# On the Design of Software Processes and Software Process Improvement in the Context of Lego Scrum Workshops

Michal Spano
Department of Computer Science and Engineering
Chalmers and the University of Gothenburg
spano@chalmers.se

***Abstract*—This paper discusses the design of a software process in the context of Lego Scrum workshops as part of the course DIT348 Software Development Methodologies offered by Gothenburg University (Sweden). Subsequently, the paper provides insights about software process improvement (SPI) techniques and proposes an SPI initiative with a redesigned process (accompanied with a SPEM 2.0 diagram). Ultimately, the paper discusses various challenges that practitioners encounter when conducting SPI efforts, and provides an additional insight into the feasibility of the Scrum of Scrums methodology.**

## I. Introduction

A software process is a goal-oriented approach which incorporates a set of tools, methods, and practices to produce a software product or a more general software service [1], [2]. With the increasing complexity of software systems, the need for a well-defined software process has become more and more significant. Basili and Rombach [3] state that software engineering processes require **tailorability** because of the diversity of the software products and the different environments in which they are developed, hence the need for a software process improvement (hereafter SPI) initiative.

This paper analyzes a designed software process and proposes an SPI initiative—based on which the process is to be improved—in the context of the Lego Scrum workshops. The text is complemented with knowledge and experiences from peer-reviewed literature and the course material alongside several research studies are conducted by the frontiers of the software engineering field (e.g., [3], [4], [5]).

## II. Process Defined for the Scrum Workshop

The defined process is titled AKAMMO and utilizes concepts from Scrum, Incremental Delivery, and, to a lesser extent, User Story Practice [6]. Scrum provides a flexible and adaptive approach to project management and encourages transparency in terms of the project's progress and the issues encountered [5]. Scrum, by definition, is meant to be applied in an iterative manner, which is why a conjunction with Incremental Delivery becomes a natural choice [5], [7], [8].

Initially, the process assumes a set of user stories that are to be implemented (and are prioritized by the Product Owner [hereafter PO]). Afterwards, the team members analyze the user stories and decide whether they require a further elicitation of requirements (which is discussed with the PO) or whether they can be implemented as they are. Moreover, at the beginning of each sprint, **sprint planning** takes place. It is an activity that aims to split up the user stories into more granular requirements, where each requirement is accompanied with a set of acceptance criteria. The requirements are then assigned to the members of the team—the delegation of requirements is carried out in a way that the team members have agreed upon, although, in case of a disagreement, round-robin, random choice, or any other method can be used.

Subsequently, the requirements are assigned **story points** that correspond to the difficulty of their implementation (i.e., the higher the number, the more difficult the implementation) based on the Fibonacci sequence. Strikingly, Tamrakar and Jørgensen [9] note that the Fibonacci scale, along with other non-linear scales, eliminates a bias present in linear scales, where individuals tend to favor the middle. Consequently, non-linear scales yield more accurate results.

Furthermore, a **Kanban board** is used throughout the workshop to categorize each requirement into following columns: (i) *To Do*, (ii) *In Progress*, (iii) *Done*. The reason why a Kanban board was considered to be a part of the process can be supported by a claim made by Ahmad et al. [10] suggesting that a Kanban board provides visibility to a [software] development process, communicates priorities, and highlights bottlenecks. However, no specific type of the board is assumed—physical or digital boards are equally viable.

What's more, a sprint lasts an hour, and as the end of the sprint approaches, a meeting, which combines a sprint review and a sprint retrospective, is held (adapted from Scrum [5]). During the meeting, the team discusses difficulties they encountered and evaluates the overall progress carried out. Additionally, this meeting should yield an **increment** which the team has produced over the course of the sprint. It is, afterwards, delivered to the PO once the meeting has concluded.

The process, by default, does not schedule any breaks; however, there is a possibility of a so-called *"emergency break"* which can be called by any team member at most once every 30 minutes. Breaks on an individual basis are allowed, but should not interfere with the team's progress. In a typical Scrum environment, stand-up meetings are held repeatedly and last for a short period, e.g., 15 minutes [5].

However, in the context of the Lego Scrum workshop, the team is not obliged to hold such meetings, because merely six members are involved in the team, and the workshop is not meant to last longer than four hours.

## III. Experiences in the Scrum Workshop

On the whole, the designed process, AKAMMO, was a viable choice for the first Lego Scrum Workshop. The team members perceived it as flexible enough to accommodate the needs of the team, while not being too rigid to hinder the team's progress [11].

The process has been followed in the desired manner (described in Section II) with very few deviations. The process has been made use of for the first time, allowing the team to identify shortcomings (that are addressed in the following subsections) propose improvements, and highlight successful aspects.

Firstly, the team members agreed that initiating a sprint with a **sprint planning** enabled them to have a clear understanding of the requirements that were to be worked on. Secondly, it was observed that merging the sprint review and retrospective into a **single meeting** was a logical choice. Usually, these meetings briefly covered team difficulties and progress, hence no need to separate them into two meetings. Omitting stand-up meetings was considered reasonable due to the small team size (six members), the short duration of the workshop (not exceeding four hours), and seamless communication among participants, because all were present in the same room. Remarkably, our process, influenced by the points mentioned above, proved highly effective, with no team member calling for *"emergency breaks"*.

However, team members identified various concerns which are explored in the following subsections with references to relevant literature. Firstly, a lack of a sensible **Definition of Done** (abbreviated as DoD) was observed. Put simply, a DoD is a set of criteria points applied iteratively to each granular increment of a product to assure that the increment is of a desired quality when considered to be *done* [12]. The DoD principle is a crucial component of the product's quality assurance, and, as Abrahamsson and Kautz [13] point out, quality assurance entails an ability to identify defects as early as possible—this becomes viably achievable with a well-defined DoD.

The absence of a Definition of Done (DoD) can lead to an undesirable product quality and increased development costs. Additionally, it may cause misunderstandings between team members and the PO, hampering overall project progress. These (and other) observations are based on a study conducted by Kopczynska et al. [12].

Furthermore, during the second and the third sprints, the team **omitted assigning story points** to user stories. This stemmed from team members being deeply engaged in working on individual user stories, leading them to skip assigning story points. Undeniably, this might lead to a serious threat in a real-word project. As Abrahamsson and Kautz [13] observe, it is a recurring issue that teams conduct improper estimations—in our case, the estimations were not made at all. The absence of estimations, especially in the face of serious risks, would complicate and, in some cases, make it impossible to evaluate the impact on the project's progress.

Lastly, it has been observed that the process did not mention any **cross-team communication efforts**. However, the nature of the Lego Scrum workshop was to produce a city environment with a set of buildings, where each team was responsible for one (or more) buildings. It is, in this case, unavoidable that the buildings are interconnected in some way, and, as a result, the teams were required to communicate with each other. This fact was not taken into account when designing the process and proved to be a bottleneck for the team's progress. This is, in fact, not an issue specific to our endeavor, albeit a rather general problem. Larsson and Larsson [14] postulate that conducting cross-team collaborative approaches to lessen integration risks is unavoidable, and should not be disregarded in any high-quality process. Cater-Steel et al. [15] further emphasize the importance of cross-team communication by stating, quote, *"[...] high quality external expertise is more critical than top management support."* It is thereby evident that the lack of cross-team communication poses a serious threat which should be addressed if the process is to be used in the future iterations of the workshop.

## IV. Software Process Improvement Techniques

In our ever-emerging world of software, there is a plethora of SPI methods, models, and techniques that are available to practitioners. Examples of some common SPI models include the Capability Maturity Model Integration (CMMI), the ISO/IEC 15504 standard, or, a more lightweight approach, the iFLAP model [16]. In this section, I will briefly describe some of the techniques that are introduced by iFLAP in relation to the process that was designed for the Lego Scrum workshop. CMMI and ISO/IEC 15504 are not discussed in this section, since our process is merely applicable to a small-scale project (with a defined time frame of no more than four hours). Thus, the aforementioned models may not be suitable for our endeavor.

**iFLAP** is an inductive process assessment framework that puts an emphasis on the involvement of practitioners in the assessment process [16], [17]. Its core principles are to: (i) construct a feasible selection of roles (and projects), (ii) gather data points, document them, and *triangulate* the improvement issues by analyzing the data, and (iii) construct an improvement plan based on the analyzed data, identify dependencies among the improvement issues, and prioritize them. These key principles (of iFLAP) are adapted from a summary provided by Pettersson et al. [16].

A central part of iFLAP is the selection of **roles** that are to be involved in the assessment process. This way, once the analysis is completed, the findings represent viewpoints of a broader spectrum of actors, and the viewpoints are not limited to the team only. Having applied this technique to our process, we would be able to obtain additional insights

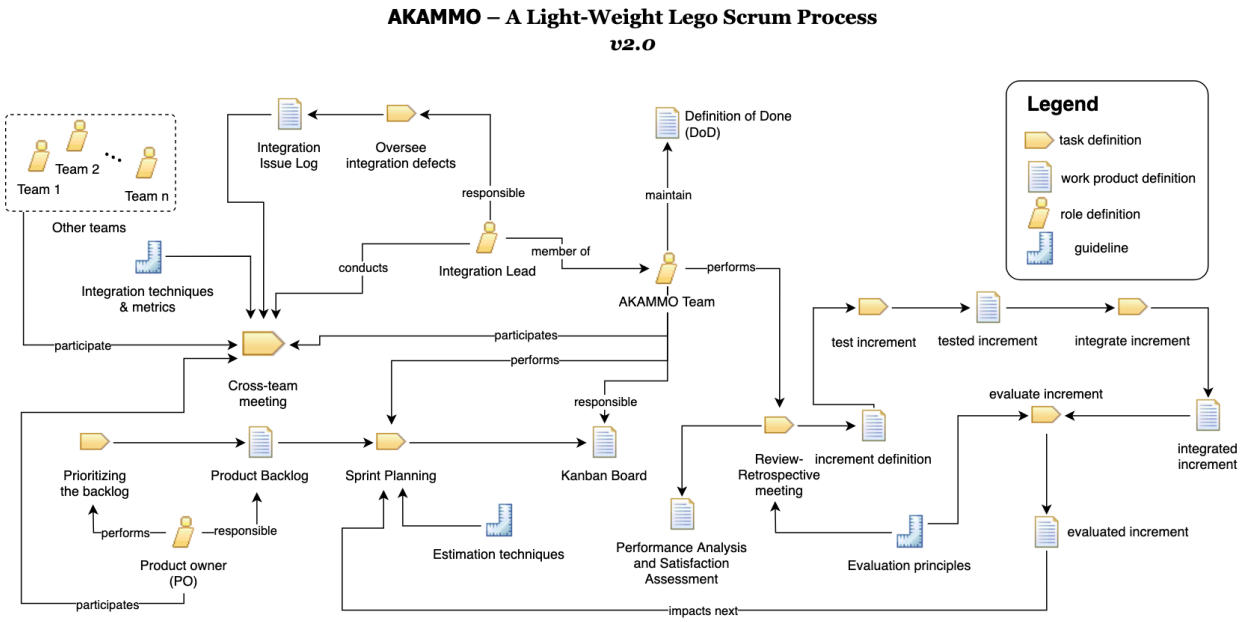


Fig. 1. SPEM 2.0 diagram of the updated process

from, for instance, the PO(s), the instructor(s), other teams, scrum masters, and so forth. This would allow us to identify any shortcomings of the process from different perspectives. Consequently, the concern discussed in Section III about the lack of cross-team communication would naturally diminish, as the technique encourages broader communication. While the technique itself appears rather straightforward, it does present several limitations. The role selection is a time-consuming process and might hinder the progress of the project. Additionally, biases and other human factors may influence the role selection. Thus, the assessment findings may provide unreliable outcomes.

Furthermore, iFLAP suggests to **gather data** from the selected roles with the help of interviews or questionnaires. This would allow the team to strengthen the process by identifying issues in a more systematic manner. However, similar to the previous technique, the primary limitation is the time that is required to gather the data. Interviews, on a general note, provide valuable insights into the issues at hand, albeit they tend to be time-consuming and tedious. Questionnaires, on the other hand, are relatively easy and quick to conduct, but they do not provide as much detail as interviews do. Luckily, an analysis of a questionnaire becomes feasible with the help of automated tools or scripts that can be written by any general-purpose programming language (e.g., Python, Julia, etc.). Nevertheless, having applied this technique to our process, we could leverage the gathered data to enhance the process in the future stages of its maturity. For instance, we could identify a new strategy on how a DoD is to be modeled, or, perhaps, we could introduce a new technique to estimate user stories.

Ultimately, iFLAP further recommends to identify **dependencies between improvement issues**, as well as *"package"* the improvement issues into improvement areas [16], [17]. For instance, the lack of a DoD is an improvement issue and corresponds to an improvement area of *quality assurance*. The identification of such dependencies allows the team to tackle the issues in a more efficient manner. That is, if a certain dependency is found, then a successful resolution of one issue might result in a successful resolution of another issue, or, at least, provide an indication of how to resolve the latter issue. This technique, under general circumstances, is not too demanding to realize. However, an extensive domain knowledge is required to be able to draw dependencies between the improvement issues, and, in a larger-scale project, this becomes a challenging task [16]. Though, in our case, the technique is relevant and feasible to implement—it must be, however, assumed that team members have a sufficient understanding of the domain, so that dependencies can be drawn between the improvement issues in a sensible manner.

## V. SPI Proposal for Future Scrum Development Efforts

In this section, the GQM tool is utilized as part of the SPI proposal. The analysis extends the work the team conducted in the assignments of the course, notably [18], [19]. Basili et al. [4] define GQM as an approach which underscores the importance of organizations defining their **goals**, linking them to operational data, and establishing a framework for data interpretation in order to **measure purposefully** their progress towards the goals. It was originally defined to evaluate defects in a set of NASA projects [4], though, over the years, it has been tailored to be applicable to a broader spectrum of projects.

Before proceeding with the SPI initiative, let us define the notation used in the remainder of this section. A goal is written as *x*). A question follows the notation Q*x.y*. Similarly, a metric follows the notation M*x.z*. A metric, in our context, provides a measurement plan with a general idea of how the data is to be collected and analyzed. In some cases, there is an example provided that illustrates the metric in a more concrete manner.

Based on the observations made in Section III, the following **goals** are defined for the SPI proposal:

1) Improve the clarity and consistency of task completion from the development team's viewpoint.
2) Enhance the estimation and planning with the help of story points from the viewpoint of the development team.
3) Strengthen cross-team communication and integration with a designated role from the development team.

- **Q1.1**: What is the team members' understanding the DoD?
- **M1.1**: Number of items in the DoD that team members share a common understanding of. The data would be obtained from the members of the team with a questionnaire – each team member would be asked to define, in their own words, what the items of the DoD are. Then, with the help of a machine learning algorithm that is able to parse human language, the responses would be clustered into a set of groups where each group contains similar responses. The number of groups would correspond to the metric. In case such an algorithm is not available, the responses can be manually clustered into groups. The information would be stored on a local machine of one of the team members and made accessible to the others (in an anonymized manner). Ultimately, the higher the obtained value, the better the team members understand the DoD.
- **Q1.2**: How consistently is the Definition of Done applied to carrying work related to user stories among the team?
- **M1.2**: A formula in the following manner is to be used: $\frac{n_d}{n} \times 100$ (%), where $n_d$ is the number of tasks that followed the DoD principles when being delivered to the PO, and $n$ is the number of all tasks expected to be worked on within a set time frame (such as a sprint). In the context of the Lego Scrum workshop, it is difficult to keep track of how many times the DoD was applied, since the team members may report inaccurate data. Yet, in a real-world software project, obtaining this metric is straightforward by checking if the DoD was attached to a *merged* merge request (or pull request) in the *main* branch of the repository where the project resides. Nevertheless, for our case, we assume that team members are honest and report accurate data. Subsequently, the aforementioned formula is applied and team members become aware of the results. The calculations are to be stored (and performed) on a local machine of one of the team members. Supposing that each sprint is evaluated separately, the obtained values may serve as a guideline for the current and future sprints. Ideally, each sprint should yield a value of $\approx 100$. Team members are encouraged to set an acceptable threshold interval for the metric, say $[90, 100]$, and strive to achieve a value within the interval. This interval can fluctuate depending on the team's needs.
- **Q2.1**: How accurately do the assigned user story points align with the actual time invested?
- **M2.1**: Decide how many minutes one story point represents (this is to be decided at the beginning of a sprint by all team members). For this example, let's say that one story point represents 5 minutes. Then, once the sprint has elapsed, a randomly chosen team member collects the total number of story points completed during the sprint, say $s$. Afterwards, they collect the total time (in minutes) the team has spent working on the user stories, say $t$. The team members would be previously instructed on how to *track* time with the help of an appropriate tool, such as *Toggl* (or any other tool that is deemed suitable). The chosen team member would then perform a simple calculation on their local machine, namely to take the ratio of the two values, i.e., $\frac{s}{t}$. In this case, the ideal ratio is $\approx \frac{1}{5}$. A threshold interval can be defined as $[\frac{1}{5} - \epsilon, \frac{1}{5} + \epsilon]$, where $\epsilon$ is some small number, say $0.1$. If the obtained ratio falls within the interval, then the story points are considered to be an accurate representation of the time invested. The ratio and interval threshold can be adjusted to suit the team's preferences, based on previous sprints and domain experience. The results are recorded in a communication channel like Slack or Discord for future reference.
- **Q2.2**: How can story points enhance the workload distribution among team members?
- **M2.2**: Establish how many story points a team member is expected to complete during a sprint (this value is decided by the team members at the beginning of a sprint). We label this value as $m$. Assume that team members should work on an equal basis, i.e., they should complete the same number of story points in a given time frame (in our case, a sprint). Keep track of the number of story points that each team member completes during a sprint. Once the sprint is over, assign them to variables $x_1, x_2, \ldots, x_n$, where $n$ is the size of the team. Furthermore, it is the responsibility of each team member to keep track of their own story points—how they do it is up to them. At the end of the sprint, a randomly chosen team member collects all data entries, and performs a calculation on their local machine in the following fashion: $\beta = \sum_{i=1}^{n} |x_i - m|$. Herein, $\beta$ is an arbitrary value that represents the workload distribution among the individual team members. The closer $\beta$ is to $0$, the more evenly distributed the workload is. In that essence, similarly to the previous metric, an acceptable threshold interval (with an appropriate $\epsilon$) can be defined. These values would be determined by the members of

the team based on their experience and intuition.

- **Q3.1**: How effective is the cross-team communication effort?
- **M3.1**: Consider an averaged satisfaction score obtained from both team members and other parties involved in cross-team meetings. A traditional Likert scale from $1 \rightarrow 5$ can be used to obtain the score, where $1$ corresponds to *"Strongly dissatisfied"* and $5$ corresponds to *"Strongly satisfied"*. Ideally, the team should strive to obtain a score of $\approx 5$. A team member who has volunteered (or has been randomly selected, if no one volunteers) will be responsible to notify all people involved in the given meeting about the survey (typically, once the meeting has ended) and provide them with an appropriate link to participate in the survey. The survey itself shall not take more than 1 minute so participants do actually fill it in. The survey can be constructed with the help of, for instance, Google Forms, and the results can be stored in a Google Sheets document. Therein, the team members can observe the results which are updated in real-time and calculated automatically. The sheet can be re-used for future meetings too.
- **Q3.2**: How well do contributions from all teams function together?
- **M3.2**: Record the number of integration problems occurring during a project (or during a sprint, if a more granular approach is desired). The data is to be obtained from other teams and the PO(s) by a volunteer (or a randomly selected team member, if no one volunteers from the team). The gathered data is to be stored in a Google Sheets document or a simple Markdown/text file (which is distributed to all team members). Furthermore, the data is continuously updated, allowing team members to take timely actions if necessary.

In terms of 1), items in the DoD are to be continuously reviewed and updated based on the needs of the team. Observations from the metrics M1.1 and M1.2 are to act as a guideline for the team members to identify possible shortcomings of the current DoD. For instance, if team members are unable to grasp the items of the DoD, then they should be rephrased in a more comprehensible manner. On the other hand, if team members consistently fail to apply the DoD to their work, then, perhaps, some of the items of the current DoD may be cumbersome to follow. As a result, the DoD should be revised and updated accordingly. Additionally, The Kanban board would be extended with a new column titled *QA* (which stands for *Quality Assurance*). The column would hold user stories that are subject to be reviewed based on the principles of the team's DoD and the PO's criteria. Afterwards, if all such criteria are satisfied, the user story is placed to the *Done* column. Otherwise, the user story is moved back to the *To Do* column. This way, the team members would be encouraged to follow the DoD principles and, in turn, enhance the quality of the product.

Regarding 2), using the metric 2.1, team members obtain a clear indication of how accurate their estimations are. As a result, they would be encouraged to investigate reasons behind possible deviations from the ideal ratio. Thereby, they would practice their estimation capabilities. This would not only improve the team's overall performance, but would also help to mature the process itself. What's more, the metric 2.2 would help to uncover any workload discrepancies among the team. Concretely, the two extrema values (i.e., the minimum and the maximum) would be taken from the set of values $x_1, x_2, \ldots, x_n$, and the team members would be encouraged to discuss the reasons behind the discrepancies. Then, for an upcoming sprint, the team member who underperformed would be assigned a smaller number of story points, and, vice versa, the team member who overperformed would be assigned a larger number of story points (team members have the authority to decide on altering the number of story points; however, in case of disagreement, $20\%$ of the story points can be added or subtracted).

To solve 3), we would like to suggest that all team schedules include several meetings throughout the workshop (e.g., once per hour) to discuss any integration issues that may occur or have been encountered. Secondly, we would have a designated role whose responsibility is to oversee any integration concerns and communicate with other teams. The role would be assigned in a round-robin fashion (this way, each team member would have an equal chance to take on the role). Moreover, the assigned person would be responsible to lead the meetings (discussed in the previous improvement step) and would utilize available metrics and measurement plans, particularly M3.1 and M3.2.

## VI. Implementation of an SPI Initiative

Because of an illness, I was unable to attend the second Lego Scrum workshop. Therefore, the alternative task is worked on in this section instead.

A challenge that is commonly encountered is related to the concept of **Change Management**. The term refers to the act of companies adapting to a change or reorganization at any CMM level [20]. It is oftentimes the case that, in the event of a low CMM level, employees are reluctant to change and do not see the benefits of the change [20]. One way to tackle this challenge is to employ a performance measurement technique that would assess the levels of performance before and after each change [21]. Furthermore, this would allow the management team to retain a sensible control over the change, as well as identify target areas for further improvement [21]. In essence, this technique is a form of a **feedback loop** that is meant to be applied to a long-term SPI initiative.

Another difficulty which is typically seen in software organizations is concerned with **knowledge management**. To illustrate, Dybå, in a paper from 2005 [22], sets out to define hypotheses that would help to understand whether a positive association between an SPI initiative and the following two learning strategies exists: (i) *"exploitation"*, the use of existing knowledge, and (ii) *"exploration"*, the acquisition of new knowledge. Furthermore, a question is raised regarding what

a reasonable balance between the two strategies is. The author concludes that the two strategies are not mutually exclusive, and that both learning strategies have a significant contribution to the success of SPI [22]. Thereby, practitioners should apply the two strategies in a symbiotic nature. Additionally, a reasonable balance (of the strategies) is to be decided upon by team members and their management team based on their domain of expertise and experience [22].

Moreover, a recurring barrier of SPI initiatives is the lack of **SPI awareness** which stems from the lack of commitment from the management team and poor communication [23]. Niazi et al. [24] conducted an empirical study to identify critical success factors associated with SPI initiatives, and, unsurprisingly, sufficient SPI awareness was identified as a key factor. Based on the implications of their study, it can be concluded that management teams should be committed to the SPI initiatives in a way that they are able to communicate benefits to the employees clearly and convincingly. This is achievable with a series of workshops and training sessions [24].

Ultimately, a set of human factors is attributed to the challenges that are encountered in SPI-oriented projects. Beecham et al. [20] postulate that human-related factors such as *"lack of motivation"*, *"lack of commitment"*, *"lack of training"*, and *"lack of experience"*, to name a few, are universally recognized in almost any industry. These, however, are not so easily dealt with, and require a significant amount of time and effort to be resolved, which is typically beyond the management team's control (i.e., other bodies of the organization need to be involved). Nevertheless, an empirical study from Pakistan [25] suggests that **recognition** is a key factor that motivates employees to perform better, which in turn, leads to a more successful SPI effort. The study further recommends that management teams should enable employees to participate in the decision-making processes (to some extent, of course) so that the employees feel that their insights are perceived as valuable and are taken into account. All in all, the implications of this study suggest that management teams should strive to establish a **culture of trust** in a prosperous environment where all parties are fairly treated and are able to communicate freely [25].

## VII. Redesign of the Software Process

An updated version of the diagram can be seen in Figure 1. In this section, some key aspects of the diagram are described in more detail.

Most importantly, the task definition *"Cross-team meeting"* has been added. This change should address the SPI initiative in terms of enhancing a collective cross-team communication effort and the detection of integration issues (see the last paragraph of Section III and 3) from Section V). Simultaneously, a new role titled *"Integration Lead"*, whose primary responsibility is represented with a new task definition *"Oversee integration defects"*, has been added. The addition of this role is supposed to realize the second improvement step of 3) (see Section V). During these meetings, an *"Integration Issue Log"*, which the Integration Lead is responsible for, is to be utilized. Furthermore, participation of the PO is encouraged, and the other teams are invited to join the meetings too—they are depicted in a *cluster* of several role definitions titled *"Other Teams"*.

The AKAMMO team members are represented with a single role definition, and the Integration Lead is a member of the team. The team, moreover, is supposed to maintain a DoD and a Kanban board (which are both work product definitions). The team also performs *"Sprint Planning"* and *"Review & Retrospective"* meetings (these remain unchanged and the descriptions from Section II are still applicable). Lastly, following the principles of **incremental delivery**, the team, after each sprint, delivers an increment that undergoes a few stages ultimately resulting in an *"evaluated increment"* which servers as a basis for the following sprint.

## VIII. Bonus Task: Scrum of Scrums

A common pattern that is observed when using a **Scrum of Scrums** (hereinafter abbreviated as SoS) approach is that the role of a Scrum Master (during an SoS meeting) resembles the role of a Project Manager [26], and this is not desirable as per the Scrum principles (using the well-acclaimed Scrum Guide [5] as a reference). Supposing that the Scrum Master does not rotate among teams, then the Scrum Master, put simply, makes decisions on behalf of the teams, so self-organization (which is a key principle [27]) is largely hindered [26]. However, this can, to a certain extent, be mitigated by rotating the role of a Scrum Master among the members of the teams. Nevertheless, other challenges are still present and are discussed next.

Paasivaara et al. in their study conclude that SoS meetings do not function well in large-scale projects with many participants that represent "disjoint interests and concerns" [28]. Typically, a stand-up is not longer than 15 minutes (in a team of 5-9 members [5], for instance) and, naturally, this time frame is not feasible for an SoS effort with many teams participating. Therefore, many of these meetings become overly time-consuming and ultimately counterproductive, cause frustration, and yield little to no value [28]. This is likely a rather unfortunate result of a phenomenon described by Dingsøyr et al. [29] who state that practitioners embrace frameworks for large-scale Agile without a proper consideration of the underlying problem(s) they are trying to solve. The authors further postulate that frameworks should never represent goals; instead, the frameworks should help to achieve particular goals [29]. As a consequence, many organizations may hurriedly adopt the SoS approach, just because it consists of the word *"Scrum"* in some way, without considering whether the approach is suitable for their needs.

It is therefore evident that SoS meetings are not a silver bullet to address inter-team coordination in larger companies. In fact, based on the previous observations, it is not recommended to solely rely on SoS meetings at all. The findings of Paasivaara et al. suggest that SoS meetings can be a viable option in the event of smaller and focused groups that share "joint goals and interests" [28]. Furthermore, highly skilled

teams with a broad understanding of the domain and the Agile practices are assumed [30]. On the contrary, adopting **Spotify Guilds** (that largely rely on self-organization, autonomy, and a sense of community) [31] may be a more sensible approach, for instance. Spotify has been employing Agile principles for many years, and the company has been able to scale Agile to great extents [32]. Needless to say, their approach may not be a *drop-in* solution for all arbitrary cases, albeit it should be treated as a viable alternative and a source of inspiration to other large-scale companies in pursuit of a **suitable** approach to inter-team coordination.

## IX. Summary and Lessons Learned

In this paper, we have discussed a software process tailored for the Lego Scrum workshops, and have identified several shortcomings of the process. As a result, an SPI effort was proposed to address the shortcomings with the help of GQM, and the process was redesigned accordingly. Simultaneously, the observations made were paired with relevant literature and studies to provide a more comprehensive overview of the topics discussed. Interestingly, it has been observed that the abundance of SPI techniques and models is rather overwhelming than helpful, and practitioners should exercise caution when selecting a particular technique or a model. In addition, SPI awareness should be made a central priority in any development endeavor. Besides, teams should consolidate their efforts to establish a culture of inter-team trust and communication. Lastly, we have discussed the SoS approach and its applicability to large-scale projects, and briefly provided an alternative approach that is endorsed by Spotify.

## References


[1] J. Münch, O. Armbrust, M. Kowalczyk, and M. Sotó, *Software process definition and management*. Springer, 2012.

[2] W. S. Humphrey, *Managing the software process*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[3] V. Basili and H. Rombach, "The tame project: towards improvement-oriented software environments," *IEEE Transactions on Software Engineering*, vol. 14, no. 6, pp. 758–773, 1988.

[4] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994.

[5] K. Schwaber and J. Sutherland, "The scrum guide," 2020.

[6] [Undisclosed Authors], "Week 1, assignment 1 in course dit348 h23," Software Engineering Program. University of Gothenburg, Sweden, 2023.

[7] A. Srivastava, S. Bhardwaj, and S. Saraswat, "Scrum model for agile methodology," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 2017, pp. 864–869.

[8] K. Schwaber, "Scrum development process," in *Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings 16 October 1995, Austin, Texas*. Springer, 1997, pp. 117–134.

[9] R. Tamrakar and M. Jørgensen, "Does the use of fibonacci numbers in planning poker affect effort estimates?" 2012.

[10] M. O. Ahmad, J. Markkula, M. Oivo, and P. Kuvaja, "Usage of kanban in software companies," in *9th International Conference on Software Engineering Advances*, 2014.

[11] [Undisclosed Authors], "The process you used during the 1st workshop and the experiences, assignment 2 in course dit348 h23," Software Engineering Program. University of Gothenburg, Sweden, 2023.

[12] S. Kopczyńska, M. Ochodek, J. Piechowiak, and J. Nawrocki, "On the benefits and problems related to using definition of done—a survey study," *Journal of Systems and Software*, vol. 193, p. 111479, 2022.

[13] P. Abrahamsson and K. Kautz, "Personal software process: Classroom experiences from finland," in *European Conference on Software Quality*. Springer, 2002, pp. 175–185.

[14] J. Larsson and L. Larsson, "Integration, application and importance of collaboration in sustainable project management," *Sustainability*, vol. 12, no. 2, p. 585, 2020.

[15] A. Cater-Steel, M. Toleman, and T. Rout, "Process improvement for small firms: An evaluation of the RAPID assessment-based method," *Information and Software Technology*, vol. 48, no. 5, pp. 323–334, 2006.

[16] F. Pettersson, M. Ivarsson, T. Gorschek, and P. Öhman, "A practitioner's guide to light weight software process assessment and improvement planning," *Journal of Systems and Software*, vol. 81, no. 6, pp. 972–995, 2008.

[17] D. Malvius, M. Ivarsson, and D. Bergsjo¨, "Increasing performance in complex product development through structured information and cross-functional collaboration," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 48999, 2009, pp. 1043–1050.

[18] [Undisclosed Authors], "Gqm, assignment 3 in course dit348 h23," Software Engineering Program. University of Gothenburg, Sweden, 2023.

[19] ——, "Software process improvement plan, assignment 4 in course dit348 h23," Software Engineering Program. University of Gothenburg, Sweden, 2023.

[20] S. Beecham, T. Hall, and A. Rainer, "Software process improvement problems in twelve software companies: An empirical analysis," *Empirical software engineering*, vol. 8, pp. 7–42, 2003.

[21] J. S. Oakland and S. Tanner, "Successful change management," *Total quality management & business excellence*, vol. 18, no. 1-2, pp. 1–19, 2007.

[22] T. Dyba, "An empirical investigation of the key factors for success in software process improvement," *IEEE Transactions on Software Engineering*, vol. 31, no. 5, pp. 410–424, 2005.

[23] M. Niazi, M. A. Babar, and J. M. Verner, "Software process improvement barriers: A cross-cultural comparison," *Information and software technology*, vol. 52, no. 11, pp. 1204–1216, 2010.

[24] M. Niazi, D. Wilson, and D. Zowghi, "Critical success factors for software process improvement implementation: an empirical study," *Software Process: Improvement and Practice*, vol. 11, no. 2, pp. 193–211, 2006.

[25] R. Q. Danish and A. Usman, "Impact of reward and recognition on job satisfaction and motivation: An empirical study from pakistan," *International journal of business and management*, vol. 5, no. 2, p. 159, 2010.

[26] C. Larman and B. Vodde, *Practices for scaling lean & Agile development: large, multisite, and offshore product development with large-scale scrum*. Pearson Education, 2010.

[27] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review," *Journal of Systems and Software*, vol. 119, pp. 87–108, 2016.

[28] M. Paasivaara, C. Lassenius, and V. T. Heikkilä, "Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work?" in *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, 2012, pp. 235–238.

[29] T. Dingsøyr, D. Falessi, and K. Power, "Agile development at scale: the next frontier," *IEEE software*, vol. 36, no. 2, pp. 30–38, 2019.

[30] J. Sutherland, A. Viktorov, J. Blount, and N. Puntikov, "Distributed scrum: Agile project management with outsourced development teams," in *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, 2007, pp. 274a–274a.

[31] D. Smite, N. B. Moe, G. Levinta, and M. Floryan, "Spotify guilds: how to succeed with knowledge sharing in large-scale agile organizations," *Ieee Software*, vol. 36, no. 2, pp. 51–57, 2019.

[32] M. Alqudah and R. Razali, "A review of scaling agile methods in large software development," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 6, no. 6, pp. 828–837, 2016.