Explorations of Software Process Improvement

ANONYMOUS ID: 122
Examination in DIT347 Software Development Methodologies
Bachelor Program Software Engineering and Management
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden

Abstract—**Software Process Improvement(SPI) is difficult in low maturity organizations. An iteration of SPI has been attempted throughout this course, and efforts to document, discuss, and reflect on this iteration has been attempted. In essence, tools and frameworks of SPI are discussed in detail, and the challenges and factors that allow SPI to succeed are critically analyzed. The first hand experience further supports aspects of these factors, enhancing the understanding of SPI in general.**

I. Introduction

Software systems are seldom created by one person nowadays, but instead developed by organizations, containing a number of teams. As the customer wants the highest value for their investment, it is essential that the final product is delivered with high quality.

A mature software process allows an organization to repeatedly deliver high quality products with limited time and resources. As such, while those with subpar software processes may still produce a worthy product, it is less repeatable due to a lack of trust, standardization, or ability to adapt to changes within the organization, amongst others.

Organizations with poor software processes, therefore, must improve if a greater consistency of delivering high quality products is desired. The field of SPI provides several tools and frameworks to help improve their development process, allowing organizations to adapt them to their specific needs.

Throughout the course, first hand experience of a poor software process was experienced, and consequently an iteration of SPI has been attempted. The purpose of this report is to document, discuss and reflect on this SPI iteration experienced throughout the DIT347 course, as well as discuss several aspects of SPI within the industry.

II. Process Applied in the Lego Scrum Workshop

Several practices were combined together to form the planned process defined for the Lego Workshop. Since our situation consisted of multiple individual teams working together towards a common goal, many practices we planned to use were taken from the Scrum process, due to its strong emphasis of individual teams working together, adapting to change. Other practices were taken from the concept of Kanban and Extreme Programming, both of which helped with having a more robust process.

One important practice we took from Scrum was the concept of sprints. Each sprint contained five phases: Planning, Development, Review, Retrospective, and Program Integration, performed in the same order. These phases would allow us to plan, execute and reflect on the process we used for each sprint. We decided on the usage of a Sprint Backlog as well, further helping us organize the way our tasks.

Other practices used included pair programming and Policy Management, taken from Extreme Programming (XP) and Kanban respectively. We decided that pair programming would increase the quality of our work products, and setting up clear policies would keep us on the same page. Emphasis was also put on communicating with other teams to better organize the overall process.

Alternatives to the planned process described above were considered, namely the Waterfall process. One strong proponent that we decided against using it was how we knew that user stories would be used during the workshop. From this, we knew that a high amount of change would occur during the workshop. Since Waterfall does not adapt well to changes at such a high speed, we decided against it early on.

It is said that no plan survives first contact with the enemy[1]. Perhaps appropriately, our applied process was rather different from our plan. Immediate alterations included the near-abandonment of pair building, instead opting to build separate pieces of the increment together, the lack of Kanban's policy management for our non-existent sprint backlog, and the discarding of the program integration phase for each sprint, effectively eliminating Continuous Integration in our process.

A significant difference which cascaded into other changes was the poor use of planned phases for each sprint. Even though we defined 5 phases concretely, more often than not we only had a Development phase for the entire sprint, ignoring or shortening the rest of the phases. In the sprints where other phases were considered or attempted, lack of time tracking made us go overtime or shorten the upcoming phases, rendering them ineffective as there was not enough to perform the objectives of the phase properly.

Another important change was the lack of continuous feedback from the product owner(PO). Our team only managed to receive feedback from the PO once throughout the

workshop, preventing us from adapting to the changing requirements. We depended on our sprints' Review phase to receive PO's approval for our increments. However, the review phases were chaotic, frequently preventing our group from approaching the POs' before the phase was over - hence the one time we managed to receive feedback.

Many of these differences stemmed from a few common reasons. The aforementioned poor tracking of time lent itself to a series of changes to each sprint's phases, and the merging of the program level's process with our planned one prevented us from fully realizing certain parts of our plan, such as the use of continuous integration.

III. Experiences in the Lego Scrum Workshop

Nevertheless, there were a few aspects which worked well during the applied process. One particular aspect that sped up production was minimum idle time. While Scrum Masters(SM) attended the Scrum of Scrums(SoS) meetings, teams grouped together to sort and filter Legos, which assisted in the management of resources for future sprints. Another aspect that worked well in the team was the decision to follow the specific roles prescribed during the plan. Each role had a responsibility, which allowed our team to quickly assign roles when the sprints started, thus knowing what to do early on.

With that said, there were far more parts that negatively affected the process. As Beecham[2] has similarly discovered for low maturity organizations, we had a plethora of organizational issues as well. Our team had no knowledge of what other teams were working on, program level decisions were constantly changing, and the sprint review was extremely chaotic. Poor time management put sprint phases into disarray. These issues created a chaotic and stressful atmosphere.

Many of these issues stemmed from or affected communication problems, which included both within the team, between teams, and with the POs. SMs were unable to propagate new decisions to the team members due to lack of time and appropriate sprint phases. Teams remained mostly isolated from others, causing teams with dependent user stories to rebuild increments, delaying development. POs were confused throughout the workshop as decisions weren't communicated to them. This caused problems during the review phase and times of availability. Another important issue was how a vocal minority dominated the scrum meetings, silencing many others' concerns or ideas.

It should be noted that the organizational issues and communication issues experienced in the workshop are not isolated from each other, as hinted previously. Instead, many communication issues led to issues with managing people and time, but the organizational issues only worsened the communication between parts of the organization due to the chaotic atmosphere, creating a cycle. As such, I agree with how Beecham has structured communication and people issues within the umbrella of Organizational Issues, as it better illustrates the nature of the spiraling effect they create.

As a result of the aforementioned problems, our group lacked awareness in many areas, which included process and documentation changes, current sprint, and dependencies of our user stories. We were not able to reach the POs because they were not aware of where to be and what to do.

It is interesting to note that even though our workshop developed with Legos rather than software, we experienced very similar issues with those described in Beecham's paper. However, it is also important to realise that our teams (or class) contained people from many different cultures, personalities, and background, which may have influenced the interaction in the SoS meetings. As shown in Niazi's[3] cross-cultural comparison, SPI barriers can be significantly different for different cultures. Given that barriers for SPI also relate to issues within the organization, it can be inferred that even though we may have had similar general issues in the workshop with those within the software development industry, why they occur may be different due to aspects such as culture. Several perspectives of the reasons for these issues in the workshop must be looked into in order to provide a clearer understanding.

IV. Software Process Improvement Techniques

There are multiple Software Process Improvement models, techniques, and methods that could effectively better an organization's current process. It is crucial that an organization chooses those suitable for the current process and maturity, as there can be adverse effects instead if techniques are poorly chosen. In order to know which methods are suitable for a certain organization, the advantages and disadvantages of a range of different frameworks should be well explored.

Several SPI frameworks are quite established and well known. As described by Pettersson et al. [4], most of these frameworks tend to follow four cyclic principles. These principles are sequential: first "Evaluate the Current Situation", then "Plan for Improvements", "Implement the Improvements, and lastly "Evaluate the Effects of Improvement". A cyclic process of improvement not only allow improvements to be done continuously, they also allow smaller improvements to be made in shorter time frames, making it more adjustable for organizations.

Although there are many different SPI frameworks that adopt these principles, most can be divided into two different approaches. Bottom-up, or inductive approaches try to have a clear understanding of the situation, and constructing the improvements on this understanding of the issues. Other frameworks aim to improve their current situation based on a collection of defined best practices in developing software. These frameworks are thus called "top-down" or prescriptive approaches as they provide concrete targets for organizations to aim for.

One inductive approach to SPI is called the Quality Improvement Paradigm(QIP), created by Basili[5]. As described by Homestudy Module [6], QIP is cyclic in nature, iteratively helping to identify critical issues with an organization's process, helping them acquire key competencies to deliver the maximum customer value. In

order to determine improvement goals and gather metrics, QIP allows organizations to pick and choose, similar to a "salad bar", as vividly explained by Laestadius[7]. The GQM(Goal, Question, Metric) model is particularly suitable for QIP, however, as both are based on setting measurable goals, and GQM allows QIP to analyze the results after the change was attempted. The seamless integration of gathering metrics is particularly important during analysis as the improvements are more visible. In other words, the positive results can be shown more easily. The solutions formulated in QIP are first evaluated in pilot projects as a test, and if results are positive, an official change is made for the organization. This "change prototyping" approach provides a lesser risk to the organization if the change negatively impacts the project. Risk management is particularly important for smaller organizations, as a small misstep can destroy the entire company, as stated by Engberg[8].

iFLAP, while similar to QIP, is another inductive framework that has a more direct style. iFLAP consists of three general steps- Selection, which focuses on choosing certain projects and roles, Assessment, which deals with eliciting and analyzing information, and Improvement Planning, which plans which improvements to execute first. iFLAP attempts to target smaller organizations with its more lightweight nature, as described by Pettersson et al.. iFLAP thus allows a more cost effective solution with less resources needed to begin the SPI process. However, with iFLAP being targeted more towards organizations still in their infancy, it may not be the best framework for more mature organizations with a more resources to dedicate to SPI, despite the fact that iFLAP is described to be scalable if desired.

It is important to realise that while inductive frameworks provide extensive focus on evaluating the current situation or the effects of improvement, they do little to provide concrete methods on *how* to perform the improvement. Prescriptive frameworks, however, allow organizations to see concrete practices that they need to achieve by providing a model of best practices. This model-based approach allow organizations to compare their current process with the "gold standard", as described by the Homestudy Module, and make changes according to it.

One well known prescriptive framework is the Capability Maturity Model Integration(CMMI), which is a amalgamation of several CMM versions[9], standardizing several models into one. CMMI provides a description of practices used in a mature organization to develop software. These practices are grouped together to satisfy goals, which in turn come together to form Process areas(PA). Many PAs depend on other PAs, allowing organizations to have a general idea of where to start improving. CMMI also provides maturity levels, which tie in with these dependencies to allow organizations to see where they lie on the spectrum. CMMI offers an in-depth look of what needs to be achieved in an organization's software process to be more mature, and attempts to take the needs of the organization into account with the process areas.

It is important to note, however, that CMMI is a "one size fits all" model, requiring organizations to adapt the PAs or goals themselves to fit their specific needs. As such, it is more difficult for smaller organizations to dedicate resources to make these adaptations. The misconceptions of Agile development methods with CMMI practices is another problem, as documented by Glazer[10]. The idea that CMMI and Agile methods, through misuse, incorrect terminology or preconceptions, are incompatible inhibits Agile organizations to adopt CMMI as a viable SPI method. As one may realise further down, this is certainly not the case.

SPICE is another prescriptive framework worth exploring. Instead of providing a single model for all organizations, SPICE contains generic process models for specific application domains, taking domain specific issues into consideration. This is especially important for safety critical systems or systems that have special needs relevant for the domain, such as international standards needing to be taken into account early on during development, for example.

Although rudimentary, the classification between inductive and prescriptive SPI frameworks provide a good general understanding for the multitude of frameworks. Mapping the categories to four principles mentioned previously, inductive frameworks allow an organization to evaluate the current situation and the effects of improvements, while prescriptive frameworks fare better within the realm of planning and implementing improvements.

With this said, the two approaches can easily be combined to improve an organization; The current situation can be better understood from inductive frameworks, which can be compared against the models in prescriptive frameworks, then creating concrete improvements based on the results. In the case of our Lego Workshop, the short time frame should be taken into account, as well as the extremely low maturity of the organization. As such, the lightweight iFLAP framework would be suitable for choosing roles and eliciting information to determine potential improvements. Although an iteration of QIP could be done instead, QIP's strengths of prototyping improvement would require more time, which is less feasible for the workshop. However, the GQM model could potentially be useful to create concrete, measurable improvements, which iFLAP does not. In particular, GQM is established to be adaptable to different environments by Basili[11], which is useful for the unique workshop domain. In terms of applying concrete improvements, CMMI could be useful, particularly due to its generic approach, since we are using Legos to develop a system rather than software, making it difficult to place in a specific domain, like SPICE would. The concept of continuous representation, focusing on individual PAs would assist in adapting CMMI for the organization.

V. SPI IN INDUSTRY

As the software industry grows, SPI becomes increasingly relevant for organizations to consistently deliver high quality products. Although companies with different domains may

have vastly different requirements, a sustainable and healthy process is equally important in both.

Safety critical systems, such as medical devices, are subject to stringent regulations and control. As such, the development processes utilized to produce such software must be able to take the standards and regulations into account early on. It should be noted that the standards and regulations mostly give indication of what work products to deliver, but not how to obtain them.

The Quality Management System(QMS) provides a process framework that helps standardize the way of working to develop the software. The QMS allows the company to learn what is required to reach the desired quality of the product or service through repeated iterations of the QMS cycle. QMS only offers a standardized way of working. Thus, companies may still deviate slightly. Engberg infers that improving the process quality would improve the software quality. This idea is similar with Chappell[12]'s paper, which explains that process quality is one of the three pillars of software quality. Thus, if process quality is improved, the software's quality would in turn improve as well. Interestingly, unlike most software, it typically takes longer for medical products to go through the regulators again if the product itself is improved due to the stringent rules. In essence, this means that continuous improvement of the product itself is not feasible. Therefore, it is more ideal to indirectly improve the software quality instead through the process.

Passing audits are central for a medical device to enter the market. Standard Operating Procedures(SOPs), a standard way of working based on regulations and QMS, among other constraints, are also used in some companies. SOPs attempt to strike a middle ground between readability for software engineers to follow and understandability for the audits. SOPs increase efficiency as teams are able to understand it without needing to update everyone individually. Since SOPs are not person dependent, individuals changing projects require no special teaching as the entire organization does things the same way regardless of which project they undertake. It is interesting to understand that while the standards set a stage-gated process, where certain steps must be fulfilled before the next step can be undertaken, companies are allowed to revisit previous steps. This may imply that Waterfall is the process most suitable to develop such systems, but it should be emphasized that companies can still develop using an Agile process, as briefly mentioned by Engberg.

Continuous evaluation is crucial for the process to ensure that the process aligns with the standards. To do so, process are measured and monitored, making sure the processes fulfill certain criteria. One key aspect of deciding the type of measurements to take is making sure everybody agrees on the measurement used. This can become an issue if people are concerned about privacy, for example. Choosing the wrong measurement for a process can also be devastating, as false positives can lead to ignorance. Dybå[13] highlights the selection of the correct measurements due to such possibilities.

Unlike the medical software industry using standards to ensure regulations are met, other companies can be much more chaotic in the process, but equally effective. Goncalves[14], describing the trials and tribulations of a "Big Bang" transition of an enterprise from a traditional plan driven process to an Agile one in her lecture, emphasizes the need of having the commitment of the entire company to make such a change.

According to Goncalves, several reasons prompted the need for the company to change their process. The first was the traditional control hierarchy. Within this hierarchy, teams reported upwards to their managers, with many layers in between and upwards as the organization grew. With the company distributed across different time zones and R&D sites, decisions took a long time, slowing development. The differing work cultures and time only contributed to the decline. Increasing working hours did not help, and instead overworked employees provided lower quality work. Each R&D site built part of the system, and later integrated it into one product. However, each site was isolated, making it more difficult to integrate due to dependency issues. Most importantly, the customers were unhappy.

The change to Agile in the company shattered many preconceptions of the traditional process. The hierarchy flipped upside down, making managers support the teams instead. Teams became cross-disciplinary, autonomous and well rounded. The cross-functional nature broke down the insulation between teams, requiring less management resources to coordinate the activities between teams. The hierarchical change also modified key roles within the organization. Product responsibilities were centralized, increasing decision times.

Goncalves and Engberg expressed the major difficulty of changing the company's team culture. The structure itself is fairly simple to change, but the minds of people are not. Initially, affected employees were displeased about the sudden changes occurring on every level of the hierarchy. Managers were confused, and the significant autonomy made some uncomfortable, causing employees unable to adapt to quit.

Many of these aforementioned issues focus on people, and makes it clear that Agile centers around people. The Agile work culture requires participants to be continuously learning, motivated, and work well with others. This mindset to take responsibility along with value driven teamwork allows an organization to be flexible, yet deliver the maximum customer value within the given constraints.

One important factor mentioned by Goncalves and echoed by Dybå was the importance of enabling a culture where SPI procedures can thrive. By creating these conditions, the organization's transformation will naturally come to be.

Many issues discussed above were experienced in the workshop as well, supporting the difficulty of reshaping the mindsets of people even within an organization of 60 people.

Although we actively attempted to make our process Agile, one underlying mindset that pervaded the workshop was how we focused on finishing an increment/product as quickly as possible without additional elicitation, instead of providing something useful for the customer. This comes back to the emphasis on creating a culture where Agile can truly grow, which we were arguably unable to do successfully.

VI. SPI Proposal for future Lego Scrum Development Efforts

In an attempt to improve our process, we proposed another SPI plan for the next workshop. GQM and CMMI were chosen as tool to perform SPI. As mentioned in Section 4, Basili has expressed the combinability of GQM "within the context of a more general approach to software quality improvement" as well as being "adaptable to different environments", which is well suited for us in this case, as CMMI is being used as well.

The compatibility of Scrum and CMMI is supported by Sutherland[15], who calls attention to how the mix "results in significantly improved performance", among other perks. As we already had a basis in Scrum, this seemed to be well suited for us as well. In essence, CMMI Roadmaps were chosen due to, as Cannegieter[16] describes it, the "guidance and focus for effective CMMI adoption" it offers. Moreover, it combines the advantages of both staged and continuous representation, which is more feasible for a lightweight organization like ours.

Taking into account central issues we had, we decided to focus on three GQM goals: Increase awareness of program decisions made from the team's point of view, improve handling of POs' time, and increase effective interaction between teams. These goals were chosen due to how we perceived many issues to resulted from a lack of communication, as documented above.

Questions and metrics were created to measure the fulfillment of goals during the workshop. Comparing the number of known and actual decisions made would let us know how much the team knew concretely. Subjective ratings would allow the team to know how they felt about certain areas. Looking at these metrics would allow the team to know potential problems during sprints, allowing the team to act upon them during the workshop.

Due to the "Lego" nature of the workshop, not all specific goals were applicable. Our GQM goals were adapted to specific process areas in the CMMI Roadmap, and to specific goals within the PA. To align both, the PAs Project Planning (PP) and Organizational Process Focus (OPF) were chosen.

To address issues such as lack of process adherence and team confusion, crucial project information would be displayed on the projector, and have an appointed individual to manage documentation. Metrics regarding these problems would be filled out by both the SM and the team. These solutions could help establish and maintain the project plan.

We also wanted to motivate POs, as they were uncaring and uninvolved in our process previously. To address this, a schedule for visiting the POs was outlined, and slack was given. To measure involvement, we decided to allow POs to record how they felt for each sprint. These solutions try to obtain a higher commitment from stakeholders. A higher commitment would mean better support for the plan.

We also felt that we had no way to deal with problems with our shortcomings in our process. We thought to have the scrummaster bring up the issues provided during retrospective in the SoS meetings to allow each member have better control of the process. The appropriate metrics would also be recorded by the end of the sprint by the Scrummaster and members. This would allow us to identify issues early on, and fix them if possible.

Finally, we felt that communication within the organization was chaotic and lacking, instigating other issues such as unknown dependencies between user stories. We decided to have rules and dependencies established in an online document accessible to everyone, displayed on the projector as mentioned before. As seen in the medical software industry, establishing standards allowed the organization to be on the same page, which was the intent here as well.

Institutionalizing the Agile process was also attempted with generic goals. As highlighted by Sutherland and Sidky and Arthur[17], Agile without discipline is detrimental. Generic goals, as well as the methods to implement them were defined as an attempt at institutionalization. The role of an ambassador was also defined to discuss dependent user stories with other groups. Practices such as Continuous Integration and Collaborative Planning were also chosen to address communication issues and increase our Level of Agility.

VII. Implementation of an SPI Initiative

It was clear that our plan would once again need to be merged with other teams' ideas to create a joint plan when the second workshop began. One unplanned aspect of the team that affected team effort was the status of the team. Although all team members were present, two members were recovering from sickness. As a result, most responsibilities were assigned to the remaining two.

From the beginning of the workshop, The Definition of Done (DoD), decisions made in the SoS meetings, and time until the current sprint phase ended was shown on the projector. A Scrum Master was appointed to be in charge of the updates to the projector, which later became an issue, as SMs had too many responsibilities already. Time keeping was constantly delayed, and only partial information was shown.

Our schedule for the POs was unaccepted, and thus the metric of the "number of teams POs met during a sprint" was skipped. Had our schedule for the POs was more concrete, it may have been accepted. Instead, POs were separated, and teams could visit them whenever they wanted for one minute. However, the POs were not explicitly notified about this schedule and our expectations, causing them to be unavailable when some sprints were not over. This indicated how the POs were still too uninvolved in the process.

The role of the Ambassador was successfully appointed. One unexpected advantage of appointing the ambassador and builder role to one member was how effective it was, since the member could instantly perform changes needed while communicating with other teams.

Bringing up issues to the SM helped improve the process to a certain extent. An exhaustive list of problems, mostly directed at the program level, was expressed by the team. Yet, resolving these problems took a long time due to discussions it needed to resolve them. It would have been smoother if some issues were phrased constructively, and a solution was described bringing it up to the SoS meeting.

To look at the level of team awareness of decisions, the number of Known Decisions(nKD) and number of Decisions Made(nDM) were measured. From sprints 1-3, nKD was 5,4,3, and nDM was 3,6,6. From this, the deviation was calculated by dividing nKD by nDM, resulting in 1.66, 0.66, and 0.5 for sprints 1,2,3 respectively. The deviation worsened during sprint 3, which suggested that it became more chaotic near the end.

Dependent user story knowledge took a toll near the end too, with Known Dependent User Stories (KDUS) being 0,1,0 and the Total number of Dep. User Stories (TDUS) being 0,1,1. Again, this shows that sprint 3 was a weaker sprint.

Yet, the team felt that they were more aware of SoS decisions made near the end of the workshop. The Awareness of SoS Decisions for the SM was 3,2,4, where 1 was "terrible" and 5 was "great". The team echoed this sentiment, with their ratings being 2,2,3, for sprints 1,2,3 respectively.

Finally, the duration of meetings with POs was recorded. In Sprint(Spr) 1, we had 3 meetings for a total of 5 minutes. No meetings were done in Spr. 2. Spr. 3 was not recorded due to program confusion.

Many of these results seem contradictory, which may be due to several factors. While the deviation in Spr.3 for the nKD and nMD shows a lack of awareness, the team felt they were more aware. This may suggest a false sense of security. However, this may also be due to how most of the changes were made earlier. With the process became more concrete near the end despite, smaller changes may not have impacted the process too much. For PO meeting duration, we certainly managed to receive more feedback than before, just in Spr. 1. This suggests that while PO management was still chaotic, the organization still improved compared to last time.

Overall, the group managed to implement a majority of our planned changes. Most issues that were brought up were slowly resolved. Appointing the Ambassador and Builder to one member was a major success, but displaying information on the projector could be handled better. The contradictory measurements may suggest that the team was lured into a false sense of awareness, but it may be due to how the process was more concretized near the end.

VIII. Summary

This report presents a first-hand experience of an SPI iteration in an Agile organization within the context of a Lego Workshop. Reflections on the various tools used, successes, and failures throughout the SPI iteration are made. The culture of enabling SPI attempts has been emphasized, along with the methods an organization can perform SPI through the use of combining frameworks, which should be adapted to the organization. It has been found that the issuing of roles related to other responsibilities allow a more efficient process, and the subjective feelings of a team regarding project issues may not always be the indicative. Further investigation is required.

In the next iteration of SPI, additional efforts for time keeping, Clearer definitions of DoD, and the division of SM's roles would be prioritized. These changes would allow greater flexibility, thus enhancing the strengths of the Agile workflow.

References

[1] R. Keyes, The Quote Verifier: Who Said What, Where, and When. New York: St. Martins Press, p.256, 2006.

[2] S. Beecham, T. Hall, and A. Rainer. (2003). "Software Process Improvement Problems in Twelve Software Companies: An Empirical Analysis". Empirical Software Engineering. 8. 7-42. 10.1023/A:1021764731148.

[3] M.Niazi, M.A. Babar, and J.M.Verner, "Software Process Improvement barriers: A cross-cultural comparison" Information and Software Technology, vol. 52, pp 1204-1216, 2010.

[4] 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.

[5] V. R. Basili, Victor R., and Gianluigi Caldiera. "Improve software quality by reusing knowledge and experience." MIT Sloan Management Review 37.1(1995):55.

[6] J.P. Steghöfer, "Prescriptive and Inductive SPI Methods,", Unpublished Video Transcript, DIT347: Software Development Methodologies, University of Gothenburg, Gothenburg, Sweden, 2018.

[7] P. Laestadius,"A salad's bar is a pick and choose configuration [...] You just select what is relevant to your needs.", "Slack Conversation", 2019.

[8] R. Engberg, "Medical Device Software Implementing, maintaining, and improving software development processes", DIT347: Software Development Methodologies, Gothenburg University, 2018.

[9] T. Gorschel, Software Process Assessment & Improvement in Industrial Requirements Engineering. Karlskrona, Sweden, 2004, p.6.

[10] H. Glazer, J. Dalton, D. Anderson, M. Konrad, S. Shrum, "CMMI® or Agile: Why Not Embrace Both!", Software Engineering Institute, Pittsburgh, United States, Tech. Note CMU/SEI-2008-TN-003, 2008.

[11] V. R. Basili, G. Caldiera, H. D. Rombach, "The Goal Question Metric Approach," Kaiserslautern, Germany, Univ. Kaiserslautern.

[12] D, Chappell, "The Three Aspects of Software Quality: functional, structural, and process", 2013.

[13] Dybå, T.. An empirical investigation of the key factors for success in software process improvement. Software Engineering, IEEE Transactions on Software Engineering, 31(5), 410-424, 2005.

[14] D. Goncalves, "Agile Enterprise", DIT347: Software Development Methodologies, Gothenburg University, 2018.

[15] J. Sutherland C.R. Jakobson, and K. Johnson, "Scrum and CMMI Level 5: The Magic Potion for Code Warriors", Hawaii International Conference on System Sciences, Proceedings of the 41st Annual, IEEE, 2008.

[16] J. J. Cannegieter, A. Heijstek, B. Linders, R. Solingen, "CMMI Roadmaps", Software Engineering Institute, Pittsburgh, United States, Tech. Note CMU/SEI-2008-TN-010, 2008.

[17] A. Sidky, J. Arthur. "A Disciplined Approach to Adopting Agile Practices: The Agile Adoption Framework", Virginia Tech, 2007.