Why do Software Teams Deviate from Scrum? Reasons and Implications

Mohamad Mortada
mortada@student.chalmers.se
Chalmers | University of Gothenburg
Gothenburg, Sweden

Hamdy Michael Ayas
ayas@chalmers.se
CSE Department
Chalmers | University of Gothenburg
Gothenburg, Sweden

Regina Hebig
hebig@chalmers.se
CSE Department
Chalmers | University of Gothenburg
Gothenburg, Sweden

ABSTRACT

Human, social, organizational, and technical aspects are intertwined with each other in software teams during the software development process. Practices that teams actually adopt often deviate from those of the used frameworks, such as Scrum. However, currently there is little empirical insight explaining typical deviations, including their reasons and consequences. In this paper we use observations to investigate selected activities of the software development process in two companies that use Scrum. We study identified deviations to understand their reasons and consequences, using a survey and interviews. We identify 13 deviations and we categorize reasons based on type. The deviations' consequences are investigated in terms of their impact. Most deviations can be found in multiple teams. Reasons are doubts of the teams, organizational structures and complexity of the work. Consequences of deviations affect product development and team work.

CCS CONCEPTS

• **Software and its engineering → Agile software development**.

KEYWORDS

Agile, Scrum, Process Deviations

ACM Reference Format:
Mohamad Mortada, Hamdy Michael Ayas, and Regina Hebig. 2020. Why do Software Teams Deviate from Scrum? Reasons and Implications. In *International Conference on Software and Systems Process (ICSSP '20), October 10–11, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3379177.3388899

1 INTRODUCTION

Besides the often-complicated technology, software development is comprising of social contexts, environments, people that are an integral part of these environments and also organizational structures and processes [1]. Scrum is one of the most commonly used software processes today [11]. However, research indicates that some development teams deviate from processes [11]. Furthermore, there is evidence for various struggles in software organizations on the adoption of process models and frameworks [6, 19]. Thus, models of software development processes might not always be adopted by the guide.

Today, little is known about the character and reasons for the occurrence of software process deviations. Some research indicates that developers tend to practice activities in the ways that they are comfortable with or closer to their experience, but far from best practice [13]. Therefore, the people responsible for the process might need to change some events or steps during the process execution.

However, even though such actions are common in software processes [10], deviating from guidelines in critical processes might have sub-optimal consequences and implications on the outcome of software projects [20]. The development of an understanding is important, because without a clear indication on the implications, it is not possible to know how effective a process is for a team [10]. Furthermore, without knowing the reasons that lead to a deviation we cannot improve the process [10]. However, there is only few empirical research on the deviations from software development processes. Consequently, there is a lack of knowledge about potential implications of deviations and reasons for their existence.

In this paper, we address the following research questions on the example of the Scrum framework:

- **RQ1:** What are typical deviations from the Scrum framework?
- **RQ2:** What are the reasons for the occurrence of the identified deviations?
- **RQ3:** What are the implications of the identified deviations?

We observed two software development teams and identified deviations by comparing on activities that the teams performed against those specified by Scrum (the targeted software development process in both cases). We further use interviews and an online survey in order to capture reasons and implications of the identified deviations, as well as to gain a first understanding of generalizability of the observations. Our findings help to better understand why software engineers deviate from the process. We further aggregate the results to provide an overview over groups of possible reasons for deviations as well as implications.

We discuss related work in Section 2. In Section 3 we present our methodology. Results are shown in Section 4. Section 5 answers our research questions and discusses the results. Finally, we conclude in Section 6.

ICSSP '20, October 10–11, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7512-2/20/10…$15.00
https://doi.org/10.1145/3379177.3388899

2 RELATED WORK

In this section, literature and related work about the main concepts of the paper are presented. Specifically, we start by describing Software Systems Processes and their challenges in actually adopting them. Then we illustrate how these challenges lead to the modifications of processes with Hybrid Processes and Process Tailoring in the attempt to improve the performance of teams.

2.1 Agile Processes and Challenges

The agile manifesto is comprised of a set of fundamentally human-centered values that have lead to the renowned agile methodologies in software development. A large number of practices, frameworks and process models have been developed, that enable the adoption of such methodologies [7]. The four basic values of the manifesto along with the twelve supporting principles facilitate strong focus on customer collaboration, frequent delivery, and iterative or incremental development through fast and light life cycles [2]. The basic values of Agile along with the supporting principles have set the foundations for process models and frameworks that claim promises for successful software development, such as Scrum and eXtreme Programming (XP) [12].

Software development teams in organisations try to adopt process models and frameworks like Scrum, that are based on the agile manifesto. The most important activities of Scrum according to Kniberg (2015) as well as the Scrum guide of Schwaber and Sutherland (2013) is the facilitation of systematic evaluation of the work done (Daily Scrum), the transparent planning of the future (Sprint Planning), the validation of the resulted products (Sprint Demonstrations) and the continuous reflection on key learning points from gained experiences (Retrospectives) [12, 18]. Some guidelines for Scrum are that:

- All team members contribute in the time-boxed (15 minutes) Daily Scrum and developers have to answer three main key questions.
- Tracking the status of sprints using burn-down charts from the backlog that should be ready before every sprint planning.
- Developers in the team decide the amount of stories to include in each sprint by answering (1) What can be delivered in the increment resulting from the upcoming sprint, and (2) How the work needed to deliver the increment will be achieved.
- Breaking down traceable stories when is needed, in order to help during story estimation, task distribution and alignment of team members.
- Present in each sprint a demo of the finished work to stakeholders, customers or other dependent teams.

From the very early days of Agile, many challenges were apparent in implementing process models and frameworks [17]. The principles and guidelines give clear directions in Software Development towards collaborative and human-centered software development, minimization of unnecessary work, customer involvement and acceptance of uncertainty [7]. These directions however lead to practices that appear good in general but are not based on substantial evidence for their benefits. They are rather based on individual cases that indicate how things should take place and do not necessarily represent how software development teams apply these directions [17].

2.2 Deviations from Agile Software and Systems Development Processes

Studies that investigated software organizations while applying process models have reported multiple struggles in adopting them [19]. Issues in process conformance are critical in applying successfully frameworks, process models and process tools [13]. According to Da Silva et al. (2011), one difficulty is due to the assumption that the process agents (companies, teams etc.) will strictly follow the adopted process model, without making any modification to perform the process differently from the specification, which is not realistic [6]. In fact, there is evidence on how the adoption of agile processes is influenced by many factors, such as the culture in the organization [14]. Additionally, another assumption is that the chosen process model captures the correct steps, stages, and roles required to achieve the company's goals [6].

However, it is not always the case that a process model is absolutely correct, since every team has its own situational factors and unique characteristics which enable high performance [4]. Just like any type of teams, the quality of teamwork in agile is perceived as influential for the learning and work of team members [16]. Characteristics of agile teams can have positive impact like perceiving positively events such as daily stand-up meetings due to opportunities for sharing information, addressing problems and discussing solutions [7]. On the other hand, teams that report their status to managers and that perceive the frequency of the meetings to be too high and the duration too long, tend to perceive negatively the practices and consequently, perform them poorly [20]. Therefore, approaching the adoption of a process with flexibility during its enactment can be beneficial to avoid some negative consequences and find a better fit [6].

In practice, it is noticed that many companies in industry are combining different processes in order to increase the suitability between process and team [8, 11]. Such tendencies is what leads to the development of approaches that emphasize on the evolution of processes and their dynamic adaptation in ways that tolerate deviations [9]. Decomposing processes into the elements of process definition, process performance and process enactment can indicate the effects of deviations on adopting a process model and help for a more focused monitoring [3]. Furthermore, the importance of improving the software process became evident and a crucial step for improving is to evaluate and measure a process [8, 21]. Even though it is clear that the activities that teams actually perform deviate from the activities suggested from process models and frameworks, there is limited evidence that illustrate how exactly the enactment of processes is different, what are the reasons of the differences and what are the implications of deviating in certain activities [10, 20]. Hence, this gap needs to be addressed with clear answers on how specific activities affect team performance, for example in terms of coordination, communication, activities and achievement of goals [8, 20].

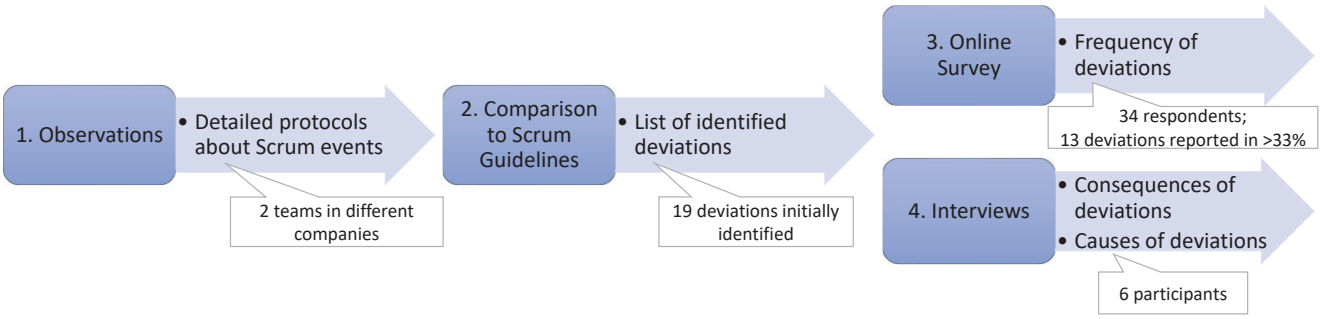

Figure 1: Overview of the Research Methodology

3 RESEARCH METHODOLOGY

We decided to use a case study approach, as boundaries of this subject, e.g. how possible deviations and their implications can look like in practice, was not clear at the beginning of the study. Hence, we conducted a multiple case study and investigated the cases of four Scrum activities in two teams (A and B) at different companies in Sweden, consisting of together 14 people. Team A consisted of five people: one agile coach and four developers. An important note is that the product owner (PO) was one of the four developers in the team. At the second company, team B was composed of nine people located in Sweden: one scrum master, one PO, one developer from Sweden, and 6 other developers from India. As shown in Figure 1, several methods were used to gather data from the teams and to evaluate the findings. This resulted to the identification of 19 deviations initially and their filtering to 13 after evaluating them with a survey.

3.1 Observations & Comparison to Scrum Guidelines

Our main motivation for using observation was the assumption that developers would not necessarily be able to recall and report on subtle deviations from the standard that became the norm within their teams. Using observations, we could record these deviations and use that information later to make developers aware about them during the interviews. Furthermore, observations offered the advantage that we could do the first round of data collection without requiring additional resources such as time from the teams.

As we were not allowed to use audio or video-records, we developed data-sheet templates specifically to protocol the different events. Tables 1, 2, and 3 illustrate the structure of these data-sheets for daily scrum, sprint planning, and retrospective. These sheets were prepared based on agile guides of Kniberg [12] and Schwaber and Sutherland [18].

Team A was observed for four consecutive days. The daily scrum in team B was observed for an entire working week every day at their usual working time (09:30 am). All developers of the teams were working from the same location and the sprints observed were representative to the typical sprints of the teams (e.g. they were not Innovation Sprints). Furthermore, the two teams were observed while doing their demonstration, sprint planning, and retrospective. The observation was performed as non-intrusive as possible. To reach that, only one of the researchers was present at the observed events. Furthermore, as observer we never interrupted, asked questions, or provided any opinions to the teams during observation. The data was directly written into the data-sheet for each agile event. The observation stage took in total three weeks to be finished.

Table 1: Daily Scrum observation template

Time for each	**Total Time**	**Contribution**	**Late comers**
To calculate the average contribution for each developer	To calculate the average time for daily stand up	To make sure if all developers contribute	To make sure that all developers attend and there are no late comers
Sprint backlog	**Awareness of owned tasks**	**Awareness of sprint status**	**Notes**
How the team update their sprint backlog?	To inspect if each developer has a task to do	Does the team trace their sprint to achieve the sprint goal?	Add more notes if any exist

After the observations, we compared the sheets to the Scrum guidelines by Kniberg [12] and Schwaber and Sutherland [18], in order to identify deviations. All 19 identified deviations were documented for the next steps.

3.2 Online Survey

To further identify which of the observed 19 deviations are likely to be generalizable and typical for Scrum, we performed an online survey.

The questions were formulated based on the identified deviations. All questions were kept short and easy to read. Many of them were formulated as statements that participants could agree/disagree with. For example, “We make sure that each member in the team

Table 2: Sprint Planning observation template

Total Time	**Is Product backlog already prepared?**	**Agenda**
How long does the Sprint Planning meeting take?	How does the team create a sprint task board? Do they have it prepared before planing?	Did the responsible of the meeting (Scrum Master) prepare for the meeting?
Presence of Product Owner	**Define Sprint Length**	**Define Sprint Goal**
How does the team pick up their stories for the next sprint and how do they get story specification?	How the team defines their sprints?	To see if the team is aware of what is prioritized for the next sprint

Table 3: Retrospective observation template

Total Time	**Used Method**	**Used Questions**	**Notes**
How long does the team retrospective meeting take?	How does the team decides the important topics to discuss further?	What questions does the team retrospective answers?	For more notes if available

contributes and we give him the turn to speak. (Yes/No)". The survey was divided into five parts:

- *Background questions* concerning the participant, such as years of experience in agile practices and their role in their teams,
- *Daily Scrum* focusing on identified deviations during the daily scrum, e.g., about the length of the daily scrum,
- *Sprint Planning* focusing on identified deviations during the sprint planning, e.g. whether a product backlog is used, how stories are selected, or whether and when goals are set for the sprint,
- *Retrospective* focusing on identified deviations during the retrospective, e.g. whether action points from previous retrospective meetings were implemented, and
- *Demonstrations* focusing on identified deviations during the demonstration.

The survey was sent out to 123 developers from different companies in Gothenburg, Sweden, using convenience sampling. The survey was answered anonymously. We received 34 responses, mainly form developers (24), but also from Scrum masters, software architects, and agile coaches. The median experience of the respondents was 8 years. 12 of the respondents have 10 of more years of experience and only 3 have less than 3 years of experience. Furthermore, 5 respondents have 10 or more years of experience specifically working with Agile and the remaining respondents' experience with agile varied between 1 and 10 years.

As a result we gained a first indication of the frequency and, with it, relevance of the identified deviations. We used the results to filter out those deviations that we identified in the teams, but which were only confirmed by few participants of the online survey. As threshold for that we decided to only move on with those deviations that were reported by at least 33% of the respondents. As a result, we settled on 13 deviations. Note that none of the deviations observed during the retrospective was frequently enough confirmed in the online survey to be reported in this paper.

3.3 Interviews

Finally, we performed semi-structured interviews to get a better understanding of the deviations and to also further confirm the correctness of our observations. The main focus of the deviations was to probe the developers' classification of the deviations and to learn more about causes and implications of the observed deviations. We grouped questions based on the observed agile events and the identified deviations. Leading and judgemental questions were avoided.

The interviews were conducted with altogether six people. These were developers from the two observed teams as well as a product owner and a scrum coach, both working with the teams. With respect for the working hours, most of the interviews took place during lunch breaks. Most interviews were performed separately, to avoid interviewees influencing and biasing each other. Only one interview was performed with 2 developers at once. Developers from both teams did not allow us to audio record the meetings. Therefore, everything they said was noted down during the interviews. The notes were shared with the interviewees directly after the interview, in order to confirm correctness. Each interview took between 45 and 55 minutes.

4 RESULTS

In this section, we present the results on typical deviations, their reasons and implications, sorted by research questions.

4.1 RQ1: What are typical deviations from the Scrum framework?

We identified 13 deviations that seem to be typical for Scrum, consisting of four deviations during the daily scrum event, seven deviations during the sprint planning, and two deviations during the sprint demonstration.

4.1.1 Daily Scrum Event. The two teams showcased some similarities but also differences in their behaviors during the Daily Scrum meetings. During the four observed days in each team, all members attended the Daily Scrum meeting except of the agile coach of Team A in one occasion. Some observed differences were the average duration of the meetings, punctuality of the developers and contribution of the developers in key questions addressed during the meetings.

Not all key questions are addressed (DS1). In both, team A and team B, we observed that key questions are systematically left unanswered by developers. Also 61.76% (21) of the survey respondents report that they do not answer all the questions. Of these, 7 respondents usually answer only parts of the key questions and 14 respondents do not answer any of the key questions. The question that is most often left out is "Do I see any impediment that hinders me or the development team from meeting the Sprint goal?".

Not all team members contribute (DS2). We observed in one of the two teams (team B) that many team members only attended the meetings without actively contributing. Likewise, 14 of the survey respondents (41.2%) reported that they are not strict in making all team members contribute. Furthermore, 22 (64.7%) respondents report that their daily scrum meetings have the form of an open discussion, rather than a structured one.

Daily scrum events take longer than 15 minutes (DS3). Another deviations we observed in team B is that the average observed daily scrum event takes longer than 15 minutes. 53% of the survey responses (18 respondents), too, report that their daily scrum event needs more than 15 minutes. In 16 of these cases, the average time needed for scrum event was even 20 or more minutes. Only 14 respondents answered that the average scrum event takes 15 minutes or less.

No fixed time for the daily scrum event (DS4). Finally, we witnessed that team A had no fixed time schedules for the daily scrum event. This made the observation of this event especially difficult, forcing us to add an additional observation day, as one meeting was missed by the observer. Moreover, 35.3% of the survey respondents report that their teams are not having a fixed time to practice the daily scrum event. An often named reason is that the team waits for late team members.

4.1.2 Sprint Planning. In both teams, sprint planning, was practiced at the same office, where the team normally sits. Product owner and all developers were present, with the exception of the agile coach in team A. It took team A around 2 hours, while team B spent around 1 hour and 15 minutes to finish their planning. The planning meetings were noticed to be mainly unstructured, without a predefined agenda and no special preparation.

Stories are not refined in the product backlog (SP1). Instead of starting from a prepared product backlog and refining stories from there, both teams were observed to only create a sprint backlog during the planning meeting. Also, 44.1% (15) of the survey respondents answered that they do not have a product backlog, but only a sprint backlog.

No calculation of resources available for the upcoming sprint (SP2). During the planning meetings, we noticed that both teams did not calculate what resources they would have available during the next sprint. This step is normally input for planning the amount of workload. Similarly, 47.1% of the survey respondents (16 of 34) reported to not discuss the available resources for the upcoming sprint.

Sprint goal is defined at the end of the planning meeting (SP3). Of the two observed teams only one defined a sprint goal. That goal definition only happened at the end of the planning meeting, when the tool forced them to do so. The other team focused instead on the question "what to demo". The results from the survey shows that 44.1% of the respondents (15 of 34) are also deviating by not defining the sprint goal at the start of their planning meeting. Even more importantly, additional 38% (12) of the respondents reported that they are not used to define a sprint goal at all.

No break down of large stories (SP4). Another interesting observation was that one of the teams did not break down large stories during the meeting. Instead stories were only defined by the team and then broken into tasks by individual developers after the meeting. Likewise, only 44.1% (15 of 34) of the survey respondents reported that they break down stories, while the remaining 55.9% answered that they do not.

No agenda used for the planning meeting (SP5). Both observed teams had no agenda or other clear structure for the planning meeting. Also in the survey, 61.8% (21 of 34) of the respondents reported that they are not using an agenda to organize their planning meeting. Of those respondents, 9 further revealed that they are having long planning meetings (8 hours or longer). Specifically, 3 of them need around 16 hours to finish this meeting.

Stories are not estimated (SP6). We observed that one of the teams did not estimate the effort of the stories. Also, 21 of the survey respondents (61.8%) report that their teams are not always estimating the stories.

Stories not formulated completely (SP7). Furthermore, we observed that both teams defined stories only using title and a small description, leaving out an index and documentation of estimation. 12 of the survey respondents (35.3%), too, reported that they do not use any fields other than name and description to formulate their stories. 52,9% of the respondents report that they are not or only sometimes adding a description to their stories. Only 50% of the survey respondents are indexing their stories.

4.1.3 Sprint Demonstration. The sprint demonstration we could observe included 12 different teams, joining in a large room. Teamnames and developers' pictures were attached to the wall. A whiteboard was placed in the middle of the room.

Sprint does not end with a demonstration (SD1). One of the two observed teams did not do a demonstration at all. Instead they just performed a review of their sprint backlog. Likewise, 15 of the 34 survey respondents (44,12%) do not end their sprint by performing a demonstration.

Demonstration to the wrong audience (SD2). The observed team that performed a demonstration did so to other teams that have no dependency on the outcome (instead of customers, stakeholders, or internally dependent teams). Also in the survey, 61,8% of the respondents answered that they do not get feedback from customers or stakeholders after each sprint planning.

4.2 RQ2: What are the reasons for the occurrence of the identified deviations?

The conducted interviews helped to understand the reasons why the deviations occurred in the two teams A and B. In the following we

Table 4: Deviations caused by human factors.

Habit and acceptance	*DS2: Developers do not want to contribute when they do not have anything to share.*
	SP3: Both teams are just used to it.
	SP5: The power position of the Product Owner and because of habit.
	SP7: The name and description are good enough for the teams.
Doubts on external factors	*DS1: Belief that no practice is required to synchronize a team's findings.*

Table 5: Deviations caused by organizational structures.

Roles of people within the team	*DS3: The number of developers and the high number of issues that should be discussed.*
	SP1: No real Product Owner and ambiguity.
The team in the organization	*SD2: No other dependent teams.*

sort these reasons into three groups: human factors, organizational structure, and work complexity.

4.2.1 Human factors. In Table 4 the causes in this group are summarized into two sub-groups. Some deviations in the sprint planning seem to be caused by an attitude of following the way of smallest resistance. Interviewees mention habit, power positions of specific roles, and a feeling of "good enough", as reasons for the occurrence of the deviations, such as defining a sprint goal at the end of the planning meeting only or not having an agenda. On the other hand we observed a deviation with a reason that was characterized by the interviewees doubts about external factors, specifically about the purpose of activities that are suggested to them as best practice. This shows a lack of conviction. However, it is also a reminder that developers are critical thinkers that do not blindly follow a practice without knowing its purpose.

4.2.2 Organizational structures. The second overarching element that classifies reasons for deviations is the organizational structure, i.e. the question how a team is organized and how it interacts with the rest of the organization, as summarized in Table 5. On the one hand, deviations can be caused by the team composition and roles of the team members. Thus, we found that a high number of developers in a team and the lack of a clear product owner leads to meetings that exceed the predefined duration and stories that are not refined on the product backlog. On the other hand, the role of the team in the organization can play a role. In this case we observed that the position in the organization can affect the ability to meaningfully perform certain activities.

4.2.3 Work complexity. Finally, as illustrated in Table 6, the complexity of a team's work is an element that groups reasons for deviations. First, planning and scheduling of resources groups several reasons. It refers to the ways that the work of teams is influenced because of challenges in planning and coordination of the work.

Table 6: Deviations caused by the complexity of work.

Planning & Scheduling of resources	*DS4: Waiting developers to arrive, and conflicting meetings.*
	SP2: Too much freedom and projected absences for vacations.
	SP4: To save time and due to the difficulty of the product.
Technical Issues	*SD1: Difficulties of the product.*
	SP6: Complexity of the product.

For example, it is difficult to have a fixed time for the daily scrum event when developers have conflicting meetings that make them arrive late. Second, technical issues can be a reason for deviations. For example, difficulties with and complexity of the product can lead to skipping demonstrations and teams not estimating their stories.

4.3 RQ3: What are the implications of the identified deviations?

Finally, we used the interviews with members of the teams A and B to learn more about the implications of the deviations and the developer's perception of these.

We analysed the implications and grouped them according to what they impact. these groups can be further split into two clusters on implications: implications on product development and implications on teamwork.

4.3.1 Product Development. The first cluster is about the implications that affect on the development of the product. We identified two types of deviations in this cluster: deviations that affect the work on value adding activities as well as deviations that affect the quality of the product and achievement of goals.

Work on value adding activities. The work that a team can put into value adding activities can be limited by deviations, which distract the teams with other tasks. For example, the interviewed scrum master reported an incident where the lack of answering all key-questions (DS1) led to identifying issues very late in the sprint. This in turn caused distraction as all team members redirected their efforts to fixing the issue before the sprint, losing time for other tasks. Furthermore, developers report that the deviation of not having a product backlog (SP1) can lead to the failure of including relevant stories into the sprint. This however, can cause other stories to get blocked, causing a failure to implement them.

Finally, our interviewees report that two of the deviations during the sprint planning (the sprint goal is defined at the end of the planning meeting (SP3) and stories are not formulated completely (SP7)) can cause the inclusion of irrelevant stories. The developers perceive deviation SP7 to further lead to ambiguity and difficulty to understand stories.

Product quality and achievement of goals. Deviations seem to be the cause of implications for product quality as well. For example, the above described consequence of deviation DS1 (not all key questions are addressed in the Daily Scrum event) was reported to

Table 7: Deviations' implications on product development

Working on value adding activities	*DS1: Issues can be raised late at the sprint which causes effort from all team members to focus on solving the issue before sprint demonstration.*
	SP1: Failure to implement important required features.
	SP3 & SP7: Inclusion of irrelevant stories.
Quality and achievement of goals	*DS1: Bugs in features.*
	SP1: Difficulties in defining sprint goal.
	SP2 & SP4: Unfinished stories during one sprint.
	SP4 & SP6: Team does not achieve the sprint goal.

lead to additional bugs in the new features. Omitting to calculating the number of available developers for the upcoming sprint (SP2) and to break down large stories (SP4), were both associated with unfinished tasks and overload of the team. Furthermore, the above already discussed deviation of not having a product backlog (SP1) was further reported by the interviewed scrum master to cause difficulties when defining the sprint goal. Finally, besides SP4 another deviation was associated with a direct failure to achieve the sprint goal: the deviation of not estimating the stories (SP6). This was directly associated to sprint failure by the agile coach of team A.

4.3.2 Teamwork. The second cluster is about the implications that deviations have on the work of teams. As summarized in Table 8, we identified three types of implications affecting teamwork: deviations that influence the morale of the team, deviations that influence the organization and planning of the team, and deviations that influence the activities of the team.

Morale. Team morale groups the more intangible consequences that deviations can have on developers. First of all, some interviewees reported that the deviation of not answering all daily scrum questions (DS1) and team members not contributing to the meeting (DS2) causes a loss of trust between the team members. Two deviations were reported to cause fatigue and loss of focus in the team members. On the one hand, this was the deviation of a daily scrum event taking longer than 15 minutes (DS3). On the other hand, the deviation of not having a product backlog (SP1) was named as a cause for boring meetings. Also the deviation of having no agenda for the planning meeting (SP5), was named a main cause for loss of focus and fatigue.

Organization and Planning. Deviations can also have an impact on how teams are organized and their work planning. Missing to calculate the resources available for the upcoming sprint (SP2), was reported to lead to an increased workload for the team. Furthermore, missing to break down large stories (SP4) was reported to increase difficulty to trace progress and stay updated on the tasks. Performing a sprint demonstration to the wrong audience (SD2) leads to confusion in the team. Furthermore, ending the sprint without a demonstration (SD1) was reported to have led to situations where changes to features were requested very late at a stage where it was hard to still modify them.

Table 8: Deviations' implications on teamwork

Team morale	*DS1 & DS2: Loss of trust between members in the team.*
	DS3, SP5 & SP1: Fatigue and members lose their focus.
Team organization and planning	*SP2: Increased workload for the team.*
	SP4: Difficulties staying aligned with updated information on tasks.
	SD2: Confusion within the team.
	SD1: Hard to modify features at late stages.
Team activities	*DS3: Cut the meeting before every team member contributed.*
	DS4: Skip the daily scrum meeting.
	SP1 & SP5: Complex and prolonged/ unfinished planning meetings.

Activities. Finally, we found that deviations can cause further deviations. For example, some interviewees reported that taking longer than 15 minutes for the daily scrum meeting (DS3) leads to teams cutting of meetings without giving the turn to everybody to speak. The lack of a fixed time for the daily scum event (DS4) can lead to teams skipping these meetings. This was perceived as negative by our interviewees. Furthermore, waiting for the meetings led, according to the interviewees to situations of surprise that the meeting happened, which in turn caused people to be not prepared. Not having the stories in the product backlog (SP1) was reported to cause difficulties when refining stories and thus longer planning meetings. Similarly, not using an agenda (SP5) was reported to cause prolonged and unfinished meetings.

4.3.3 Varying perception of deviations. As a final note, it is to be said that not all developers perceived each deviation as necessarily problematic. Specifically, deviations that take place deliberately are recognised by some team members as aspects of improvement. For example, the deviation to not break down stories (SP4) was perceived positive by some, as it helped to shorten already too long planning meetings. Furthermore, the deviation of not answering all daily scrum questions (DS1) and the deviation of not time-boxing the daily scrum (DS3), are perceived as neutral by some interviewed developers. They expressed the opinion that the deviations are harmless and that there is no significant advantage in answering the questions or in being very strict with the duration.

5 DISCUSSION

Our results uncovered 13 deviations from the Scrum framework that seem to be typical, i.e. can be found in different companies. Furthermore, we were able to learn about reasons for the occurrence of the deviations and about implications of the deviations. We grouped these reasons and implications to allow an abstraction over their types.

In the following, we discuss how our results can be utilized by practitioners, relations to related work, implications for research, and threats to validity of our results.

5.1 A method for debugging deviations

Our results can directly be utilized by practitioners, who observe problems in their daily work and want to identify the reasons for that. First of all, the findings of this study can be used to increase awareness about implications of deviations. Furthermore, the above uncovered relations between reasons, deviations, and implications can be used for analysing observed problems and identifying potential changes that can be made to improve the situation. Therefore, we formulate a simple method for practitioners to "debug their process" shown in Figure 2. This method can be used for a preliminary root-cause-analysis of symptoms that are potentially a consequence of a deviation. Practitioners can apply the following steps:

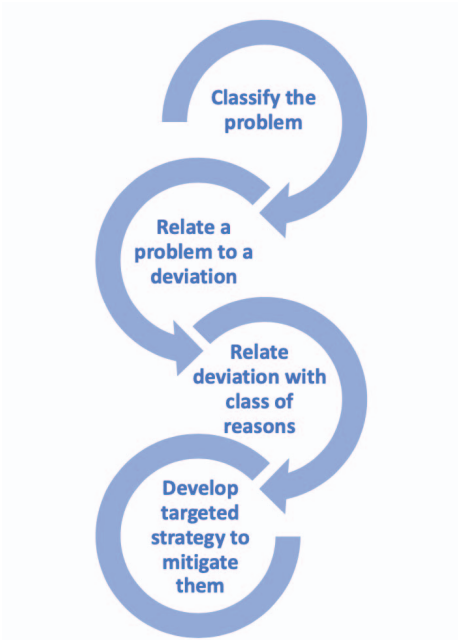

Figure 2: Method for Process Debugging

5.1.1 Step 1 Classify the problem. Practitioners who observe problems in their teams can use the Tables 7 and 8 for a first classification of the problem. Problems that fall in one of the 5 implication groups ('Quality and achievement of goals', 'Working on value adding activities', 'Team moral', 'Team organization and planning', and 'Team activity') are candidates for a further investigation. It is possible that the problem is identical to one of those listed in Tables 7 and 8 under these groups. However, it is also possible that the problem is just similar to one of these already known problems. For example, a problem might be that team members are more and more unaware about the work progress of others and get stuck because they are not aware that the needed input is already finished. This could be classified as a problem of team organization and planning.

5.1.2 Step 2 Relate a problem to a deviation. Now that the problem is classified practitioners can use the tables to start identifying potential deviations causing the problem. The problems identified in this study can be used as a hint about what part of the process to analyze. In the example, the practitioners might start to look out for deviations of the types SP2, SP4, SD2, and SD1, as these are already associated to problems in the 'team organization and planning' group. If a deviation is identified, practitioners can discuss as a team whether they perceive this deviation to be the cause for the observed problem.

5.1.3 Step 3 Relate deviation with class of reasons. If such a deviation is identified as the cause for the problem, the next challenge is to understand where the deviation comes from. Here practitioners can use Tables 4, 5, and 6, to identify what types of reasons are associated with this or similar deviations. In the example above, the deviation identified as cause for the problem could be a missing break-down of stories. Given the deviations we identified in this paper, this could be a hint to look at planning and scheduling of the team. It could be that developers are trying to save time during the planning meetings.

5.1.4 Step 4 Develop targeted strategy to mitigate them. Once a reason for the occurrence of a deviation is identified, practitioners are in a position to develop mitigation strategies. Unfortunately, this can so far not be supported by the result of our study and, therefore, needs to be solved in each team individually. However, we argue that being aware of the cause of a problem is crucial to enable a solution. Of course the deviations identified in this study are just an initial set. Future work will have to further complete the list in order to make the debugging method even more applicable for practitioners. The contents of the groups of reasons and implications can be enriched further with more points that are related to the same thing.

5.2 Relation of results to related work

It has been evident from previous studies that it is challenging to apply process models and frameworks such as Scrum by the guide [4, 6, 9]. However, specific reasons of this difficulty are rarely explored with focus on specific activities of the processes [8]. Literature on deviations focuses rather on the process level, with scope on the overall process enactment [3, 5]. This study's results analyse deviations in depth and decompose them into indicative reasons and implications. With this decomposition, elements are identified that might alter existing explanations for the difficulties on applying process models and frameworks. For example, it is evident that processes following multiple frameworks can be more suitable to teams [11] and deviations can have positive or negative impact on performance [20]. We also observed this, as some deviations were perceived as useful or at least neutral by developers. Nonetheless our sample mostly included deviations with negative impact.

Furthermore, our findings are distinct from those of process tailoring research [10], as deviations are not planned and can have negative effects on the process. Nonetheless, it would be interesting to further study the relation between both fields in future work, as deviations might be a hint that a further tailoring of the process is required. Moreover, tailoring of a process with the intention to improve it, can also have unintentional implications that are negative and risks [8]. Deviations of a process do not always happen in order to improve it, but because it is more convenient and thus, they are unintentional deviations. For example, this could happen because it fits better with the team culture [14].

The proposed simple method for debugging of deviations, provides a way to identify both intentional and unintentional deviations and trace them based on their implications. This shows how knowledge about the deviations and their implications can be utilized for process improvement [21].

5.3 Implications for future research

We highlight implications that our findings have for future research.

5.3.1 Construction and design of process models. Researchers who construct new or compose different process models can benefit from understanding and considering deviations. Specifically, our results provide the opportunity to consider reasons for the occurrence of deviations from the design phase of process models. Thus, common characteristics that lead to such might be avoided deviations or mitigation strategies could be built into the process. Having in mind the human aspects that lead to deviations can prevent the inclusion of activities and practices that cannot be followed [15]. Also, taking into account the effects of deviations can give a different perspective about design choices that are needed. For example, if particular updates on activities are noticed in the identified reasons (or are equal to known deviations) then the projected implications should be considered. Process designers cannot assume that developers are going to follow a process by the guide [6], but it is more accurate to assume that developers will deviate. Hence, the design approach of the process can be changed accordingly. Taking into account that people are going to deviate means that it is useful to study deviations from the start and identify which deviations are to be expected and what is the potential impact. Therefore, when moving to a new practice, the preliminary assessment is more educated with the perspectives of the developers and their work.

5.3.2 Empirical research on deviations. Empirical research in software processes can benefit from the consideration of deviations and their underlying elements to give more comprehensive explanations of various phenomena that appear in processes and their activities. This study explores the existence and underlying elements of deviations. However, there is a need to further explore what types of deviations occur in other processes. We identified relations between reasons, deviations, and implications. Future research will have to show whether the same deviation can be caused by different reasons and whether implications of deviations are stable with changing contexts. Moreover, different contexts have the potential to further elaborate the existing types of deviations or even enrich with new types.

Additionally, this study touches upon the qualitative exploration and identification of deviations, without a substantial quantitative analysis. Therefore, an important direction for future work would be to quantify the ways in which teams deviate from processes as well as the impact of deviations. Such a quantification has the potential to showcase whether agile teams deviate from suggested practices intentionally in order to specialize an adopted process or whether teams deviate unintentionally due to other factors that lead to a lack of conformity. Another example of quantifying the impact of deviations would be to relate deviations with specific characteristics of artifacts that teams develop. Specifically, our suggestion for a future study is to measure specific quality attributes of developed products and investigate how these measurements relate to certain deviations. A potential way to achieve this besides experiments, is with comparing the results of this study with results from analysing data-sets like the one in the HELENA study [11].

5.3.3 Deviations in hybrid processes. We expect deviations to become more complex when considering hybrid processes, i.e. processes that follow a combination of models and frameworks. For example, deviations that appear in hybrid situations might be cause by different process philosophies. Furthermore, such deviations might reveal a teams tendency to adopt a different model or framework than agreed upon. More importantly, during the adoption of hybrid processes or the tailoring of a process, it is beneficial to have an indication of the potential consequences [10]. Such an analysis of deviations like the one of this study, can reveal the unintended consequences of intended changes.

5.3.4 Problem prediction. In future, relating deviations with their reasons and implications along with the potential quantification of their relations, can be used as an input for creating models that predict specific consequences and implications that certain behaviors of teams can result to. Therefore, the area of Artificial Intelligence can use the reasons to make predictions about teams and their work. However, this predisposes that causality is further quantified and proved between reasons and implications.

5.4 Threats to Validity

In the following we discuss internal and external threats to validity.

5.4.1 Internal threats to validity. During the observation activity, one threat is changes in behaviors of participants while being observed. This could lead to having some incorrect measurements. To mitigate this, we aimed to be as non-intrusive as possible during the observations. It is still possible that our presence with the team has had an effect. We further confirmed in the interviews that observed deviations occur also without our presence in the team.

Another threat is the observer's bias on what was noticed. The presence of the researcher was as discrete as possible and thus, it was avoided to ask developers during practicing their scrum events to clarify anything since this could affect their actions and decisions. Thus, the initial results are based entirely on what the observer perceived, which makes it possible that deviations were missed during the observation. To enable the observer to collect as much information as possible, templates were prepared to capture the observations. This allowed for a structured gathering of information and to hopefully minimize amount of deviations not identified during the observations.

One threat that was encountered during the interviews was team members' tendency to talk about the practices they were supposed to follow, instead of the practices they are actually following. Here it helped that we performed the observations ahead of the interviews as it helped us to point to specific events during the interview.

To avoid misunderstandings in the online survey, we kept the questions as simple and short as possible to avoid confusion and make them easy to comprehend. However, it is difficult to fully exclude the possibility of misunderstandings. Therefore, we mitigate the risk by triangulating with the observations and interview data. Only deviations recorded via all three data collection methods were

included in this paper. Two other threats during the interviews can be the wish of the interviewees to please the interviewer with their answers or interviewees' fear of retributions when sharing some information. While the first one leads to the risk that consequences of deviations are overstated, the latter can cause participants to avoid talking about these consequences. In addition, in one of the teams the product owner is also a developer, which can have its influences for certain deviations. Future research will have to be performed to further support or dismiss the generalizability and relevance of the collected implications.

5.4.2 External validity. In terms of external validity, there are also some threats that were identified. Of course, observing a larger sample size of two teams could have lead to even deeper insights. Similarly, a larger number of participants in the online surveys could have further increased our understanding of the frequencies of the deviations. Nonetheless, gaining access to two team already revealed a large amount of detail about how deviations from Scrum look like. In addition, both teams are based in Sweden and therefore, the results are specific to the Swedish Software Development culture, which might have some differences from other cultures.

Furthermore, generalizability of the results needs to be further shown in future work. While our triangulation shows that the results might be valid across different companies, it is possible that there is an undisclosed cultural component to deviations. Similarly, we only studied scrum. Thus, it is not clear how much our results apply to other process models.

Finally, one of the observed teams was reorganised shortly after the study which means that it could be a case with very specific characteristics. However, again the used triangulation with the survey and the other team indicates that the here presented deviations are not specific to that case.

6 CONCLUSION

In this study we addressed the question of how deviations from Scrum look like in practice. Using a mix of observations, online survey, and interviews, we identified 13 deviations from the scrum framework. We further studied reasons for the occurrence of deviations, which were grouped into three clusters of two types each. In addition, we identified implications of the deviations and sorted them into two clusters that together have five types. We believe that these insights will help development teams to become more aware about how deviations work and what they can cause. Thus, we want to enable practitioners to improve their processes, e.g., by understanding deviations from best practices it is possible to address challenges and consequently tackle these challenges in a targeted and personalized way. For example, through training initiatives of software development team members.

ACKNOWLEDGMENTS

Special thanks to Cybercom for facilitating the access to the 2 software teams. Also, we would like to thank the participants of the study for their support and cooperation during the observations, interviews and for their time to answer the survey.

REFERENCES

[1] Gordon Baxter and Ian Sommerville. 2011. Socio-technical systems: From design methods to systems engineering. *Interacting with Computers* 23, 1 (2011), 4–17. https://doi.org/10.1016/j.intcom.2010.07.003

[2] Kent Beck, James Grenning, Robert Martin, Mike Beedle, Jim Highsmith, and Steve Mellor. 2001. Manifesto for Agile Software Development. *The Agile Alliance* (2001). http://agilemanifesto.org/

[3] Sorana Cîmpan and Flavio Oquendo. 2000. Dealing with software process deviations using fuzzy logic based monitoring. *ACM SIGAPP Applied Computing Review* 8, 2 (dec 2000), 3–13. https://doi.org/10.1145/373975.373979

[4] Paul Clarke and Rory V. O'Connor. 2012. The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology* 54, 5 (2012), 433–447. https://doi.org/10.1016/j.infsof.2011.12.003

[5] Gianpaolo Cugola. 1998. Tolerating deviations in process support systems via flexible enactment of process models. *IEEE Transactions on Software Engineering* 24, 11 (1998), 982–1001. https://doi.org/10.1109/32.730546

[6] Marcos Aurélio Almeida Da Silva, Reda Bendraou, Jacques Robin, and Xavier Blanc. 2011. Flexible deviation handling during software process enactment. In *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC*. Institute of Electrical and Electronics Engineers Inc., 34–41. https://doi.org/10.1109/EDOCW.2011.37

[7] Torgeir Dingsøyr, Sridhar Nerur, Venugopal Balijepally, and Nils Brede Moe. 2012. A decade of agile methodologies: Towards explaining agile software development. , 1213–1221 pages. https://doi.org/10.1016/j.jss.2012.02.033

[8] Veli Pekka Eloranta, Kai Koskimies, and Tommi Mikkonen. 2016. Exploring ScrumBut - An empirical study of Scrum anti-patterns. *Information and Software Technology* 74 (jun 2016), 194–203. https://doi.org/10.1016/j.infsof.2015.12.003

[9] Mohammed Kabbaj, Redouane Lbath, and Bernard Coulette. 2007. A deviation-tolerant approach to software process evolution. In *International Workshop on Principles of Software Evolution (IWPSE)*. 75–78.

[10] Georg Kalus and Marco Kuhrmann. 2013. Criteria for software process tailoring: A systematic review. In *ACM International Conference Proceeding Series*. 171–180. https://doi.org/10.1145/2486046.2486078

[11] Jil Klunder, Regina Hebig, Paolo Tell, Marco Kuhrmann, Joyce Nakatumba-Nabende, Rogardt Heldal, Stephan Krusche, Masud Fazal-Baqaie, Michael Felderer, Marcela Fabiana Genero Bocco, Steffen Kupper, Sherlock A. Licorish, Gustavo Lopez, Fergal McCaffery, Ozden Ozcan Top, Christian R. Prause, Rafael Prikladnicki, Eray Tuzun, Dietmar Pfahl, Kurt Schneider, and Stephen G. MacDonell. 2019. Catching up with Method and Process Practice: An Industry-Informed Baseline for Researchers. In *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019*. Institute of Electrical and Electronics Engineers Inc., 255–264. https://doi.org/10.1109/ICSE-SEIP.2019.00036

[12] Henrik Kniberg. 2015. *Scrum and XP from the Trenches*. Lulu. com.

[13] Filippo Lanubile and Giuseppe Visaggio. 2000. Evaluating Defect Detection Techniques for Software Requirements Inspections. *International Software Engineering Research Network (ISERN)* (2000), 24.

[14] M. R.R. Lazwanthi, Abeer Alsadoon, P. W.C. Prasad, S. Sager, and Amr Elchouemi. 2016. Cultural impact on agile projects: Universal agile culture model (UACM). In *2016 7th International Conference on Information and Communication Systems, ICICS 2016*. Institute of Electrical and Electronics Engineers Inc., 292–297. https://doi.org/10.1109/IACS.2016.7476067

[15] Per Lenberg, Robert Feldt, and Lars Göran Wallgren. 2015. Behavioral software engineering: A definition and systematic literature review. *Journal of Systems and Software* 107 (2015), 15–37. https://doi.org/10.1016/j.jss.2015.04.084

[16] Yngve Lindsjørn, Dag I.K. Sjøberg, Torgeir Dingsøyr, Gunnar R. Bergersen, and Tore Dybå. 2016. Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software* 122 (dec 2016), 274–286. https://doi.org/10.1016/j.jss.2016.09.028

[17] Bertrand Meyer. 2014. The Ugly, the Hype and the Good: an assessment of the agile approach. In *Agile! The Good, the Hype and the Ugly*. Springer International Publishing, Chapter 11, 149–154. https://doi.org/10.1007/978-3-319-05155-0_11

[18] Ken Schwaber and Jeff Sutherland. 2013. The scrum guide-the definitive guide to scrum: The rules of the game. *www.scrumguides.org, 2017* (2013).

[19] Borislava I. Simidchieva, Leon J. Osterweil, and Alexander Wise. 2009. Structural considerations in defining executable process models. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 5543 LNCS. 366–376. https://doi.org/10.1007/978-3-642-01680-6_33

[20] Viktoria Stray, Dag I.K. Sjøberg, and Tore Dybå. 2016. The daily stand-up meeting: A grounded theory study. *Journal of Systems and Software* 114 (apr 2016), 101–124. https://doi.org/10.1016/j.jss.2016.01.004

[21] Michael Unterkalmsteiner, Tony Gorschek, A. K.M.Moinul Islam, Chow Kian Cheng, Rahadian Bayu Permadi, and Robert Feldt. 2012. Evaluation and measurement of software process improvement-A systematic literature review. , 398–424 pages. https://doi.org/10.1109/TSE.2011.26