Evaluation of Problem Driven Software Process Improvement

MARCUS LYTH
HANNES CARLESON
Software development is constantly growing in complexity and several new tools have been created with the aim to manage this. However, even with this ever evolving range of tools and methodology, organizations often struggle with how to implement a new development-process, especially when implementing agile methods. The most common reason for this is because teams implement agile tools in an ad-hoc manner, without fully considering the effects this can cause. This leads to teams trying to correct their choice of methodology somewhere during the post-planning phase, which can be devastating for a project as it adds further complexity to the project by introducing new problems during the transition process. Moreover, with an existing range of tools aimed at managing this process transition, none of them have been thoroughly evaluated, which in turn forms the problem that this thesis is centred around.

This thesis explores a method transition scenario and evaluates a Software Process Improvement method oriented around the problems that the improvement process is aiming to solve. The goal with this is to establish if problem oriented Software Process Improvement is viable as well as to provide further data for the extensive research that is being done in this field. We wish to prove that the overall productivity of a software development team can be increased even during a project by carefully managing the transition to new methods using a problem driven approach.

The research method used is of qualitative and inductive character. Data is collected by performing a case study, via action research, and literature studies. The case study consists of iteratively managing a transition over to new methods, at an organization in the middle of a project, using a problem driven approach to Software Process Improvement. Three iterations of method improvement are applied on the project and each iteration acts as an evaluation on how well Problem Driven Software Process Improvement works.

By using the evaluation model created for this degree project, the researchers have found that problem driven Software Process Improvement is an effective tool for managing and improving the processes of a development team. Productivity has increased with focus on tasks with highest priority being finished first. Transparency has increased with both development team and company having a clearer idea of work in progress and what is planned. Communication has grown with developers talking more freely about user stories and tasks during planning and stand-up meetings. The researchers acknowledge that the results of the study are of a limited scope and also recognize that further evaluation in form of more iterations are needed for a complete evaluation.

**Keywords**

Problem driven Software Process Improvement, agile process management, process transition
Sammanfattning

Mjukvaruutveckling växer ständigt i komplexitet och många nya verktyg har tagits fram i syfte att hantera detta. Även med denna växande ström av verktyg och metoder, kämpar företag ofta med hur de ska implementera en ny utvecklingsprocess, speciellt när det gäller agila metoder. Den vanligaste orsaken till detta är att de agila metoderna implementeras ad hoc, utan att fullständigt betrakta konsekvenserna som detta resulterar i. Detta leder till arbetslag som försöker rätta till deras val av metoder någonstans efter planeringsfasen, vilket kan vara förödande för projekt då det tillfogar ytterligare komplexitet genom att introducera nya problem under övergångsfasen. Vidare finns det idag ett antal existerande processförbättringsverktyg som alla har gemensamt att de inte blivit noga utvärderade. Denna brist på utvärdering är sålunda det problem som den här avhandlingen bygger på.

Den här avhandlingen utforskar detta scenario och utvärderar en förbättrings metod för mjukvaruprocesser med utgångspunkt i de problem som kan uppstå till följd av förbättrings metoden. Målet är att etablera riktlinjer för hantering av olika situationer samt bistå med information för vidare undersökning inom området. Vi vill bevisa att den övergripande produktiviteten hos ett mjukvaruutvecklingsteam kan öka även under pågående projekt genom att noga hantera och styra en övergång till nya arbetsmetoder.

Forskningsmetoden som användes var av kvalitativ och induktiv karaktär. Information hämtades in genom en fallstudie, via aktionsforskning samt litteraturstudier. Fallstudien utgjordes av hantering av övergång till nya arbetsmetoder hos en organisation som var mitt i ett projekt genom att applicera etablerade processmodeller samt analysera hela processen.

Genom att använda den utvärderingsmodell som skapades för det här projektet har forskarna kommit fram till att den problemdrivna processförbättringsmetoden är ett effektivt verktyg för att styra och förbättra mjukvaruprocesser hos ett utvecklingsteam. Produktiviteten har ökat med fokus på att arbete med högst prioritet blir klart först. Transparensen har ökat då både utvecklingsteamet och företaget har bättre inblick i planerat och pågående arbete. Kommunikationen har förbättrats med medarbetare som pratar mer obehindrat om tasks och user stories under planerings- och stand up-möten. Studenterna som skrivit den här avhandlingen har tagit i beaktning att resultaten av den här forskningen är begränsade samt föreslår att vidare forskning i form av fler iterationer behövs för en fullständig utvärdering.

Nyckelord

Problemdriven processförbättring, agila processhanteringsmetoder, processövergång
# Table of contents

1 Introduction.............................................................................................................................................. 3  
1.1 Problem.............................................................................................................................................. 3  
1.2 Purpose ............................................................................................................................................... 4  
1.3 Research question ................................................................................................................................. 4  
1.4 Goal ..................................................................................................................................................... 4  
1.5 Research methodology ......................................................................................................................... 4  
1.6 Delimitations ....................................................................................................................................... 5  
1.7 Beneficiaries and ethics ......................................................................................................................... 5  
1.8 Outline ............................................................................................................................................... 5  

2 Process Improvement ............................................................................................................................. 7  
2.1 Reference Model Driven SPI ............................................................................................................... 8  
2.2 Problem Driven SPI ............................................................................................................................. 10  
2.3 Related Work ...................................................................................................................................... 11  

3 Research Methodology ........................................................................................................................ 13  
3.1 Overview of the Research Strategy .................................................................................................... 13  
3.2 Research Phases ................................................................................................................................. 14  
3.2.1 Literature Study Phase ..................................................................................................................... 14  
3.2.2 Preparation and Case Study ........................................................................................................... 14  
3.2.3 Incremental Process Improvement with Action Research .............................................................. 15  
3.3 Research Type and Methods ............................................................................................................. 15  
3.3.1 Qualitative Research ......................................................................................................................... 15  
3.3.2 Inductive Reasoning ........................................................................................................................ 16  
3.3.3 Alternative Research Methodology ................................................................................................ 16  
3.4 Research Instruments ........................................................................................................................ 17  
3.4.1 Process Management Methods ....................................................................................................... 17  
3.4.2 Process Improvement Methods ....................................................................................................... 17  
3.4.3 Literature ....................................................................................................................................... 17  
3.4.4 Evaluation Model ............................................................................................................................. 18  
3.4.5 Interviews ...................................................................................................................................... 18  
3.5 Sampling Method ............................................................................................................................... 18  
3.6 Experience Gained .............................................................................................................................. 18  

4 Process Improvement Evaluation Model ............................................................................................ 20  
4.1 Overview ............................................................................................................................................. 20  
4.2 Evaluation Criteria .............................................................................................................................. 20  
4.2.1 Productivity .................................................................................................................................... 21  
4.2.2 Transparency ................................................................................................................................. 21  
4.2.3 Communication .............................................................................................................................. 21  
4.2.4 Well-being .................................................................................................................................... 21  
4.2.5 Interview Questions ....................................................................................................................... 21  

5 Results of Problems Driven Software Process Improvement .......................................................... 24  
5.1 Preparations for incremental process improvement ........................................................................... 24  
5.1.1 Overview of course of action ......................................................................................................... 24  
5.1.2 Process problems identified ........................................................................................................... 24  
5.2 Incremental problem driven Software Process Improvement .......................................................... 25  
5.2.1 Iteration 1 of process improvement ............................................................................................... 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.2</td>
<td>Iteration 2 of process improvement</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Iteration 3 of process improvement</td>
</tr>
<tr>
<td>6</td>
<td>Results summary and evaluation with discussion</td>
</tr>
<tr>
<td>6.1</td>
<td>Incremental process improvement results</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Summary of Iteration 1</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Summary of Iteration 2</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Summary of Iteration 3</td>
</tr>
<tr>
<td>6.2</td>
<td>Overview of evaluative interview results</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Productivity</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Transparency</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Communication</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Well-being</td>
</tr>
<tr>
<td>6.3</td>
<td>Evaluation and discussion</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Analysis</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Quality assurance</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion and further research</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary of research results and discussion</td>
</tr>
<tr>
<td>7.2</td>
<td>Conclusions</td>
</tr>
<tr>
<td>7.3</td>
<td>Impact of Software Process Improvement</td>
</tr>
<tr>
<td>7.4</td>
<td>Future work</td>
</tr>
</tbody>
</table>

References

Appendix A – Glossary of agile management terms
1 Introduction

The software development process can be divided into two main parts; software engineering and software management. The software engineering aspect, also known as the software lifecycle, consists of requirement analysis, system design, implementation, integration and testing, deployment, as well as operation and maintenance [1, p. 12]. These individual activities have evolved over time to meet more modern standards, but the overall structure of the activities has remained.

Although agile software development methods can be dated back as far as 1957 [2, p. 47], the majority of today’s most recognized methods such as Scrum, Kanban and XP were introduced in the mid 1990’s, quickly gaining a large following in the software industry [3, p. 27]. Common attributes of these methods are that the software lifecycle is carried out in an iterative manner where each iteration produces quick feedback, a deliverable part of the product as well as enough documentation for refinement and development of the product in the next iteration.

The agile methods of today are implemented with varying degree, with companies and organization either using them as a tool for guidance, otherwise called the adaptive approach, or by rigorously following the rules of the method, implementing the regulatory approach [4, p. 8].

Regardless whether an adaptive or regulatory approach is implemented, problems occur during multitasking, process transitions and foremost due to lack of theoretical understanding [4, p. 8]. The health of above mentioned factors are dependent on how initiated management, team leaders and development team are in both practical and theoretical process methodology.

1.1 Problem

To successfully implement and customize process management methods is no easy feat, and often introduce new complexity and overhead to an already problematic situation. Therefore, a model that helps the team reach a streamlined and well working methodology should be used. Today there are several different tools that does this, all falling under the category of Software Process Improvement. The problems is that Software Process Improvement, especially Problem Driven SPI, focusing on improving the process with the current problems as a basis, have not been evaluated on a large enough basis.
1.2 Purpose
The purpose of this thesis is to evaluate Software Process Improvement that is built from a problem perspective. The evaluation will be performed by following a set of activities that iteratively will help a team to adapt new methods to their current set of development processes using observed problems that the team is experiencing.

1.3 Research question
The research question that guided the overall research explores the lack of evaluated problem driven process improvement models and was defined as:

How well does problem driven model for Software Process Improvement work?

1.4 Goal
The goal with the thesis is to provide a basis for further evaluation in the problem area so that a valid method for implementing new methods using a problem driven approach can be used.

1.5 Research methodology
To guide the research and work conducted in this thesis, a set of research methods and methodologies were chosen in order to achieve correct and well-founded results.

The research was mainly of qualitative character with corresponding qualitative research methods such as case study and action research. During the case study, the researchers gathered information using interviews, literature studies and observations, whereas the during the action research the researchers both observed and participated in the iterations of the incremental process improvement. This led to a useful combination of academic and practical research.

Throughout the entire research, a set of literature studies were conducted in parallel with the research phases. During the case study, literature concerning agile process management as well as process improvement was read in order to prepare for the iterative process improvement. Towards the end of the action research, related work concerning process improvement and software method adoption, were studied in order to create the evaluation model.

As the research was of qualitative character, inductive reasoning with an interpretivism approach was chosen. Thus, the research was initiated with focused observations and ended with new theory concerning process improvement.
1.6 Delimitations
This study has been done at a single organization, observing a single team during one project. Therefore, the results presented will be incomplete and catered towards this specific team. More research has to be conducted before a valid evaluation of Problem Driven Software Process Improvement can be established.

1.7 Beneficiaries and ethics
The outcome of this research is meant to be beneficial for companies, organizations and groups that are in need of improving or re-organizing their software process. The implementation and results of this research can be utilized as a guide that ultimately helps the beneficiaries focus on business value, which in turn could lead to experiencing positive changes in productivity, transparency and communication. Furthermore, the writers of this thesis hope to contribute to the global research and evaluation of Software Process Improvement.

As the researchers conducted the study on the company, some sensitive information such as company long term strategic business plans and structure of systems were presented. Thus, a confidentiality agreement was signed between the company and the researchers. Moreover, by leaving out names of employees and interviewees, the researchers further provided confidentiality towards the contributors of the research.

1.8 Outline
This thesis consists of the following chapters:

- Chapter 2: Software Process Improvement: This chapter presents a more detailed background description of Software Process Improvement.

- Chapter 3: Research Methodology: This chapter presents an overview of the research strategy implemented to guide the researchers through the whole study.

- Chapter 4: Evaluation model: This chapter presents the evaluation model that was used for measuring process improvement, as well as a motivation for the usefulness of each of the evaluations models criteria.

- Chapter 5: Results of problems driven process improvement: This chapter presents a detailed description of the three iterations of problem driven process improvement.

- Chapter 6: Results summary and evaluation with discussion: This chapter gives a summary of the iterative process improvement as well as an overview of the results of the evaluative interviews.

- Chapter 7: Conclusions and Further Research: This chapter presents the conclusions drawn from analysing the results that were collected during this study.
2 Process Improvement

Business processes are the set of activities and routines adopted at an organization that defines how that organization conduct their business [1]. Among these, the software development process can be considered to be one of the most important, as well as one of the most complex component of a software development organization.

The software development process is the distinct activities and phases in the software engineering discipline that concerns all aspects of development and maintenance. It is a professional activity where software is developed by a team, for a specific business purpose, and generally to be used by others than the developers [2].

Managing business processes is an initiative to increase an organization's success [10] [11], by making development faster, cheaper and better [12]. There are hundreds of different tools that guides businesses in how to manage their processes. However, only some are accepted today as viable and established methods [1]. Among these, Agile and Lean methods is one category that is continuously increasing in popularity [13]. The key concept shared among these methods is iteratively creating a clear value for the business and end-user. This value, however defined, is then used as core factor when defining backlog prioritization and during project planning [14].

Each company must develop a process that is customized according to their own needs, which can change depending on its organizational size, the team that apply the process, what is being developed, who the customer is, and the company culture. Process improvement is therefore not an easy feat, as adopting established methods or models will not guarantee improvement because these local aspects might affect the results [1].

Software Process Improvement (SPI) is a catalogue of models that helps the organizations to increase their business success by guiding them during the improvement of different aspects of the processes such as quality, schedule and budget adherence [15], [16]. There are several methods of SPI to choose from and they all implement their own type of implementation cycle and range of activities. There is, however, a common core cycle among all models that is similar to “Kaizen”, the Japanese model for industrial improvement. This cycle, which is known as PDCA, consists of four activities:

- Plan - Weaknesses are identified and a process change plan is created.
- Do - Change is implemented.
- Check - Study what impact the change had.
- Act - If improvement was observed, this is now standard.
SPI methods can be divided into two main categories; Reference Model Driven SPI and Problem Driven SPI.

2.1 Reference Model Driven SPI
Reference model driven SPI is based on the implementation of already established methods and best practices advocated by an already defined model. There are three well-known models in this category: CMMI, SPICE and IDEAL. Capability Maturity Model Integration (CMMI) is one of the most well-known Reference Model Driven SPI models [17]. It was developed by the U.S Software Engineering Institute (SEI) and is built on a retired version known as CMM as an initiative to sort out some of the problems of using multiple CMMs [1], [11]. CMMI for Development (CMMI-DEV) provides a set of best practices applied to the software development process to help the organization increase the quality of their processes, and indirectly improve the quality of their product. CMMI consists of two representations: continuous or staged [15].

The continuous representation of CMMI allows the organization to select the order of improvement and use capability levels to measure process improvement. There are six capability levels, ranging from 0 to 5, were each capability level corresponds to a set of goals and specific processes [15].

The staged representation of CMMI heavily resembles the original CMM model and allows for organizational comparison across maturity levels and use a single rating to summarize process improvement. Maturity level apply to the overall organizational maturity and range from 1 to 5. Each level of maturity consists of a set of process areas [15].

Software Process Improvement and Capability Determination (SPICE), also known as ISO/IEC 15504, is another reference driven model for process improvement. SPICE provides guidelines for process assessment and capability determination [18, p. 372] and is similar to the continuous version of CMMI in its structure. Similarly to CMMI, SPICE provides the organization with
capability levels used to rate different processes. There are six of these levels and they are as following: Incomplete process, Performed process, Managed process, Established process, Predictable process, and Optimizing process [18]. Each capability level is measured by different attributes that must be applied before reaching a certain capability rating. With the help of these guidelines, the SPICE model provides the company with the means to reach a higher level of process maturity.

Both CMMI and SPICE help the organization with assessing the maturity of current processes and determining what processes needs to be implemented into the business model. They do not, however, guide the organization in how to introduce these processes. The IDEAL model is a process implementation model that specifically guides the organization in how to implement these new processes into the business model. The model consists of five phases that also make up the name: Initialize, Diagnose, Establish, Act, Leverage [19].

**Initialize:** an SPI plan is created to guide the organization, approvals and other administrative aspects are handled in this phase.

**Diagnose:** establish a baseline of the current state of the system.

**Establish:** create a solution strategy and complete the SPI action plan.

**Act:** create and implement the solutions that address the areas of improvement.

**Leverage:** Collect lessons learned, metrics defined, and information gathered for future improvement initiatives.

---

*Figure 2 – The IDEAL Model*
2.2 Problem Driven SPI

Problem driven SPI is an effort to improve processes in a customized manner by focusing on the needs of the team. This is done by isolating obstacles in the processes that prevent the business to generate its desired value, and then by adapting or altering methods the model aims to remove these problems and improve the process. There are two well-known models that are often used in problem driven SPI: QIP and Six Sigma.

The Quality Improvement Paradigm (QIP) is a continuous process improvement model that allows the organization to improve upon their processes without implementing an established reference model. This way of performing process improvement encourages the company to learn from experience by experimenting and customizing models according to the local aspects of the company, and then packaging these best practices in a reusable form [20]. The QIP model consists of two cycles: the project cycle, which provides feedback during the project, and the organizational cycle, which provides feedback after a project has finished [20].
Six Sigma is a problem driven approach to improving development processes that focuses on eliminating waste-cost by improving customer-value and efficiency [22]. It was first introduced by Motorola in the mid 1980's as a process improvement initiative by Bob Galvin, their CEO at the [22], [23]. The model defines how well a process is performing via a measure-based strategy and suggests improvements to the processes. The suggested improvements are introduced continuously via a model known as Define-Measure-Analyze-Improve-Control (DMAIC) [22], [23]:

- **Define** the problem and goal of the improvement activity
- **Measure** the current system
- **Analyze** obstacles and causes
- **Improve** the system
- **Control** the new system and ensure sustainability

### 2.3 Related Work

The theoretical information provided in this chapter regarding business processes and Software Process Improvement provides an overview of the current field and what theories we based our work on.

Business processes is an important concept for all organizations and studying them is important if the company want to keep a competitive edge. Managing these to optimize them is a core concept of Software Process Improvement, which cannot be done unless they are properly understood.

Software Process Improvement is an enormous field with a wide range of established methods and research. This has been the foundation that our thesis was built on top of, and it is therefore important to understand these concepts.

Furthermore, it is important to understand the differences between different models when evaluating one of them, so that they can be compared effectively. Therefore it is important to have knowledge about both Problem Driven and Reference Driven SPI.
3 Research Methodology

This chapter contains the research methodology utilized during the work of this thesis. Section 3.1 presents the research strategy which worked as a guide through the whole study. The research process and its phases are accounted for in Section 3.2, whereas the type of research conducted and motivation for research methods is presented in Section 3.3. Section 3.4 presents the research instruments used and Section 3.5 shows sampling method considered for data collection. Section 3.6 presents experiences gained, whereas validity threats are accounted for in Chapter 6.

3.1 Overview of the Research Strategy

The research strategy was created to guide the researchers through the whole project. Much like a business strategy used when starting a company, the research strategy worked as a plan that the researchers could use as a roadmap in the initial stages, then as a user manual for conducting and evaluating the research, as well as a retrospective index used for confirming if the research was going the planned direction.

A majority of the research strategy was produced before and in the initial stages of the research process. Research instruments such as tools taken from Scrum and Kanban as well as relevant literature were studied early in order to get a clear idea of how to start the research. Then, some parts of the research strategy such as sampling methods were decided upon in the middle of the research. Lastly, during the end of the research, an evaluation model and validity threats were considered.

As seen in the Figure 5, the research strategy is directed by three fundamental and philosophical components; Qualitative research, Inductive reasoning and Interpretivism, all of which guides how remaining methodology is implemented. Then there are the Research methods, Research phases, Research instruments, Validity threats, and Sampling method that constitute the actual research components.

Figure 5 – Research Strategy
3.2 Research Phases

This section presents the research phases that were passed during the study. With a total of four phases, starting with Literature study followed up by Case study and Action research, and ending with Evaluation and analysis, which are presented in Sections 3.2.1 - 3.2.3.

3.2.1 Literature Study Phase

In order get a steady theoretical basis at the beginning of the research, a literature study was commenced with subjects ranging from agile development methods and Software Process Improvement to related work done by both students and professionals.

The initial wide literature study helped the researchers locate and categorize what problems the development team had, thus giving a firm starting ground for the action research phase, while also providing enough theoretical understanding to begin writing the thesis.

Later on, more specific literature studies were conducted to complement each iteration in the action research depending on what problems that were found. Lastly, a study of related work was done for creating the evaluation model and preparation for analysis and discussion.

3.2.2 Preparation and Case Study

The purpose of the case study phase was to give the researchers a clear idea of how the company and development team operated. The company’s systems and workflows were inspected through interviews and by presentations given by the team leader and lead developer. During this period, the researchers also studied what process tools that were already implemented by looking at the state of the scrum board and definition of user stories while also observing the communication within the development team and the quality of their stand-up meetings. In parallel, preparatory literature studies concerning agile methods and company culture were conducted in order to prepare and find problems for the action research phase and its iterations.
3.2.3 Incremental Process Improvement with Action Research

During the action research phase, problems identified at the end of the case study phase were used for the first iteration of process improvement. In total, three iterations were succeeded, all including preparatory work of literature studies and observations, as well as participation and management by the researchers themselves.

At the end of the action research phase, an evaluation model began to take form, and interviews were conducted whose contents reflected the criteria of the evaluation model. Then, the results of the observations and interviews, as well as the researchers own experiences, contributed to the evaluation and analysis phase.

3.3 Research Type and Methods

When undertaking a degree project, it is crucial for the researchers involved to have a set of methods and methodologies to use as a roadmap for planning, conducting and validating their research [5, p. 3]. Moreover, the researchers have to contemplate and determine whether the area that is researched is formed by social contexts, proven concepts and models, by technical and statistical data, or a combination.

This section starts with presenting motivation for conducting a qualitative research, along with appropriate research methods, in Section 3.3.1. In Section 3.3.2, the concept of inductive reasoning is introduced and motivation for why it was implemented. Lastly, alternative research types and methods, and why they were not chosen, are accounted for in Section 3.3.3.

3.3.1 Qualitative Research

The research type of this study was in a large majority qualitative. During the whole research phase, known concepts such as Scrum, Kanban and Software Process Improvement have been studied and implemented, but the outcomes of these concepts, when applied to this particular real life context, have depended on people’s opinions and understanding, as well as the researcher’s own understanding of textual and social contexts.

Both case study and action research were used as qualitative research methods, while also constituting the two main research phases of the degree project. As explained in Section 3.2.2, case study as a qualitative research method provided both retrospective and current information about the company and related theoretical work, were we as researchers had to use qualitative data collection methods such as interviews and literature studies to obtain a solid knowledge base. With case study having this exploratory character, it was best utilized in the initial stages of the research [6].

Action research in turn, as mentioned in Section 3.2.3, gave the researchers the possibility to actively and iteratively participate and manage, while also observe
the company and the development team using qualitative data collection methods and inductive reasoning, thus also conforming to the interpretivism research paradigm mentioned in the research strategy. Together, the case study and the action research provided a symmetry between theoretical and practical insights.

### 3.3.2 Inductive Reasoning

There are several reasons for inductive reasoning being the best choice for a research approach during the work of this study; the company and its development team are being studied within a limited time of two months whereas the action research phase only lasts for six weeks. Moreover, the data that is being collected during observations, participation and literature studies is of qualitative character and limited scope, and may thus be evaluated in a subjective manner, where both researchers and employee’s opinions affect how theories and conclusions are being determined.

Moreover, Inductive reasoning, also called the “bottom-up” approach [7] was applied as starts with specific observations and theories where patterns may be detected that then forms a tentative hypothesis which is answered with general open-ended conclusions or theories which are merely likely [8].

### 3.3.3 Alternative Research Methodology

Although some of the data measured in this study has been of quantitative character, such as estimation of user stories and tasks in point or hours, the analysis and evaluation of this data has not been conducted with quantitative methods such as statistics or mathematics, but rather with qualitative methods explained in Sections 3.2.1-3.3.1. Moreover deductive reasoning, which is best applied to quantitative methods, could not be used as it requires absolute certain assertions [8], where for instance it can be proved that using Scrum in a development team has overall positive effects on productivity. However, as the outcome of implementing Scrum is based on human factors and opinions and not proven concepts or evidence, these assertion could not be determined.

![Figure 7 – Inductive Reasoning](image)
3.4 Research Instruments

This section presents process management and improvement methods utilized for this study in Section 3.4.1-3.4.2, then literature is accounted for in Section 3.4.3. Lastly, the evaluation model and corresponding interviews are presented in Section 3.4.4-3.4.5.

3.4.1 Process Management Methods

During the implementation phase of the action research, different parts of agile process management methods were applied in order to solve the problems identified. Since the approach of the process improvement was problem driven, the researchers would never utilize a certain management method in full, but rather the parts of it that the researchers thought would solve the particular problem for the iteration. Eventually, spanning all three iterations of the action research, tools from management methods such as Scrum, Kanban and Extreme Programming were utilized.

3.4.2 Process Improvement Methods

Whereas parts of the management methods were used to handle work during the iterations, the process improvement methods were implemented in between the iterations. As the method name implies, it is meant for analyzing and extracting signs of improvement.

Since our approach to process change was problem driven, the majority of the tools taken from process improvement methods were from the established problem driven, such as Six Sigma and the Quality Improvement Paradigm. However elements of Software Methods Adaption were also implemented.

3.4.3 Literature

The literature study was an important part of the research throughout the whole degree project. Although the researchers had some knowledge of agile development from previous courses, the theory had to be carefully rehearsed since the methods were going to be implemented in a real development team. Books concerning agile management methods such as, Kanban och Scrum: det bästa av två världar, Scrum and XP from the trenches, and Software Engineering by Ian Sommerville were read to give a broad theoretical ground at the beginning and throughout the whole project.

Literature concerning process improvement had to be even more carefully studied since the concept was new to the researchers. Here, articles and related research were the main source of theoretical knowledge. The related work gave the researchers an idea of how software improvement had turned out in previous projects while also functioning as a guide during the action research.
3.4.4 Evaluation Model
The evaluation model was an important instrument used in different parts of the research. During the action research, the main criteria categories of the model worked as a focal point for determining and analyzing problems and solutions for the next iteration. The interview questions were formed to reflect the evaluation model so that the results would be appropriate and not produce irrelevant feedback. Lastly, the evaluation model and its criteria were used in the final stages of the research as reference for final evaluation and discussion.

3.4.5 Interviews
Before and during the case study phase, informal and semi-structured interviews were held with the product owner and lead developer in order to get a quick and wide overview of both company culture and systems. Later on, during the action research and evaluation phases, more structured interviews were conducted with pre-planned questions reflecting the evaluation model’s criteria. The majority of the interviews were individual, however, unstructured group interviews were also held at retrospective meetings at the end of each sprint to get initial feedback concerning the process improvement.

3.5 Sampling Method
The selection of interviewees for all of the interviews were done with convenience sampling [9]. This non-probability method was implemented simply because all members of the development team had sufficient background knowledge to provide both subjective opinions and could provide enough theoretical relevant feedback concerning process change and agile methods. In addition to convenience, judgment sampling, which is an extension of convenience sampling [9], was applied when interviewing the team leader about more theory heavy questions during the final evaluation period, since he had previous experience in both practical and theoretical process improvement cases.

3.6 Experience Gained
At first, the main focus of this research was on implementing agile development methods such as Scrum and Kanban, and then evaluating how difficult that process would be. Then, after a tutoring session, the researchers were recommended to narrow their scope to a more precise problem area, which led to focusing on software requirements and how they were handled. This approach was decided to be narrow enough, and after talks with the team leader, we decided to focus on user stories and how they could be used to improve both the internal issue tracker system and tasks managed in sprints.

After a second tutoring session, the researchers were recommended to utilize Software Process Improvement as an evaluation tool, which in turn would allow using agile methods, software requirements, process change and more in a controlled manner, which also could be systematically evaluated.
4 Process Improvement Evaluation Model

This chapter gives an overview of the evaluation model created for this thesis, as well as an in detail description of each of the evaluation criteria.

4.1 Overview

To evaluate the results of this study we decided to focus on the improvement displayed during the SPI process by the team. This evaluation was done by us participating in the project using action research as well as mapping any findings against four evaluation criteria specifically chosen to measure improvement in development. During the study the researchers collected data in the form of status reports by several correspondences with the team, such as Sprint retrospect’s and interviews, as well as by performing an analysis by observing and participating in the development process.

4.2 Evaluation Criteria

This section presents each of the evaluation criteria and motivates their relevance to this thesis as well as presents an overview of the interview questions related to each criteria. Each of the criteria should in one way or another be associated with increasing the generated business value of the company.

![Evaluation Model Diagram]

Figure 8 – Evaluation Model
4.2.1 Productivity
Productivity refers to the ratio between the quantity of product created and the cost spent creating it. Increasing the productivity in software development means increasing the velocity in which the team can develop software programs. Many improvements can be done to the productivity so that the team can reach their full potential, however only increasing the productivity will not inherently increase the business value, as what is being produced is equally important.

4.2.2 Transparency
In process management transparency is the aspect of openness, communication, and accountability [24]. These qualities improves productivity and communication by making sure everyone on the team and in the organization knows as much as possible about the work that is being done. Having good transparency in a company makes sure any dependencies between activities is easily identified so that a higher level of synergy can exist in the company.

4.2.3 Communication
Communication is one of the key aspects of any team based work. The more effective it is, the better, and faster, the team can work together. This further improves productivity, as a higher collaboration effort can be achieved. Effective communication also decreases the amount of waste produced, as the team more easily can identify problems and unwanted features, and therefore make sure the project is oriented correctly.

4.2.4 Well-being
With more organized work, there is less stress, which directly benefits the well-being of each member of the team. With little stress, it is easier for each team member to focus on their respective task, as the temptation to multitask is reduced. This also makes it easier for the team to handle critical situations such as last minute high priority additions to the project.

4.2.5 Interview Questions
When finalizing the status report of the case study each member of the development team participated in an interview where they were asked to answer questions regarding the evaluation criteria. The interview questions have been translated to better fit with the text, but each interview was conducted in Swedish.

Most of the questions followed the same pattern, which was:
- How has <evaluation criteria> changed since the implemented changes to the process was introduced?
This question was asked after each evaluation criteria had been explained to the interviewee. However, some of the criteria required more questions to properly be evaluated, these were:

**Productivity**
- *How has the productivity changed after the implementation of...*
  - Pair Programming?
  - Test Driven Development?
  - Organized estimation of tasks?
  - Separation of tasks and user stories?

**Communication**
- *How has the communication changed since...*
  - Stricter estimation was implemented?
  - We started with planning poker?
  - Defining user stories and demos together?
5 Results of Problems Driven Software Process Improvement

This chapter presents the results of the Problem Driven Software Process Improvement initiative. In Section 5.1, problems identified during the preparatory phase are accounted for. Section 5.2 presents the outcome of the incremental process improvement.

In Appendix A, a glossary for terms concerning agile process management tools mentioned in this chapter is presented. A summary of the results accounted for in this chapter as well as an overview of the interview answers is presented in Chapter 6.

5.1 Preparations for incremental process improvement

In Section 5.1.1, an overview of the work conducted to identify initial process problems is given. Then, Section 5.1.2 presents the problems that were found.

5.1.1 Overview of course of action

The initial week of the research was spent preparing the iterative process improvement phases. During this time, the researchers studied theory concerning agile process management and process improvement methods in order to be able to identify problems and suggest solutions. In parallel, the researchers observed the development team’s process tools, such as state of Scrum board, stand-up meetings, retrospective and planning meetings, as well as communicational health within the development team.

5.1.2 Process problems identified

Each of the identified problems were sorted into long term and short term problems. The long term problems were those that had existed for an indeterminable period of time, whereas the short term problems were a direct result of changes made during the previous iteration of process improvement. As no previous iteration of process improvement had been conducted yet, all initial problems found were long term and are presented in Table 1.

<table>
<thead>
<tr>
<th>Table 1, Process problems found during preparatory phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>No estimation of tasks or user stories</td>
</tr>
<tr>
<td>No clear prioritization of user stories</td>
</tr>
<tr>
<td>No evident distinction between user stories and tasks</td>
</tr>
<tr>
<td>User stories are not divided into tasks</td>
</tr>
<tr>
<td>Casual commitment; committing to work that is not planned in the sprint</td>
</tr>
<tr>
<td>In-house requests need to be structured as a user story</td>
</tr>
<tr>
<td>No demo of user story when finished</td>
</tr>
</tbody>
</table>

Table 1 - Process problems found during preparatory phase
Table 1, *Process problems found during preparatory phase*, was produced towards the end of the preparatory phase and functioned as the input for Iteration 1 of the incremental process improvement as shown in Figure 9.

### 5.2 Incremental problem driven Software Process Improvement

This section presents the results of the incremental process improvement phase. Each iteration followed the PDCA cycle model described in chapter 3, in order to identify problems, suggest solutions, assess improvement and sustain the new process standard.

Section 5.2.1 shows the results of Iteration 1, with suggested solutions to problems found during the preparatory phase, as well as results of the implemented solutions. Section 5.2.2 and 5.2.3 presents the outcome of Iteration 2 and 3 with corresponding new problems found and results of the implemented suggested solutions.

#### 5.2.1 Iteration 1 of process improvement

After reviewing the newly created backlog of problems identified during the preparatory phase, the researchers prioritized over which of the problems to solve and suggested solutions consisting of agile process management tools.

Table 2 presents an overview of the prioritized problems and their corresponding suggested solutions. The table also shows which problems that were not considered for solving during Iteration 1.
Table 2. Prioritized problem list with suggested solutions for Iteration 1

<table>
<thead>
<tr>
<th>Problem</th>
<th>Suggested solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>No clear prioritization of user stories</td>
<td>Prioritize user stories during sprint planning meetings</td>
</tr>
<tr>
<td>No estimation of tasks or user stories</td>
<td>• Estimate user stories with planning poker</td>
</tr>
<tr>
<td></td>
<td>• Implement burn-down chart to show sprint velocity</td>
</tr>
<tr>
<td>User stories are not divided into tasks</td>
<td>Discussion of what tasks belonging to each user story during planning meeting</td>
</tr>
<tr>
<td>No evident distinction between user stories and tasks</td>
<td>Educate development team. Use distinction in planning meetings and on Scrum board</td>
</tr>
<tr>
<td>Casual commitment; committing to work that is not planned in the sprint</td>
<td>No suggested solution for Iteration 1</td>
</tr>
<tr>
<td>In-house requests need to be structured as a user story</td>
<td>No suggested solution for Iteration 1</td>
</tr>
<tr>
<td>No demo of user story when finished</td>
<td>No suggested solution for Iteration 1</td>
</tr>
</tbody>
</table>

Before implementing the suggested solutions, a presentation of problems identified and theory about agile management tools meant to solve them was held with the development team. Then, implementation began with a planning meeting where estimation, prioritization and task discussion was conducted. Moreover, the researchers estimated a Sprint Velocity and created a Burndown chart. Towards the end of the sprint, the researchers could observe the results, which are presented in Table 3.
Table 3, Results of suggested solutions for Iteration 1

<table>
<thead>
<tr>
<th>Problem</th>
<th>Suggested solution</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>No clear prioritization of user stories</td>
<td>Prioritize user stories during sprint planning meetings</td>
<td>Solved</td>
</tr>
<tr>
<td>No estimation of tasks or user stories</td>
<td>• Estimate user stories with planning poker&lt;br&gt;• Implement burn-down chart to show sprint velocity</td>
<td>Solved</td>
</tr>
<tr>
<td>User stories are not divided into tasks</td>
<td>Discussion of what tasks belonging to each user story during planning meeting</td>
<td>Solved</td>
</tr>
<tr>
<td>No evident distinction between user stories and tasks</td>
<td>Educate development team. Use distinction in planning meetings and on Scrum board</td>
<td>Solved</td>
</tr>
</tbody>
</table>

Table 3 - Results of suggested solutions for Iteration 1

As shown in Figure 10, Table 2, a prioritized list of problems and their suggested solutions was created at the start of Iteration 1. Then, Table 3, which presents the results of the suggested solutions, was created at the end of Iteration 1.
5.2.2 Iteration 2 of process improvement

At the beginning of Iteration 2, the researchers had identified short term problems that were the result of Iteration 1, as well as having unprocessed problems identified during the preparation phase. Therefore, an updated list of unsolved problems was produced in order to prioritize which problems that were most severe and suggest appropriate solutions. The updated list, which is presented in Table 4, did not include problems that were considered completely solved during Iteration 1.

Once again, as during the initial phase of Iteration 1, the researchers prioritized over which problems that were most important to try to solve during Iteration 2. Then solutions were suggested consisting of agile process management tools taken from Scrum, Kanban and Extreme Programming. Some of the unsolved problems remaining from the previous iteration could now easier be processed since the researchers had had a whole sprint to contemplate over an appropriate solution. The list of prioritized problems and their suggested solutions is presented in Table 5.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Found during</th>
</tr>
</thead>
<tbody>
<tr>
<td>Under commitment with too few user stories planned in the sprint, developers wondering what to do towards end of sprint</td>
<td>Iteration 1</td>
</tr>
<tr>
<td>Stop in Scrum board flow with too many tasks stuck in &quot;merge request&quot; column</td>
<td>Iteration 1</td>
</tr>
<tr>
<td>Unclear titles of user stories</td>
<td>Iteration 1</td>
</tr>
<tr>
<td>Casual commitment; committing to work that is not planned in the sprint</td>
<td>Preparation phase, before Iteration 1</td>
</tr>
<tr>
<td>In-house requests need to be structured as a user story</td>
<td>Preparation phase, before Iteration 1</td>
</tr>
<tr>
<td>No demo of user story when finished</td>
<td>Preparation phase, before Iteration 1</td>
</tr>
</tbody>
</table>

Table 4 - Updated list of unsolved problems for Iteration 2
<table>
<thead>
<tr>
<th>Problem</th>
<th>Suggested solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Under commitment with too few user stories planned in the sprint,</td>
<td>Make a visible backlog of user stories next to the Scrum board</td>
</tr>
<tr>
<td>developers wondering what to do towards end of sprint</td>
<td></td>
</tr>
<tr>
<td>Stop in Scrum board flow with too many tasks stuck in &quot;merge request&quot;</td>
<td>• Utilize Kanban numbers to reduce work flow congestion</td>
</tr>
<tr>
<td>column</td>
<td>• Implement Pair Programming to reduce merge request time</td>
</tr>
<tr>
<td></td>
<td>• Implement Test Driven Development to simplify code review.</td>
</tr>
<tr>
<td>Unclear titles of user stories</td>
<td>• Make titles less technical and more self-explanatory</td>
</tr>
<tr>
<td></td>
<td>• Add subtitles to which product they belong</td>
</tr>
<tr>
<td>Casual commitment; committing to work that is not planned in the sprint</td>
<td>Implement prioritization model for unplanned tasks</td>
</tr>
<tr>
<td>No demo of user story when finished</td>
<td>Prepare and present demo of user stories related to the sprint goal</td>
</tr>
<tr>
<td>In-house requests need to be structured as a user story</td>
<td>No suggested solution for Iteration 2</td>
</tr>
</tbody>
</table>

Table 5 - Prioritized problem list with suggested solutions for Iteration 2

As with the previous iteration, the researchers studied the effects of the implemented solutions by observations and participating in planning, daily stand-up and retrospective meetings held during Iteration 2. The outcome which is presented in Table 6, now consisted of solved, partially solved and unsolved results.
Table 6, Results of suggested solutions for Iteration 2

<table>
<thead>
<tr>
<th>Problem</th>
<th>Suggested solution</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Under commitment with too few user stories planned in the sprint, developers wondering what to do towards end of sprint</td>
<td>Make a visible backlog of user stories next to the Scrum board</td>
<td>Solved</td>
</tr>
<tr>
<td>Stop in Scrum board flow with too many tasks stuck in “merge request” column</td>
<td>• Utilize Kanban numbers to reduce work flow congestion&lt;br&gt;• Implement Pair Programming to reduce merge request time&lt;br&gt;• Implement Test Driven Development to simplify code review.</td>
<td>Solved</td>
</tr>
<tr>
<td>Unclear titles of user stories</td>
<td>• Make titles less technical and more self-explanatory&lt;br&gt;• Add subtitles to which product they belong</td>
<td>Solved</td>
</tr>
<tr>
<td>Casual commitment; committing to work that is not planned in the sprint</td>
<td>Implement prioritization model for unplanned tasks</td>
<td>Not solved</td>
</tr>
<tr>
<td>No demo of user story when finished</td>
<td>Prepare and present demo of user stories related to the sprint goal</td>
<td>Partially solved</td>
</tr>
</tbody>
</table>

Table 6 - Results of suggested solutions for Iteration 2

As seen in Figure 11, Table 4, updated list of unsolved problems for Iteration 2 and Table 5, Prioritized problem list with suggested solutions for Iteration 2 were produced at the start of Iteration 2, whereas Table 6, Results of suggested solutions for Iteration 2 was produced at the end of the iteration.
5.2.3 Iteration 3 of process improvement

Iteration 3 was the final phase of the incremental process improvement. Although some problems remained from the previous two iterations, Iteration 3 also functioned as an evaluative phase where the researchers ascertained that the already implemented solutions still functioned. Additionally, it was towards the end of this iteration that final evaluative interviews were held with development team members and product owner.

As can be seen in Table 7, only one short term problem was identified during the previous iteration. Two of the remaining process problems had already been processed during previous iterations but with unsatisfactory results. Therefore, improved solutions were suggested as can be seen in Table 8.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Found during</th>
</tr>
</thead>
<tbody>
<tr>
<td>Over commitment, too many user stories planned for sprint</td>
<td>Iteration 2</td>
</tr>
<tr>
<td>Casual commitment; committing to work that is not planned in the sprint</td>
<td>Iteration 2 and Preparation phase, before Iteration 1</td>
</tr>
<tr>
<td>In-house requests need to be structured as a user story</td>
<td>Preparation phase, before Iteration 1</td>
</tr>
<tr>
<td>No demo of user story when finished</td>
<td>Preparation phase, before Iteration 1</td>
</tr>
</tbody>
</table>

Table 7 - Updated list of unsolved problems for Iteration 3

<table>
<thead>
<tr>
<th>Problem</th>
<th>Suggested solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Over commitment, committing to unplanned urgent tasks</td>
<td>Estimate unplanned task and perform task switching</td>
</tr>
</tbody>
</table>
| Casual commitment; committing to work that is not planned in the sprint | • Educate about multitasking (improvement)  
  • Educate about discipline in request acceptance (improvement)  
  • Prioritization model for unplanned tasks                             |
| In-house requests need to be structured as a user story                | Restructure in-house request template                                              |
| No demo of user story when finished                                   | No suggested solution for Iteration 3                                              |

Table 8 - Prioritized problem list with suggested solutions for Iteration 3
During Iteration 3, a majority of the development team became devoted to solving the unplanned urgent task which was introduced in Iteration 2. The researchers tried to estimate the new task and performed task switching in order to balance sprint velocity, but with minor results. In parallel, the researchers started reforming the in-house request template, but the project never got past the prototype stage. Instead focus now lay on managing and analysing the already implemented solutions from the previous iterations, as well as preparing for handing over the roles of being Scrum masters.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Suggested solution</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Over commitment, committing to unplanned urgent tasks</td>
<td>Estimate unplanned task and perform task switching</td>
<td>Partially solved</td>
</tr>
<tr>
<td>Casual commitment; committing to work that is not planned in the sprint</td>
<td>• Educate development team about multitasking</td>
<td>Not solved</td>
</tr>
<tr>
<td></td>
<td>• Educate about discipline in request acceptance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Implement prioritization model for unplanned tasks</td>
<td></td>
</tr>
<tr>
<td>In-house requests need to be structured as a user story</td>
<td>Restructure in-house request template</td>
<td>Not solved</td>
</tr>
</tbody>
</table>

Table 9 - Results of suggested solutions for Iteration 3
Figure 12 shows where in Iteration 3 each of the tables were produced. Table 7, updated list of unsolved problems for Iteration 3 and Table 8, Prioritized problem list with suggested solutions for Iteration 3 were produced at the start of Iteration 3, whereas Table 9, Results of suggested solutions for Iteration 3 was created at the end of Iteration 3.
6 Results summary and evaluation with discussion

Section 6.1 presents a summary of the incremental Software Process Improvement, whereas a compilation of the conducted evaluative interviews is presented in Section 6.2. A detailed account of the iterative process improvement is presented in Chapter 5. Lastly, Section 6.3 presents evaluation and discussion concerning the iterative process improvement as well as quality assurance.

6.1 Incremental process improvement results

This section presents an overview of the results of the incremental process improvement. Each of the three iterations, with identified process problems, suggested solutions, when each problem was found as well as results of the suggested solutions are presented in Sections 6.1.1-6.1.3.

6.1.1 Summary of Iteration 1

At the initial phase of Iteration 1, the researchers reviewed the process problems found during the preparatory phase. Then, a prioritized list of problems that were to be processed for the iteration was created with suggested solutions in form of agile process management tools. The results of the iteration is presented in Table 10.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Suggested solution</th>
<th>Found during</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>No clear prioritization of user stories</td>
<td>Prioritize user stories during sprint planning meetings</td>
<td>Preparation phase</td>
<td>Solved</td>
</tr>
</tbody>
</table>
| No estimation of tasks or user stories | • Estimate user stories with planning poker  
                                           • Implement burn-down chart to show sprint velocity | Preparation phase | Solved |
| User stories are not divided into tasks | Discussion of what tasks belonging to each user story during planning meeting | Preparation phase | Solved |
| No evident distinction between user stories and tasks | Educate development team. Use distinction in planning meetings and on Scrum board | Preparation phase | Solved |

Table 10 - Summary of Iteration 1
6.1.2 Summary of Iteration 2

At the beginning of Iteration 2, an updated list of process problems was created. The prioritized list now consisted of both newly discovered and previously unprocessed problems with corresponding suggested solutions. The results, which now included solved, partially solved and unsolved problems, is presented in Table 11.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Suggested solution</th>
<th>Found during</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Under commitment with too few user stories planned in the sprint, developers wondering what to do towards end of sprint</td>
<td>Make a visible backlog of user stories next to the Scrum board</td>
<td>Iteration 1</td>
<td>Solved</td>
</tr>
</tbody>
</table>
| Stop in Scrum board flow with too many tasks stuck in "merge request" column | • Utilize Kanban numbers to reduce work flow congestion  
  • Implement Pair Programming to reduce merge request time  
  • Implement Test Driven Development to simplify code review. | Iteration 1  | Solved    |
| Unclear titles of user stories                                         | • Make titles less technical and more self-explanatory  
  • Add subtitles to which product they belong                           | Iteration 1  | Solved    |
| Casual commitment; committing to work that is not planned in the sprint | Implement prioritization model for unplanned tasks                                 | Preparation phase, before Iteration 1 | Not solved |
| No demo of user story when finished                                     | Prepare and present demo of user stories related to the sprint goal                | Preparation phase, before Iteration 1 | Partially solved |

Table 11 - Summary of Iteration 2
Table 12, Summary of Iteration 3

<table>
<thead>
<tr>
<th>Problem</th>
<th>Suggested solution</th>
<th>Found during</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Over commitment, committing to unplanned urgent tasks</td>
<td>Estimate unplanned task and perform task switching</td>
<td>Iteration 2</td>
<td>Partially solved</td>
</tr>
</tbody>
</table>
| Casual commitment; committing to work that is not planned in the sprint | • Educate development team about multitasking  
• Educate about discipline in request acceptance 
• Implement prioritization model for unplanned tasks | Iteration 3, Iteration 2, Iteration 1, Preparation phase, before Iteration 1 | Not solved       |
| In-house requests need to be structured as a user story      | Restructure in-house request template                     | Preparation phase, before Iteration 1 | Not solved  |

Table 12 - Summary of Iteration 3

6.1.3 Summary of Iteration 3

During the final iteration of process improvement, the researchers shifted focus on sustaining and managing already implemented suggested solutions. However, some problems that were unresolved from previous iterations now received improved suggested solutions. It was also during this iteration that individual evaluative interviews were held with product owner and development team member. The results of Iteration 3 is presented in Table 12.

6.2 Overview of evaluative interview results

The individual interviews were held during the final iteration of the incremental process improvement. In total, five people were interviewed, whereas one was a system administrator, three were software developers and one was the product owner and technical chief of the company. The interviewees had various experiences working with agile process methods and years working with information technology ranged from 3.5 to 16 years.

All of the interviews were designed to obtain individual feedback concerning how the evaluation model’s four criteria; productivity, transparency, communication and well-being had changed over the course of the incremental process improvement.
6.2.1 Productivity
Although none of the interviewees thought that more work was being finished within the sprints compared to before the process improvement had begun, four of five people thought that they now worked with more prioritized tasks. Moreover, two of five people thought it was more evident when work was actually finished.

Furthermore, two of five people thought that the breakdown of user stories into tasks had led to smaller but more important features being finished faster. However, one person thought that there should be more focus on the user story's business value during the planning meetings, whereas another person thought that prioritization of user stories should be done separate from the planning meetings.

Lastly, two of five people thought that the problems of causal commitment had been reduced following the introduction of prioritization of user stories, since it now was more evident what had highest priority during the sprint.

6.2.2 Transparency
Concerning transparency, three of five people thought it had increased within the development team. Especially the system administrator now had more insight of what the rest of the team was working with. Furthermore, the people who had not experienced any change in transparency motivated this by adding that they believed they had just as good track of work in progress now as before, but did appreciate the better standard of the Scrum board.

Lastly, none believed transparency had increased towards the rest of the company since no one at the other departments had been instructed about the method changes that had been implemented.

6.2.3 Communication
Thanks to the introduction of dividing user stories into tasks, three of five people thought that the communication within the team had become better, since they now discussed more openly about the work. However, two of five people thought the communication was the same, whereas everyone thought the communication had always been good. Moreover, one person thought Pair Programming had led to better communication and two of five believed the that they now better perceived the dependencies between the user stories in progress.

6.2.4 Well-being
Two of five people thought that overall stress had reduced since it now was clearer what each team member was working with during the sprints. However, all interviewees felt just as motivated now as before. Additionally, three of five people believed that the introduction of Pair Programming had made it possible to experience new programming languages and thus broadening the competence, although one person thought Pair Programming was awkward but could be improved with better scheduling.
Lastly, four of five people were concerned with the sustainability of the improved Scrum standard and expressed worry about who was going to be responsible after the researchers were finished at the company.

6.3 Evaluation and discussion

This section presents a subjective evaluation done by the researchers. In Section 7.1, an analysis of how well the Software Process Improvement worked is given and Section 7.2 discusses validity threats concerning quality assurance.

6.3.1 Analysis

In this section, we analyze the results of our work. The analysis was conducted by collecting data from interviews combined with our own observations and problem identifications. All of the data collected have been mapped towards the defined evaluation criteria.

How to measure success in a Software Process Improvement initiative varies between models and can therefore not easily be compared. However, in this study we have constructed a set of evaluation criteria that can be used as a success metric regardless of methods used. This increases the transferability of the research and made it so that we could discuss the results with the team without educating them in more abstract concepts such as maturity levels.

During the research we conducted several interviews to collect data and get a status report on the business process improvement. This was done partly using the Sprint retrospectives from Scrum, where we discussed the results of implementing tools and methods as well as what should be kept or removed from the business process. When we finalized our research we also conducted individual interviews with each of the members to get a last overview of their thoughts and concerns. This approach allowed us get data during the process that we could compare with our own observations and helped in confirming that we were correct in our conclusions.

The Software Process Improvement was conducted in an iterative manner in parallel to the Sprints that the team conducted their work during. Before each Sprint the researchers would identify problems with the current software development process, these problems were then prioritized and solutions were suggested. To increase the successfulness of introducing new methods we decided to not implement all of the solutions so that the impact of change could be properly monitored. This way, if any of the implemented solutions had a negative impact on the team, it could quickly be identified and solved in the next iteration.

The first iteration was introduced with only long-term problems, as no problems related to the impact of implementing new methods could have been identified. This also meant that the first iteration overall had the most significant change to the process. However, some of the long-term problems that were identified in the preparatory phase introducing the iteration was not handled until later, as these were deemed to have a lower priority.
The problems identified for the first iteration was deemed to be a result of the ad-hoc implementation of agile methods that the company had implemented, as well as a result of the lack of theoretical understanding of the methods across the team. Therefore, before implementing any solutions we held a lecture on the theoretical aspects of Software Process Improvement as well as on agile methodology.

For the second iteration we discovered short-term problems directly related to the previous implementations. Some of these problems, such as under-commitment, could have existed during the preparatory phase but were not visible until after because of the implementation of a method, such as velocity and burndown charts. Furthermore, after the implementation and status report of this iteration, one of the problems were only partially-solved. This is because even if a solution was suggested and implemented, it was not carried out fully and therefore the problem could not be considered as solved, even though it had a significant impact on improving the process.

During the second iteration the team got a critical task after committing to a Sprint. This resulted in over-commitment as this task had to be committed to the locked Sprint backlog. This led to the implementation of task-switching and education of how to handle this type of unplanned features in the future. However, because of the implementation of the new methods and organization of the development process, the team could handle the critical task and identify how it would impact the Sprint and notify stakeholders that there would be a change in deliverables early.

6.3.2 Quality assurance

Quality assurance is concerned with the validation and verification of the research material. Because the approach was of a qualitative nature, this research was exposed to quality threats such as validity, transferability, dependability, and conformability. In this chapter, we will discuss these threats and how they have been accounted for.

Validity determines the credibility of the interviews and the collected answers. Each of the interviewees was a member of the team and had actively participated in the business processes that the improvement project managed. Furthermore, each of the interviewed subjects had years of experience with working with software development.

Transferability refers to what degree the study’s findings can be used beyond the scope of this case study. This research evaluates the usefulness of an established model and uses a general set of evaluation criteria so that the results can be compared across different SPI models. This makes it possible to continue the research with another scope, in another context, and still compare the results.
Dependability is the process of judging the correctness in conclusions, it concern the repeatability of the study. However, qualitative research often lacks in dependability, and this research should be performed in different contexts before the correctness in the conclusion can be fully confirmed.

Conformability confirms that the research has been performed in good faith without any personal assessments that affected the results. This was done by repeating all the answers collected from the team members and confirming with them that they were correct. Furthermore, data was constantly collected from the team both in group and individually to get a complete understanding.
7 Conclusion and further research

This chapter presents the conclusions of the research drawn from the results and analysis, an impact summary of the SPI initiative, as well as recommendations for future research. Section 7.1 presents a summary of the results of the process improvement and discussion. Section 7.2 presents the conclusions, 7.3 gives a detailed explanation of the impact results, and 7.4 presents a recommendation for future work.

7.1 Summary of research results and discussion

This research has consisted of a preparatory phase followed by an incremental process improvement phase with three iterations. During the preparatory phase, the researchers observed the state company culture and process tools in order to identify process problems. Then, during each of the iterations of the problem driven process improvement, identified problems were prioritized, suggested solutions were given and results of their implementations were observed. In order to analyze and evaluate the process improvement, agile process management and process improvement literature was studied.

A qualitative research approach with corresponding research methods such as case study and action research were utilized in order to balance both academic and practical studies. Inductive reasoning in combination with an interpretivism approach has directed the research, starting with focused observations of the development team and leading to new theory concerning process improvement.

Following the evaluation model created to measure the success of the process improvement, the productivity, transparency, communication and well-being have been analyzed through observations and interviews. Most of the process problems were identified during the preparatory phase and solved during the first iteration. However new problems arose during the following iterations and were solved with varying results.

7.2 Conclusions

To answer our research question, we performed a case study by managing the development process of a software development team, which was done using Problem Driven SPI. The improvement initiative was performed in an iterative manner by implementing suggested solutions to the most important of the identified problems and then monitoring the results.

From the results gathered from the interviews as well as our own observations we could conclude that problem driven Software Process Improvement indeed is a valid model for implementing new methodologies in a development team. Furthermore, we believe that focusing on the problems in the current processes and how to solve them is more effective than changing the process completely in an already working team and ongoing project. This allowed us to, in a controlled manner, monitor and manage the improvement process and make sure the team could adapt to and understand the changes.
Furthermore, the defined evaluation model was effective for monitoring the improvement in the team. We also believe that using a generalized model for evaluation instead of the defined metrics in the method will allow for easier comparison between studies as well as minimize the overhead when teaching a team without deep knowledge in the theoretical background of the used model.

Because of the limitations of our research, we had to adapt not only the way in which the maturity of the improvement process was measured but also how the company culture was handled and on what scale the improvement process was conducted. However, we believe that SPI is an effective model for process improvement even after limiting us to a small scale closed team and using generalized metrics.

### 7.3 Impact of Software Process Improvement

This section will present the impact that the process improvement initiative had on the software development team and their business processes. The impact will be measured against the evaluation criteria with status reports gathered from the retrospectives as well as the interviews.

Productivity in what was developed according to the velocity did not implicitly increase in any significant way. The development team was already working at the best of their ability and delivering features as fast as they could. However, after the implementation of backlog-prioritization the team could more easily identify and deliver the correct features faster. Furthermore, the implementation of Velocity and Burndown allowed the team to identify and track their work which improved the planning and obstacle identification before and during sprints.

However, the implementation of Test Driven Development as well as Pair Programming did introduce a slight increase in productivity, as less time had to be spent during code review, allowing code to be deployed faster, giving the team more time to actually develop new features. This also increased the code ownership, which allowed for a broader understanding of the system across the team.

The implementation of better User Stories and Tasks as well as moving the Scrum-board from a digital issue tracking system to a post-it based wall significantly increased the transparency in the team and towards management. It allowed the team to track their progress during a Sprint, as well as to identify dependencies between features. However, because of the company culture and a lack of understanding from the rest of the company regarding the task management, transparency was not increased in that aspect.

Furthermore, the implementation of a more structured development process introduced an improvement in the overall well-being among the members of the development team. Overall, this lessened the burden from planning and managing priorities, which decreased the amount of stress among the members.
of the team. This was confirmed by each member respectively during the interviews but also noticeable during the second iteration when an unplanned critical task was introduced, as the team could easily introduce the task and still know what they would be able to finish.

7.4 Future work

Because of the qualitative nature of the research and the fact that it was a case study performed on a single team, the results and conclusions are very specific for the team that the evaluation was performed on. More studies carried out at different companies and teams has to be conducted to reach a correct conclusion.

Furthermore, as Software Process Improvement is such a large field with millions being spent at different companies this type of research should absolutely be performed at more companies, in different contexts, so that even more effective methods can be constructed.
References

# Appendix A – Glossary of agile management terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint</td>
<td>The basic unit of development in Scrum. A sprint is a predefined period of time in which the team commits to working on a specific set of features.</td>
</tr>
<tr>
<td>User Story</td>
<td>A short, simple description of a feature told from the perspective of the person who requested the feature.</td>
</tr>
<tr>
<td>Burndown</td>
<td>A chart that represents work done versus time.</td>
</tr>
<tr>
<td>Demo</td>
<td>A short presentation of a feature.</td>
</tr>
<tr>
<td>In-house request</td>
<td>Features that are requested from inside the company, often from another department.</td>
</tr>
<tr>
<td>Story Point</td>
<td>An estimated measurement of time-resources spent on a user story.</td>
</tr>
<tr>
<td>Task</td>
<td>An individual part of a feature that can be developed by one person.</td>
</tr>
<tr>
<td>Velocity</td>
<td>The speed in which the development team can produce new features. This is the amount of Story Points that fit in one Sprint.</td>
</tr>
</tbody>
</table>