Agile in a small project

A refinement of a framework for agile requirements engineering
Abstract

Requirements engineering (RE) is the process to eliciting, analyze, document and validate the requirements of a system. This is always a risk factor in system development, especially when agile methods are applied, which does not emphasize this part of the process. Ramesh et al. (2010) developed a framework to aid agile RE, consisting of practices and challenges that have an effect on the risks regarding RE. In this paper I operationalize this framework by using it in a case-study on a small-scale project.

I found that all challenges was encountered to some extent, but only two of the risks, namely presenting requirements in forms of design, and modelling only functional requirements.

The other risks were mitigated, mostly by the first (of six) practice in the framework: communication over written specifications. The practices prototyping and frequent meetings also helped mitigate the risks.

The theoretical contribution of this paper is an operationalization of the framework in a specific setting. The paper is also useful for practitioners looking for guidance in how to implement RE in an agile environment, and pinpoints the most important agile RE practices to consider in a smaller project.

**Key words**: agile system development, requirements engineering, small project
# Table of contents

Glossary ....................................................................................................................... 1

1 Introduction ............................................................................................................... 2

2 Agile system development ....................................................................................... 3
   2.1 XP and Scrum ...................................................................................................... 3

2.2 Theoretical framework ......................................................................................... 3
   2.2.1 The best practices ........................................................................................ 4
   2.2.2 The challenges ............................................................................................. 5
   2.2.3 The overall picture ...................................................................................... 5

3 Method ..................................................................................................................... 7
   3.1 Case description ............................................................................................... 7
   3.2 How the framework was used ......................................................................... 7
   3.3 Analysis process .............................................................................................. 8
   3.4 Limitations ....................................................................................................... 9

4 Result and analysis ................................................................................................. 9
   4.1 Challenges ....................................................................................................... 9
      4.1.1 Problems with cost and schedule estimation (C1) .................................. 9
      4.1.2 Inadequate or inappropriate architecture (C2) ..................................... 9
      4.1.3 Customer access and participation (C3) ............................................... 10
      4.1.4 Minimal documentation (C4) ................................................................. 10
      4.1.5 Prioritizing on a single dimension (C5) .................................................. 10
      4.1.6 Inadequate requirements verification (C6) .......................................... 10
      4.1.7 Neglect of non-functional requirements (C7) ..................................... 10
   4.2 Manifested risks ............................................................................................... 11
      4.2.1 Presenting requirements in forms of design ........................................... 11
      4.2.2 Modelling only functional requirements .............................................. 11
   4.3 The risks not manifested in my project ............................................................ 11
      4.3.1 The lack of requirements existence and stability ................................... 11
      4.3.2 Issues with users’ ability and concurrence among users .................... 11
      4.3.3 Inadequate user-developer interaction .................................................. 12
      4.3.4 Overlooking crucial requirements ......................................................... 12
      4.3.5 Not inspecting requirements ................................................................. 12
      4.3.6 Intrinsic schedule flaws ........................................................................ 12
      4.3.7 Attempting to perfect requirements before construction .................... 12

5 Discussion .............................................................................................................. 12
   5.1 So, what lessons can we learn from this? ...................................................... 14

6 Conclusions .......................................................................................................... 15

7 References ............................................................................................................ 16
<table>
<thead>
<tr>
<th>Annex A</th>
<th>Extreme programming</th>
<th>Scrum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex B ................................................................. 20
Annex C ................................................................. 21
Annex D ................................................................. 22
Glossary

**CMS** – Content Management System, a system where end-users easily can create, remove and modify the content of a web site, without having any programming experience.

**Functional requirements** – the requirements on an information system regarding the functionality, e.g. the operations the users should be able to perform with the system.

**Lo-fi prototype** – an early prototype without any functionality, just giving an indication of how the final system will look like.

**MVC** – Model View Controller, a design pattern used for structuring code in a system with a graphical user interface.

**Non-functional requirements** – the requirements not regarding actual functionality, like security, usability and scalability.

**Product backlog** – a prioritized list of requirements on a high abstraction level used in Scrum.

**Refactoring** – making changes to the code in order to make it more structured and readable.

**Requirements validation** – making sure that the right features are implemented.

**Requirements verification** – making sure that the features are implemented the right way.

**Review meeting** – a meeting with the customer after a sprint is finished to review what has been implemented during the sprint.

**Scrum** – an agile method for system development.

**Scrum board** – A board were all work items in a sprint is visualized, digitally or by a real board with post-its.

**Scrum master** – The leader of a Scrum-team.

**Sitemap** – A model of what pages and information should exist on a particular web site.

**Spaghetti code** – Large amounts of code written without structure ending in a code-base entangled like a bowl of spaghetti, making it impossible to maintain it.

**Sprint** – a fixed period in time during a system development project, with a goal to be achieved by implementing requirements.

**Sprint backlog** – a part of the product backlog selected for a particular sprint.

**XP** – extreme programming, an agile method for system development.
1 Introduction

A requirement can be defined as “any desired feature of an information system” (Beynon-Davies, 2002). They can be expressed and organized in many different ways, but no matter what method is used, they are a risk factor when developing a system (Lamsweerde, 2000). Requirements engineering (RE) is the process where developers try to understand what the users need and how the system can help them achieve their goals (Sommerville, 2005). The process is different depending on the project, but the following activities are always involved in some way (Sommerville, 2005; Ramesh, Cao & Baskerville, 2010; Paetsch, Eberlein & Maurer, 2003):

- Elicitation, where the requirements are discovered
- Analysis and negotiation, where requirements are investigated and clarified
- Documentation of the requirements in some form
- Validation, to make sure that the requirements are what the stakeholders needed.

Traditionally, RE has been a software engineering process taking place before the construction phase. This is assuming that it is possible to capture all requirements before-hand and that they are stable and non-ambiguous, giving the developers a distinct specification to implement (Paetsch, Eberlein, & Maurer, 2003). But the system development (SD) field has seen time and time again that this is not the case – requirements are not stable and frozen in time, but volatile and constantly changing. For the past fifteen years the trend has moved from a traditional RE to a more agile approach, with an iterative development cycle, constantly managing changes that occur (Hochmüller & Mittermeir, 2008).

However, the agile approach has matured and the hype that surrounded the agile paradigm has somewhat settled (Janes & Succi, 2012), and studies show that agile RE are aiming for the same goals as a traditional approach (Paetsch, Eberlein, & Maurer, 2003). The four key activities are still present, but adapted to an agile process and done iteratively throughout the development cycle (Sommerville, 2005).

As a solution to this sticky situation – adapting RE into an agile process – researchers has developed best practices (Hofmann & Lehner, 2001), methods (Mahmud & Venezianot, 2011) and patterns (Hagge & Lappe, 2005). In this paper I will focus on a framework of best practices, challenges and risks (henceforth referred to as just “the framework”), developed by Balasubramaniam Ramesh, Lan Cao and Richard Baskerville, published in Information Systems Journal 2010 (Ramesh, Cao, & Baskerville, 2010). The framework is a result of studies on medium to large companies and intended to be used by both practitioners and researchers to provide “a foundation for developing bundles or collections of practices to improve software development processes in specific contexts” (Ramesh, Cao, & Baskerville, 2010, p. 476) The framework consists of three parts – best practices to follow, challenges that can arise and risks regarding RE that are either mitigated or exacerbated by the practices and challenges. To implement this, i will follow the best practices. I will explain the framework more in detail in chapter 2.2.

The framework is mentioned in approximately 30 articles1 and most of them only briefly mention the framework, without any real use of it. There is however a few of these articles that explores the framework more in-depth. Bjarnson et al (2012) investigates the best practices with the purpose to see how “agile RE practices impact the causes and effects of overscoping” in large scale SD projects (Bjarnason, Wnuk, & Regnell, 2012, p. 1108). Haugset et al uses the risks stated in the framework to extend it and encompass automated acceptance test-driven development in relation to agile RE and a traditional approach (Haugset & Stålhane, 2012). Bjarnson et al (2011) uses the best practices as a startingpoint for further investigation of best practices in large-scale RE and how agile practices can mitigate the challenges faced in traditional RE (Bjarnason, Wnuk, & Regnell, 2011).

---

1 The database Scopus had 31 citations of the article, and in Web of Science the article had 11 citations, with overlapping results.
According to my literature review, the framework has not been applied on a small-scale project. My aim with this paper is therefore to apply the best practices stated by the framework in such a project and investigate what challenges arise and how the risks are affected, in order to refine the framework for this specific context. The project where the framework is to be used is the development of a web site for Örebro student union. During the process I will use parts of agile SD methodology together with the framework.

My research questions are: When following the best practices in the framework designed by Ramesh et al. (2010) on a small-scale project, what challenges from the framework arise? How do the challenges and practices affect the risks stated in the framework?

2 Agile system development

2.1 XP and Scrum

In February 2001, a group of people later known as the Agile Alliance, had a meeting that resulted in a statement called the Agile Manifesto, as seen to the right. This was a reaction to document heavy, plane driven methods and has been a foundation for several method instantiations. Two of the most recognized methods are Scrum and Extreme Programming (XP) (Highsmith, 2001).

XP’s foundation is a set of five values that basically is ideas of what is important in the SD process. At the other end, there is a set of practices to be incorporated in the day to day work. To bridge the gap between values and practice, there is a set of principles to be followed – domain-specific guidelines for the SD process (Beck, Andres, & Gamma, 2004).

Scrum is an equally lightweight method, with no explicit “recipes” for how different activities in the SD process is done, but where XP relies on values, principles and practices, Scrum consists of a set of roles, activities and artifacts. The development team is a self-organized unit that is in great control of what is to be done, supported by a scrum master and guided by a sprint backlog, which is a prioritized and estimated list of features to be implemented in the following sprint, a preset time-box with an explicit goal (Cohn, 2010).

A detailed description of these agile methods is outside the scope of this paper. To read more about what parts of the methods I have used in this one-woman-project, check out Annex A.

2.2 Theoretical framework

Ramesh et al. (2010) wanted to get an answer to three questions:

- What RE practices are adopted in software development in agile environments?
- What problems are encountered when these practices are used?
- Do these practices mitigate or increase requirements risks?

To do so they conducted a field study on 16 software development organizations, resulting in a framework consisting two main parts: a set of best practices used in agile environment to handle the constantly changing requirements, and a number of challenges, resulting from these best practices. In their analysis, they conducted a literature review searching for risks regarding the RE process, and mapped these risks to the practices and the challenges: some risks are mitigated by the practices and some are exacerbated by the challenges. It is not a simple 1-to-1 relationship between risks and
practices/challenges and some of the risks are even worsened by the practices. In the 16 companies, all practices were adopted, but not all challenges were encountered.

There were a total of six practices, seven challenges and nine risks identified. The following two sections go into detail on the practices and the challenges. The third section shows how they relate to the risks by a picture intended to show the mappings.

### 2.2.1 The best practices

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Description (Ramesh, Cao, &amp; Baskerville, 2010)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Face-to-face-communication over written specifications</td>
<td>The focus is to quickly transfer ideas from the customer to the developers, and that is done by detailed discussion and feedback rather than writing specifications. It requires the customer to be very involved in the development process and a frequent dialog with the development team.</td>
</tr>
<tr>
<td>P2</td>
<td>Iterative RE</td>
<td>The requirements in an agile environment are not stated beforehand, but emerging along the process. The development team has in the beginning a vague idea of what is to be implemented, they start to build, and then through an iterative process the requirements are refined as the building process is moving forward. The requirements are never meant to be perfect at the beginning – one always has to go back and improve/change them.</td>
</tr>
<tr>
<td>P3</td>
<td>Requirements prioritization goes extreme</td>
<td>To prioritize requirements early in the process helps delivering value to the customer in form of software. Business value is often the single parameter to consider in agile RE. Since the requirements change, prioritization takes place during the entire development process.</td>
</tr>
<tr>
<td>P4</td>
<td>Managing requirements change through constant planning</td>
<td>This is a core principle in agile development – to adapt quickly to change. Through constant planning, changes become a part of the daily work and therefore, the changes are not that big. The constant communication is also a reason to that major changes not occurring.</td>
</tr>
<tr>
<td>P5</td>
<td>Prototyping</td>
<td>Prototyping as a way of communicating requirements to the customer is very effective. IKWISI (I Know It When I See It) is a common term in software development – the customer may have difficulties expressing requirements, but when she sees it, it becomes very clear. Prototyping can happen on many levels – from paper and pen to working code.</td>
</tr>
<tr>
<td>P6</td>
<td>Use review meetings and acceptance tests</td>
<td>Frequent meetings are important due to requirements validation. During a review meeting, which takes place at the end of a sprint, implemented features are demonstrated and discussed. Acceptance tests are also a way of validating the implemented requirements. TDD (Test Driven Development) is yet another way. However, the most important validation technique is constant communication with the customer.</td>
</tr>
</tbody>
</table>

---

2 I have given the practices and challenges an id to make the picture more clear and compact due to limited space in this paper. The id has no other purpose than this and there is no ranking among the items.
### 2.2.2 The challenges

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Description (Ramesh, Cao, &amp; Baskerville, 2010)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Problems with cost and schedule estimation</td>
<td>Since an agile project starts developing before all requirements are known, there is really hard to estimate cost and time. The estimates serves as starting points that are adjusted over time, and gets more and more accurate for each sprint.</td>
</tr>
<tr>
<td>C2</td>
<td>Inadequate or inappropriate architecture</td>
<td>If the little knowledge in the beginning of a project results in a certain architecture is found to be inappropriate later on in the project, this is a problem since it is a big change to make later in the process. Refactoring of code is a big part of XP, but it can be both difficult and costly to make big changes, especially if the code has evolved to entangled spaghetti code.</td>
</tr>
<tr>
<td>C3</td>
<td>Customer access and participation</td>
<td>For an agile approach to work, the customer must be available, consistent and trust the team, at least in the beginning of a project since the estimates are vague at this point and there is no written specification of what to be done. A customer representative on site is preferable, but can be hard to attain since that person won’t have the time to do her regular duties.</td>
</tr>
<tr>
<td>C4</td>
<td>Minimal documentation</td>
<td>Obviously, agile RE does not focus on documentation. This works well as long as the communication is good – but if something happens to the team, there can be a real problem if there is no documentation of the system or what is to be built.</td>
</tr>
<tr>
<td>C5</td>
<td>Prioritization on a single dimension</td>
<td>The extreme focus on business value can lead to missing functionality and problems with other aspects of the system.</td>
</tr>
<tr>
<td>C6</td>
<td>Inadequate requirements verification</td>
<td>Validation is often done in an agile process, but verification is much rarer. There is a possibility that a requirement implementation is supporting the right feature, but has been programmed in a way that not works in the long run.</td>
</tr>
<tr>
<td>C7</td>
<td>Neglect of non-functional requirements</td>
<td>Since prioritization is done together with the customer, and the parameter is business value, it is easy to forget about the non-functional requirements, like security or performance. If this is ignored in the early phases, there will be a problem later in the process.</td>
</tr>
</tbody>
</table>

### 2.2.3 The overall picture

Below you can see the framework in its entirety according to Ramesh et al (2010). To the left we have the practices, which result in challenges. The practices mitigate the risks regarding RE, and the challenges exacerbate them. The first six risks are seen as tractable and other three as intractable by agile RE.

The framework is not explicit on the difference between a challenge and a risk. As I see it, a challenge is an obstacle that one can get around without any loss in time. A risk however, is more serious and cannot be solved without some loss, even if it is minor.
Unstable, ambiguous requirements
This is probably the most well-known risk regarding SD projects and the very reason for agile methods' existence. The risk is tractable.

Attempting to perfect requirements before beginning construction
This is a problem that agile RE is directly aiming for—there is no such thing as being able to freeze a requirements specification. The risk is tractable.

Overlooking crucial requirements
This is also a well-known risk regarding SD: traditionally there has been a danger in simply implementing the wrong features. This is tractable when doing agile RE.

Intrinsic schedule flaws
Since there is hard to estimate cost and time, there is a risk that errors are built in to the project schedule. However, constant planning mitigates this risk and it is considered tractable.

Inadequate user-developer interaction
The users are more involved than ever using an agile approach, and is the core of the SD process. Problems can arise if the customer does not trust the developer, but since it is the heart of the agile RE (user interaction), the risk is tractable.

Presenting requirements in forms of design
Since the requirements in agile RE are not documented, they are presented indirect as design and code and can be a risk if changes has to be done. However, changes are expected and embraced, making this risk tractable.

Not inspecting requirements
Since there are no written specifications, it is hard to inspect and verify requirements after they have been implemented. This is an even bigger concern if the non-functional requirements are extra important. This risk is intractable in bigger systems, but usually not a problem in smaller projects.

Modelling only functional requirements
The practices make sure that crucial functional requirements are not forgotten, the non-functional ones unfortunately can be, as we have seen in the discussion of the challenges that arise. Intractable risk.

Inconsistent, unable users
Agile methods are highly dependent of the users and if they are not qualified or consistent, the project is in danger. This is a risk even more current in agile methods and is seen as intractable.
3 Method

“A computer-based system is built for people and by people. It is intended to serve some human purpose, and it exists in some human context” (Goguen, 1993, p. 1). This means that the RE process is a social process and can never be entirely formal. (Goguen, 1993) Ramesh et al. share these assumptions when creating the framework and says that the RE process is a social-political one that is highly influenced by human factors and largely depends on the context (Ramesh, Cao, & Baskerville, 2010). This makes a case study approach suitable, since it is investigating one instance of something with a purpose of getting a deeper understanding of the phenomenon in its real-life context (Oates, 2006). In my situation it is one instantiation of an agile RE process that is to be investigated, with the purpose to find out how the framework works in the specific setting.

I want to get a deeper insight in how the framework is applicable and why it produces a certain outcome, making a case study with a descriptive approach feasible. (Oates, 2006).

3.1 Case description

Örebro Student Union (ÖS) is working for improvement of the study time for the students of Örebro University (ÖU). Their vision is to promote their members interests and be characterized by an open and accessible organization. As a part of their operational plan for 2014/2015, they want to continue the work of clarity and transparency, and a web site is an important part in this work. Their previous web site was a contemporary solution with just a fraction of the intended content.

ÖS needed a web site mostly to spread information to their members about their rights and obligations, the student organizations, student residences, and all of the work that ÖS does for the students. They needed a Content Management System (CMS) so that they themselves could put up articles and edit content in an easy manner.

The stakeholders in ÖS for this project have been:

- The Student Union Board, which is ÖS executive body.
- The presidium, which is the president and vice president of the Student Union Board.
- Three officials working full-time
- Two full-time trustees

When I heard that ÖS needed a new web site, I thought it was a perfect opportunity to try a scientifically formulated framework and apply that on a real world situation. Researchers says that more practical anchoring is needed within the field (Kaindl, o.a., 2002) and a strong recommendation is to as a researcher work with practitioners when investigating RE (Cheng & Atlee, 2007). Small companies have also been overlooked in the RE field of research (Jorge Aranda, 2007), even though they can be just as critical (Russ & McGregor, 2012).

This case serves as a “Test-bed for theory”, meaning it is suitable for testing an existing theory, in this case the framework, to see if it is applicable or needs to be modified for the particular context (Oates, 2006, p. 144).

3.2 How the framework was used

As described in previous sections, the framework consists of practices that cause challenges and both of these have a positive or negative effect on risks regarding RE. For the purpose of evaluating the best practices, I have beforehand decided how I would implement them, as seen in the table below.

---

3 If the reader wishes to know more about ÖS, I recommend visiting the website, where their vision document, opinion document, operational plan and statutes is located. My knowledge of the organization comes from these documents, but most of all from discussions with the presidium and my previous engagement in one of the Student Union Division of ÖS. The documents can be found at: [http://orebrostudentkar.se/ → Om Oss → Vad är ÖS?](http://orebrostudentkar.se/ → Om Oss → Vad är ÖS? (Swedish))
Many of the best practices are closely connected to the practices advocated by XP and Scrum, and the integration between the SD methods and the frameworks best practices are overlapping.

<table>
<thead>
<tr>
<th>ID:</th>
<th>Practice:</th>
<th>How I used it in my project</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Face-to-face communication over written specifications</td>
<td>I have had access to the full-time trustees by ÖS during the entire project, and the only written specifications is my initial product backlog (see Annex C), that also served as a support to the agreement I had with ÖS on what should be implemented to consider the web site done. Unfortunately, I could not be sited with the customer due to technical issues (I needed my stationary PC for the development) but we have had frequent meetings and in periods daily contact by mail and Facebook. I have written comments to let future programmers know what the different parts do, without have to read any additional documents. I want the system to be scalable and easy to maintain in the future, even for someone unfamiliar with the code.</td>
</tr>
<tr>
<td>P2</td>
<td>Iterative RE</td>
<td>We started off the entire project by holding a workshop where we discussed what content should be on the web site. As a result of this, I draw a site-map on the whiteboard and from this I created a prototype. At the next meeting, I showed the prototype and I got relevant feedback for next iteration, and so on.</td>
</tr>
<tr>
<td>P3</td>
<td>Requirement prioritization goes extreme</td>
<td>During the process I have let business value be the guide for prioritization of what features that needed to be implemented. There have been minor re-prioritzations during the process, regarding the content on the web site.</td>
</tr>
<tr>
<td>P4</td>
<td>Managing requirements change through constant planning</td>
<td>To aid my planning, I have used an online tool, which supports Scrums note system. When changes have been made, I have dropped or added work-items and always kept my work-board up to date, making the planning clear and flexible.</td>
</tr>
<tr>
<td>P5</td>
<td>Prototyping</td>
<td>I have worked with prototypes throughout the entire project. From the first site-map, I created a lo-fi prototype using an online tool. Then I made a html-based prototype, and from the feedback on this, I made another, more complex html-prototype. In the last sprint I have been working with the final version, which has a CMS-backend.</td>
</tr>
<tr>
<td>P6</td>
<td>Use review meetings and acceptance tests</td>
<td>We have had five review meetings during the process. At each of the meetings, I have shown what I have done, received feedback and we have discussed everything from function and design to text and images. I have also used a test-domain so that the full-time trustees could click around the site themselves and do acceptance testing, before I have put it online at the official domain, orebrostudentkar.se.</td>
</tr>
</tbody>
</table>

3.3 Analysis process

During the project I have implemented the best practices as explained in previous section. I have created a simple time-line for the SD process, and in this timeline I have noted if I have encountered the challenges and approximately when it happened, see Annex D. Time-line also shows the meetings I have had with ÖS: the first two were focused on initiation and planning of the project, the third was a workshop where the requirements for the web site were discussed. The other five meetings were review meetings. During the process I have noted what risks occurred and during the last sprint I started my evaluation.

The evaluation process was very straight-forward. I started with the list of risks and noted which risks I had encountered and which I did not encountered, and tried to make connections between them.
and the practices and risks. To do this, I analyzed the outcome (i.e. my notes on the risks and challenges that had occurred) by performing the following step for each risk:

1. I looked at the framework to see if it was tractable or intractable.
2. I noted what had caused or prevented the risk.
3. I compared the note to the framework to see if there were any connections to the practices or challenges.

As a result of this process, I noted a pattern on which practices were most important and which challenges that caused the encountered risks. To illustrate this, I made a new model, as seen in section 5.

3.4 Limitations

As recognized by Oates (2006), there are some limitations to a case study approach: the conclusions may be tied to the specific setting and lack credibility when generalized. There is also no set of rules to follow in conducting a case study (Oates, 2006), making it harder for an inexperienced researcher to conduct it in a proper, scientific manner.

Another important factor is that I played the role of both researcher and developer in this project. I evaluated the framework by using it my-self, in a one-woman-project, and that leads to a possibility of affecting the case. On the other hand, I had a very rich insight in the case by being so involved and the customer was never influenced by my research. To them I was only a developer, even though they of course knew about my research project.

4 Result and analysis

During this project, I have encountered all of the challenges in some way, mostly during the two last quarters of the project. Out of the nine risks, two was encountered, but not to the extinct that the project itself was in danger. However, it did cause some time waste.

In the next sections, I will go into the challenges and risks more in detail, and how they manifested themselves in my project.

4.1 Challenges

4.1.1 Problems with cost and schedule estimation (C1)

This was the first challenge I encountered. ÖS wanted to know exactly how much time it would take to build this web site. I had to make an estimate, but also add that this is one of the hardest part of the SD process and that I cannot make any guarantees about the dates. I estimated the tasks roughly in the beginning, and then I adjusted the estimate as the project proceeded.

The same approach is noted by Ramesh et.al. (2012): the estimate is a starting point and frequently updated, with more proper planning for each sprint.

4.1.2 Inadequate or inappropriate architecture (C2)

This also occurred in the beginning of the process, and I did not know what architecture to choose for the application. My first choice was MVC, and I investigated if the CMS of my choosing supported this, and it did. However, I later discovered that the web hosting did not support the latest version of MVC, which meant that I had to reconsider my choice.

This is also recognized by the framework: “The architecture chosen by the development team during the early cycles may become inappropriate or inadequate as newer requirements become known.” To refactor can be a costly task, especially if it is discovered late in the process. (Ramesh, Cao, & Baskerville, 2010)
4.1.3 Customer access and participation

ÖS have been an engaged client and have always been able to answer my questions. I thought that it would be a problem when the new fiscal year started July 1st, since all the full-time trustees would be replaced, but good communication between the old and the new staff at ÖS made this transition seamless. However, when everyone had vacation a couple of weeks later, I could not get a respond to my email, but I could contact the president on Facebook so this challenge was easy to overcome.

A big part of the ease of which this challenge could be averted, is an important factor that Ramesh et al brings up: trust. I have throughout the process felt comfortable in that my customer will be as involved as needed. I also feel that the trust has been mutual: even though ÖS has no experience with being a requirements specifier or agile system development, they trusted me as a developer. This positive attitude has a major impact on the project (Ramesh, Cao, & Baskerville, 2010).

Another factor in this is that there has not been that many stakeholders involved, and the core organization of ÖS is small – approximately 7 people is working during the summer, so consensus has not been a problem.

4.1.4 Minimal documentation

This is a challenge when thinking about the future; some time there will be need to make changes to the code, and without me or any documentation this will be hard. Another thing that could be a problem in the future is that the design is made for the content we have now, and major changes in content can result in strange behavior of the design.

The first challenge I have addressed by writing clear code with lots of comments, so that a developer easy can get an understanding of the code. The second issue is harder, since the user does not see (or understand) the code – all she sees is the break in design.

The lack of documentation can in these cases cause lots of problems (Ramesh, Cao, & Baskerville, 2010), and by taking this actions I have hopefully made sure that these problems will not occur.

4.1.5 Prioritizing on a single dimension

Ramesh et al recognizes that a rush to coding, without considering other dimensions of requirements prioritization, can result in problems. Requirements that can appear as trivial at the beginning of the project, becomes crucial later on.

This challenge I encountered in a very straight-forward way: I chose a CMS that I thought would bring the most business value, but discovered later that the web hosting company did not support that.

4.1.6 Inadequate requirements verification

Ramesh et al points out that there is no formal modelling of the requirements beforehand and no detailed specifications, which can result in requirements implemented in ways that not work in the long run. Nevertheless, with emphasis on testing, agile RE can work around this challenge.

For some parts of the coding, I felt safe in my own knowledge and am certain I implemented it correctly. However, some techniques were new to me and I had to spend some time study how to program it correctly.

4.1.7 Neglect of non-functional requirements

This challenge is a major concern in agile RE, and Ramesh et al reports that many organizations ignore the non-functional requirements in the beginning of the process, and makes them harder to implement afterwards.

I encountered this challenge as soon as I published the first version of the site – it looked really bad on every device but a large screen PC. ÖS had of course counted on that it would look good on mobile, since the mobile browsing is extremely common nowadays. But this was nothing we discussed, and since I develop on a large screen PC, this was the device I designed for.
4.2 Manifested risks

4.2.1 Presenting requirements in forms of design
In the early stage of the project, I and ÖS discussed how the contact site would look like, with the contact info to all personnel working in ÖS office. I pitched the idea of a contact form and interpreted our discussion as if they agreed. However, at the next meeting, when I had implemented a contact form on the site, they said that they did not want a form, only the e-mail addresses. There was nothing for me to do but undo the work I put in the form, and remove all code connected to it. If we instead of a working prototype had expressed the requirements in form of text, they would have been able to tell me right away that they did not want the contact form, and no time had been wasted. But since the requirement already was implemented (i.e. presented in forms of design), this was not possible.

Just as Ramesh et al states: this technique is very efficient for speedy implementation; it may impede the SD process when requirements change. My situation was not really a change, more of a misunderstanding from my part, but the result was the same – I had to undo a bit of code and had wasted about a day’s work.

This risk is considered tractable by the framework. Yet, it occurred in my project. Why is that? I can think of one main reason: The communication failed between the customer and the developer.

4.2.2 Modelling only functional requirements
This is a risk exacerbated by the agile practices and is considered intractable. Ramesh et al says that there is a risk for the non-functional requirements to be poorly understood and not dealt with until late in the process. They are also harder to implement if discovered late in the process, and difficult to change. Since the customer is very much involved and in control of the process, it is up to the developer to take the non-functional requirements to consideration. The customer seldom have any idea of what requirements needed for security, scalability or usability – she just assumes that the system will be secure, scalable and usable. If the developer does not realize this situation, the system probably won’t have any of these aspects. (Ramesh, Cao, & Baskerville, 2010)

I encountered a few smaller problems regarding the non-functional requirements, with just hours wasted in the process. The biggest concern though, was due to the web hosting: I chose a CMS that was not supported by the web hosting company, and wasted days of the process on building something that could not be used.

4.3 The risks not manifested in my project
Seven of the risks in the framework were not a problem in this project. I will cover them very briefly here.

4.3.1 The lack of requirements existence and stability
As mentioned earlier, the lack of requirements existence and stability is the very reason for an agile process, and several of the practices mitigate it. For this project, I believe that the practices to emphasis communication and the prototyping mitigated this risk – with constant feedback, the developer can keep track, even if the requirements do change. It was also a small project, with a well-defined scope, i.e. a web-site mainly for information, based on a CMS, and that also contributed to the stability of the requirements.

4.3.2 Issues with users’ ability and concurrence among users
ÖS is a non-profit organization, with seven people working full-time. All of these have been more or less involved in the RE process, and they all share a good knowledge about the organization. They are also working tightly together and are very cohesive, making this risk a no-show, even though agile RE exacerbates it. Again, the practice of communication is at place – this time both between the users and the developer, and among the users.
4.3.3 Inadequate user-developer interaction
An ideal situation had been for me to work on site, in the customer’s office, but since I need my stationary PC for the development, this was not possible. However, constant communication and a very good, trusting relationship with ÖS, eliminated this risk. From the very start of the project, I emphasized communication as the solely most important factor in a customer-developer relationship, and I urged them to talk to me if there was something they felt was unclear. I believe that this approach helped mitigate the risk – as Ramesh et al says: this interaction should be in the center of agile RE, than it makes the risk tractable. The prototype also helped, since it was easy for the customer to look at what had been done and discuss from that, rather than a list of features, which tend to be a bit abstract.

4.3.4 Overlooking crucial requirements
This was never a problem in this project, and we were continuously able to capture the important requirements, much thanks to frequent meetings and acceptance tests, and once again – the communication.

4.3.5 Not inspecting requirements
In this small project, there really was no need for a formal inspection of the requirements. The features were pretty much straightforward and a small part advanced functionality to code, making an inspection unnecessary.

4.3.6 Intrinsic schedule flaws
As shown in section 4.1.1, the estimation was a challenge. However, there were no real flaws in the schedule – it all evened out since some tasks took more time to complete than I estimated, and some took less.

4.3.7 Attempting to perfect requirements before construction
Since this is a well-known risk and something I was aware of, I never made such an attempt. I know that it is not possible to freeze the requirements before construction, and used an iterative approach based on prototyping to prevent this risk.

5 Discussion
When following the best practices from Ramesh framework in a small, one-woman-project, I encountered all challenges to some extent, and two of the risks. Three of the best practices showed to have larger, positive impact than others. The following image illustrates the outcome of this study, i.e. how the framework manifested itself in this particular context:
Starting from the top: I implemented all of the best practices, and in my experience, three of the practices (P1, P5 and P6) had a larger positive effect than the other ones.

**P1 – Communication over written specification**

This was the most important of the practices, and imbued the entire work, even if we didn’t meet face to face more than for the planned meetings. The contact has been mostly by email, and through quick responses and a constant discussion, the SD process has been keeping on track.

**P5 – Prototyping**

The prototypes have served as a ground for discussion, and have been an important part of the process. The most value it brings is to make requirements and features more concrete and direct the discussion with the customer in the right way, i.e. keeping the focus on requirements and features.

**P6 – review meetings and acceptance tests**

This practice has tied the previous ones together – the meetings has been milestones in the project, serving as a place for discussion, a review of what has been done, approval of features and planning for the upcoming sprint. The meetings have formed natural sprints in my work, and reshaping the sprint plan I had from the start to some extent, and they have been the biggest resource for capturing changes in the requirements along the way.

The challenges did however appear, but in a manageable way. No challenge was bigger than I could handle it without endangering the SD process. In my opinion, it was the challenges C4 – C7 that exacerbated the two risks that occurred:

**C4 – Minimal documentation**

If I had some documentation of the requirements, I would probably not have made the mistake to implement a contact form that was not wanted.
**C5 – Prioritizing on a single dimension**

If I had prioritized requirement on more parameters, I probably would have chosen a more suitable CMS from the start.

**C6 – Inadequate requirements verification**

If I had verified the requirements before they were implemented, I probably could have avoided both of the risks.

**C7 – Neglect of non-functional requirements**

If I had taken a look at the non-functional aspect of the requirements, I probably would have chosen the right framework from the start.

5.1 So, what lessons can we learn from this?

My study pretty much confirms what others have concluded before: an agile approach helps dealing with previous problems regarding RE, but the SD process also needs a dose of planning and documentation (Boehm, 2002; Mahmud & Venezianot, 2011; Paetsch, Eberlein, & Maurer, 2003) - even if it is a very small project. Paetsch, Eberlein and Maurer also points out an important aspect: agile methods require skilled people;

“Good” developers will do the “right thing” (p.6).

When there is no documentation, modelling or verification to lean on, the developer has to make many decisions on the fly, while coding. An experienced developer will more often make the right choice than an unexperienced one.

I think that this is something worth to have in mind when choosing a strategy for RE – it is all dependent on the specific context. The team, the client and the situation has to be considered before making a choice. This also requires skill from the project manager, or whoever is making this decision, to take a smart approach from the beginning. This is also recognized by Ramesh et al, that the project settings should be analyzed, and based on that one can chose agile practices or a more traditional approach.

Another point to take in notion, which may be obvious, is that a small project often means small challenges and small risks. In this case, it never was a matter of weeks or months of waste, since the total scope and iterations are smaller. The stakes are lower, and I also believe that the smaller the project is, the less skill is needed from the developer. Compare maneuver a huge ferry from Sweden to Finland to paddle a canoe over a small lake – it’s just easier with a small project. However, you cannot fool around even if you are in a canoe; there are still challenges and risks that have to be considered.

I also believe that a small project makes it possible for a much more personal relation to the client – I am the only developer so I will be on all meetings and take care of all contact. The customer organization was not that big either, so I have had contact with almost everybody and the contacts and meetings has always been casual and friendly. Such a climate makes it easy to bring up topics for discussion and the customer feels freer to talk about changes that might occur.

It should also be noted that I have been engaged in one of the Student Union Divisions for two years now, and I have a pretty good insight in ÖS organization, which probably has been an advantage in my work.
6 Conclusions

My initial research questions were: *When following the best practices in the framework designed by Ramesh et al. (2010) on a small-scale project, what challenges from the framework arise? How do the challenges and practices affect the risks stated in the framework?*

The result from this case study, showed that all of the challenges were met to some extent, and that two of the nine risks regarding RE occurred, namely:

- Presenting requirements in form of design and
- Modelling only functional requirements

The challenging factors that led to the risks being exacerbated was:

- lack of documentation,
- the prioritizing with focus only on business value,
- a neglect of non-functional requirements and
- the fact that there is no formal verification of the requirements.

There is no silver bullet when it comes to RE – the agile practices have both pros and cons, and it all depends on the context what approach to use. However, the study shows that three of the practices were more important than others:

1. Frequent communication between developer and customer. This was the single-handedly most important practice and served as a foundation for the whole project.
2. Prototyping was also an important factor – it is so much easier for the customer to talk about something when she sees it in front of her, instead of some abstract idea or words written on paper.
3. Frequent meetings tied the other two practices together, making a forum for showing the current prototype and discuss about it.

This conclusion shows what is important when operationalizing the Ramesh et al framework in a small-scale project. In section 5, I have processed the framework and highlighted what was important in this project. Despite the limitations of this study, I believe that the result is generalizable on other projects, since the best practices and challenges are not specific to the context.

The theoretical contribution of this paper is an operationalization of the framework and a deeper understanding of how the framework applies to a small-scale project – in particular, what risks and challenges are less important in this type of project. The paper also serves as a guidance for practitioners in the field in determining the RE approach for a small SD project.
7 References


Annex A
In this section I will explain what parts of Scrum and XP I have used. Since a detailed explanation of the methods is outside the scope of this paper, this section may require some knowledge of the methods.

I wrote these tables before I started the projects, and I have been following my plan all the way.

Extreme programming
XP has values, ideas of what is important, as a foundation of the method, and practices that guide the daily work, combined by principles to follow regarding system development. The practices are hard for me to implement in a one-woman-team, so I will mostly follow the basic values and some of the principles. The five values are:

<table>
<thead>
<tr>
<th>Values:</th>
<th>How I will consider the value:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>The key to fix problems in a system development project is to communicate, and I have told ÖS that this is a crucial part of the development process. Clear, frequent communication will be implemented through email and meetings.</td>
</tr>
<tr>
<td>Simplicity</td>
<td>I will ask myself: “What is the simplest thing that could possibly work?” to eliminate wasted complexity.</td>
</tr>
<tr>
<td>Feedback</td>
<td>I will mostly get this with my prototype/review-meetings from the customer, but I plan to involve the end-users, i.e. the students that will visit the web site, later in the process to get their feedback to.</td>
</tr>
<tr>
<td>Courage</td>
<td>When in doubt, I will try to be brave – but not take unnecessary risks. If something goes wrong, or not as planned, I will have the courage to talk to my client about it. I will also encourage the customer to talk to me about everything, especially if they are unhappy about something or if they think I am doing something wrong in their eyes. We will have the courage to have an open communication throughout the development process.</td>
</tr>
<tr>
<td>Respect</td>
<td>I will respect my client, and other stakeholders, and by doing so, hopefully get respected back. This is the foundation of XP values.</td>
</tr>
</tbody>
</table>

There are a total of 14 principles in XP, and I have chosen the following 9 as relevant for my project:

<table>
<thead>
<tr>
<th>Principle:</th>
<th>How I will follow the principle:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humanity</td>
<td>I will see to that my basic human needs are satisfied. 😊</td>
</tr>
<tr>
<td>Economics</td>
<td>Since ÖS are paying me for this website this principle becomes important. I need to keep the business value and needs first in my prioritizing of the functionality.</td>
</tr>
<tr>
<td>Mutual benefit</td>
<td>All choices made should benefit all concerned stakeholders and I will not do tasks that have a negative effect on me or my customer.</td>
</tr>
<tr>
<td>Self-similarity</td>
<td>I have made web pages before and as a starting point, I will repeat a solution from a previous project.</td>
</tr>
<tr>
<td>Improvement</td>
<td>“Perfect is a verb, not an adjective.” I will do as well as I can today, and strive to do it better tomorrow. I will not wait for perfection and just find a starting point and get going, and improve my work from there.</td>
</tr>
</tbody>
</table>
Reflection
Don’t just work – think about how and why you work. I will analyze success and failure, but remember that action comes before reflection; it should not take time from the coding.

Flow
I will strive for a continuous flow of deliveries, and not release in big chunks.

Opportunity
I will try to see problems as opportunities to reach excellence, and embrace change in the development process by being open to the customers’ suggestions.

Quality
When I leave the finalized product, I will make sure it reaches the highest possible quality. High quality == the customer should be satisfied with the GUI and the look and feel of the web site, and I should feel proud of the code behind it.

<table>
<thead>
<tr>
<th>Scrum</th>
</tr>
</thead>
<tbody>
<tr>
<td>The different team roles of Scrum does not apply to my one-woman-project, and because of the highly changing organization in ÖS, I will not have a dedicated product owner. Since their financial year ends in June 30, the whole organization is being replaced from July 1 – mid way in my project. This, in combination with being an extremely inexperienced client, leads me to believe that a role of product owner only will cause more work and confusion for them. Daily scrums are of course not applicable, and the sprint retrospective will be a continuous reflection of my work, according to XP.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part to use:</th>
<th>How I will use it:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint</td>
<td>As a time-boxed period where a sprint goal is to be reached through planned activities. I have eight sprints in my project, and the duration varies to match up the organization and my schedule for the summer.</td>
</tr>
<tr>
<td>Sprint planning</td>
<td>Not as a meeting, since it’s only me, but as a planning before each sprint that should answer: what can be done in the following sprint and how can it be done? The sprint goal is a guiding star in the planning.</td>
</tr>
<tr>
<td>Product backlog</td>
<td>As a very high leveled, prioritized and roughly estimated list of work items.</td>
</tr>
<tr>
<td>Sprint backlog</td>
<td>A list of work items for the upcoming sprint, which completed together, will achieve the sprint goal. This I will use my planning tool Kanban Flow for.</td>
</tr>
<tr>
<td>Sprint review</td>
<td>Unfortunately the customer was not able to have a formal sprint review at the end of each sprint, but we had five review-meetings where I demonstrated what I had done so far and got feedback on my work.</td>
</tr>
</tbody>
</table>
Annex B

This is my sprint planning for the project. I have named the sprints by color to easily be able to visualize them in the calendar below, and in the time-line in Annex C.

<table>
<thead>
<tr>
<th>Name of sprint:</th>
<th>Sprint goal</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>The brown sprint</td>
<td>A first understanding of the project and a project plan.</td>
<td>Yes</td>
</tr>
<tr>
<td>The pink sprint</td>
<td>A site-map for the website content and a first prototype</td>
<td>Yes</td>
</tr>
<tr>
<td>The green sprint</td>
<td>A complete plan for the project work (both essay and construction part) and starting of second prototype</td>
<td>Yes</td>
</tr>
<tr>
<td>The purple sprint</td>
<td>A high-level prototype in HTML</td>
<td>Yes</td>
</tr>
<tr>
<td>The blue sprint</td>
<td>Release of the first version of the website (a refined prototype)</td>
<td>Yes</td>
</tr>
<tr>
<td>The orange sprint</td>
<td>Get understanding for Wordress and transfer the prototype to a stable back-end version.</td>
<td>Yes</td>
</tr>
<tr>
<td>The turquoise sprint</td>
<td>First release of the final website</td>
<td>Yes</td>
</tr>
<tr>
<td>The yellow sprint</td>
<td>Refinement of the final website, demonstration of admin tools for end users.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Grey days = I will be out of town
Red days = holidays
Star = release
**Annex C**

This shows my initial product backlog. It contains the requirements on a very high level and are roughly estimated in relation to each other.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Description</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crucial</td>
<td>As a visitor I should be able to see relevant information about ÖS in an intuitive manner. The relevant info and how it should be arranged is visible in the site-map and in the mock-up.</td>
<td>XXXL</td>
</tr>
<tr>
<td>High</td>
<td>As a ÖS-user I should be able to edit all text on the mother-site.</td>
<td>XL</td>
</tr>
<tr>
<td>High</td>
<td>As a section-user I should be able to edit all information on my section.</td>
<td>XL</td>
</tr>
<tr>
<td>Medium</td>
<td>There should be a similar structure for all sections web pages, with approximately the same info.</td>
<td>XXL</td>
</tr>
<tr>
<td>High</td>
<td>As a ÖS-user I should be able to post news on the start page.</td>
<td>L</td>
</tr>
<tr>
<td>Low</td>
<td>All posts should be able to share on Facebook.</td>
<td>M</td>
</tr>
<tr>
<td>Medium</td>
<td>As a visitor I should be able to find out which section I belong to in an easy and intuitive way.</td>
<td>M</td>
</tr>
<tr>
<td>Low</td>
<td>There should be a map of the organization with hotspots where you can click and get more info on the clicked object (e.g KS, FUM)</td>
<td>L</td>
</tr>
<tr>
<td>Low</td>
<td>There should be a map of the organization with focus on possible commitments and hotspots where you can click and get more info on the area of interest.</td>
<td>L</td>
</tr>
<tr>
<td>Low</td>
<td>Every section should have one email-address, possibly connected to other addresses.</td>
<td>S</td>
</tr>
<tr>
<td>Low</td>
<td>The web address to all sections should always be orebrostudentkar.se/sesam</td>
<td>S</td>
</tr>
<tr>
<td>High</td>
<td>On the start page, there should be a Instagram-flow visible.</td>
<td>M</td>
</tr>
<tr>
<td>Medium</td>
<td>Connect what the student paper Lösnummer writes about ÖS to the news feed on the start page.</td>
<td>M</td>
</tr>
<tr>
<td>Medium</td>
<td>The final version of the web site should be responsive.</td>
<td>XL</td>
</tr>
</tbody>
</table>
Annex D
This timeline shows the different challenges and when they occurred in my project, together with the meetings I had with ÖS.

### Challenges

<table>
<thead>
<tr>
<th>Week</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
<th>32</th>
<th>33</th>
<th>34</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Notes
- March 13th: First meeting with the Student Union Board of ÖS, were they told me about the website they wanted.
- April 9th: Second reconciliation meeting with the president of the Student Union Board.
- April 15th: Workshop with all stakeholders.
- April 29th: Demonstration of the first prototype.
- May 15th: Reconciliation meeting with ÖS president and vice president.
- June 19th: Demonstration of the prototype in HTML, which also were the first version of the website.
- July 3rd: Demonstration of the second version of the website, with approval to publish.
- August 13th: Demonstration of the final version of the website and walkthrough of the administrator interface.