business Rules

In companies like Heijmans business rules are very important and useful. However very few companies are actively specifying business rules and documenting them. This also appears to be the case at Heijmans. Although interviews pointed out they are being discussed when trying to establish requirements, they are rarely being formalized and documented. At most, they are used to construct work flow diagrams. As one of the business consultants explained:

> *"Yes, they are being discussed in the design phase, but they are rarely documented using a specific method."*

5.2.4 Techniques used in a functional design

When constructing a functional design, very few specific techniques are used. The requirements are always categorized in order of priority, and in most cases people try to specify the requirements according to the S.M.A.R.T. standard, but as a project manager told us:

> *"Not in all projects the requirements are conform S.M.A.R.T. and sometimes we come to the conclusion it would have been more useful to use the S.M.A.R.T. method when a project is finished."*

As a result of not always using the S.M.A.R.T. method, the requirements are lacking detail sometimes:

> *"The details of the requirements are not very high. Also the scope is sometimes a bit vague."*

Here we are talking about functional requirements. Non-functional requirements are not specified in most projects. The following quotes are from an information manager and project manager:

> *"They (non-functional requirements) are being considered most of the time in consultation with the architects or the security officer, but there is no real focus on them. Almost no non-functional requirements are being written down."*

> *"Documenting non-functional requirements is something we can improve. It is known to bring problems in the implementation phase when they are not written down."*

Also specific techniques like use cases or UML(-like) diagrams are not used very often. Subjects like architecture have just started to appear in functional designs in recent years, but are still seen as less important, as can also be concluded from the previous quotes. Furthermore testing is just started to be more professionalized at Heijmans. Basic functional test scripts are still absent in functional designs. And last, a stakeholder analysis is absent is most cases, although it was mentioned some of the information managers do a stakeholder analysis.

5.2.5 Project methods

Actually there are three types of projects at Heijmans, which are divided in two categories. First we have the most common project method, which is the Waterfall method. Second we have a method which is used in recent projects more often, the Agile method. These are the official projects, but then we have the unofficial projects, this third kind follows the Waterfall method more or less. Many of the interviewees had their doubts about if the Agile method is suitable for IT projects at Heijmans.

> *"The business finds it hard to accept that not all requirements are implemented in the end when the proverbial 'jar with money' is empty. Actually the jar should be variable, but Heijmans is used to 'fixed price' projects, as is the case in construction projects."*

Said one of the project managers. So some people in the business units find it hard to comprehend when the budget is used and not all requirements are implemented. In most of the Agile projects, it was the supplier who chose this project method.

5.2.6 Responsibilities and experts

As mentioned earlier, topics like testing, architecture and security are becoming more mature at Heijmans. The question that remains is who is responsible or who is accountable for specifying this topic in design documents. One of the business consultants told us:

> *"Still too often there is discussion about who has to write about a specific topic in the functional design."*

Again the business consultants might say: "The architects, the testing manager and the security officer are the experts on these areas, so they have to specify these topics in the functional design.". On the other hand, the experts might say: "The business consultants are responsible for the functional design, so they have to specify these topics and they can consult with us about the content.". These topics could also have separate documents in the design phase.

Furthermore, the people responsible for customizations would like to be involved in an earlier stage, say the design phase, when work has to be done by them. Also it appears in some projects people would

have wanted a business consultant involved, while there was no consultant involved. Someone from team Integration mentioned an example:

> *"For some system we were developing a customization. We were done implementing and presented the product to the users. But then they told us out of the blue, that apparently there was a second system that had some links with the first one, that made the developed customization unusable in practice. We then had to start all over again. This wouldn't have happened if a business consultant was involved in the project and would've analyzed their way of working and what systems are used."*

5.2.7 Uniformity

Uniformity helps keep the expectations realistic, templates for instance help guard uniformity. In the interviews we found out that the existing templates are not used very often, or are used wrong. Uniformity helps keep ambiguity as low as possible. It also makes it easier to compare different projects, different performances with each other. One of the business consultants told us:

> *"The templates are always the basis for the documentation."*

This could very well be the case, but the information manager told us the following:

> *"There is often ambiguity in the documents, but this however corrected in most cases by working together very closely."*

From earlier discoveries we can conclude that the above is not possible in all projects, because in many projects requirements management is not part of the requirements practices. One of the project managers also had the following to say:

> *"There is many differences between the delivered documents of the information managers. Sometimes it is really summarily, which leaves many question marks. But sometimes every detail is worked out entirely, which is also wrong, because this is not their job."*

This lets us believe the input for the requirements process is also not always of good quality.

5.3 Case Studies

Besides the interviews we also looked at two cases of real finished IT projects at Heijmans and we studied some templates that are used in IT projects. In consultation with the project managers, we chose two projects. The first was considered successful, the second can be considered "less successful". Saying this we do not mean the second project failed miserably, but the result could have been more satisfactory. One very important note that has to be made, is the fact that both projects did not follow the official 'Heijmans IT projects' route entirely or for some parts. This is not unusual at Heijmans, because some projects are so small, they may only take a few days from start to finish for instance. When a project does not follow the official route, it is likely there is no business consultant involved, and sometimes the project manager is only involved in a later stage. All available documentation about these projects was provided by the project managers. It contained emails, a presentation and text documents. The focus was on the documents which could be considered functional or technical design documents.

5.3.1 Project 1

Because accidents happen more often than wanted in the construction industry, Heijmans has started a project to register dangerous situations on projects that could lead to accidents. Because the current method is not accessible enough for employees to register dangerous situations, the IT department received an assignment to develop a mobile phone app where employees can easily register dangerous situations. Because the entire application has to be developed, one should expect a business consultant involved. Also a proper design (both functional and technical) should be available.

There is good documentation available about the plan of action and the functional design. In this project there was no business consultant involved, which is the first thing that raises questions. Was the person who created the design trained to do so? Let us start with the plan of action.

The plan of action gives a clear explanation of the problem and the objective. There is no description why the objective is a proper solution for the problem, but then again, this could also be placed in the functional design document. The scope is demarcated well, describing not only what is part of the project, but also what is not. Critical issues are listed, but a more analyzed section for this should benefit projects like this. Furthermore the document contains the typical chapters like 'milestones', 'project responsibilities', 'a planning of the activities' and 'a financial overview'.

As mentioned before, we would expect an analysis of the chosen solution for the described problem. This is clearly absent in the functional design. Is an mobile app the best solution? Does everybody on the project locations have a mobile phone or other mobile device which can access this application? These are questions which are crucial for the success of this product, but there is no trace that someone has analyzed this. On the first page we find an architecture map. This is a good thing, although it is maybe a bit simplistic, missing crucial information about 'what' is communicated between the different systems and 'how' this is done. Next we find the 'global actions', which is some kind of 'basic course of events'. This is something where use cases are especially beneficial for clarification as mentioned by *Gallardo-Valencia et al. (2007)*. Unfortunately a free format is chosen. Last is the section with requirements. Functional requirements and non-functional requirements are ransacked in one chunk of information, not using priority or the S.M.A.R.T. criteria. The requirements do describe how the application should work and how it should look like, but we can only imagine many questions because of ambiguity and vagueness in requirements like these:

- "The app has a help function"
- "The app is user friendly and intuitive in its use"
- "... an email is send to the reporting person, stating that his registration of a dangerous situation is being treated by person X after Y days etc."

We cannot help but think that a developer would put his hands on his head and stare at the document for some time, not knowing where to start asking questions. Our fear of not having the right person construct this design document may have been well-grounded. Nevertheless Heijmans considers this project to be a success.

5.3.2 Project 2

Project 2 was about changing an existing application for 'leave applications'. The changes were necessary for new country policies, and creating uniformity in leave applications systems across the Heijmans business units. Again we would expect a decent design and maybe even some kind of architectural analysis to map the different leave applications in the Heijmans application landscape. We would expect a business consultant involved in the project and maybe even a consultant with expertise about the country policies.

For this project there was some kind of functional design available, although it cannot be considered an official document. In this functional design, which is constructed by a business consultant, the background of the project is described and there is a clear problem statement. There is also a clear description of the deliverables present. Concrete requirements are not noted however, and although there was a scope described, it was very minimal. It just mentions the deadline, the objective and the departments the project will affect.

Deliverables show exactly what everybody has to deliver in the project. However, the interpretation of these deliverables is left blank. This will not create problems necessarily, but it does increase the risk of ambiguity. Some examples of deliverables in the project:

- "Create usability trainings"
- "Communicate the changes"
- "Create the report with booked/withdrawn hours"
- "Create admin accounts"
- "Give users access to the application"

Again, non-functional requirements, use cases, or UML diagrams are absent.

Although the mission was clear in this project, the content of the mission or the means to reach the goal could be considered a bit vague to some people. It is not clear if clear requirements were not documented, or simply not there. It is also unknown if requirements for 'uniform leave applications systems' at Heijmans are in existence.

5.3.3 Templates

For this part of the case studies we looked at templates for 'program of demands', 'functional design', and 'project proposition'.

First of all the project proposition template. This document is very complete. There is a complete project definition present, with descriptions of the goal, the scope, the deliverables and a stakeholder analysis. There is also a section for the possible solutions and a section for the project structure.

For the program of demands there are several templates available, with different sections. It is not clear which template is used in which situation. There are sections for the scope, the stake holders and the current process. Furthermore there are chapters for the demands on the following areas: functional, technical, implementation and administration. Then there is a template for the comparison of several options, and/or suppliers. Last there is another template for the program of demands. This one differs from the first one. The demands are categorized on the following areas: functional, technical and supplier. Also there are less guidelines for what has to be present in the introducing chapters.

What is absent in all templates for this category are the non-functional requirements, like security or usability. Also there is no section for architecture and no specific sections for UML-like models and use cases.

Last we looked at the template for a functional design. This one shows similarities with the first template for a program of demands, which should be seen as a good thing, since this reduces the chance for ambiguity. Again the scope, stake holder analysis and context sections are present. Although it is not clear how thorough the stake holder analysis is done, since the accompanying text lets us believe this is only done very marginal. The chapter on functional requirements looks very complete. There are examples of UML diagrams and use cases present. Also there is a paragraph for testing on functionality. After the chapter on functionality comes the chapter about entities, or information handling if you will. This is the last chapter in the template, leaving again no room for non-functional requirements or a chapter on architecture.

The templates are almost complete, but are missing some small topics like mentioned.

6. Results

6.1 Requirements Engineering at Heijmans

Let us start by stating that we concentrated our research on the aspects that go wrong or could go better in the RE practices at Heijmans. Naturally most things do go right, because the result at the end of most projects is satisfactory. For instance the amount of hours spend on the RE activities appear to be in line with the 25% that *Eberlein & do Prado Leite* mention and we believe there are enough experts for the different roles in IT projects present in the organization. However, in the context of Lean development, the turnaround could and should be reduced. We believe this is something that is already in the minds of the business consultants and we expect that improvements will be made on this topic. Furthermore Heijmans has embraced the concepts of software architecture, digital security and software test management. This shows there is a sound basis for these aspects in IT projects. And let us not forget that most products from the IT projects at Heijmans, do represent a solution to the problems or needs that were the occasion of those projects.

Then again, one would say nothing is perfect. So let us get back on what could be improved. As mentioned in the previous chapter we can distinguish seven topics where improvements can be made:

1. Expectations
2. Requirements management
3. Business rules
4. Techniques
5. Project methods
6. Responsibilities and experts
7. Uniformity

Expectations could be considered in a broader way, including culture management (i.e. differences in mentality between the two IT sub departments), but this is not in the scope of this research. The same is the case for project methods. We have focused in the role of RE in the different project methods, but because Agile is relatively new for Heijmans, there might also be interesting improvements possible on the project setup of the Agile projects, for instance training people on the SCRUM method. Again, this was not in the scope of this research.

6.2 Improvements on Requirements Engineering at Heijmans

Let us use the key areas introduced in chapter 4 to introduce possible improvements on RE practices at Heijmans.

6.2.1 The process

The RE process at Heijmans is lacking one most important thing: requirements management. This could be implemented by getting the business consultants, who created the functional design, involved in the implementing phase. In this phase they should be responsible for:

- Guarding the requirements, making sure the product is conform the requirements
- Updating the functional design when something changes, when priority changes or a requirement changes
- Making sure ambiguity is minimized

Furthermore there is very little documentation in some projects if not none. Transferring knowledge in face-to-face talks is great, but having documentation as some sort of spine of the project helps reduce the risks of unclear requirements, even in Agile projects.

Also Heijmans should consider how the RE process in Agile methods should look like. Although requirements can be vague beforehand, it is still wise to create a functional design with high detail, to make sure there is a sound basis for a project to start.

6.2.2 The people

Expectations and responsibilities, they are intertwined. Responsibilities give expectations. An architect is responsible for the coherence of the application landscape and everything that has a link to that, so also new projects. A business consultant is responsible for the functional design. The architectural consistency should be part of the design. Responsibilities have an overlap here. Heijmans should clarify which of the two is responsible for that overlap and this should be one person only, where the other can have the consulting role.

The other way around expectations might give responsibilities, but this is not always the case in companies. Especially in bureaucratic companies, where everyone has a specific set of tasks. Again, if a task comes up that cannot be allocated to an existing set of tasks (and a person with that set of tasks), Heijmans should locate that task to an existing set, or create a new job with a new set of tasks (as they have recently done with test management).

Above all, Heijmans should make sure all expectations are realistic and in line with each other. This is no simple task indeed, but a workshop between programmer and requirements engineer about what the needs are, what can be delivered and how a gap in between can be filled should provide a start.

6.2.3 The products

The biggest chances for improvement lie in the process and the people. Still there are improvements possible that can boost quality of the products. Business rules can improve knowledge retention, not only inside projects, but also for future projects and when looking at the past. Formalizing and documenting them is key herein.

Then there is uniformity. Keep evaluating the templates and always use them as a basis. Do not remove chapter you do not need, instead show readers that you handled the topic and deemed it not applicable or not important for that specific project, and most important: 'why?'.

Last we have the used techniques in the RE development phase and in the functional designs. It is very difficult to say Heijmans should use this technique or that technique. It is very dependent on the type of project, and at Heijmans every project is different in some way. *Gallardo-Valencia et al. (2007)* state that the use of use cases is an absolute must, but one can imagine that in some situations something like a flow chart can also be suited. What we can recommend is the use of a priority on all requirements and the use of the S.M.A.R.T. method at all times. Furthermore non-functional requirements should always be considered (in consultation with the expert) and documented beforehand and not afterwards.

7. Conclusion

In this research we have tried to construct an advice for Heijmans, on how to improve their Requirements Engineering practices. To come to this conclusion we started with creating a sound basis for assessing the situation at Heijmans, by doing a literature study on known bottlenecks in the three areas: process, people and products. Many different indicators for problems were found. From the importance of good requirements management, to the use of clarifying methods like the use of use cases. In particular the love-hate relationship between requirements and Agile methods was examined. It is clear they can go hand in hand, as long as you find the proper place for requirements in those Agile methods. With the foundation from the literature, the next step in the research was mapping the Heijmans situation on the known indicators. This was done by conducting interviews with different people who come in contact with the RE process in any way, and by analyzing two real projects from the past and the templates used for the RE products. This provided us with interesting results that are in line with the literature, but also results that were not encountered in the literature. In all of the three key areas improvements can be made at Heijmans.

In the process it is clear that requirements management has to get a more important role. There is no literature to be found that states that requirements management can be skipped or deems it less important. All papers state the opposite, that requirements management is just as important as requirements development, maybe even more important. Furthermore documentation should be constructed more often and more precisely, even in Agile projects, to provide all projects and face-to-face communication with a solid backbone.

The proposed improvements for the people (or more general: the roles) are the least tangible. Heijmans should further crystallize the tasks each role or expert has in the RE process and who is responsible for which product or sub product. This will help keeping expectations realistic and expectations management will not only help keep the RE practices healthy, it will also keep synergy in the whole department in the end.

For the products there are also improvements possible, although they will have less impact than those proposed before, but this might make them easier to implement. The opportunities herein lie in living up to using the S.M.A.R.T. standard for every requirement (official project or not), formalizing business rules and documenting them for future use (both in the project as outside), always providing information about non-functional requirements (like security, durability, usability, etc.) or at least mentioning they are not considered important and last trying to maintain uniformity in all products. The last could also apply to products other than RE products, like a plan of action or a project proposal.

The RE practices at Heijmans can in no way be called ‘bad’. Yet with implementing the proposed improvements, Heijmans can further professionalize its IT projects, to maximize the amount of successful projects.

8. Additional findings

Because we did some research on the different development methods, to examine the place of RE in those methods, we can also state some findings about the methods per se. We have to emphasize this is not an answer to any of the research questions and thus not part of the original research, however it would be a pity not to mention these findings. Heijmans has not yet developed a prescription when to use the Waterfall method and when to use an Agile method. In this chapter we will provide Heijmans with some handles to use when choosing one of the project methods for some project.

We have discussed the up- and downsides of the different project methods. Waterfall is slow and inflexible, but it is a safe method where details can be elaborated in depth. Agile is fast, flexible and improves innovatism, but the scope can go in all directions if not maintained correct and the product might not be what you needed beforehand. Let us sum up what we found out in the literature used for the original research.

8.1 Strict Waterfall method

You can (not per se should) choose for the strict Waterfall approach when:

- There is a clear problem or need, that is analyzed in details.
- There is a clear view on what the solution should be.
- A great amount of clear requirements can be formulated.
- Non-functional requirements are of crucial importance for the success of the product.
- There is no fixed deadline set.
  (a fixed deadline herein is for instance when legislation changes)
- It is important to get the product right in one go.
- Enough people with the right expertise are available.
  (i.e. a requirements engineer for the design phase, a programmer for the implementation phase and a project manager for the whole project)

8.2 Strict Agile methods

You can (not per se should) choose for the Agile approach when:

- The problem or need is analyzed and proven vague.
- There is no clear view on what the solution might be.
- Very few requirements can be formulated and they are quite vague.
- There is a fixed deadline set, the product has to be ready before that deadline with no exceptions.
- Relatively few people are available for the project and these people have a skill set that suits multiple roles in the project. (i.e. project manager is also the designer or programmer, or the designer is also the programmer)
- Innovation is an important factor in the project.
- The sprint team can meet on a regular basis.
  (when the SCRUM method is chosen on a daily basis)
- The customer is available for most of the meetings or a representative as a product owner.

8.3 An hybrid method

The previous paragraphs clearly build up to this third method. As mentioned before in the paragraph Waterfall vs. Agile, you can also use an hybrid solution. *Boehm (2002)'s* vision already shows that a combination of the two is possible and maybe even preferred. The following aspects must be present in any way when choosing this format:

- Problem analysis has to be done before the first sprint.
  (however, this should also be the case in a regular Agile method in our opinion)
- The first version of the design documents have to be developed before the first sprint, containing as much details as possible.
- Documents should be kept up-to-date during the different sprints.
- A requirements owner should be appointed as the responsible person for the requirements during the sprints, this could be the product owner.
  (this person is typically the person who created the first requirements)
- The requirements are always the basis for the product, so if the product has to change in a sprint after the first one, the requirements have to change first.
  (updating the documents can be done simultaneously to cut waiting times)
- Even if there is only one sprint, regular meetings are still recommended.

Besides the handles listed above, there are many more do's and don'ts thinkable. Experimenting to improve this methodology is key for successfully applying this hybrid method. In short, you could look at it as multiple iterations of Waterfall routes (sort of sprints), with the important parts of the Agile method that make that method flexible and efficient, incorporated into those iterations.

9. Discussion

As with any research, it is wise to take a critical look at the methodologies used and the results. Let us start with the literature study. Thanks to the literature study of Chris Kleine Staarman we had a good list of papers that formed the basis of this literature study. However, it was difficult to find useful papers on specific RE topics, like for instance business rules. Still, we believe the current literature basis is strong enough to uphold this research.

As mentioned several times before, we chose an informal setup for the interviews, because this suited the situation best. This was possible because the interviewer already worked together with his interviewees. The downside to this were the difficulties transcribing the interviews. A lot of good information was provided that was not an answer to any of the questions, so this was especially hard to transcribe. But in the end this extra information provided us with interesting insights, that lead to recommendations for improvements. A seconds criticism on the interviews is the number of people we interviewed. More people from certain roles, like business consultant, or people from the implementation phase could have been interviewed, giving even more evidence for the conclusions from the interviews. We feel however, that more interviews would not change the conclusions in the end.

One critical issue for the conclusions are the unofficial projects. Because these projects have a different setup every time and cannot be compared to the projects that are part of the official project portfolio route. So the proposed improvements mainly apply on the official projects, but we can state that Heijmans should try to critically assess if a project can or should be started outside the project portfolio loop.

The case studies are the prove of concept. There are definitely relationships with the conclusions from the interviews. But because the literature study and the interviews were the most important part of the research, less time has been spend on the case studies. First of all it was difficult to choose two projects which lend themselves for an analysis that could contribute to this research. Especially in the second project there was not much documentation, which made it even harder to analyze the RE practices in those projects. If Heijmans were to further analyze projects in the past, conducting interviews to support the available data would be a good thing.

Overall the research could have used more evidence to support the conclusions, but in the end, the conclusions would not have changed if more researching was done.

10. Literature

Boehm B. (2002); Get Ready for Agile Methods, with Care; *IEEE 02;* 64-69

Cao L. & Ramesh B. (2008); Agile Requirements Engineering Practices: An Emperical Study; *IEEE 08;* 60-67

Cohen D., Lindvall M. & Costa P. (2004); An Introduction to Agile Methods; *Advances in Computers Elsevier 62*

Daneva M., van der Veen E., Amrit C., Ghaisas S., Sikkel K., Kumar R., Ajmeri N., Ramteerthkar U. & Wieringa R. (2013); Agile requirements prioritization in large-scale outsourced system projects: An emperical study; *The Journal of Systems and Software Elsevier 86;* 1333-1353

Eberlein A. & do Prado Leite J.C.S.; Agile Requirements Definition: A View from Requirements Engineering

Gallardo-Valencia R.E., Olivera V. & Sim S.E. (2007); Are Use Cases Beneficial for Developers Using Agile Requirements? (research presentation slides)

Gottesdiener E. (1999); Capturing Business Rules; *Software Development Magazine: Management Forum 7, 12*

Kapocius K. & Butleris R. (2006); Repository for Business Rules Based IS Requirements; *Informatica 17, 4*; 503-518

Kwak Y.H., Anbari F.T. (2004); Benefits, obstacles, and future of six sigma approach; *Technovation Elsevier*; 1-8

Paetsch F., Eberlein A. & Maurer F. (2003); Requirements Engineering and Agile Software Development; IEEE 03

Poppendieck M. & Poppendieck T. (2003); *Lean Software Development: An Agile Toolkit*; Addison Wesley

Rising L. & Janoff N.S. (2000); The Scrum Software Development Process for Small Teams; *IEEE 00*; 26-32

Saiedian H. & Dale R. (2000); Requirements engineering: making the connection between the software developer and customer; *Information and Software Technology Elsevier 42*; 419-428

Shams-Ul-Arif, Qadeem Khan & Gahyyur (2009-2010); Requirements Engineering Processes, Tools/Technologies, & Methodologies; International journal of Reviews in Computing; 41-56

Szalvay V. (2004); An Introduction to Agile Software Development; Danube Technologies, Inc.

Turk T. & Rumpe B.; Limitations of Agile Software Processes

Wupper H. (2012); *Architecture for the digital world!*; *epubli GmbH*

10.1 Background information sources

http://en.wikipedia.org/wiki/Six_sigma

http://en.wikipedia.org/wiki/Lean_manufacturing

http://en.wikipedia.org/wiki/Agile_software_development

http://en.wikipedia.org/wiki/Waterfall_software_development

http://en.wikipedia.org/wiki/SMART_criteria

http://reinertsenassociates.com/about/

http://business901.podbean.com/2010/10/19/creating-flow-with-don-reinertsen/ (podcast)

11. Appendix

Interview questions for the business consultants

1. Can you please describe how the process of requirements development works at Heijmans and what the role of business consultant is herein?
2. Is there a business consultant involved at every project?
3. Is a functional design constructed in every project?
4. Are business rules being used and documented besides regular requirements?
5. How frequent are the priorities of requirements being adjusted in projects?
6. Do you think it is important to construct an architectural design in every project?
7. Are requirements always developed according to the S.M.A.R.T. methodology?
8. Are requirements always given a priority, like for instance the MoSCoW methodology?
9. Are non-functional requirements always documented?
10. What is your role in requirements management?
11. What is the average percentage of requirements that is actually implemented? (note: only the demands, not the wishes)
12. Are questions being asked frequently about the requirements from someone in the implementation process?
13. Are business consultants being consulted when choosing for a project method?
14. What is the average time spend developing requirements in a project?
15. Do you use Use Cases in a functional design?
16. Do you use User Stories in a functional design?
17. Do you use scenarios in a functional design?
18. Do you use UML diagrams in a functional design?
19. Do you think the Lean methodology can improve the requirements development process and how? (note: the business consultants all had training in the Lean methodology)
20. Do you always use a template as the basis for a functional design and how precisely do you follow the chapter division?
21. Do you always perform a stake holder analysis?
22. Is a test plan part of the functional design?
23. Is the product always tested on functionality with the actual requirements?

Interview questions for the information manager

1. Can you please describe the process from IT problem or need to actual project and the role of information manager in this process at Heijmans?
2. How frequent are the priorities of requirements being adjusted in projects?
3. Is there cooperation with the IT architects when developing the project proposition?
4. Are non-functional requirements being considered when developing the project proposition?
5. Would you prefer involving business consultants in more projects?
6. Would you prefer involving business consultants in more aspects in projects?
7. What is the average percentage of requirements that is actually implemented?
   (note: only the demands, not the wishes)
8. How many times has the conclusion been made that another project would have been more suitable for the project than the method chosen?
9. What is the average time spend developing requirements in a project?
10. Are questions being asked frequently or are there ambiguities about the requirements from someone in the implementation process?
11. How good is the uniformity when it comes to the design documentation?
12. Is the product always tested on functionality with the actual requirements?
13. Is the product in most cases a solution to the problem or the need of the business, or is it what the business wanted to have?
14. Do you always use a template as the basis for a project proposition and how precisely do you follow the chapter division?
15. Do you always perform a stake holder analysis?
16. Who determines which people get a role in the project?
17. What is the percentage of projects that can be seen as successful?

Interview questions for the project manager

1. Can you please describe the process of project management at Heijmans?
2. How frequent are the priorities of requirements being adjusted in projects?
3. How frequent do architectural problems come up in the implementation phase?
4. Are requirements always developed according to the S.M.A.R.T. methodology?
5. Are non-functional requirements always developed and documented?
6. What is the average percentage of requirements that is actually implemented?
   (note: only the demands, not the wishes)
7. Do you consult business consultants when choosing for a project method?
8. What is the average time spend developing requirements in a project?
9. Are questions being asked frequently or are there ambiguities about the requirements from someone in the implementation process?
10. How good is the uniformity when it comes to the design documentation?
11. Do you find it difficult to maintain the budgets when the requirements were vague in the beginning (i.e. in Agile projects)?
12. Is the product always tested on functionality with the actual requirements?
13. Is the product in most cases a solution to the problem or the need of the business, or is it what the business wanted to have?

Interview questions for the team integration

1. Can you please describe at what stage you are involved in projects and what your role is in projects at Heijmans?
2. At what type of projects are you involved?
3. Is a technical design always developed?
4. To what extent is the functional design good input for the technical design?
5. Do you consult with the author of the functional design often?
   (note: in most cases this is the business consultant)
6. Is ambiguity a heavy issue in functional design documents?
7. Are functional design documents always uniform?
8. Are functional design documents constructed from the perspective of the problem/need or from the perspective of the solution?
9. Do you always use a template as the basis for a project proposition and how precisely do you follow the chapter division?
10. To what extent do people analyze what is possible and what is not on a technical area when constructing a functional design?
11. How often do problems appear in the implementation process because of ambiguity or inaccuracies in the functional design?

Functional specification for the safety app (Dutch)

Versie: 4

Datum: 17-12-2013

De voorziene oplossing bestaat uit 4 componenten

1. Twee frame app's
2. Een responsive website
3. Een webservice die de koppeling vormt tussen de website en het ... systeem
4. Aanpassingen in de reeds bestaande ... ongevallen applicatie in ...

Algemene uitgangspunten

- De website is responsive en werkt vanaf verschillende aparaten: PC, tablets, smartfoons.
- De website werk vanuit de gebruikelijke browsers: IE, Chrome, Safari, Firefox
- Afhankelijk van de mogelijkheden van de toestellen (hardware), de versie van het besturingssysteem en van de gebruikte browser zal bepaalde functionaliteit wel of niet of in beperkte mate beschikbaar zijn. Er wordt naar gestreefd een zo breed mogelijk publiek te bedienen, binnen de technische mogelijkheden
- Beveiliging is op basis van HTTPS verkeer
- De applicatie moet snel werken ook bij beperkte datalijn capaciteit. Extra aandacht voor het comprimeren en verkleinen van de te versturen data.
-

Overall architectuur

ARCHITECTURE PICTURE

Globlale werking

1. Een nieuwe gebruiker gaat via zijn browser naar de GO-website en meldt zich aan als gebruiker. Na het ingeven van zijn gebruikersgegevens en bevestigen van de aanvraag in de bevestigingsmail is het systeem klaar voor gebruik.

   NB: IOS en Android gebruikers kunnen de Go-app ook eerst downloaden uit de betreffende appstore. Na openen van de app komt de gebruiker dan op de GO-website.

2. De gebruiker maakt een melding aan en stuurt deze op

3. De website controleert de melding op volledigheid, accepteert deze en stuurt deze door naar de Heijmans webservice.

4. De webservice zet de ontvangen melding door naar de ... ... Database.

5. In ... wordt de workflow gestart die zorgt dat de melding opgepakt, behandeld en afgewerkt wordt.

6. Vanuit ... worden de aanmelder, de verantwoordelijke en de website per mail geïnformeerd over de status voortgang van de melding.

Detail werking per onderdeel

Onderstaand worden per component de gewenste functionaliteiten beschreven voor zover die nu bekend zijn. Aangezien dit voor zowel Heijmans als ... nieuwe ontwikkelingen zijn zal rekening gehouden moeten worden met voortschrijdend inzicht.

1. De Apps

- De Apps zijn frame-apps wat betekent dat de apps niet zelfstandig werken maar primair links zijn naar de GO-website.
- De apps werken dus in principe alleen als er internet verbinding is met de GO-Website. Wel is het zo dat als de app al openstaat er een melding gemaakt kan worden. Alleen het versturen werkt dan niet en moet gebeuren als er weer verbinding is. Dit betekent ook dat als de verbinding tijdens het registeren van de melding tijdelijk wegvaklt er wel gewoon doorgegeaan kan worden met de registratie.
- In verband met performance voordelen is er voor gekozen de apps wel te voorzien van foto verkleiningsfunctionaliteit. Dit betekent dat de te versturen foto's client side (op de telefoon) proportioneel verkleind worden naar een vooraf gekozen maximum formaat. Hiermee wordt voorkomen dat er grote bestanden verstuurd worden.
- De App heeft het Heijmans GO label
- De App is beschikbaar voor IOS en Android en kan zowel uit de betreffende appstore worden gedownload als vanaf de GO-website.
- De app werkt gebruikersvriendelijk en intuïtief waarbij de app de website zo veel mogelijk als App presenteert inclusief de gebruikelijke controls zoals die voor het betreffende besturingssysteem gelden (IOS look, android look).
- De app heeft een helpfunctie

2. De GO-website

ALGEMEEN

- De GO-website is een responsive website. Dit betekent dat website zich zo veel mogelijk aanpast aan de mogelijkheden van:
  - het apparaat (PC, tablet, smartfone),
  - het besturingssysteem (IOS, Android,..) en
  - de browser (IE, Firefox, Safari, Chrome).

  Ook de gebruikte versies van het besturingsysteem en de browser zijn van invloed op de mogelijkheden. Waarbij geldt dat de nieuwste versies over het algemeen de meeste functionaliteit ondersteunen en grafisch de beste mogelijkheden bieden.

  Met bovenstaande wordt bereikt dat een zo breed mogelijk publiek wordt bereik met waar mogelijk zoveel mogelijk functionaliteit. Hierbij worden natuurlijk wel de grenzen van het redelijke in ogenschouw gehouden. Niet elke uitzondering zal afgedekt kunnen worden en een aantal zaken zal ook lopende het project uitgevonden / ontdekt moeten worden.

AANMELDEN / INLOGGEN

- Als een gebruiker de website voor de eerste keer bezoekt wordt hij gevraagd zich aan te melden. Na registratie van een aantal "persoonlijke instellingen" als naam, e-mail, 06-nummer en relatie tot heijmans) ontvangt hij een bevestigingsmail met inlogcode, wachtwoord en bevestigingslink.
- Na het klikken op de bevestigingslink komt de gebruiker op een inlogscherm waar hij met inlogcode (zijn e-mail adres) en wachtwoord kan inloggen. De inloggegevens worden (zo lang mogelijk) lokaal bewaard, dus bij een volgend bezoek hoeft de gebruiker niet opnieuw in te loggen (single sign on). Als de gebruiker een nieuw toestel krijgt of de app / browser op een andere wijze opnieuw geïnstalleerd wordt, gaan de inloggegevens lokaal verloren en zal weer eenmalig ingelogd moeten worden.
- De website geeft een wachtwoord vergeten link waardoor de gebruiker indien nodig eenvoudig weer een nieuw wachtwoord krijgt toegemaild.
- De gebruiker kan meerdere "projecten" aanmaken met een verantwoordelijke (uitvoerder). Door bij het aanmaken van een melding een project te kiezen wordt ook de verantwoordelijke uitvoerder aan de melding gekoppeld (degene die in ... de meldingtaak krijgt).

MELDING MAKEN

- Na het openen van de app / website komt de gebruiker direct in het melding maken scherm.
- De gebruiker vult de volgende velden in (allemaal verplicht).
  - Kiest een project en daarmee de verantwoordelijke
  - geeft de lokatie aan. Standaard komt de App met het adres van de lokatie waarop de medewerker zich bevind (indien dit uit staat wordt gevraagd of de GPS coördinaten mogen worden gebruikt). Gebruiker kan ger voorgestelde adres / lokatie overschrijven met eigen tekst. Er wordt indien mogelijk een kaartje getoond met daarop de lokatie
  - Beschrijving incident
  - Upload max 2 foto's van onveilige situatie VOOR "verhelpen"
  - Geeft Ja / Nee aan of hij de onveilige situatie terplekken heeft kunnen oplossen
  - Geeft de genomen maatregelen aan
  - Upload max 2 foto's van onveilige situatie NA "verhelpen"
  - Geeft aan of hij Ja / Nee de verantwoordelijke (uitvoerder) telefonisch gesproken heeft
  - Datum en tijdstip worden op de achtergrond door de app meegestuurd in de melding.
  - Voor de foto's geldt dat deze altijd uit het foto album te halen zijn en indien toestel / OS dit toestaan er ook direct een nieuwe foto kan worden gemaakt.

- Na verzenden van de melding krijgt de gebruiker een bedanktekst. Hierin wordt ook aangegeven dat er bij dringende zaken direct gebeld moet worden. Deze tekst is door Heijmans te beheren in de APP beheer tool.
- Voor het verzenden wordt gecontroleerd of alle verplichte velden volledig en juist zijn gevuld. Indien dit niet zo is wordt via kleur en tekst aangegeven wat nog van de gebruiker verwacht wordt.

- Op dit moment is er maar één type melding, namelijk “onveilige situatie”. Het type is op dit moment dus geen keuze veld.

BEHEER FUNCTIES

- Beheerders loggen met aparte code en password in in de beheer module van het systeem
- Beheerders kunnen verschillende teksten (o.a. de bedanktekst) wijzigen
- beheerders kunnen vaste waarde lijsten beheren die in de website vastliggen (o.a. lijst “relatie met Heijmans”)
- het kunnen aanpassen van een meldt tekst op de startpagina
- Bij misbruik van het systeem is het voor beheerders mogelijk om gebruikers te black-listen waardoor zij niet meer kunnen inloggen (met het betreffende e-mail adres).

OVERIGE ZAKEN

- Ook foto’s die niet via de apps wordten ge-upload worden waar mogelijk zoveel mogelijk client side verkleind. Indien dit niet mogelijk blijkt gebeurd dit server side. Naar … gaan altijd verkleinde foto’s.

- Kleurstelling en grafics worden door Heijmans aangeleverd. De webbouwer maakt op basis van zijn ervaring een eerste ontwerp. Tijdens het testen zal dit beoordeeld worden en zullen hieraan wijzigingen gedaan (kunnen) worden. Het grafisch design ligt dus niet op voorhand vast. Tevens geldt dat het design er per apparaat (PC, tablet, smartfoon) of per OS (o.a. android of Apple) anders uit kan zien.

- Er is een overzicht met alle reeds door de gebruiker gemelde meldingen incl de status. Deze status wordt geupdate aan de hand van een door … verzonden mail naar de website met daar in de meldingID en Status.

- Gebruikers hebben een uniek ID in het systeem (niet hun e-mail adres). Gebruikers mogen hun e-mail adres dus wijzigen. Wel zal dan weer een verificatuie mail verstuurd worden om dit adres te controleren.

- De website biedt de melding aan aan de webservice. De website blijft aanbieden totdat de webservice aangeeft de melding volledig te hebben verwerkt. Dus als de webservice tijdelijk niet beschikbaar is worden de meldingen bewaard en weer aangeboden als de webservice wel weer beschikbaar is.

3. De webservice

- De webservice draaid bij … en ontvangt de meldingen. Na ontvangst worden de meldingen toegevoegd in de … database. Hierna wordt binnen … een workflow gestart waarmee een taak aan de verantwoordelijke wordt toegekend. De precieze werking van de communicatie tussen website, webservice en database wordt nog nader uitgewerkt.

De onderstaande velden worden uitgewisseld.

Naast bovengenoemde velden wordt en nog “proces” informatie uitgewisseld tussen de website en de webservice m.b.t. de communicatie tussen deze systemen.

4. Aanpassingen in ...

- In ... wordt een rol toegevoegd voor de “verantwoordelijke”. Dit zijn medewerkers die in … de taak krijgen de meldingen rond onveilige situaties af te handelen (met name uitvoerders). Deze medewerkers worden aangemaakt in … via een éénmalige import vanuit Excel

- Bij meldingen die via de app binnen komen wordt het veld “soort incident” default gevuld met de waarde “Onveilige situatie”. Op moment van live gaan worden de soort incidenten “sos melding” en “bijna ongeval” inactief gemaakt in ….

- Onveilige situaties zijn ook handmatig te registreren in … (zonder app).

- In ... wordt de melding obv “e-mailadres verantwoordelijke” aan een medewerker gekoppeld die daarmee een taak krijgt. Afhankelijk van het vinkje “onveilige situatie is direct verholpen” wordt de taak “controleer de tijdelijke

oplossing" (bij Ja) of los de onveilige situatie op (bij nee) gestart.

- Als de melding aan een verantwoordelijke gekoppeld is dan ontvangt deze een mail met daarin een link naar de melding in … waarmee de verantwoordelijke (single sign on) de melding kan openen.

- Als de melding aan een verantwoordelijke gekoppeld is dan wordt er ook een mail verstuurd naar de melder dat zijn melding door X in behandeling is genomen en na Y dagen etc…

- Op basis van de GPS coördinaten kan in … een kaartje met de lokatie getoond worden.

- Indien het e-mail adres niet bekend is in … dan wordt een taak aangemaakt (koppel melding aan verantwoordelijke) die toegewezen wordt aan een nog nader te bepalen functionaris (K&V?) die de melding alsnog toewijst aan een medewerker. Obv de medewerker is ook de organisatorische eenheid bekend en daarmee de K&V medewerker bekend.

- Indien een taak van de verantwoordelijke overschreden wordt gaat er een signalering (mail) naar de bijbehorende K&V medewerker die de voortgang bewaakt (conform huidige logica toewijzen van meldingen).
- Indien de verantwoordelijke al zijn taken bij een melding gereed meld / afsluit, dan ontstaat er een taak voor de K&V medewerker (registratie controleren en aanvullen)

- Verantwoordelijke legt indien mogelijk de koppeling tussen de melding en het project (vullen veld project). Pas bij afsluiten van melding door K&V wordt gecontroleerd of dit veld gevuld is (verplicht is).

- Bij de melding staat in … het vinkveld "ik heb mondeling contact gehad met de melder).

- Indien een leidinggevende (verantwoordelijke) de melding niet zelf kan oplossen kan hij deze doorzetten naar een andere oplosser die dan de taak krijgt toegewezen.
- Als K&V de melding sluit gaat er een mail naar de melder dat zijn melding is afgesloten. In de mail (sjabloon) worden een aantal velden gevuld met waarden uit … (nader te bepalen welke).

- Tevens heeft de K&V medewerker een overzicht van alle lopende meldingen (bestaat reeds) waarin nu ook de meldingen "onveilige situatie" getoond worden. Obv "soort incident" is dit te filteren.

- Via zoekopdrachten een overzicht maken waarop per verantwoordelijke of per organisatorisch onderdeel de openstaande meldingen worden getoond.

- Ontwerp toevoegschermen: inhoud van schermen conditioneel maken (afhankelijk van soort melding). Velden wel / niet tonen. Doel is gebruikersvriendelijkheid. Ook voor bewerkschermen. … maakt ontwerp obv screenshots.

- Koppelen / exporteren van meldinggegevens naar Excel tbv rapportages (wegen en civiel), nader uit te werken welke gegevens geëxporteerd moeten worden.

- Op incident rapport (I03, I05 en I07) de kolommen "sos" en "bijna ongeval" verwijderen. Nieuwe kolom "onveilige situatie" die de onveilige situaties toont inclusief de oude sos en bijna ongeval meldingen.

- Zie verder het ontwerp "aanpassingen schermen ..." zoals dat door … is aangeleverd