G

DIT347 – Final Report

- SPI plan has no relation to CRM
- Questions and metrics implicit
- Improvement steps vague
- No measurements
- Unclear which improv steps from plan were used

ANONYMOUS AUTHOR
Examination in DIT347 Software Development Methodologies
Bachelor Program Software Engineering and Management
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden

***Abstract*— This document is a template. The various components of your essay (title, headings, etc.) are already defined here. Please note that the headings must NOT change unless explicitly stated below. That is, you MUST use the predefined headings. Make sure to retain font sizes, line spacing, column width, and margins. Not following this template in either structure or layout will lead to an automatic FAIL. Remember: the criteria for attaining a grade Pass (G) and a grade Pass with Distinction (VG) are defined in the Course PM.**
You can replace this abstract with your own text.

I. Introduction

This document is written in order to report my experience and learnings from the Software Process Improvement (SPI) course as the final examination report.

This course places emphasis on the actions and mindsets of people working on software development projects and introduces us to the guidelines and methodologies created through years/decades of research and experience. These can be applied on software projects in order to create a structured workflow, provide descriptive plans, execute and improve development processes and analyze the progress and the outcome.

II. Process Applied in the Scrum Workshop

Since the workshop was set up in a way to accommodate many teams of 6-7 people to work together to build a digital Lego city (Minetest city) where the teams would build separate structures on the same map as others, my team decided to use Scrum Methodology [1] for the first workshop as it provided concrete guidelines to establish concrete plans, workflows, and analysis. The team didn't choose to go with Kanban or Extreme Programming methodologies, because the workshop was not expected to be large and complex enough to adopt rather complex and detailed guidelines. Even though a version of a Kanban board was used, nor a Work in Progress (WIP) limit was present, neither any measurement was planned.

Scrum board

why?

The team adopted user story, product and sprint backlog, incremental delivery, and pair programming practices. While the user stories provided in the product backlog by the Product Owner (PO) were expected to contain all the requirements for a functional and standalone building that can be completed in a sprint, the team was prepared to divide take up one of these tasks and divide it into smaller and easily manageable tasks in the sprint backlog. We were not aware of the fact that we were supposed to sync up with other teams in the timeline, i.e. there is a single timeline that defines the start/end times of the sprints, including the planning, development, and retrospective phases. We also did not know whether there would be a single sprint or multiple sprints in the whole workshop duration, therefore we kept the time plan abstract in that sense, so that it could be flexible enough to satisfy both situations.

schedule?

how?

The team chose a scrum master within the members, who was not a volunteer but a recommendation of another member. Since there was not any other volunteer and the chosen member was happy and content with the decision, this role has been unanimously decided. This person's responsibilities as the scrum master were to be the spokesperson of the team that communicates with the other teams and the product owner when necessary, keep track of the timeline and oversee the task status, and help the team with building tasks if he/she was available and the team needed help.

The team used a randomizer tool found on the internet to help create sub teams of 2 members each. These teams would be working in pairs in order to complement each other in case one of them didn't know how to do something, besides, a partner would be helpful in successfully completing a task, working as a double check mechanism, assuring the quality.

The team described a team retrospective phase in detail, which would be held after the development phase, where they could evaluate their development phase by rating their feelings on a scale of 10, and everybody would need to provide a couple of statements regarding what was positive or negative in the previous phase. The team also needed to vote on these statements to eventually find out the most important matter to focus on, which was presented to the team to address the issues and potential solutions.

The process that was planned [1] and how it ended up [2] is as following (- indicates the plan, + indicates what happened):

- As soon as the sprint plan starts, the scrum master contacts the PO, picks up a product backlog item, identifies the requirements, comes back to the team chat and assigns the tasks to the members.
+ It took the scrum master longer than expected to come back to the team, and the requirements weren't completely identified.
- When the development phase starts, builders would start a separate chat with their pair and start working on their task. They needed to finish their tasks before the retrospective starts.

?

+ In the first sprint, the team started collecting materials and started building their tasks, but the unclear requirements prevented the team from having the expected building. They had to break down some parts and rebuild them. Pair work did not happen in this phase, as the allocated time for development has ended right after deciding how to proceed. In the second sprint, the team paired up and completed their tasks. In the third -final- sprint, they decided to pass on the pair work since it took a considerable amount of time to switch between voice chat applications and mute/unmute themselves on the other side.

- The team retrospective phase would let the team members express their opinions and vote on them to pick the worthiest one to tackle in the next sprint.

+ Throughout the whole communication within the development phase, it's been realized that there were a couple of issue which were agreed upon without any objection, thus the voting system didn't seem necessary. The team focused on solving the apparent issues.

- The team expected full participation of the members.

+ One of the members was not able to join the workshop, while another member did not have a mouse or any prior gaming experience which was unknown to the rest of the team and this rendered him powerless in the workshop.

III. Experiences in the Scrum Workshop

The teams were given an hour before the team works officially started, and in the meantime, they have been introduced to the assignment that they were expected to accomplish within the next 3-hour period. Teams were asked to identify the meaning of "Definition of Done" which they haven't been able to unanimously agree on, so they tried to build a new Slack channel where the scrum masters would meet and agree on the Definition of Done. Even though the result was not found completely satisfactory, it was good enough that the teams finally had a common understanding of how to proceed onwards. As David Chappell [3] emphasizes, this is important for the collaborative teams to agree on. Without some shared understanding or values, it's difficult to build something big or connect two pieces of a single entity.

In the first sprint's planning phase, the scrum master took longer than expected to pick a task and get back to the team, which was due to a lacking method to reach the PO. The team simply waited for the PO to join their channel, and then all of their questions have been answered, but the team was too eager and tried to be as complete as possible, so they asked too many questions and asked for unnecessary details due to the excitement of the moment, and some of these details were not even part of their assigned task, but they wanted to make a perfect building with a lot of components inside. So, they lost a precious chunk of their development time while gathering the details. It can be seen throughout the software projects that the requirement analysis and various issues related to the initial phases are one of the most common problems, also as explained by Sarah Beecham et al [4]. The following planning phases went well, learning from the first plan turned out to be beneficial and the mistakes were not repeated.

In the first development phase, most of the time was wasted on finalizing the requirements, so the team barely started working on the development, and did not manage to finish the task in time. In the team retrospective meeting of this sprint, the team decided to keep the questions clear and just detailed enough to satisfy the requirements. There hasn't been any voting, as the team seemed to agree on the same issue. Since it was difficult to contact the PO in the previous phases, they decided to be more insistent and make the scrum master chase the PO rather than waiting for him to join the team channel. This proved to be useful, but since other teams also decided to do the same thing, the communication was still slow and insufficient.

The team felt more relaxed in the second sprint. They managed to complete the task and got their PO's approval. The most important thing to mention in this sprint is that the team decided to bail on using the pair programming method by the end of the spring. In the development phase, setting up the pair channel and moving between the team and the pair channels felt very difficult and time consuming as they had to use two different applications and needed to manage their audio in both of them, which resulted in a messy inter-team communication. Also, the difference in the pairs' skill levels were quite different, which resulted in an unbalanced workload on the teams, as some of them finished fast and had to wait, while others finished closer to the deadline. Ivar Jacobson [5] mentions that this kind of pairs are difficult to manage in real life and experimenting with this idea has no place in the spring planning. If it's crucial to work in this manner, then the processes need to be defined explicitly to remove any doubt. So, it's been decided to stick to the team channel and keep the communication to minimum to avoid chaos.

In the third sprint, the team needed to collaborate with another team, which wasn't realized at the time of picking the task. Scrums masters of these two teams eventually managed to find each other and agreed upon the requirements. In the meantime, the development team gathered as much material as possible rather than building the required structure, since the location and material details were unknown. Once the questions have been answered, they managed to complete the task. Talking to the PO just two minutes before the deadline has been proven to be a mistake, as one of the building materials was of an incorrect type, and a piece was missing. The team didn't have enough time to rebuild and correct their mistakes at this point.

In summary, the issues can be called as undefined behaviors. None of these issues had been thought of in advance and caught the team off-guard. Therefore, the team felt the need to improvise to save the day, to complete the building. Although, this made it almost impossible to analyze some parts of the plan, such as the plan to time-box all events and keeping close track of the time, thus it's unknown whether it would have benefitted the team more if they followed the plan in a calm manner.

IV. Software Process Improvement Techniques

There are many SPI methodologies that exist today. We can explain them under two categories based on the way they work; inductive (bottom-up) methods and prescriptive (top-down) methods. Inductive methods focus on the company's urgent needs and provides solutions to these specific needs, while the prescriptive methods provide more generic practices which can be described as one-size-fits-all policies [6].

Prescriptive models can be overly resource consuming and not even solve the company's urgent needs, as the analysis and improvement actions are defined in an order which needs to be followed and the company's actual needs can be overlooked. On the other hand, since the inductive models focus on the company's most important issues, it can be a more straight-forward solution. Most of these methods follow a cycle of four steps: analyze current situation, plan for the improvements, implement the improvements, evaluate the result.

Improvement framework utilizing light weight assessment and planning (iFLAP) is one of the well-known methods that fall into inductive model category and consists of three steps. First, a project of interest and related roles are selected. Then, data from multiple sources, such as documentation and interviews, are gathered and analyzed [6]. And finally, improvement plan is made, and tasks are prioritized.

Goal, question, metric (GQM) approach is another inductive model which focuses on improvement of specific goals. It requires the formulation of a clear question, which identifies the improvement area and the perspective, and a way(s) to measure the improvement [7]. This is the SPI method used in our workshop process improvement assignment [8], as it provided clear and concise guidelines to solve the issues. The project team managed to formulate multiple goals, questions, and metrics in a matter of hours, which in the end was proven to be beneficial.

Capability maturity model integration (CMMI) falls under prescriptive models. This model has two versions known as staged representation (SR) and continuous representation (CR). Even though these two versions derive from the same key process areas (KPA), they approach SPI in different ways. While SR tries to improve the situation from the organizational perspective, CR addresses the improvement from process perspectives; such as improving the practices used in the related process [6]. Using this methodology would have solved the issues that we faced in the first workshop, but it would have required more time to be spent on analyzing the whole event, while wasting valuable and limited time on analyzing non-problematic areas such as inter-team communications.

ISO/IEC 15504, also known as software process improvement and capability determination (SPICE) is one of the prescriptive models, as well. This model contains specific sets of practices explicitly defined for various domains. The main difference of this model compared to CMMI is that it can only use external assessment group members, while CMMI can use both internal and external assessment group members [6].

V. SPI in Industry

In the SPI course, we had two guest lecturers explaining us their experiences of SPI in the industry, former from a medical device company called 1928 Diagnostics (1928D), latter from Siemens, a rather well-known tech company.

In 1928D, the company had been using a mix of waterfall methodology required by the standards and regulations, and agile methodologies where the company needed to be more flexible and adaptable to the regulations that govern the location where the device/software is used. They had to develop the same product with some modifications based on the use area. By using agile methodologies, they expected to be able to re-use more components they had developed and make the software adjustable based on the location and changing requirements. When the regulations required that a process exists to ensure product safety, the company had difficulties coming up with agile methods as they usually lack concrete standards [9]. On the other hand, when it was applicable, iterative prospect of the agile methodologies helped them gather feedback quickly and more often, resulting in more successful increments. They also adapted teams and resources applicable to the agile ways, which is similar to the structure we used in the workshops. This lecture also gave us insights regarding our expectations; there is no guarantee that our first solution to SPI is going to work, just because we want to improve our process, we need to tackle the problems while considering the requirements.

The situation in Siemens was different. They wanted to completely transform from waterfall to agile model. When they were using the waterfall model, they had a chaotic environment, where people were stressed out due to projects not meeting the deadlines, frequent overtime works, and issues surfacing with last minute code merges. The change process was not easy, as they had to change the people's mindsets to be open to new practices, such as retrospectives. The HR department took the responsibility to motivate the employees through some of the changes, which eventually turned out to be good, as the people's morale had been increased. At some point, there were both agile teams and waterfall teams, where the waterfall teams did not manage to meet the deadlines unlike the agile teams, which also caused problem in team collaborations.

There were other problems, as well. It was an expensive procedure to change the development methodology, people had to be trained. At first, the teams were not assigned in accord with the agile guidelines. The scrum masters were chosen from the management roles, and they tried to manage the projects rather than protecting and coaching the teams. Product owners did not collaborate with the teams as they should have, they expected the teams to take all the product backlog responsibilities. The employees were also confused and surprised as they had to take more responsibility for the actions they have made, in contrast to the past where they followed a hierarchical model and were free of some of them. Customers were a whole different story, as they were both happy for having their thoughts being taken into consideration through feedback loops, they were also upset due to being partially responsible of the development process.

In our workshop, we had similar problems with the SPI at Siemens. Our members were on different levels of skillset, which caused an unbalanced workload, and difficulties with the other teams where we needed to merge our final products, but one team was ready while the other one was not [9].

VI. SPI Proposal for future Scrum Development Efforts

I (and my team) considered the lack of requirement specification as the biggest problem [10], as it caused the team to lose a significant amount of time waiting or rebuilding some parts of the assigned structures. For this problem, we can formulate a GQM SPI practice, as the problem can be focused on its own, without meddling with the other processes where no issue has been spotted. Our first goal would be "Improve the understanding of requirements from the developers' perspective.". Since the user stories in the product backlog did not have enough detail regarding the requirements, we need to clarify them with the PO that is responsible for the related task, since he/she is the person that can approve the final product. We can define a more detailed process for the operation, rather than vaguely stating that the scrum master gathers the requirements. Scrum master should make a list of assumptions regarding the assignment, identify the questions regarding the materials, size, location of the structure, and then verify these with the PO. Once the development starts, PO should be contacted, and the progress should be presented to them. Then the PO may give a score to the product on a scale of 0-10 in two perspectives; correctness and completeness. Repeating this process about every 10 minutes in a 30-minute sprint should result in 3 outputs, where they can be quickly compared, and rebuild/continue decisions can be made.

Another big problem was the difference in skill levels of the team members. Some of our members had never played Minetest or a similar game before, which made them unable to contribute to the team while the rest of the team either worked on their tasks or tried to teach them the game. This is very similar to the agile training issue I mentioned in the previous section. In the industry, the issue was the cost, but in our case it's very insignificant as the sufficient training can be completed in a matter of a few hours. Being software engineering students with interest in computer games, many of us wouldn't consider this as much of an effort to teach a student that is willing to learn. We can formulate the second goal as "Improve the team knowledge of operations in Minetest from the perspective of the development team." [2]. In this case, we want to identify how much game knowledge the development team has. Provided with a list of basic game operations (gathering, building, crafting, etc.), the developers choose the number of operations they are capable of doing. In another question, we can ask the members how much time they have spent in the game (non-idle time, approximate duration they have spent building things). After training the less experienced/knowledgeable members a few hours, which can be estimated from the in-game time spent question and a rough total expected time can be decided, the same questionnaire will be repeated to measure the game knowledge of the developers. This training does not need to take place in the workshop itself, but it could be taken care of within the weeks between the two workshops.

VII. Implementation of an SPI Initiative

The team decided to focus on fixing communication problems, not only being able to contact the person we were seeking, but also to clarify the requirements analysis issue with the Pos.

Through collaboration with the other teams that shared the same POs, a common understanding of communication practices has been defined. POs were given their own chat rooms, where member(s) of teams could join, but with the rule where they must state that they are waiting in line to talk to the PO. Then, the PO names the team he is going to deal with, and answers their questions in a calm manner, where none of the teams are stressed out as the process is transparent and accessible to everyone. Through these meetings, every team had many chances to talk to the PO in a single sprint, where they could ask questions, share their screens and show the building progress for a feedback, while the other teams could listen in and realize some of the questions they haven't figured out yet.

Scrum masters (SM) realized that the previous SM channel was not known by all SMs, so they made an announcement where everybody learned about it. This channel was set to be the place for SMs to contact each other, be that for common questions or team collaborations.

The shared communication channels and frequent feedback from the PO seemed very useful, as the team started building their structure as soon as the development phase started. The PO has been contacted many times, and the progress has been presented, therefore mistakes and misunderstandings have been diminished and the team managed to complete their task within the sprint, in every sprint.

On the negative side, the team did not handle the training requirements of the member with no gaming experience, due to lack of initiative from the member in subject. The team did not feel the need to pressure him or put more effort into convincing him to participate in the game or personal training sessions, as the available skill level was sufficient to accomplish the foreseeable tasks.

In the sprint retrospective phase, just as planned, the team followed the plan and made statements regarding what went well, what went bad. They also answered a short questionnaire concerning the team's mood and their opinions of the changes in the process, all of which turned out positive and the team did not have any concerns left.

VIII. Summary and Lessons Learned

This report demonstrates my knowledge of SPI through my experience in the team activities, workshops, lectures and researches, and expresses my thoughts of the implementation of SPI in the industry and its relation to my experience in the workshops.

In Section II and III, I've described my experience in the planning and execution of the first workshop. I mentioned some of the issues that I and my team have encountered, and how we were caught off-guard with unexpected events. I explained the differences between our process plan and what we have done in the workshop. There are always surprises, but it's obvious that the negative impact can be diminished by having a detailed plan. It also helps identify the problematic areas as it becomes easier to pinpoint the issues in a clear plan.

In Section IV, I demonstrated my knowledge of SPI methodologies and their usages. It can be seen that there is no single solution that solves every company's all problems. The solutions are derived from the issues and need to be managed with the company's values in mind.

In Section V, I demonstrated what I've learned from the guest lectures where professionals came to our class (online meeting) and told us their experiences. I made connections to my own workshop experiences from these stories. Even though I was not concerned with satisfying customer values, I was concerned with satisfying my team's needs in accord with the course requirements.

In Section VI, I explained the actions we have taken, and what I would have done in order to improve our workshop process. Now, in introspect, I can see that we, as a team, managed to succeed. Were I the dictator of the team, some issues may have been solved, but the overall happiness wouldn't have been as high as our end result. Since it is a team of different people, with different thoughts and desires, making decisions together empowered us, we became a self-managing team, and required less of micro-management as the weeks went on.

In Section VII, I explained my experience in the seconds workshop with the implemented SPI practices in mind. Taking one matter at a time, trying to answer a clearly identified question improved our process.

If I were to do this project again, I would put more effort into the workshop plan, as it would make it easier to identify the problems. I would also focus on communication between the teams and the POs so that there is no question in that subject to begin with.

REFERENCES

[1] [Undisclosed Authors], "Process Plan", Assignment 2 in course DIT347 H20, Software Engineering Program, University of Gothenburg, Sweden, 2020

[2] [Undisclosed Authors], "Applying a Process", Assignment 3 in course DIT347 H20, Software Engineering Program, University of Gothenburg, Sweden, 2020

[3] David Chappell, Three Aspects Of Software Quality: Functional, Structural, and Process

[4] Sarah Beecham et al. Software Process Improvement Problems in Twelve SoftwareCompanies: An Empirical Analysis

[5] Ivar Jacobson et al., Software Engineering Essentialized, The Rise of Software

[6] F. Pettersson, M. Ivarsson, T. Gorschek, and P. Öman, "A practitioner's guide to light weight software process assessment and improvement planning" Journal of Systems and Software, 81(6), pp. 972-995, 2008.

[7] Victor R. Basili, Gianluigi Caldiera, H. Dieter Rombach, "The Goal Question Metric Approach", Institute for Advanced Computer Studies, Department of Computer Science, University of Maryland, College Park, Maryland, Universität Kaiserslautern, Kaiserslautern, Germany

[8] [Undisclosed Authors], "Process Analysis and Improvement Goal Formulation", Assignment 4 in course DIT347 H20, Software Engineering Program, University of Gothenburg, Sweden, 2020

[9] [Undisclosed Authors], "Process in the organizational context", Assignment 5 in course DIT347 H20, Software Engineering Program, University of Gothenburg, Sweden, 2020

[10] [Undisclosed Authors], "Concrete process improvement", Assignment 6 in course DIT347 H20, Software Engineering Program, University of Gothenburg, Sweden, 2020