CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Chalmers University of Technology Master's thesis

Deviations From Scrum Framework: Root Causes And Consequences

Master's thesis in Computer science and engineering

Mohamad Mortada

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

Master's thesis 2019

Deviations From Scrum Framework: Root Causes And Consequences

Master's thesis in Computer science and engineering

Mohamad Mortada

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Deviations From Scrum Framework: Root Causes And Consequences

Mohamad Mortada

Supervisor: Regina Hebig, Department of Software Engineering
Advisor: Cybercom
Examiner: Riccardo Scandariato, Department of Software Engineering

Master's Thesis 2019
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2018

Deviations From Scrum Framework: Root Causes And Consequences

Mohamad Mortada
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

This research is a multi-use case study that investigates companies that are using the Scrum framework. It aims to study how companies are implementing the Scrum framework and practice its agile events as daily scrum, sprint planning, sprint retrospective, and sprint demonstration.

Observations and interviews have been made with different companies and developers to discover these deviations, negative and positive experiences, the causes, and its consequences on the team. These results are confirmed by an online survey that has been sent to more than 100 developers at different companies. As a result, some respondents teams are also deviating in practicing the scrum events as daily scrum, sprint planning, sprint retrospective, and sprint demonstration.

At the second round interviews, scrum masters from different teams were have reported the consequences of these deviations in their teams and how they affect their teams negatively or positively.

Keywords: Agile, Scrum, practices, values, principles, negative experiences, positive experiences, deviation.

Acknowledgements

Most importantly, I would like to thank my supervisor Regina Hebig who gave me so much time to guide me through and help me with my Thesis. I would like to thank the two teams who gave me all the help and support during interviewing them and observing their agile practices. To the interviewee, the respondents from all teams, and to the scrum masters that provided me all their support and help. Special thanks for my company Cybercom who provided all support and courses to be able to learn as much as possible about agile and achieve the goal which is becoming an Agile Scrum Master.

Mohamad Mortada, Gothenburg, June 2019

Contents

List of Figures

List of Tables

1
Introduction

The agile manifesto is comprised of four fundamental values: "Individuals and Interactions Over Processes and Tools", "Working Software Over Comprehensive Documentation", "Customer Collaboration Over Contract Negotiation", and "Responding to Change Over Following a Plan" which lead the agile approach to software development. In addition, twelve supporting principles provide more guidance and support to the four main values that focus on customer collaboration, frequent delivery, and iterative and incremental development through fast and light life cycles. Each agile process model applies the four values in different ways.

Struggles have been reported in software organizations while applying process models [15, 16, 25] The difficulty is mainly caused by two strong assumptions made during software processes adoption. The first one is that the process agents (companies, teams..) will strictly follow the adopted process model, without making any modification to perform the process differently from the specification. The second is that the chosen process model captures the correct steps, stage, and roles required to achieve the company's goals.

Researchers try to understand why process agents do not follow the process and the reason behind the two assumptions' failure in the software development domain. They realized that processes depend on many human factors and resources, which invalidates the two assumptions [25]. For example, sometimes, due to unexpected project limitations, a process agent needs to change events or steps during the process execution. In critical processes, having agent deviating from guidelines is illogical, however, as a study [24] states, this is common in software processes. This is caused by the fact that developers' may be confronted with unexpected conditions that are not described in the process model. Another common reason is the unexpected project constraints or simply the developers' willingness to practice the activity in a way that they are comfortable with and closer to their experience but far from the process specification [24].
Despite the presented reasons, we still know little about the root causes of these deviations or even how these deviations are treated by the process agent. Are these deviations planned by the process agent or are they resulted by other factors, beside, how these deviation are effecting the process agent (positively or negatively)? This is a problem, as we don't know the answers of its questions, therefore this study will try to investigate in what cases the process agent deviate, the root causes, and the consequences on the company and software quality.

For this study we define *deviation* as a repeated/systematical divergence from an agile practice's specification and guidelines according to the agile scrum manifesto. A deviation is not necessarily negative or positive.

Currently there is scarce information about typical deviations and their consequences. The conditions under which deviations occur are still not understood. Due to this lack of knowledge, it is difficult for process managers to prevent, detect, and handle deviations.

This thesis focuses on teams that are implementing the agile scrum framework and investigates whether they are deviating. Therefore, the thesis targets four important scrum events, which are: daily scrum, sprint planning, sprint demonstration, and retrospective.

Statement of the problem

Studies [3, 5, 4] have confirmed that agile methods are not perfect and face many challenges such as difficulties in scheduling tasks, difficulties in managing knowledge and difficulties in managing large and multi-site projects. Also, researchers [22], [23] and [20] examined how deviations occur between the executed process by the agents and the process model. They revealed the challenge of allowing agents to deviate from the process specification and correct these deviations later during the process. However, the mentioned studies did not examine the consequences or even the perceived impact of these deviations on the teams.

Moreover, in this thesis, I investigate root causes of deviations and whether it is negatively or positively affecting the team.

Research Questions

This paper will answer the following questions:

- RQ1: How are companies practicing the scrum framework?
- RQ2: In cases where teams deviate from a specific scrum event compared to literature, what are the factors that cause deviation?
- RQ3: What are the positive and negative consequences of the deviations reported by the developers?

The thesis is structured as follows: Chapter 2 will explain the challenges and issues in agile that have been already discovered by previous studies and the guides provided by the agile manifesto. Chapter 3 represents the used methodology to collect the data from observation, interviews and online surveys to compare it further with the scrum guides. Chapters 4, 5, 6, 7 represent the results of the observation, interviews and online survey. Each Chapter is composed of 7 subsections: Observation represents the result of the observation, Summary Identified Deviations represents the discovered deviations during observation, Frequency of the deviation represents the result and the deviation's frequency from the online survey, Perceptions of the deviation and Identified root causes of deviations represents the perception of the

deviations and it's root causes based on the interviews conducted with the developers. Section Identified consequences of deviations represents the results of the second interviews conducted with PO's and Scrum masters to identify the consequences of the discovered deviations. The last section Classification Of The Deviations Based On Root causes represents the classification of the resulted deviations upon its root causes. Finally, Chapter 8 includes the results, discussion, future work, and thread of validity.

2
Review of literature

This section represents both challenges and issues of agile that have been already discovered by previous studies and the guidelines provided by the agile framework that explains how to apply an agile event.

Firstly, previous studies that address the issues and challenges in Agile software development are used to understand if the discovered negative experiences during observation and interviews are common with what already has been studied and to get more knowledge about the issues that an agile team can face such as cultural issues such as study [9].

Secondly, the guidelines and its methods are studied. During scrum planning meeting, the team should declare an initial assessment of how much work is needed to implement a story compared to other stories this attribute is called the initial estimate. To the team be able to determine a certain initial estimate, they should follow a certain method to determine it e.g. “Poker method”. Studying this method would give an estimation of how a team is deviating during practicing it and whether this deviation is positive or negative.

Finally, literature was the source of collecting data about how to practice and apply a certain agile event in a Scrum team. This data is used later to conduct a pure comparison with the collected data from observations and interviews.

2.1 Challenges And Issues In Agile Software Development

Peggy Gregory et. al [1] has developed a "Challenge Wall" method and deployed it at five agile conferences and events between 2013 and 2014 to find out what agile topics are most challenging for organizations. They used this method to collect agile challenges from a range of attendees. They collected around 193 challenges, after analyzing it and came out with four observations:

- It is hard to understand the cultural change which is highly contextual and complex. This challenge has persisted for many years.
- Some challenges that have persisted for many years has changed and matured with the agile maturing as distributed teams.
- Some challenges areas were not being researched.
- Some challenges have been addressed but some were not aware of its evolution.

Kai Petersen et al. [7] investigated the issues and advantages when using agile within industry where agile practices are applied. They found that implementing agile leads to benefits in one part of the process. For example, small team might increase the control over a project, however it leads to new issues on the management level where the coordination of the projects should take place. They identified further examples in their study. They presented four issues that has been identified and agreed on earlier in literature: *" 1) challenges in regard to realize continuous testing, 2) increased maintenance effort with increase of the number of releases, 3) management overhead due to the need of coordination between teams, and 4) detailed dependencies are not discovered on detailed level due to lack of focus on design"*. Furthermore, three of eight found issues was related to their study context which are: *"e 6) high amount of testing documentation, 7) long durations for developing the configuration environment realizing product customizations, and 8) increase in product-packaging effort"*.

Study [3] discuss the challenges that exists in Agile Project Management (APM). In agile, specifications and plans are made at each cycle and not for the whole project, hence agile makes it difficult for the team to estimate a duration for the entire project leading to scheduling problems. Thus, the lack of clarity regarding the completion time makes it hard for the organization to plan other projects and may lead coast changes.
Also, by changing project requirements may makes it hard for the team to monitor the project scope, it makes it difficult for them to differentiate between changes that are necessary and other requirements that are good to have.
As a result, the study concluded that APM is limited by challenges such as difficulties in scheduling tasks, difficulties in managing knowledge and difficulties in managing large and multi-site projects.

Barry Boehm et Al. [8] identified three measures that are critical challenges for software managers to successfully combine agile methods and practices in their agile: (1) development process conflicts, (2) business process conflicts, (3) people conflicts which is the heart of the agile transformation. They categorized these barriers into three categories: Nonproblems, Problems only in term in terms of size or scope and Significant issues.

In another study [9], they revealed cultural issues that might affect practicing agile. They distinct between culture of an organization and culture of the individual which reflect on software development process. They concluded that *it is important to understand the cultural issues related to each process model of the agile culture and the agile practices and techniques that are necessary to address these issues.*

2.2 Process Deviations And Hybrid Processes

Paul Robinson et Al. [14] reported that the Sony Interactive Entertainment (SIE) team varied their scrum practices from the scrum descriptions. This ran the team into a number of difficulties when using scrum to manage their development; the

team failed to complete scrum sprints where tasks required interaction with the open source community. There are two major differences: the first is that the team was not co-located and they had one virtual team with unified process and backlog, and the second is the inherent conflict between how scrum is intended to operate and the requirements of the open source software development process. They tried to adapt scrum through lengthening the sprint, commit locally, handle the review off-sprint, and handle upstream review as its own tasks, as a summary, the burn-down data confirmed that sprints were not working as indented what caused incomplete sprints.

Papers [15] [16] [17] [18] reveals how "deviation" occurs between the executed process by the agents and the process model. They all reveal the challenge of how to allow agents to deviate from the process specification, and correct these deviations later in the process.
Cugola et Al. [15] examine how to deal with deviations during the process model, they distinguish between the software process as specified in a process description and their actual performance by humans. They explore how the two differ and illustrate how a temporal logic-based approach can be used to capture and tolerate deviations from the process description during execution. They reveal how the process model is important in practice during software development to reduce the danger of ambiguity and improve communication among peoples, and the problem of how to adjust the predefined model to enhance it is practice during software development. Also, they focus on the problem of how to remain the process model consistent even if a deviation occurs during software development.
Marcos el Al. [17] mention in their paper how the Process-centered Software Engineering Environments (PSEEs) tool is limited and forbid the process agents to follow the process model and that their deviations get detected and handled on time to avoid the two mentioned issues by this paper. On the one hand, process models are inherently incomplete, since companies can not capture all possible situations in a single model. On the other hand, managers can not force process participants (agents) to strictly follow these models.

Papers [19] [20] [21] represent the challenge of what contextual factors influence the use and combination of hybrid software development approaches. The hybrid approach is a combination of agile practices and traditional approaches that an organization has adopt and customize it for it is own context need to avoid regulations during software development.
In 2011, West et al. [20] revealed the term “Water-Scrum-Fall” and concluded that hybrid development methods will become the standard. A systematic review by Theo-charis et al. [23] tried to collect evidence to confirm West’s claim. They revealed a gap in the literature: while research on agile software development is rich, traditional processes are widely ignored in recent research. Therefore, data and evidence on combination patterns and contextual factors driving the creation of hybrid approaches are missing, e.g., how do standards affect the use of agile methods, or do company size and industry sector matter?
The software development process in practice is studied by Regina Hebig et Al. [22]. They found out that 76.8% of the companies are implementing hybrid meth-

ods. They reveals that companies that are combining planned improvement program with process evolution can increase their process suitability by 5%.

2.3 Agile Guide

"Scrum and XP from the trenches [10]", "The Definitive Guide to Scrum [11]" and "Safe Scaled Agile" was my guides to learn about Agility. They try to define Scrum and explain how to practice each event to achieve agility in a team.

The following sections will only cover the important agile events that should be used during the software development process.

2.3.1 Sprint Planning

Almost all reviewed literature confirms the importance of this event in Scrum which is applied to give the team all needed information about the upcoming sprint.

Sprint planning is a time-boxed event to a vice time frame that varies between all three reviewed resources, [9], [10] stated 8 hours as maximum hours for one Sprint.

All three guides confirm that Sprint planning should address the following:

- A list of stories included on the backlog.
- A sprint goal.
- A defined sprint demo date.
- Available team members and their commitment level for the upcoming sprint.

Also, before the sprint planning meeting, the product backlog should exist including all stories that the Product Owner (PO) sees it important for the upcoming sprint. It is the PO and Scrum Master responsibility to understand the importance of each story and explains it to team members during the meeting.

Each story should have some attributes to describes it, e.g. a story should have an ID for traceability and a description to be understandable.

Furthermore, Sprint planning starts with preparing a preliminary agenda for the meeting to reduce the risk of breaking the time frame. Also, it is important that all team members attend including the PO to define the sprint planning goal. During the meeting, attendance is free to use any available methods as Gut Feel, Velocity Calculation or any other to decide which stories to include in the sprint as Henrik stated in his book [10]. They are also free to use any available method to estimate the working hours for a vice story e.g using planning poker cards.

It is important to keep the meeting inside the planned time frame for the meeting to do not cause fatigue for the contributors.

2.3.2 Sprint Demos

Or sprint Reviews as it's called in some guides, it is one of the main objectives of using Agile is to demonstrate team achievement at the end of each sprint, the demonstration should summarize the sprint goal decided at the start of the sprint.

Doing demos is good to change the atmosphere inside both the team and company: it makes developers feel good when they deliver and it push the team to finish stories for demonstration, it is a way to communicate team achievement to other teams inside one company, finally, it helps developers to get feedback from stakeholders.

Some guides as [10] stated that some teams do two reviews: a short review for public aimed at external stakeholders followed by an internal review with more technical details and decisions.

Safe Guide defined Sprint review or Iteration Review as it is called in their framework that the team demonstrates each story accomplished, culminating with the team's increment of value for that iteration [12].

2.3.3 Sprint Retrospective

Human nature wants always to move forward with the next thing to do once the job is done, hence it's important to do not dismiss sprint retrospectives. The most important things of retrospectives are to evaluate the sprint and improve the negative feedback provided by team developers.

Henrik [10] considered that without retros the team will keep doing the same mistakes, hence he sees retros as the second most important event after sprint planning.

It is not specified exactly how to do retros, there is no specific guide or method that explains how to practice it. However, the three guides state that retros should answer three questions:

- What was good in the sprint?
- What are the things that could have been done better?
- How can we improve?

The team is going to be brainstormed and write their comments on Post-its. Comments should be added in the specific column corresponding the previous questions on a whiteboard or whatever the team is using.

Almost all guides use the dote voting methods to determine which improvements to focus on during the next sprint.

2.3.4 Daily Scrum

The daily scrum is one of the main important events in Agile, recently the Scrum guide updated the three questions to target this [10], [11]:

- What did I do yesterday that helped our team meet the sprint goal?
- What will I do today to help our team meet the sprint goal?
- Do I see any impediments that prevent me or our team from meeting the sprint goal?

In the daily, all team members should attend for 15 minutes as the maximum recommended time frame from almost all guides. Team members should respect the time and be there at the determined time. Also, all team members should agree about how to deal with latecomers and make sure that each team member knows his task of the day.

Different opinions were found about when is the right time to update the task board, is it before, during or after daily Scrum. Henrik stated in his book that updating the task board should happen during the daily's while Ken Schwaber et al. [11] argue this by defining this time for team communication and not administration. I found no statement that recommends doing the update after the daily.

2.3.5 Sprint Dashboard

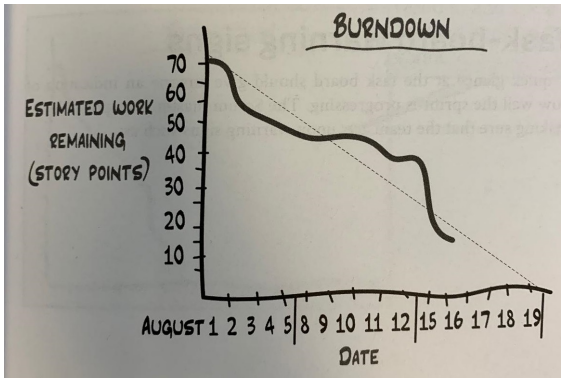


Figure 2.1: Sprint backlog Burndown Chart [10]

Dashboard design varies from a company to another, some uses Excel, Jira, VSTS (Azure-DevOps) and others, there is no opinion that claims which one is the preferred. However, the main reason for having it is to allow story tractability for team members. They should be able to see the not committed stories, the ongoing and the done stories.

The dashboard should be composed of at least three columns where the first contains the "Not checked out", the second "Checkout out" and the third is "Done". This pattern will help the developers to trace their activity during the sprint. Other design forms may add more columns as "In Testing" for example or any other, it depends on how the design suits the team.

The "Burndown chart" (fig 2.1) is a common valuable thing that should be added to the dashboard to visualize stories estimated work and to indicate how well the

sprint is progressing. For example, the bellow chart indicates from the first glance how to sprint progress is going, the progress is aligned with the line drawn from Y into the X-axis. It is a bad sign if the curve is above or under the line so the team should respond to this by reducing or increasing the number of stories to achieve sprint goal. Having the sprint goal on the dashboard is also important as a reminder for the team members.

3
Research Methodology

The case study method is used in this thesis. Case studies are used when the boundaries of a subject are not necessarily clear which is the case in this thesis since we do not necessarily know what character deviations have, what positive or negative experiences are made and what potential reasons for experiences other than deviations might come up in the interviews.
This thesis is considered a multiple case study since it studies cases at two different companies. The more cases are studied the better generalizability results.
Four phases are used in this thesis, each phase consists of a goal, data collection, analysis and an addressed research question as shown in section 3.2. Additionally, I used various methods to help gathering data from team developers and evaluating it as described in section 3.1

3.1 Data Collection Methods

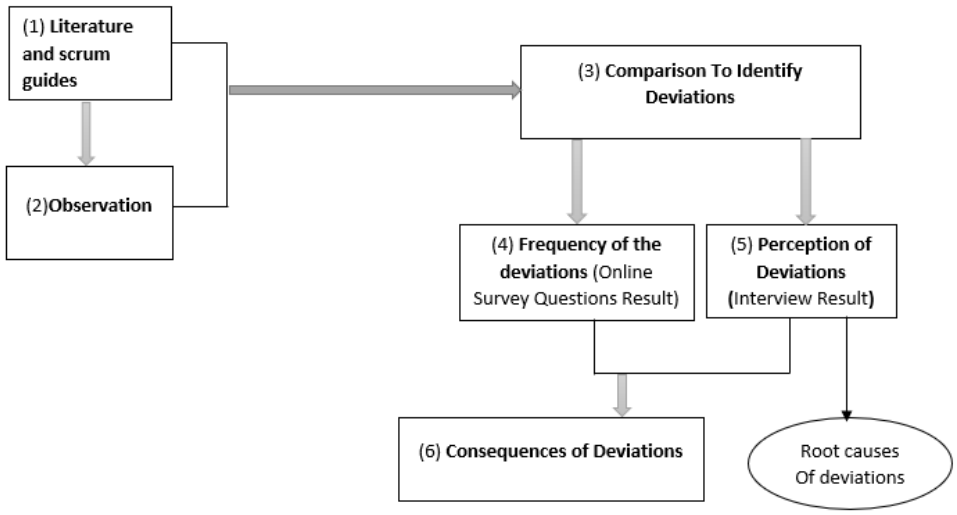


Figure 3.1: Methodology Overview

Figure 3.1 explains the overview of the research methodology, (1) literature and scrum guides are reviewed to get more understanding on how scrum events should be practiced. The output of step (1) is used to create the observation templates at step (2). These templates focus on the provided guides and methods by the scrum

Team member	Role	Company	Agile years of experience
AA	Developer	A	2
AB	Developer	A	6
AE	PO and Developer	A	4
AD	Developer	A	1
AE	Agile Coach	A	
BA	Developer	B	
BB	Scrum Master and Developer	B	3 years as SM and 6 years Agile
BC	Developer	B	2
BD	Developer	B	5

Table 3.1: Teams at both companies

manifesto for each scrum event. Once observation step (2) is conducted, it's output and the output from (1) are then used as an input to start answering the research first question (RQ1) and to create a pure comparison between how the scrum guide specifies practicing each agile event and how the teams are applying it.
The discovered deviations are then used for further investigation by doing an online survey to calculate the frequency of the discovered deviations and by doing interviews with the developers at both observed teams to discuss the positive and negative experiences and the root causes of the deviations. The online survey questions are prepared upon the resulted deviations at step (3).
Furthermore, the results of steps 4 and 5 are used to discuss the consequences of deviations and how they effect the teams with POs and scrum masters at step (6).

Company A, the team is composed of five developers: one agile coach, four developers. Notice that the product owner (PO) is one of the four developers in the team. At the second company, team B is composed of nine developers: one scrum master, one PO and one developer from Sweden, and 6 developers from India.
Table 3.1 represents the information regarding the interviewed team members (AC and AD were not interviewed).

3.1.1 Observation

The main reason for using observation is to inspect how the team practices the agile events thus, increase the validity of collected data.

The second reason is that I want to reduce the time of the interview, for example, some questions can be answered by observations instead of a direct question.

Time for each	Total Time	Contribution	Late comers
This attribute is used 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

Table 3.2: Daily Scrum observation template.

Sprint backlog	Deals With I do not know what to do	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 sprint goal?	To add more notes if any exists

Table 3.3: Daily scrum observation template.

Finally, observation will help to collect data silently without bothering the team while their working hours.

I signed a confidentiality agreement with the teams to respect their private information, therefore, neither video nor voice or picture was recorded or taken during observation.

Also, During the observations, I tried to investigate the participant's behavior, contribution, time, etc. while practicing a vice agile event.
The following tables represent the data sheet I prepared for the daily scrum, sprint planning, and retro respectively and the reason behind using these attributes for each agile event. For example, the "total time" attribute is used in almost all agile events to study how long does each event consumes from the teams working hours. These sheets were prepared upon the outcome of reading literature and guides that describe how to practice agile events.
During the observation, the template in table 3.6 is used to inspect if each developer answers the three key questions of the daily scrum.
Also, I tried to summarize everything the team did during planning, for example,

Total Time	Used Method	Used Questions	Notes
How long does the team retro takes?	How the team decide the important topics to discuss further	What questions does the team retrospective answers?	For more notes if available

Table 3.4: Retrospective observation template.

Define Sprint Length	**Define Sprint Goal**	**Notes**
How the team define their sprints?	To see if the team is aware of what is prioritize for the next sprint	To add more notes

Table 3.5: Sprint Planning observation template.

What I did yesterday	**What I will do today**	**Any hinders that keep the team away from the sprint goal**

Table 3.6: Daily scrum key questions observation template.

how does the meeting agenda look like? do they follow it? in the case where the team schedules their planning meeting etc. The same was made for the retrospective.

Team A was observed for four consecutive days, however, at the second day the team changed their daily scrum to another time without informing me hence, I requested to make the last observation at the last day of their sprint. The daily scrum in team B was observed for an entire working week every day at 09:30.

Furthermore, I was with the two teams the days where the team did their demo, sprint planning, and retrospective.

During the observation, no interruption or opinion was given for the teams. The data was directly written into the template for each agile event.

The observation toke around three weeks to be finished.

Total Time	**Is Product backlog already prepared**	**Agenda**	**Presence of PO**
How long does the team sprint planing takes?	How do 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?	How the team pick up their stories for the next sprint and how do they get story specification.

Table 3.7: Sprint planning observation template.

3.1.2 Interviews

Semi-structured interviews were conducted to confirm the deviations discovered during observation and to get more clearance regarding the unclear observed points. For example, the feeling of one developer cannot be observed if the team has a long daily scrum. Furthermore, I aimed face-to-face interviews because I want to freely ask my questions and clarify the terms I used instead of sending it via email or any other tool.

I used the same questions for different developers separately to get transparent answers as much as possible and to avoided potential bias caused by participants influencing each other.

I followed the interview guide provided by [13]:

- "Yes/No" questions were avoided.
- Leading questions, i.e. I avoided questions that imply an answer from developers.
- I Grouped my question depends on each agile event, for example, I asked all questions related to standing up before I ask about sprint planning.
- I used a non-judgmental question to make the developer feel that there is no specific wrong answer.

I wanted to reduce the interview time and duplicated question that can be answered either by interview or observation, hence observation phase was made before the interview.

Questions were composed upon the agile guide and observation phase. The questions were semi-structured which contained open and closed one. The Interviews were conducted with six developers from both teams A and B with respect for their working hours, most of the interviews were at their launch time. Also, one interview was placed with two developers at the same time.

It takes around two weeks to finish all interviews: developers from both teams did not allow me to record the meeting hence I made sure to note everything they say and document it in the report before conducting the next interview.

Once the interview and documentation are done, an email was sent to the developer to confirm what he said: the result was that two developers wanted me to fix some sentences that I have noted wrongly and four of them agreed.

Each interview takes approximately 45 to 55 minutes. Working at the same company with some developers made them afraid to answer freely. I tried to stay neutral and confirmed that the data is just used for study purposes and nothing else. Only two developers were hard to interview while the remaining were open.

3.1.3 Online Survey

A Google form questioning survey was sent out to 123 developers from a different culture, different roles, different experiences, and has a different assignment at different companies in Gothenburg.

The questions were formulated upon the output of the observation and interview with developers. I tried to keep the questions so short, easy to read and answer. The survey was anonymously answered.

The survey was divided into six parts: Background, Daily Scrum, Sprint Planning, Retrospective, Demo and Sprint Backlog.

Background: To get some information about the developer such as years of experience, years of experience of practicing agile and the role in the team.

Daily Scrum: this section was prepared upon the deviation resulted by the observation and interviews, for example, a question about the length of the daily scrum is proposed to collect more data about the teams that have also the same deviation as team A and B.

Sprint Planning: in this section, I tried to get more information about how the teams do their meeting: if they have product backlog, how they select their stories, if they set up a goal for their sprint and more information that are relevant to the resulted deviation of team A and B.

Retrospective: also, the questions of this section are more to confirm whether teams are doing retros or even implementing their taken action points.

Sprint backlog: during observation, both teams A and B use just the title and description to formulate their stories. The guidelines request more hence I want to get more data about other teams that could be using the same approach as team A and B.

I was uncertain about sending question regarding the negative and positive experiences through the survey for two reasons: 1) some developers would not like that, and they may just avoid the survey, 2) I wanted to keep the survey as short as possible to encourage the developers and get as many answers as possible.

The online survey results 34 responses from different respondents at different companies, **24 of them are developers, two of them are also Scrum Master**, **7 Scrum Master, 1 Agile Coach**, 2 Architect, 1 UX, and 1 Business analyst.

The years of experiences vary between the respondents five of them have 20 or more years of experience, seven between 10 and 20 years of experience and the remaining vary between 1 and 10 years of experience.

Five respondents have 10 or more years of experience working with Agile and the remaining vary between 1 and 10 years. Figure 3.2 represents both the years of experience and the agile experience for each respondent.

Role at the team	Years of experience / year	Agile experience / year
Developer	20	10
Scrum master	9	7
Scrum master	3	5
Developer and scrum master	22	9
agile coach	20	10
Architect	15	15
Developer	1	1
Scrum master	13	6
Developer	10	2
Developer	25	4
Developer	7	6
Developer	6	3
Developer	6	1
Developer	3	2
UX	20	8
Developer	9	6
Developer	8	5
Developer	7	7
ScM + developer	0,5	1
Architect	17	10
Developer	8	5
Developer	5	2
Developer	6	3
Developer	7	4
Developer	5	3
Developer	9	6
Developer	8	6
Scrum master	15	12
Developer	10	7
Developer	12	9
Developer	3	2
Developer	1	1
Business analyst, interim SM sometimes	9	4
Scrum master	6	6

Figure 3.2: Contributors years of experience and Agile experience

3.2 Analysis

Table 3.8 represents the activity of each phase, it's output, and the interview participants. Also, table 3.9 represents both participants contributed during the second round interviews.

Phase 1: The goal of the first phase is to learn more about the values and principles provided by agile scrum guides to prepare for the next phase observation. The output of this phase will be the observation templates that will be used to note the team mechanism during practicing scrum events.

Phase 2: two teams are going to be observed to record their activity during practicing: daily scrum, sprint planning, retrospective, and sprint demonstration. During observation negative and positive experiences or the root causes of deviations are hard to inspect, such information is retrieved by doing interviews with some developers from both teams. The outcomes of this phase are the comparison between how teams are practicing scrum events and scrum guidelines, deviations, and the questions for the online survey.

Phase 3: The outcomes of both previous phases are used to proceed with the interviews and the online survey at this phase. Interview questions are based on the observation results to discuss the positive and negative experiences and the root causes of the deviations. In parallel, an online survey was sent out to 123 developers from a consultant company where it's employee works at different companies in Gothenburg. The survey questionnaire is based on the result of both previous phases. The outcomes of this phase are the frequency of deviations, some consequences, and the root causes of the deviations.

Phase 4: at this phase discovered deviations are discussed with Scrum Masters to analyze the consequences and the effect of the deviation on the teams. One agile coach (AC) from team A and a scrum master from a new company are interviewed to discuss the consequences of the discovered deviations and it is effect on the teams (see table 3.9). The outcome of this phase is the consequences of identified deviations.

Phase 5: this phase is more as an analysis phase to classify deviations upon the reported root causes. The reported root causes can categorize deviations into four categories: planned deviations, inherited deviations, inevitable deviations, and unclassified deviations.
The **planned deviations** are defined as deviations that the team plan and benefit or loose from it's consequences. However, we have to take into consideration that the same deviation can have another root cause with another team which sorts it differently.
The **inherited deviations** are defined as deviations that are caused by another deviation.
The **inevitable deviations** are defined as deviations that the team is aware of but

they cannot find a resolution to it.
The **unclassified deviations** are defined as deviations that the team reported as a suspected root cause. To expand, these are deviations that has no exact specification of the real cause.

Team member	**Role**	**Company**	**Agile years of experience**
A3	Agile Coach	A	10 years
S2	Scrum Master	New Company	8 years

Table 3.8: Interviewee from second round interviews

Phase	**Activity**	**Output**	**Interviewee(s)**
1	Review of literature and related work	- Observation templates	
2	Observation	Comparison between scrum guide and practice Deviations Question of the questionnaire survey	No interviews, observation of team A and B
3	First round interviews Questionnaire	Positive and negative experiences, deviations root causes, and some consequences of deviations Frequency of deviations	Developers D1, D2 from team A and D4, S1, D6, and D7 from team B 123 developers
4	Second round interviews	Consequences of identified deviations	The agile coach A3 from team A and Scrum Master S2 from a new team
5	Analysis	Deviations' Classification	

Table 3.9: Phases summary

4
Daily Scrum Event

This chapter focuses on the resulted deviations from interviews, observation and the online survey of daily scrum (DS) event.

4.1 Observation

This section represents the observational results for both teams A and B at two different companies when practicing their daily scrum event (daily's). The bellow table 4.1 shows the results of the four observed days.

Team	**Average Daily's Time**	**Late Comers**	**Answers The Key Questions**	**Sprint Status**
A	9 minutes	No later comers	All team members answer the daily's key questions during the 4 days.	None
B	approx. 18 minutes	Yes, two developers were late for two minutes at two different days in a row	Not all members answers the daily's key questions during the 5 days	None

Table 4.1: Result of Daily Scrum observation at team A and B

4.1.1 Team A

The total time-average shown in table 4.1 is the sum of the DS total time divided by the total observed days' number. In which the last daily meeting took around 9 minutes because of the team's occupation with another meeting in front of their sprint demonstration. Moreover, during the four observed days, all members attended the DS except with the exception of the agile coach. The team is used to have the sprint backlog presented on a big monitor while practicing the DS. Only the two key questions was answered: *"What I did yesterday? What will I do today?"* Except one developer that did not answer the second question on day 1. None of the developers mentioned anything about the sprint status and I did not notice any tool or method to trace the sprint status. None of the developers answered the third question which is about the existence of any impediments that prevent the

What I did?	What I will do?	Any impediments that prevent the team from meeting the sprint goal	Update Sprint backlog
All members answered with exception of one developer	Not all team members	None	None

Table 4.2: Result of answering the daily's key questions during observation at team A

team from meeting the sprint goal during the four days. All technical issues and others were raised by the PO. I also noticed that he tried to confirm if the team was having difficulties in implementing their tasks by directing the question to each one of them. Table 4.2 represents the result of answering the daily scrum key questions during the four observed days at company A.

4.1.2 Team B

As shown in table 4.1 the comparison between both teams shows that team B needs more time to practice their DS. Their DS was unstructured and was mostly as an open discussion meetings. Thus, recording the time of each speaker is unfeasible, the assessed times are only approximated. Also, the same developer came late twice in a row (day 1 and 2) while only one came late at the first day. I also noticed that neither all developers answer the daily's key questions nor contribute in the open discussion.

C and **Q** are two used shortcuts in the table 4.3 with the following explanation:

- **C** : the team member does not speak during the daily scrum event. He stays totally silent.
- **Q**: the team member does not target all daily scrum key questions but he might contribute in the discussion.

The resulted data in table 4.3 shows that during the five days almost the same developers have the same behavior in answering the DS key questions and in contributing in the discussion.

Day	**C** :Did not contribute in the discussion	**Q**: Did not answers all key questions	**Note**
1	2 developers	3 developers	Same two developers that did not contributes in the discussion did not answer the key questions.
2	2 developers	2 developers	One developer from day 1 and Q contributes during the discussion. Also, Same developer from C and day 1 did not contributes.
3	3 developers	3 developers	Same developer from day 2 and Q also contributes in the discussion. One developer from Q and C have recently came back from vacation.
4	2 developers	3 developers	Same developer from days (1, 2, and 3) and Q contributes in the discussion and raise again the issue he mentioned earlier at day 2. Also, Same developer from days (1,2) and C did not contributes.
5	2 developers	1 developers	Same developer from all previous days and Q but he contributes in the discussion.

Table 4.3: Result of contribution and targeting the DS key questions for team B

What I did?	**What I will do?**	**Any impediments that prevent the team from meeting the sprint goal**	**Update Sprint backlog**
Answers variate from a day to another, below are the number of developers who did not answer this question per day: Day 1: 3 developers Day 2: 2 developers Day 3: 3 developers Day 4: 3 developers Day 5: 1 developers	Not all developers answered this question. Below are the number of developers who did not answer this question per day: Day 1: 4 developers Day 2: 3 developers Day 3: 5 developers Day 4: 5 developers Day 5: 3 developers	The team did not discuss if there are any impediments that prevent to achieve the sprint goal during the five days.	None

Table 4.4: Result of answering the daily Scrum observation at team B

Table 4.4 shows how developers variate their answers concerning their DS key questions. In one observation, one of the developers answered the first question but not

on the second question. The same developer neither target the first question, nor the second or the third in the other observation. The numbers represented in table 4.4 are the numbers of developers who did not answer the DS key questions. Nothing was mentioned or noticed regarding the sprint status and regarding the tools that represented them.

4.2 Summary Of Identified Deviations

Id	Deviation:	Agile Guide:	Team
DDS1	**A team does not answer all daily scrum key questions**	The Scrum guide counts three questions that should be answered during the daily scrum (see section 2.2.4)	Both
DDS2	**A team does not have any tool or use any method to trace the sprint status**	one of the key aspects of daily scrum is to track the status of the sprint. Sprint status can be traced for example by using burndown chart (see section 2.2.5)	Both
DDS3	**Not all team members are contributing in the daily scrum event**	All team member should contribute (see section 2.2.4)	B
DDS4	**Daily scrum takes more time than 15 minutes**	Daily scrum are time boxed event that should not be more than 15 minute per day	B
DDS5	**No fixed time for the daily scrum**	The time and place of daily meeting should be defined at the planning meeting	A

Table 4.5: Resulted deviations in daily scrum event upon observation at Both teams

Table 4.5 summarized the observed deviations for both teams in comparison with the scrum provided guidelines. The guidelines state that developers have to answer three main key questions during the DS. Team A answered the first two key questions but not the third key question. Also, the DS in team B is not structured, not all developers contribute, not all developers answer the three key questions, and targeting the key questions vary from a day to another. For example, a developer can answer the first two questions on the first day but not on the second day, he/she skips answering the first or maybe the second question.
Developer D1 reports during the interview that the team focuses on answering the first two questions of the DS key questions. However, developer D2 from the same company states that they are not strict with answering the key questions, he also claims that a developer does not need to contribute to the discussion if he has nothing to share. Also, all developers from the second team confirm that they are not strict in answering these questions they focus more on the raised issues during the

daily scrum. All developers with the exception of the scrum master do not see a problem with not contributing during the discussion if a person does not have any information to share. In addition, during the interview both teams have confirmed that they do not use any tool or method to check their sprint status. Therefore, **DDS1** and **DDS3** are considered as deviations : "both teams deviate in answering the daily scrum key questions" and "not all members are contributing in the daily scrum event".

The guidelines highlight the importance of tracing the sprint status through discussing the impediments and using the burn-down chart during the DS to be aligned with the sprint goal. However, both teams have reported that they do not use any method or tool to trace the sprint status. Hence, **DDS2** is considered as a deviation where teams do not trace their sprint status.

Almost all agile guides recommend on having a 15-minute daily scrum meeting and not more, however team B needs more than 15 minutes to finish their daily scrum meeting. Team A reported that sometimes their meeting can take longer than what I observed but they try to keep it into the limit of 15 minutes. Team B also, confirms that their daily scrum meetings are long despite the effort of shorting them. Consequently, some teams need more than 15-minutes to practice the daily scrum event as described in the deviation **DDS4**.

Finally, During the sprint planning the team should decide the time and the place for the daily scrum event as the agile guide states. However, team A starts their daily scrum once all team members are united or whenever the PO decided that. The two interviewed developers from team A have confirmed during the interview that they do not decide on a specific time for the daily scrum event in advance. Therefore, **DDS5** is considered as a deviation. Thus, the team does not have a fixed time for their daily scrum event.

4.3 Frequency Of Deviation (Questionnaire)

Figure 4.2 shows the total time used by the respondents' teams to practice their daily scrum. 14 contributors responded that 15 minutes is their daily scrum limit, while 12 contributors responded that 20 minutes is their daily scrum limit and finally 4 contributors responded that 25 minutes consumes their daily scrum activity.
In order to calculate the percentage of the members' responses who are practicing their daily's within 15 minutes range, the total average and minimum average are calculated. The minimum average is the sum of respondents who report their total daily scrum time within the 15 minutes range, while the maximum average is the sum of respondents who report their total daily scrum time outside the 15 minutes range per day divided by the total number of respondents. Bellow is the result:

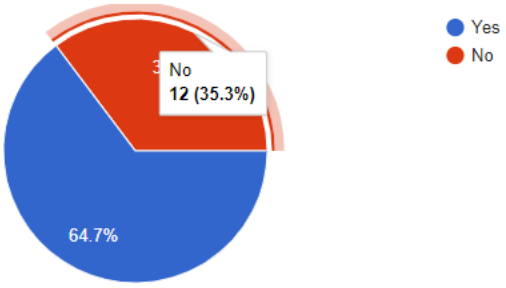
- Minimum Average = 47% responses report that their daily scrum event is within the 15 minutes range.
- Maximum Average = 53% responses report that their daily scrum event needs more than 15 minutes.

The responses show that teams other than team B are also deviating while practicing their daily scrum, the 18 (53%) responses that are outside the 15 minutes range verify the deviation DDS4 discovered during the observation.

Further, 22 respondents (64.7%) do their daily scrum everyday at the same time while 12 of them don't (see figure 4.4-d). The following are their responses:

- 4 responded that they wait for the other team members to arrive; one developer responded: *"People arrive too late and we have to wait for them".*
- One responded said that they are not strict with the time of the daily scrum: *"We are not strict with the time".*
- Another developer related the reason to "other meetings"

Thus, the 35.3% of respondents shown in figure 4.2 report that their teams are not having a fixed time to practice the daily scrum event confirms the deviation DDS5 discovered at team A.



(a) Responses of: "We are used to have the daily scrum everyday at the same time"

Figure 4.1: Respondent answers of daily scrum section

Furthermore, the below histogram (figure 4.3) represents the responses regarding answering the daily scrum key questions: 20 respondents report that they answer both the first two questions (7 of them do not answered the third question) while 13 respondents answered the third question and 14 (41,2%) respondents did not answer any of the key questions. Therefore, the two deviations DDS1 and DDS2 can be confirmed: teams do not answer all daily scrum key questions and teams do not focus on answering the impediments that prevent them from achieving the sprint goal.

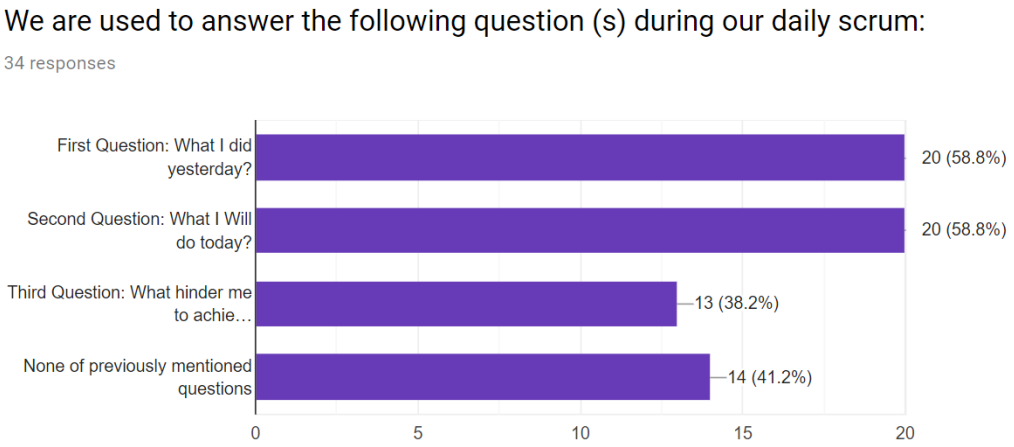
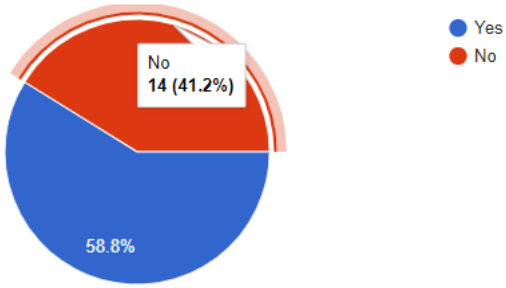
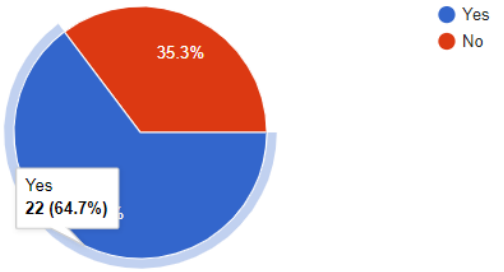


Figure 4.2: Result of Daily Scrum key questions

Finally, Deviation DDS3 is verified by the provided responses shown in figure 4.4-a, 14 of them (41.2%) report that they are not strict in making all members contribute. Also, 22 (64.7%) respondents report that their daily scrum meetings are more as an open discussion more than structured (figure 4.4-b). Therefore, some teams have the same issue as team B where some developers contribute neither in the open discussion nor in answering the scrum key questions.



(a) Responses of: "We make sure that each member in the team contributes and we give him the turn to speak"



(b) Responses of: "Our daily scrum is more of an open discussion than a structured meeting"

Figure 4.3: Respondent answers of daily scrum section

4.4 Perceived Impact Of The Deviations (Interviews)

DDS1: developer D2 from company A declares that avoiding the key questions is not so problematic for them. He believes that daily scrum is meant to focus on discussing the finished and unfinished stories of the sprint with high prioritization and the raised issues of it. This claim is almost confirmed by all developers from team B, for example, developer D6 and D7 do not care about answering the key questions

rather than solving the issue they are facing during implementing a feature.
Furthermore, developers D4, S1, and D7 from company B declare that it is neither crucial to follow the agile guide rules nor to answer the scrum questions, for example, D4 said: *"we are not strict with answering these questions, we are not strictly following the agile guide"*. The scrum master S1 is not happy with the way they are performing their daily scrum meetings. He is aware that not all team members are answering the key questions and he try to encourage the team to be more concerned about participating with the daily scrum questions, however, they did not respond. Accordingly, S1 felt bewildered in that situation. He claims that the reason behind this issue is the cultural background of the team members and their experience of practicing agile.
The answers shows how developers are neutral about the deviation DDS1 except for the scrum master S1. They believe it is harmless and they do not give any advantages of not answering the key questions. However, S1 sees it as a negative deviation since not all team members are going to report their occupation, finding, and impediments to achieve the sprint goal.

DDS2: Further, deviation DDS2 is considered as positive for developers from team B and neutral for developers from team A. Team B developers report that tracing the sprint goal is meaningless since some tasks take more than 2 or 3 weeks to be done because of the difficulty or missing information. Also, the scrum master S1 reason the difficulty of tracing the sprint status for the big number of dependent teams (16 teams) and the complexity of some stories. He also states that sometimes one story can take up to one or two months depending on the available resources or the issues they face during the implementation process. At some cases, the available tools for the teams are not sufficient for them to speed up the process of implementation and testing and thus not efficient. Developer D6 and D7 explain the way of tracing their sprint status by observing the number of finished and unfinished tasks during one sprint.

DDS3:The deviation DDS3 is considered negative for S1 since he insists that each developer in the team has to give feedback on what he is being occupied with lately and the status of his findings. However, as he mentioned, it is hard to force all team members to contribute during the discussion even during answering the key questions because of their cultural background. Also, developers D2, D6, D4, and D7 were neutral regarding this deviation, they only believe that not everyone within the team needs to speak during the meeting if he does not have informative information to share with others.

DDS4: Moreover, the deviation DDS4 is considered as neutral for some developers for both teams with the exception for D4 and S1 that pointed out the issue of having long daily scrum meetings that leads to consuming time, power, and focus from developers. However, developer D6, D7, and both developers from team A did not mention any experiences or feeling about this deviation, for example, developers D6 and D7 explains the reason of having long daily's because the high number of team members in the team. D6 said: *"I don't think about the length of daily scrum*

I just want to solve the issue I have" while D7 said: *"we are eight developers in the team so it's ideal to have 20-minutes as daily scrum, we try to keep it short"*
DDS5: Finally, regarding the DDS5 deviation, both developers from team A report their negative experiences towards not having their daily meeting at a fixed time. Developer D1 dislikes to wait for the other developer to arrive while developer D2 dislike unplanned daily meetings. Therefore this deviation is considered as negative.

4.5 Identified Root Causes Of Deviations (Based On Interview Results)

The below table 4.6 represents the root causes of the deviations extracted from the interviews.

D.Id	Deviation	Root Cause
DDS1	A team does not answers all daily scrum key questions	Developers believe that no needed pattern or method are required to synchronize their findings.
DDS2	A team does not have any tool or use any method to trace the sprint status.	Not breaking down stories
DDS3	Not all team members are contributing in the daily scrum event	Developers don't want to contribute when they don't have anything to share.
DDS4	Daily scrum takes more time than 15 minutes	The number of developers in the team and the high number of issues that should be discussed
DDS5	No fixed time for the daily scrum	Waiting developers to arrive, and conflicted meetings.

Table 4.6: Deviations root causes

DDS1: both teams insist that the main focus of the daily scrum is to synchronize and share their findings or issues without taking into consideration any method or pattern to share this information. For example, the scrum master from team B stated that he lets the team to have an open discussion to let developers share as much information as possible. Also, one developer from team A reports that the most important outcome of daily's is to share the findings and issues if exists instead of focusing on answering the key questions. Therefore, the root cause of the deviation DDS1 is the confidence of developers that believe that a pattern or a method is not required to share information within the team member. They believe that they can target the required information without answering their daily scrum key questions.

DDS2: team B explains the difficulty of tracing their sprint goal by having difficult tasks that might take too long to test and finish. Developer D4 stated that

the product is too heavy, some selected tasks are too broad and cannot be broken into multi-tasks. Also, S1 identifies the reason for the leak of professional hardware tools that the team needs during the testing phase, he also reasons it to the big number of dependent teams that are geographically distributed. Team A doesn't explain the reason for untracing their sprint status. However, the agile coach A3 pointed out that it is the reason to unestimating the stories which makes it difficult to calculate the measurements related to the burn-down chart. Accordingly, not breaking down stories, and not estimating them is the root cause of the deviation DDS2 at the team.

DDS3: no direct cause is reported by the interviewers however, analyzing the responses suspect that the deviation DDS3 is caused by the developers who don't prefer to speak when they don't have any information to share. Similarly, developers D2, D4, D6, and D7 claim that there is no need to contribute when one doesn't have any information to share.

DDS4: the cause of this deviation vary from one team to another, developer D7 in team B reports that they are many developers in the team so it's ideal to have such a long daily scrum meeting. However, D4 and S1 relate the reason of a long daily scrum meeting to the number of raised issues of one story and it's dependencies with other teams, hence daily time is extended to discuss finding solutions for the blocked item to move further with the implementations. Consequently, the root cause of the deviation DDS4 might be the high number of the developers within the team or the high number of raised points during the meeting.

Finally regarding **DDS5**, developers in team A state that the PO is causing this deviation because of his frequent occupation with other meetings. Also, the online survey reports some other root causes of this deviation such as: waiting for others to arrive and the occupation of few members with other meetings. Hence, the root cause of deviation DDS5 is: waiting for others to arrive, and occupation with other meetings.

4.6 Identified Consequences Of Deviations (Based On Interview Results, Second Interview Round)

Table 4.7 represents the consequences of the resulted deviations from the second round interviews.

DDS1 and DDS3: The agile coach from team B pondered about the developer's reported time who do not contribute or even answer the daily key questions as "What I did yesterday?". It made him at many cases lose the trust in these developers who don't proclaim how they consumed their reported time. Also, in some cases, he felt that developers who are not contributing are not aligned with the team. Thus, the reason behind the developers' miss-contribution with the daily scrum questions is the repetition of some questions during the meetings. The scrum master S2 describes another effect that his team has encountered. S2 reported that not answering

daily scrum key questions especially the third key question, has led the team to raise issues (a problem while implementing a task) at late stages just a few days before the sprint demonstrations. This makes the team leave all other tasks and focus on the raised issue to fix it before the sprint demonstration which is time-consuming for the team. Also, he continued and said: "Not communicating an issue at early stages and reporting it few days before the demonstration, has led to speeding up the process of implementation of this task which resulted in bugs that can drag through multiple sprints before it is fixed." To summarize, the following has been reported from the interviewee, not answering the daily scrum questions or not contributing during the daily scrum has led to the loss of trust in team members, bugs within a feature, and lost time that should be used for implementing other tasks.

DDS2: The agile coach A3 answered shortly about this deviation and reported that his team has at many cases failed to achieve the goal of the sprint. He used the word *" The sprint fails"*. When being asked about details, he added: *"The team wasted 2 weeks for nothing."*. The scrum master S2 reported that not tracing the sprint has been a problem for them in some cases. It led to discovering issues late which made them put extra work to finish before the sprint demonstration. As a conclusion, not tracing the sprint status has led some teams to under achieve their sprint goal and put extra work on the team.

DDS4: The scrum master S2, reported that at many cases the team had to cut the long daily scrum meeting without giving the turn for the remaining developers to speak. In some cases, the team had to rush to other meetings or other activities, and in turn no sufficient time was left for other developers at the team to participate.The agile coach from team A has noticed that developers became unfocused and tired of having long daily scrum meetings. They checked their mobiles and watches more frequently than before. Therefore, long daily scrum may lead to the following consequences: fatigue, unfocused members, cutting the meeting before it actually ends, and not giving the turn for the remaining developers to speak.

DDS5: according to scrum master S2, both interviewees reported that the absence of a fixed schedule for daily scrum meetings resulted with less attendance due to the developers' occupation with other activities.

D.Id	Effect	Interview With
DDS1	- Loss of the trust in team members. - Issues can be raised late at the sprint which causes effort from all teams to focus on solving the issue before sprint demonstration. - Bugs in features	Agile coach A3 Scrum Master S2
DDS2	- Team does not achieve the sprint goal. - Discover that sprint is not going well at late stages what leads to the workload on team members	Agile coach A3 Scrum Master S2
DDS3	- Some developers are not aligned with others	Agile coach A3
DDS4	- Fatigue and members lose their focus. - Do not give the turn to remained members to speak. - Cut the meeting with remained members who do not contribute.	Agile coach A3 Scrum Master S2
DDS5	- Skip the daily scrum meeting.	Agile coach A3 and Scrum master S2

Table 4.7: Deviations effects from second round interview.

5
Sprint Planning

This chapter focus on the deviations resulted by observation, interviews and online survey of the sprint planning meeting (SPM)

5.1 Observation

This section represents the result of inspecting both teams A and B at two different companies while practicing sprint planning. One observation is made for this event for both teams. Team A needs around 2 hours, while team B needs around 1 hours and 15 minutes to finish their planning.

5.1.1 Team A

The sprint planning is practiced at the same office where the team sits with the presence of the PO and all remaining developers with the exception of the agile coach.

The planning was unstructured, no agenda was set and no preparation for the meeting was noticed during the observation. They start the meeting by checking the unfinished stories on their sprint backlog which is deployed on Azure-DevOps. Then, they confirmed that all stories are placed correctly and migrate the unfinished stories to the newsprint backlog. The team has a mirror backlog placed on the wall inside the room to facilitate creating the stories during the planning.

The PO starts the meeting by asking the developers about what they want to demonstrate for the next upcoming sprint. He wrote their opinion on the whiteboard and tried with the team to figure out what are the most important and relevant opinions to select for further discussion. Once they agreed about their demonstration target for the next sprint, they checked the existence of any related stories on the wall, if not they created the ticket and attach it to the wall to move it later into Azure-DevOps. Only the title field is used to define one story.
Also, once the meeting is done, each developer selected a story, added it to the digital sprint backlog and broke it down separately.

I also noticed that the PO is the decision maker in the team while the remaining seem to accept his decisions. In one observe situation, developers have defined and agreed about one story for one feature to include in the sprint. However, the

PO refused their decision and created two stories instead of this one feature. In another situation, all developers agreed on including one feature in the upcoming sprint except the PO who selected what he believed is more prioritized and important for the sprint. The team did not react to that and stays calm.

Finally, before closing the meeting, the PO triggered a discussion on how to demonstrate the included stories by dividing the whiteboard into two sections: "How" and "What to demo" and started writing developers' opinions to decide the relevant. Following are some more observed activity during the planning meeting:

- One developer sited on his desk occupied with something else, the PO tried to get his attention, but it turned out that the developer had another meeting with another developer from China.
- The meeting was unfocused and partially turned into a retrospective meeting. For example, the team opened a discussion that toke around 9-minutes discussion about mob-programming and the reason behind things not working out for them during the sprint and how to improve it. This happened multiple times with different other topics.
- The team is independent of the other twelve teams in the organization. This was confirmed by the PO when asking about team dependencies.
- The team put a lot of time on reformulating unfinished stories from previous sprints.

5.1.2 Team B

No agenda is used for their sprint planning meeting. The PO, scrum master, and developers presented in this meeting. The PO connected his computer into a big projector screen, listened to developers opinions regarding the stories to include and wrote it into the backlog deployed on Jira platform. Before adding any new stories, they included all unfinished stories into the new backlog and start pointing out the blocked ones.
The PO was in charge to write developers new recommended stories, he added a title and description for each story while the developers were waiting until he is done typing so they can start another story discussion.

For estimation they used the following method: each team member voted from using the Fibonacci sequence, the estimation is the sum of what all the team member vote divided by the number of attended developers which is 5, only 5 developers including the scrum master and the PO attended the planning meeting.

Moreover, before closing the meeting, the PO and developers tried to clarify each ticket and make sure that everyone understood what the tickets were about, also the PO made sure to break down the tickets if needed. They used the prioritization method to include stories into their sprint, they started with the most important stories and included the remaining once the most prioritized ones were done.

Finally, the last activity for the team was to define sprint length and sprint goal. Following are some more observed activity during the meeting:

- The PO was leading the meeting and not the scrum master, he wrote the stories into Jira but he did not affect the team decisions.
- Too many quiet moments during the meeting because of waiting for the PO to type the stories.
- One developer was around 15 minutes late to the meeting.
- The developers decided which story to include and described it to the PO to include it into the backlog.
- The meeting took around one hour and 15 minutes. The reason for having a short meeting was because of having a lot of blocked stories.

5.2 Summary Of Identified Deviations

Table 5.1 represents the deviations observed at both teams during the implementation of the sprint plan. Nine deviations are discovered according to the agile guides.

Deviation **DSP1** results due to the abscence of a prepared product backlog before the sprint planning among both teams. Even though preparing a product backlog is of great importance and this in confirmed by Henrik Kniberg's guide. Kniberg states in his guide that *"The product backlog is the heart of Scrum. This is where it all starts"*. Both teams created their sprint backlog stories during the planning meeting instead of having it ready on the product backlog. Refining the stories is a different scrum activity than refining them during sprint planning.
During the interview, both teams confirm that they usually neither have an agenda nor prepare for this meeting, they just run it randomly. Comparing the scrum guide with the team's behavior confirms that teams do not have a refined product backlog before starting the planning meeting. In conclusion, the following deviation **DSP1** is summarized: teams are not refining their stories on the product backlog.

The scrum guide explains the importance of calculating the number of available resources for the next sprint to include the correct number of stories in the sprint and avoid work overload. However, both teams didn't apply that during the planning to identify their workload. Inaddition, asking the interviewee about this confirms the observed deviation, for example, both developers in team A explain that they are not used to check the number of available resources during the planning. Also, in team B, the scrum master confirms the availability of a vacation list for the team attached to the wall in their room but the remaining developers BA, BC, and BD think the opposite and that they are not used to calculate the available developers for the next sprint. Accordingly, the following deviation **DSP2** is summarized: teams are not calculating the number of available resouces for the upcoming sprint.

Id	Deviation:	Agile Guide:	Team
DSP1	**Teams are not having product backlog**	Product backlog should be available before sprint planning. section 2.2.1	Both
DSP2	**Teams are not calculating the number of available resources for the upcoming sprint**	A list of team member that are available during the sprint. See section 2.2.1	Both
DSP3	**The PO is affecting team decisions**	It is the developers in the team that decides in how many stories to include in the sprint not the PO or anybody else.	A
DSP4	**Teams are defining the sprint goal at the end of the planning meeting**	In each planning meeting the sprint length and sprint goal should be defined.	Both
DSP5	**A team is not breaking down stories**	Included stories on the new sprint backlog should be broken down by all team members.	A
DSP6	**Developers are having other meetings during sprint planning**	No other meetings during sprint planning.	Both
DSP7	**Teams are not using agenda during their planning meeting**	Planning meeting should be prepared and an agenda for the meeting can help to avoid long planning meetings.	Both
DSP8	**Some important fields as Id is missing when formulating stories within teams**	The manifesto states that stories should at least have an id for traceability for testing and so on.	Both
DSP9	**Teams are not estimating their stories**	During planning, team member should calculate the estimation time for each task.	Both

Table 5.1: Deviation in Sprint Planning event upon observation at both teams.

Further, regarding deviation **DSP3**, the PO avoids developers' recommendations and applies his thoughts instead, while the agile guide gives the developers the power of determining the stories to include in the upcoming sprint. This behavior is confirmed from the interviewee of team A. They explained that the reason of that is the higher experience of the PO as the fact that he takes in roles in the project (developer and PO). As a result, the following deviation **DSP3** is summarized: the PO is affecting team decisions.

The agile guide reveals that the sprint planning answers the following two key questions: *(1) What can be delivered in the increment resulting from the upcoming sprint* and *(2) How will the work needed to deliver the increment be achieved.* In team A, I did not observe that they define the goal of their sprint. The "What to demo" question that they raised, aims to identify their demonstrations of the upcoming

sprint. Team B also starts their planning by creating stories without defining the sprint goal, however, they define it at the end of the meeting when Jira prompted them to add a goal. It is good to note that, in Jira, a user cannot start a new sprint if the goal of the sprint is not set.
Both teams confirm that they usually add the sprint goal at the end of their meeting during the interview. When asking team A about the reason for not defining the sprint goal during my presence, developer A reports that they miss it and that they made a short meeting the next day to decide the goal of the sprint and attach it to the wall. Therefore, teams are defining their sprint goal at the end of their meeting while it should be determined first. In conclusion, the following deviation **DSP4** is summarized: teams are defining the sprint goal at the end of the planning meeting.

The scrum guide insists to break down stories when the story requires that. Breaking down stories helps during story estimation and task distribution. Also, the guide confirms that breaking down stories should happen as a team to share as much possible information between team members. However, during the observation, for example, team A only defines their stories (add a story title on a post-it) and attach it to the wall. Afterward, each developer selects one story, migrates it to their digital backlog, and breaks it into tasks. This was done separately. Developers AA and AB confirm this behavior during the interview. AB report that: *" breaking stories separately would save us so much time"*. As a result, the following deviation **DSP5** is summarized: a team is not breaking down stories.

Also, the scrum guide asks for the presence of all developers during the planning meeting and demands sufficient focus on it. Developers from both teams leave the meeting to attend another meeting. To confirm this behavior, I asked the developers at both teams about that behaivor during the interview, they replied that this can always happen because of the support they have to provide for customers or other teams. Hence, the following deviation **DSP6** is summarized: developers are having other meetings during sprint planning.

Furthermore, **DSP7**: teams are not using an agenda for their planning. The meeting is summarized as a deviation because both teams did not use an agenda or structured their meeting while the agile guide clarified the importance of preparing the meeting and it's agenda to avoid double meetings or even postponing them.

Moreover, A team can experiment whatever fields to add to their stories as the scrum guide says but it's important that each story at least is indexed for traceability, have an estimation field, and a short description on how to demonstrate it. Both teams are having just a title and a small description of what the story consists of. This is confirmed during the interview, developers confirm that they only add a title and a description for one story. Thus, teams are deviating in formulating their stories (**DSP8**).

Finally, Team A don't estimate their stories while I noticed that team B do some sort of estimation for each story. However, asking the scrum master about it ex-

plains that this estimation is not related to the time it is just used to compare the size and the complexity of one story related to another. The agile guide explains the importance of estimating stories to calculate velocity which is a measurement to determine the "amount of work done". Therefore, **DSP9** results as a deviation for some teams

5.3 Frequency Of Deviation (Questionnaire)

Only three respondents report that their team need around 16 hours to finish their sprint planning while the remaining respondents claim that the planning vary between 4 and 8 hours.

The respondents' answers regarding **DSP1** deviation shows that 19 of 34 (55.9%) don't use a product backlog (figure 5.1). Also, another question in the survey is requested to examine if stories are already defined and existed on respondents teams product backlog before starting their planning meeting. The responses show that: 18 of them create their stories during the planning and include it to their sprint backlog, 9 of them have already their stories defined on the product backlog, 4 of them do both previously mentioned methods. However the remaining respondents have another methodology. Referring to the agile guide, stories should be already prepared and included to the product backlog instead of creating them during the planning. Hence, It is clear from the result that teams other than A and B are deviating in the absence of a defined product backlog (**DSP1**).

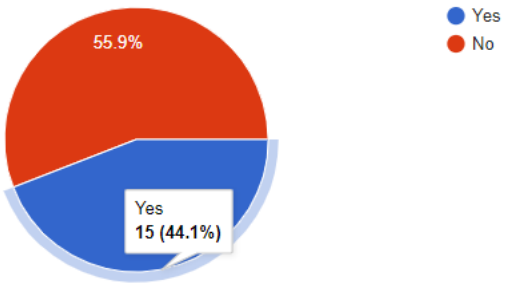


Figure 5.1: DSP1, responses regarding question : We do not have a product backlog, we only have a sprint backlog

Figure 5.2 shows that 16 respondents out of the teams which is equivalent to 47.1% of the respondents do not discuss the available developers for the upcoming sprint while the remaining does that. During the planning, the team should calculate the number of available resources to distribute the workload on all the members and to correctly include the number of stories in the upcoming sprint. The 47.1% of respondents result and the results of observation and interviews confirm that teams are deviating by not calculating the number of available developers of the upcoming sprint, thus, deviation **DSP2** is confirmed.

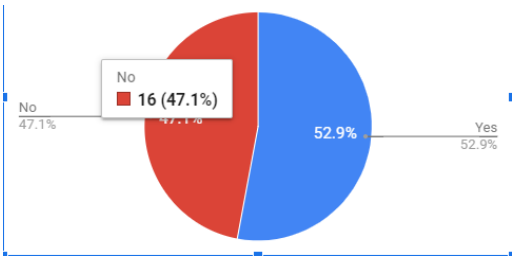


Figure 5.2: DSP2, responses regarding question: We calculate the number of available resources for the upcoming sprint.

The result of figure 5.3 shows that 15 teams which is 44.1% of the total teams other than team A and B are also deviating by not defining the sprint goal at the start of their planning meeting. Also, what is more important to mention is that 13 respondents reported that they are not used to define a sprint goal. For this reason, it's clear that some teams deviate in defining their sprint goal and thus, deviation **DSP4** is confirmed.

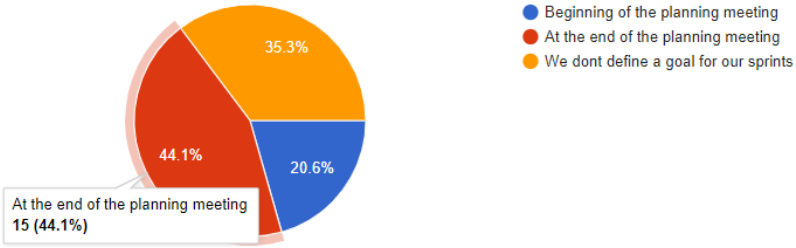


Figure 5.3: DSP4, responses regarding question: We define the the sprint goal at.

Further, one question targets the deviation **DSP5** where teams are not breaking down stories. The result shown in figure 5.4 confirms how some other teams than A and B are also not breaking down stories; 14 respondents (44.1%) report that they break down stories while the remaining 55.9% of the teams answered that they do not and thus, the deviation DSP5 is confirmed.

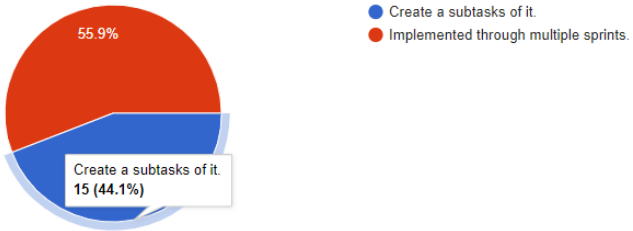


Figure 5.4: DSP5, responses regarding question: Sometimes stories are too large so we used to.

Furthermore, the 61.8% of respondents that report that they are not using an agenda to organize their planning meeting as the pie diagram in figure 5.5 show confirm the

deviation **DSP7**. The scrum guide proposes the agenda as a solution to avoid drag on and on meeting, to organize the meeting and to minimize the length of it. Analyzing the results explains that 9 respondents' teams are having long planning meetings. 3 of them need around 16 hours to finish this meeting, those teams are not having an agenda. For this reason, not having an agenda can affect the length of the planning meeting.

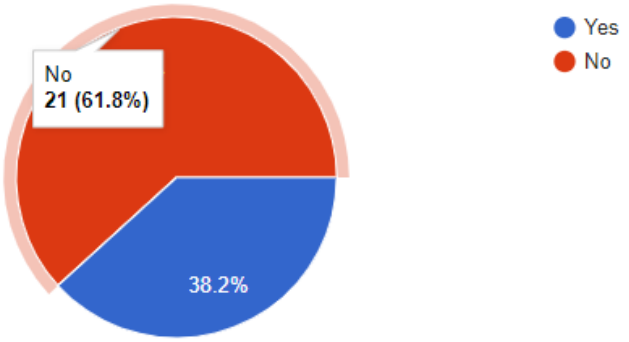


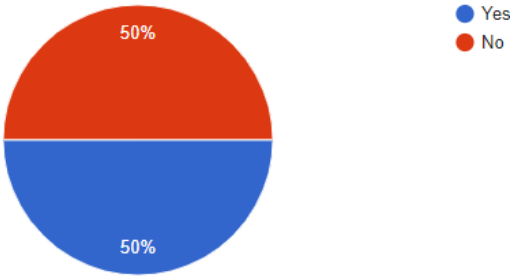
Figure 5.5: DSP7, responses regarding question: We are used to have an agenda for the sprint

Moreover, Figure 5.6 represents the responses regarding story formulating:

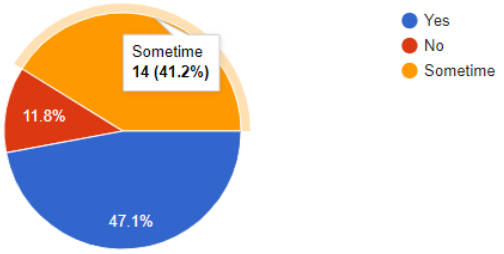
- 50% (fig 5.8-a) of the teams are indexing their stories.
- 47.1% are adding description to their stories.
- 41.2% (14 responses) report that they sometimes add description (fig. 5.8-b).
- 21 respondents (61.8%) are not estimating their stories as shown in figure 5.8-c.
- 35.3% of the responses reveal 12 that their teams are not having any other fields more than the name and description.

From the previous results, two deviations are confirmed: **DSP8** due to the situation where some teams (12 teams form 34) do not formulate their stories as they should and **DSP9** because some teams are not estimating their stories (21 teams are not estimating their stories and 27 teams don't use any technique to calculate the estimation time).

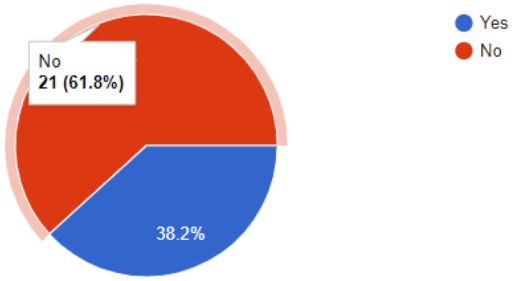
Finally, the numbers do not confirm that the two deviations deviations **DSP3** and **DSP6** resulted from section 5.2 are very frequent. However, this does not mean that they are not valid. Almost all responses confirm that the developers are deciding the stories of the upcoming sprint without any intervention from the PO (only 2 respondents think that their teams' decisions are affected by the PO). Also, almost all developers replied that they make sure that all team members attend the sprint planning and avoid other meetings (only three developers report "No").



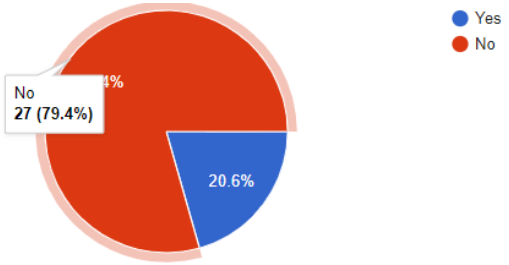
(a) Responses of: "Our Stories are indexed"



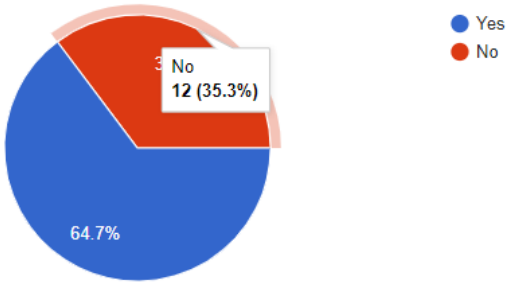
(b) Responses of: "We always add description on our stories"



(c) Responses of: "We always estimate each story"



(d) Responses of: "We use specific techniques to calculate the estimation for each story"



(e) Responses of: "Our stories contains more fields than the name and description for example: estimation, how to demo, notes ...."

Figure 5.6: Response summary of Backlog section

5.4 Perceived Impact Of The Deviations (Interviews)

Not having a product backlog (**DSP1**) makes developers wonder about the reason of the usage of scrum framework. For example, developers D4, D6, and D7 were confused for not using the product backlog. In which not having a product backlog effect their working process and make it complex. They believe that their product is complex and they are having difficulty sometime in defining stories during the

planning.
Almost all developers in this team consider their meeting to be boring because of the quiet moments they counter while waiting for the PO to finish writing instead of having the stories already prepared.
Also, developers from the same team explained that not having the stories already defined can lead to miss some important feature that is dependent with a current feature. Developer D4, D6, and D7 mention that some stories get blocked because it requires the implementation of another feature that they have missed to implement in a previous sprint.
Hence, deviation **DSP1** is suspected as negative because of the below consequences: not having a product backlog which can lead in having boring meetings, difficulties in defining stories at the sprint planning meeting, and stories that get blocked because it requires another dependent story to be implemented first.

DSP2: Both developers from team A are neutral about not determining the number of available developers for the upcoming sprint. However, developers from team B have another thought about it, they all almost report with the exception of the scrum master that might lead to having work overload within the team. Developer D4 from team B dislikes that, and he thinks that it is unfair to do extra work due to not calculating the number of available resources. They also claim that one of the reasons for not finishing their stories might be this. Therefore, deviating in calculating the number of available developers for the upcoming sprint might lead to unfinished tasks and work overload within the team. Thus, the deviation **DSP2** is sorted as negative.

DSP3: Developer D1 from team A also had some difficulties at the beginning because he felt isolated and all his opinions are thrown by the PO, he thought about leaving the team at many cases, but developer D2 is neutral about this. Consequently, the deviation **DSP3** is considered negative because it might lead the team members to be uncomfortable within the team.

DSP4: Both teams are neutral about defining the sprint goal at the end of their planning meetings, however, developer AB mentions that he feels confused at some sprint planning meeting because of having some stories included in the sprint without defining firstly the reason of why we want to have it included. He believes that at some cases some stories are not relevant to the defined goal at the end of the meeting. Therefore, deviation **DSP4** is sorted as negative, not defining the goal at the start of the meeting can lead to include irrelevant stories.

DSP5: Developer D2 from team A is happy about the way they break down their stories, he claims: *"Doing this would save us so much time"*. They use this method to shorten their planning meeting. Developer AA is neutral about it. Therefore, the **DSP5** deviation is sorted as positive as the developer D2 report.

DSP5: Developer D4 and S1 from team B explain the reason for not breaking down stories due the difficulties of their product. Developers D6 and D7 said that they try to break down stories if it is so complex but the team is not so good in doing that.

They all agree that they have to be better in that to avoid unfinished stories per sprint and avoid having so much blocked stories within sprints. They also seem to have a neutral feeling about this since they are almost convinced that the difficulty of the product is the reason behind not breaking down stories. However, they report that they dislike having unfinished stories from a sprint to another. Therefore, we sort this deviation **DSP5** also as negative.

DSP6: Other meeting that conflicts with the planning meeting of team B are required in some cases for some developers in this team. They can not avoid such meetings because of it is important to provide support for other teams. Developers in this team have neutral feeling about it. Two exceptions developer D6 and D7 who claim that developers who do not attend the planning meeting because of their occupation with something else might miss important information about included stories. This information should be recovered from the other developers who attend the planning. Therefore, the deviation **DSP6** is considered as mixed neutral and negative.

DSP7: Developer D2 dislikes the way how the team is not preparing for their planning meeting and not organizing the meeting by having an agenda to be more specific about the important discussion points of the day. He claims that sometimes their meeting drags on and on and take more time than expected which makes him tired and wants to leave the meeting. Developers from second team B are neutral about not having an agenda for their meeting. Accordingly, the deviation **DSP7** is considered as negative because of the reported feedback of developer D2.

DSP8: Both teams are happy with the way how they formulate their stories when asking about it. They do not complain about it and never had any issue with having just the title and the description as only fields for their stories (DSP8).

Only developer A from team D1 said that sometimes the description is not understandable so he requests more explanation from his teammate. Therefore, this deviation is considered neutral for both teams.

DSP9: Asking S1 from team B about estimating their stories (deviation **DSP9**) explains that their estimation is not related to the time, it is just used to compare the size and the complexity of one story related to another. Additionally, Both developers in team A said that they don't follow the agile guide to estimate their stories, each story has no deadline time to finish. As a conclusion regarding this deviation, both teams are neutral about this deviation, they did not provide any negative or positive feedback.

5.5 Identified Root Causes Of Deviations (Based On Interview Result)

DSP1: Developer D2 from team A reports that their team doesn't have a real PO that prepares the product backlog and play the role of a real PO. None from team B could clarify the reason of not using the product backlog even though they have a PO, almost all report the reasons to the difficulty of the product and that they together have to create the stories.

DSP2: Both developers from team A report that they have so much freedom in the team so one developer can take a day off whenever he/she wants. Also, S1 in team B reported that the team has a vacation list in the room so the developers write their day off. The remaining interviewee mentions the opposite, they report that they do not check the number of available developers for the upcoming sprint. The scrum master reason this deviation to the vacation list they have in the room, but the remaining developers were not sure about the reason.

DSP3: Developers at team B think that the PO has the most experienced developer and knowledge in some technical decisions that the team divide during implementation, therefore, the cause of the deviation DSP3 is because of the diversity of experience within the team.

DSP4: Both teams didn't report any explanation for this deviation.

DSP5: Team A follows their own methods to break down stories to save time and make planning meetings shorter as developer D2 mentioned. Almost all developers in team B explain the reason of not breaking down stories to the difficulties of their product. For example, developer BA and BB report that sometime a task can take up to one month because of the dependencies with other teams, and because of not having available testing tools within their team.
DSP6: Both teams relate the reason for having another activity during practicing the sprint planning event (DSP6) to the support they have to provide for other teams or customers. For example, team A has to help their customer during an accidental issue while team B has to also give support to 16 other teams that are dependent on them.

DSP7: Developer D2 relate the reason for deviation DSP7 to the PO. He says that they don't have a real PO in the team to take the role of a product owner and that the agile coach doesn't have so much freedom/power in the team. He sees the PO as the decision maker in the team. Team B didn't specify any specific reason for this deviation, they are simply used to run their meeting without any agenda or preparation.

DSP8: Both teams are happy with the fields they have, they never needed an Id for example to trace a story. They both answered too short about it.

DSP9: The PO in team A explain this deviation towards not completely following the scrum guide, they want to have their freedom when implementing a feature without having estimation into consideration. Also, He says that they are not calculating velocity measurement nor using a burn-down chart so he does not see the reason for using the estimation methodology. Team B is calculating the complexity of stories and explains the reason for not having the time as a measurement is due to the complexity of the product they implementing.

The below table (table 5.2) represents a summary of the deviations and it is root causes.

D.Id	Deviation	Root Cause
DSP1	Teams are not refining their stories on the product backlog	No real PO in team A and no clarification from team B
DSP2	Teams are not calculating the number of available resources for the upcoming sprint	Too much freedom in team A. Vacation list in team B
DSP3	The PO is affecting team decisions	diversity of experience in team A
DSP4	Teams are defining the sprint goal at the end of the planning meeting	Both teams didn't specify any reason they are just used to it.
DSP5	A team is not breaking down stories A team is not correctly breaking down stories	To save time in team A and because the difficulty of the product in team B.
DSP6	Developers are having other meetings during sprint planning	The support that should be provided by team A to their customers and from team B to 16 other dependent teams.
DSP7	Teams are not using agenda during their planning meeting	The PO power in team A. Team B didn't specify any reason for that.
DSP8	Teams are having only title and description as fields for one story	The name and description is enough for them.
DSP9	Teams are not estimating their stories	PO from team A relates the reason towards they are not following the scrum guide. Complexity of the product in team B.

Table 5.2: Deviations root causes

5.6 Identified Consequences of Deviations (Based On Interview Result, Second Interview Round)

D.Id	Effect	Interview With
DSP1	- Unfocused team - Long planning meeting - Difficulties in defining sprint goal	Scrum Master S2
	- Hard to refine stories - Complex planning meeting - Miss to implement important required features	Developers D4, D6, and D7
DSP2	- Unfinished stories during one sprint	Agile coach A3 of team A and all developers of team B
	- Workload on the team	All developers in team B
DSP3	- Unhappy team members	Agile coach A3 of team A and Developer D1
DSP4	- No effect, seen as a positive deviation	Agile coach A3 of team A, and Scrum Master S2
	- Include irrelevant stories	Developer D2

Table 5.3: Deviations effects from second Both round interviews.

Both tables 5.3 and 5.4 represent the consequences of the deviation after the two round interviews and by developers, agile coach, and the scrum master.

DSP1: The scrum master S2 reported the load of focus and energy required by the team to define new stories for the upcoming sprint. The failure of having stories refined on the product backlog has led to hanging in long discussions that require time and cause long meetings. Furthermore, this situation made it hard for them to decide the goal of the sprint because of the difficulties of defining the stories and the awareness of what stories have to include when they don't have them ready on the product backlog. Developers D4, D6, and D7 from team B find it hard to define stories in the planning meeting because of the complexity of their product which has led to have long and complex planning meetings. They report that at some cases they missed to add some important stories that are related

or dependent on another which lead in turn to miss the implementation of some important required features. As a result, the below table 5.3 shows the effects of not having an already refined product backlog.

DSP2: The agile coach of team A and all developers from team B report that not calculating the number of available developers for the next sprint has led to have unfinished stories during one sprint. In addition, team B developers report that additional workload resulted from this deviation. The present developers need to put extra effort when others are not there because the team has included the wrong number of stories for one sprint. Therefore, avoiding the calculation of the available resources' number has led to unfinished stories by the end of one sprint and added extra work on team developers to finish within a sprint.

DSP3: The agile coach A3 and the developer D1 from team A report the same consequence for this deviation where the team decisions are affected by the PO. The agile coach has noticed that some members became displeased with this behavior. The developer D1 has confirmed the agile coach's claim when he reported his dissatisfaction about the PO affecting the team decision. One more effect is reported by the agile coach which is affecting the internal quality of the product because of pushing the team to make implementations faster. However, the agile coach did not confirm that the team has met such a situation before. In conclusion, affecting the team decisions by the PO has led to unsatisfied team members.

DSP4: Both scrum master and the agile coach did not specify any effect of defining the goal at the end of the sprint. They see it as a positive deviation because the teams are substandard in determining the goal of the next sprint, they try to include the stories first so they decide the goal upon it later. In contrast, developer D2 from team A has reported that defining the goal at the end of the meeting has not been the best practice for the team because some irrelevant stories are included for the sprint. As a result, according to the scrum masters and the agile coach, defining the goal at the end of the sprint has both positive and negative consequences as it was reported.

DSP5: The agile coach A3 has reported two different statements regarding breaking down stories separately, he reported that he noticed that some team members became unaligned with the information related to one task first. However, when asked about this deviation again, A3 reported that it has helped them to save time and shorten the planning meeting's time.
The scrum master S2 has reported that not breaking down stories has led to have some unfinished stories that drag on and on through multiple sprints. Moreover, he reported the failure of tracing the process of closing stories that migrate between sprints. Instead the scrum master S2 triggered the team to wonder about a way finish the story. Team B, has also the same claim as the scrum master S2, did not break down stories led to unfinished stories that blocked another related story. Therefore, the following consequences have been noticed in some teams while deviating in breaking down stories: unfinished stories,

unaligned team members with latest task information, drag on and on stories through multiple sprints, loss of tracing of a way to close a story, and blocked stories.

D.Id	Effect	Interview With
DSP5	- No effect, he sees it as a positive deviation	Agile coach A3 of team A
	- Unaligned team members with latest information for each task	
	- Unfinished stories	Scrum Master S2
	- Drag on and on stories - Loss of tracing about how to finish one story	
	- Unfinished and Blocked stories	Team B
DSP6	- No Comment	Agile coach A3 of team A
	- Missing leading information	Scrum Master S2
	- Difficulties of requesting missing information	Developers D6 and D7
DSP7	- Miss some important key points.	Agile coach A3 of team A
	- Unfinished meetings - Unfocused and tierd developers	
	- No noticed effect	Scrum Master S2
	- Long meetings - Unfocused and tierd	Developer D2
DSP8	- No Comment	Agile coach A3 of team A
DSP8	- Not understandable stories	Developer D1
DSP9	- Sprint failure: Not achieving the goal	Agile coach A3 of team A
	- No Comment	Scrum Master S2

Table 5.4: Deviations effects from second Both round interviews.

DSP6: The scrum master S2 report that having some developers occupied with

other activities during sprint planning has led the occupied developers to miss some important information. This claim is confirmed by developers D6 and D7 who reported also that in some cases it is hard for them to find someone in the team that can explain this missing information because of their occupations or because of any other reasons. Therefore, having occupied members during sprint planning has led to miss some important information for some developers, and difficulties in getting explanation regarding that information from another team member.

DSP7: Unprepared sprint plans and unprepared agendas has led to miss some important details that the agile coach A3 wanted to bring up during the meeting. Also, in some cases, he noticed that the meeting became unfocused which confuses the developers and causes their fatigue (e.g. developers ask for coffee break). In addition, he reported that in some cases they had unfinished planning meetings. Developer D2 from team B has also reported that he gets tired during an unfocused meeting, sometimes he felt urge to leave the meeting. In conclusion, working in a none prepared meeting and the absence of an agenda has led to the following consequences in some teams: unfinished meetings, unfocused meeting, unfocused developers, and missing some important key points.

DSP8: Developer A reported that in some cases the stories are difficult to understand. He claimed that in some cases the included information in the description is not enough or hard to understand. Hence, this deviation has led developers in some teams to have difficulties in understanding stories.

DSP9: Only the agile coach AC claimed that not estimating the stories has led to not achieving the goal of the sprint what he called "Sprint Failure". He did not provide so much information about this claim but he reported how developers are not interested in estimating stories. In conclusion, not estimating stories has led to not achieving the goal of the sprint in some cases.

6
Sprint Demonstration

This chapter focuses on the resulted deviations regarding the sprint demonstration event.

6.1 Observation

The department at company A is composed of twelve teams that unite in a big room where all team-names and developers' pictures are attached to the wall. A whiteboard is placed at the middle of the room where each team writes their demonstration scope.
They have three demonstration rounds of 12 minutes each, therefore, a developer can only attend in three different demonstrations.

During the demonstration, I noticed that only one developer from other teams attended team A's demonstration.

Team B doesn't make demonstrations they only review their sprint backlog and check the finished and unfinished tasks.

6.2 Summary Of Identified Deviations

Table 6.1 represents the deviation that resulted during the observation at both companies compared to the scrum guide.

The scrum guide states that each sprint has to end with a demo to represent the finished work to stakeholders, customers or other dependent teams. Team B didn't demonstrate its implemented stories, thus, team B deviates by not doing a demonstration at the end of their sprint (**DSD1**) which is confirmed by the developers from team B during the interview. They report that they only check their finished and unfinished stories of the sprint.

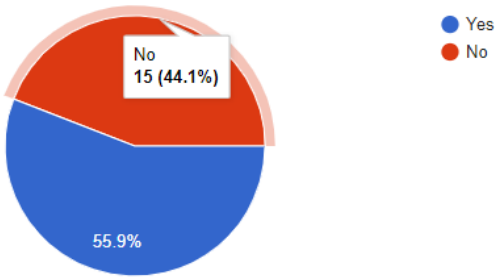
Also, **DSD2** results as a deviation in team A because they present to other teams that have no dependency on it. The scrum guide says that demonstration should target customers, stakeholders or internal dependent teams, however, team A has only one developer attending their demonstration. Developers from other teams are disinterested in team's A achievements concerning a sprint. This led to the lack of alliance of work and interest between team A and other developers. This situation

Id	Deviation:	Agile Guide:	Team
DSD1	**A team are not ending their sprint with a demo**	The agile manifesto insist that each sprint end up with a demo	B
DSD2	**A team is not presenting to the right target**	agile is meant to get feedback from customer or stakeholder at the end of each sprint	A

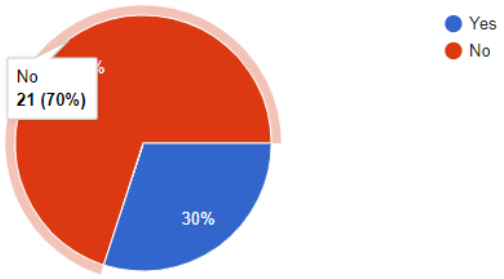
Table 6.1: Deviation in Sprint Demo event upon observation at both teams.

was confirmed during the interview with developers D1 and D2. To sum up, they assert that their team works independently from others in the organization.

6.3 Frequency Of Deviation (Questionnaire)



(a) Responses of: "We always finish our sprint with a demo"



(b) Responses of: "Number of attendee below 3"

Figure 6.1: Response summary of sprint demo section

Figure 6.1 represents the summary of the responses for the sprint demo section. 15 respondents do not end their sprint by performing the demo as shown in figure 4.9-a. The result indicates that other teams than team B are also deviating by not doing a demonstration at the end of their sprint, therefore, the deviation **DSD1** is confirmed to be common.

Also, 15 (51.7%) of the respondents report that the number of their attendance is less than three developers/customers when they demonstrate. Also, 21 (70%) contributors answered that they don't get feedback from customers or stakeholders after each sprint planning as shown in figure 4.9-b. These results show that not only team A are struggling but also teams are struggling too in order to correctly target their demonstration attendance. Therefore, **DSD2** is considered as a deviation at the teams.

D.Id	Deviation	Root Cause
DSD1	Team B is not ending their sprint with a demo	Difficulties of the product
DSD2	A team is not presenting to the right target.	No other dependent teams.

Table 6.2: Deviations root causes

6.4 Perceived Impact Of The Deviations (Interviews)

Developer D1 from team A wondered about the reason behind demonstrating for teams that are not related or interested by their product. He felt like their demo is meaningless sometimes. Both developers from the same team report that they never get feedback regarding the demonstration because of the independency that their team has. They also mention that they only had once or twice a demonstration for customers in almost 2 years. Developer D1 has a more negative feeling about it and suggested that the team would extend their sprint length while developer D2 is more neutral about the sprint length. Based on this, the deviation DSD2 is considered negative for some developers and neutral for others within the team.

The difficulty of dealing with the product is again the reason for not doing the sprint demonstration as developers D4 and S1 stated: *"Some tasks take more than one month. It is not possible to demonstrate anything when tasks are unfinished"*. They all are neutral about it, except developer D6 and D7 who ask for sprint demonstration at the end of each sprint. Therefore, the deviation **DSD1** is also considered as negative for some and neutral for others within team B.

6.5 Identified Root Causes Of Deviations (Based On Interview Result)

Table 6.2 represents the root cause of the deviation at both teams. **DSD2**: Team A doesn't have cooperative teams that attend their demonstration and it doesn't happen often that they demonstrate for their customers or stakeholders. The team members are not aware of the reason but they know that they are breaking a fundamental aspect of the scrum by not getting any feedback from customers or stakeholders. Also, team B is explaining deviation **DSD1** with the difficulties of the product and that sometimes one story can take up to multiple weeks of implementation or some story gets blocked through sprints. Hence they avoid making sprint demonstrations.

6.6 Identified Consequences Of Deviations (Based On Interview Result, Second Interview Round)

Both interviewees did not contribute with enough information about the effect of the identified deviations of this section. Regarding **DSD1**, the scrum master S2 reported that not having a demo for the customers has led in many cases to difficulties in modifying the feature. Not demonstrating has let at many cases to dissatisfaction of the customer once the team presents the product. The team finds difficulties in re-implementing the feature as the customer requires especially if this feature is dependent on another.

Further, regarding the deviation **DSD2**, the agile coach A3 reported that he does not not understand the reason behind using the scrum framework when not presenting to the correct customers. There is no cooperative team in the organization that relied on their findings.

In conclusion, the reported consequences for both deviations are as following: the team gets confused about demonstrating to the wrong customers (deviation DSD2), and the team finds it hard to modify features as costumers require if they do not demonstrate their sprint findings (deviation DSD1).

D.Id	**Effect**	**Interview With**
DSD2	- Confusion within the team	Agile coach A3
DSD1	- Hard to modify features at late stages	Scrum master S2

Table 6.3: Deviations effects from second Both round interviews.

7
Retrospective

7.1 Observation

7.1.1 Team A

The agile coach asked the team if they notice any improvements from the previous sprint, in addition he raised the discussed issues from the previous retrospective. The agile coach tried to confirm with each developer if he/she makes any progress with his/her assigned action point. For example, in the previous retrospective, they raised the issue of the internet connectivity and committed it to one developer to investigate the problem and contact the responsible authority to solve that issue.

Then the agile coach asked the developers if they were satisfied with the sprint results, they were all happy with the new method that they are using in the team "mob-programming" to speed up the implementation process. Only one developer declared his dissatisfaction because he felt that he is not committing code by following the mob-programming pattern.

The agile coach split the whiteboard into three sections: **"Went Well", "Did not work out" and "Need to improve"**. It takes around five minutes for the developer to write their opinions on post-its and then attach it to the specific section with a short explanation. After 20 minutes, the agile coach sorted all posts and tried to give a title for each group. The agile coach requested to vote on the groups to select the most important issue to starts the discussion with. Each developer has 3 available dots that he can vote on as the most important issue to discuss.
Groups with the most voted dots will be selected first from the **"Need to Improve"** section. As a result, two topics were selected from **"Need to Improve"** section and one from **"Did Not Workout"**. The following are the results:

- Mob programming took around 25-minute of discussion
- Some tasks were dropped because the developers were not aware of the methodology they need to use in implementation. The discussion did not take long time because it was the PO's decision to drop these tasks.
- Demo did not go well for this sprint because the audience were not interested in what the team did. The team stated that: *"they felt that they waste 3 days of preparation for the demo"*. This discussion took around 14 minutes.

The three selected groups were documented by the agile coach as new action points for the next sprint.

Also, the following was noticed during the retrospective:

- The agile coach tries to make sure that all team members contribute by giving their opinions through redirecting the question to the none contributing developer.
- The agile coach advise the team on a procedure to solve the issue of mob-programming by contacting another team that practiced it earlier within the organization.
- The team decided to continue using mob-programming despite the dissatisfaction of some developers in the team.

7.1.2 Team B

The 6 available developers start the meeting with rating the sprint from 1 to 5, each developer writes his vote and add it to the whiteboard. Five of them rate the sprint with 3, and one developer votes with 1. As a summary of their discussion regarding the vote result, they agreed that they included too many stories for the sprint which burdened the team with work-overload. The developer who voted with ”1” relates the reason for the number of high meetings and the workload of the sprint. He also complained that not all developers are committing with at least one task during the sprint.
Another developer relates the reason for the workload within the team to the absence of many developers who are on vacation in the other cooperative teams which negatively affects them and increases the work load.

Further, the scrum master raises the previous action points from the previous sprint and confirms with the developers if there were any improvements The following are the raised points:

- Knowledge sharing inside the team, the members are satisfied with the improvements and with increase in communication between them.
- the daily scrum consumes less time now, even though the scrum master suggested more communication between the developers during the work day in order to minimize more the daily scrum time.
- More documentation on the wiki. They mention that they need to improve it more.
- Too many modularity, this problem is still not solved.
- Sync meeting problem still not fixed, not all developers attend this meeting, the scrum master asked for more attendees.

The discussion took around 20 minutes.

Furthermore, the scrum master divides the whiteboard into four sections: **"Drop, Add, Keep and Improve"**. It takes around 12 minutes for the developer to write his/her opinion on "post-its". Once they are done, each developer stands beside the whiteboard and attaches the post to the specific section with a short explanation. Table 4.5 represents the result of the attached "post-its": 1 in the drop, 15 in the add, 3 in keep, and 10 in improve section. Around 55 minutes is needed for all

developers to post their thoughts and explain it. Once they are done, the scrum master requests from the team to create new action points to add them into the improvement list for the next sprint. He also asks the team to write a private personal improvement. For example, they can write "What can I do to act better in the next sprint?" and then they can note it on post-its and save it on their screens so they can always see it.

The scrum master closes the meeting by filling in what they call the "KPI paper", where each team member gives a percentage of his occupation during the ending sprint as shown in below figure 7.1.

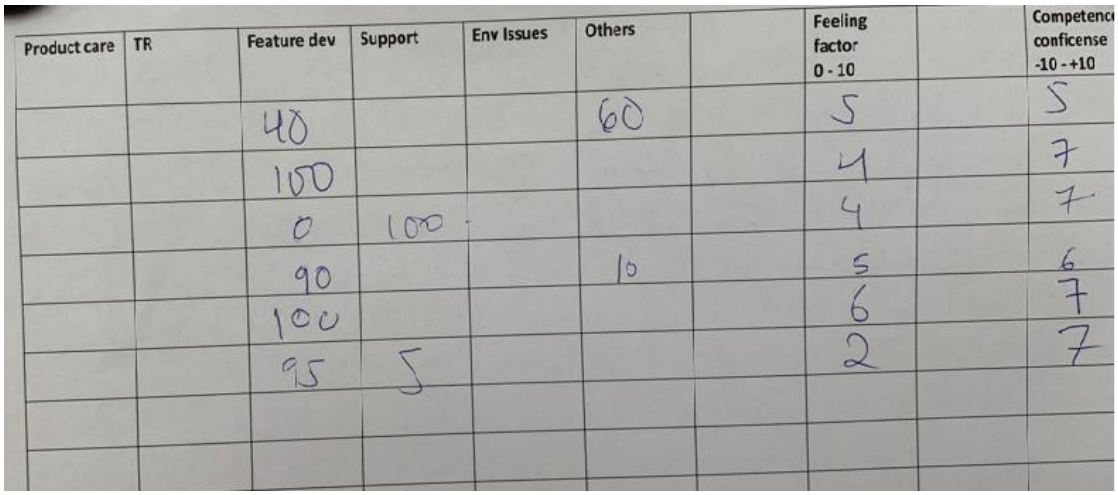


Figure 7.1: Team B KPI paper

The important raised problem that I observed during the retrospective:

- They mentioned that they need to start breaking down their stories.
- item High level of workload and meetings during the sprint.
- Their meetings are unstructured and they require an agenda as they report during this discussion. However this issue is still not fixed.
- Language problem: Three developers are from Sweden and the remaining are from India. The problem is that sometimes they speak their native language during meetings and in working areas. They tried to hinder this behavior through accounting 1 Swedish Crown bill for the one who breaks the rule, but that step is not effective.
- One developer reports that not all developers are helping within the team, he is not satisfied, he feels overwhelmed by the fact that all the work is being done by him.
- They mention the problem of having long daily scrum, all the plans that they tried to solve the problem did not help.
- Late developers for the retrospective meeting.

7.2 Summary Of Identified Deviations

DDR1: The scrum guide insists that all developers should be available and focused during the retrospective in order to evaluate the sprint and to improve the next sprints. In team B some developers arrive at the end of the retrospective due to their busyness with another meeting. Add to it, another developer was not focused on the meeting and was instead occupied with typing on his computer.

DDR2: Team B complains about the high number of meetings that they add to their scrum framework. During the interview with the developers confirm that having many meetings is distressful due to the cooperation with other teams that rely on daily synchronization. These meetings cannot be avoided as developers report.

DDR3: During the observation it is not possible to notice if the team has implemented their action point or not and it is not possible to improve it. However, both teams confirm in the interview that the lack of improvements' achievements after their retrospective is disappointing.

Id	Deviation:	Agile Guide:	Team
DDR1	**Teams developers are occupied with other meetings during the retrospective**	All team member including the PO have to attend the retrospective	B
DDR2	**A team is adding more activity to the scrum framework**		B
DDR3	**Improvement are not discussed**	Retrospective is the heart of agile to improve things when needed	A and B

Table 7.1: Deviation in retrospective in both teams.

7.3 Frequency Of Deviation (Questionnaire)

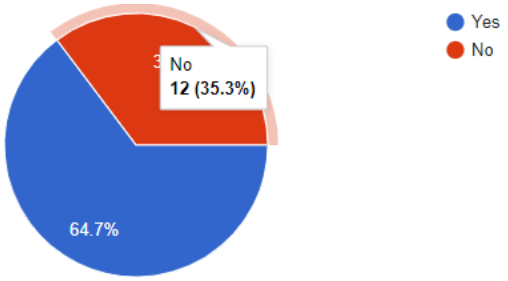
12 respondents report that they don't do the respective at the end of each sprint (figure 7.2-a) while the remaining respondents are doing it.
Further, more questions are tackled in the questionnaire that are concerned with the team's atmosphere. The interviewee responses and claims are as follows:

- 4 respondents feel that they don't say whatever they want during retrospective (figure 7.2-b).
- 9 respondents answer *"Disagree"* on whether they are aware of the issues they meet during the sprint while 10 answer *"Neutral"*, and 5 *"Agree"* (figure 7.2-c).
- 16 respondents confirm that they do discuss the positive and negative experiences they meet during the sprint and they try to improve it (figure 7.2-d and -e).

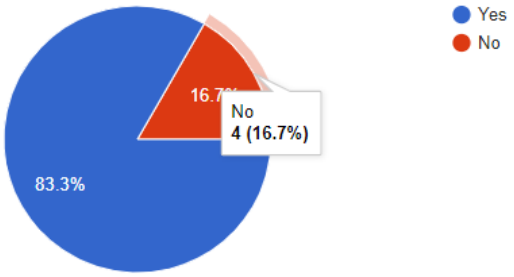
Regarding deviation DDR3, 6 respondents report that they do not create action points, 3 respondents report that they **sometimes** create action points and 15 respondents report that they always create action points as shown in figure 7.2-f. Also, 5 respondents report that they are not seeing any improvement of their retrospective, 9 respondents report that they **sometimes** notice the improvement, and 10 respondents confirm that they are seeing the result of their retrospective. The numbers do not completely confirm the results of the interview and observation, only 5 developers from 22 are not sensing any results of their retrospective while 9 others say that they **sometimes** see it. Hence, the **DDR3** deviation cannot be completely confirmed.

Furthermore, deviations DDR1 and DDR2 cannot be confirmed to be common. The results show that only 4 respondents report that they might be occupied with something during the retrospective and 5 respondents report that several more meetings have time conflict with the sprint. Accordingly, some teams are deviating in DDR1 and DDR2 but with low frequency and thus, both deviations cannot be completely confirmed.

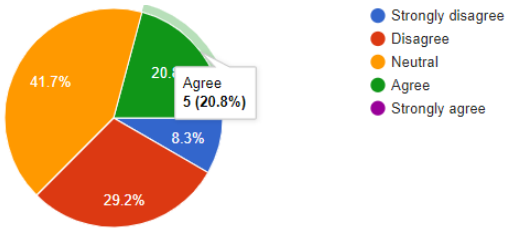
Finally, one more deviation results from the questionnaire, **DDR4**: "Teams are not doing retrospective" this is caused by the situation where 12 respondents report that their teams are not applying the retrospective event by the end of their sprint.



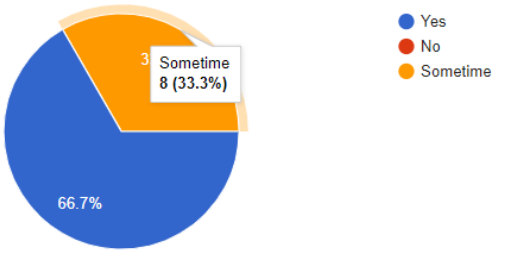
(a) Responses of: "We always do Retrospective"



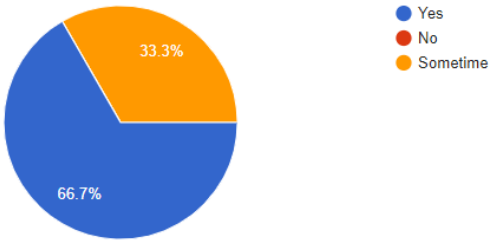
(b) Responses of: "I can say what ever I feel during retrospectives"



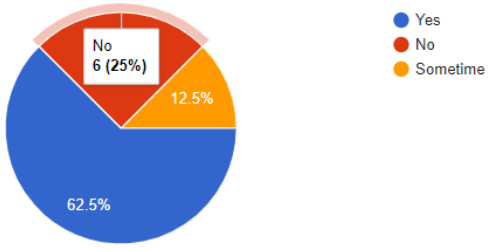
(c) Responses of: "We are always aware of issues that we might meet during one sprint"



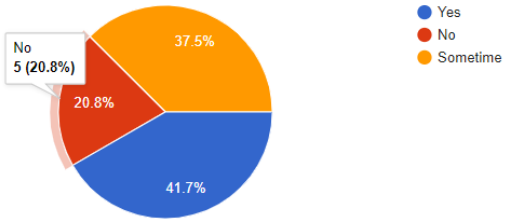
(d) Responses of: "We discuss the negative and positive experiences we experienced during the sprint"



(e) Responses of: "We always try to improve"



(f) Responses of: "We create action point from our improvement discussion"



(g) Responses of: "We make sure to confirm our improvement action point on each retrospective"

Figure 7.2: Response summary of Retrospective section

7.4 Perceived Impact Of The Deviations (Interviews)

DDR1: Only the scrum master S1 in team B is dissatisfied with the developers disrespect to the retrospective in the team. Knowing that the scrum master can't improve the situation without feedback. For instance, developer D4 is neutral about this deviation, but he asks the team members for better time management in order to engage more in the retrospective. Also, Developers D6 and D7 cannot affect this deviation because sometimes they are obliged to support other cooperative teams no matter what they were working with. D6 and D7 have neutral feedback about it. Therefore, this deviation is neutral for some developers at the team and negative for others as for the scrum master.

DDR2: Developers D6 and D7 consider these meetings as beneficial because of the help that they provide for the developers in the synchronization of the latest findings with other teams orally instead of putting their effort on documentation. D4 and S1 consider this as a deviation, but they believe that it cannot be avoided due to the importance of the meetings in synchronizing with another 16 cooperative teams. Therefore, this deviation is considered a neutral deviation.

DDR3: Both teams report that they are not sensing results from their retrospective. For example, developer D1 reports his discontent with absence of the expected results of their retrospective, which causes the same issues to raise through multiple retrospectives which in turn causes repetition. Also, developer D2 from the same team reports that the pre-selected action points are not always implemented, he also has the same claim as developer D1 that is the same issue can be raised through many sprints retrospective.
Add to all that, developers from team B who complains from the same problem, they all report that they do not sense noticeable improvement. For example, developer D4 lost trust in his team because of this lack of major changes within the team. Therefore, the deviation DDR3 is a negative deviation because of the complaints reported by both developers at team B.

7.5 Identified Root Causes Of Deviations (Based On Interview Result)

Both deviations **DDR1** and **DDR2** are caused by the high number of cooperative teams who always need to synchronize with team B, one member from team B needs always to give support for the other team at random times. Also, The high dependency between teams lead to many blocked stories that need virtual discussion which is time consuming and difficult to apply.

DDR3: Developer D4 reports that the agile coach's authority upon the team is relatively weak. In the same contact, same developer reports that the scrum master

needs to step up and gets more involvements. In addition, S1 from the same team claims the cause of involvement to the cultural background which affects the team and makes things difficult to fix, as for example the problem of the language within the team. Some developers in the team are discontent with the "bill solution" and do not apply the taken decisions, hence problems are left unsolved.
Below table 7.2 represents the root causes reported at both companies.

D.Id	Deviation	Root Cause
DDR1	Developers are occupied with other meetings during the retrospective	Developers need to give support to other dependent teams.
DDR2	A team is adding more activity to the scrum framework	Synchronizing with 16 other dependent teams and so many blocked stories.
DDR3	A team are not seeing any improvement of the retrospective	Scrum master need more involvement, and cultural background

Table 7.2: Deviations root causes

7.6 Identified Consequences Of Deviations (Based On Interview Result, Second Interview Round)

D.Id	**Effect**	**Interview With**
DDR3	- Uncomfortable members	Developer D1, Agile Coach A3, and Scrum master S2
	- Replicated issues	Both teams A and B
	- Loss of the trust in the team	Developers D1, D4, D6, and D7
	- Leave the team	Agile coach A3, D1, and Scrum master S2

Table 7.3: Deviations effects from second Both round interviews.

DDR3: Not implementing the discussed "improve/action points" during the retrospective has led to the following consequences listed in table 7.3. The agile coach A3 and scrum master S2 have noticed and reported that developers became resentful when the team did not implement the "action points". Developer D1 confirmed this

claim when he reported his irritation when things are not improving in team A.
Also, all developers in team B and developer D1 from team A reported how the same issue can be discussed multiple times in different retrospectives. This made developers wonder about the benefits of having the retrospective meetings. It has also led to the loss of trust within the team, for example, the developer D4 reported that he did not often see an effect of what they have decided during the retrospective.
Developer D1 from team A reported that in some cases he was thinking about leaving the team because he was dissatisfied with the team's decisions and not seeing any improvements. The scrum master S2 and the agile coach have confirmed this claim when they reported that some developers left the teams because of their dissatisfaction and irritability.
As a conclusion, not acting according to the taken decisions during the retrospective has led to a lack of improvement and thus, to the listed consequences in table 7.3.

8

Discussion And Conclusion

8.1 Classification Of Deviations Based On Root Causes

Developers from both teams couldn't specify or give a direct reason for the three unclassified deviations presented in table 8.1, they only reported the reason for unfollowing the scrum guide which makes it difficult to classify these deviations.

Four deviations DDS2, DSP4, DSP5, and DSP9 are affected by another deviations which classifies it as inherited. For example, teams has reported the reason of not having any tool for tracing sprint status for not breaking down stories, deviation DSP5.

Four deviations are classified as planned since the teams decided mandatory to follow this process model to benefit from it. For example, developers in team A has their own method to break down stories to avoid long planning meeting and makes it shorter. They aim to positively affect the software development process, however, same deviation's consequence has been reported negatively from another team, team B suffer from work overload, unfinished stories, blocked stories, etc. when not breaking down stories. Also, another planned deviation as DSP9 has negatively affected the software development process, it led to not achieving the sprint goal. Therefore, its noted that teams are not always benefiting from the planned deviations.

Six deviations are classified as inevitable since developers/teams are aware of the problem but they are having some difficulties in evaluating those deviations and finding solutions for its consequences. Team B tried to solve deviation DDS3 but unfortunately they still suffer from it despite all improvement they tried.

This classification identifies that sometimes teams are subjected to these deviations without trying to develop the process they are following in order to avoid the negative effects which negatively affects the software development process. In addition, the unclassified deviations is considered the most dangerous since the team doesn't know the main reason behind those deviations. This survey consider the consequences as not enough in order to know the dangerousness of these unclassified deviations.

ID	Deviation	Root Cause	Category
DDS1	A team does not answers all daily scrum key questions.	Not following the scrum guide by book.	Unclassified
DDS2	A team does not have any tool or use any method to trace the sprint status.	Not breaking down stories	Inherited from DSP5
DDS3	Not all team members are contributing in the daily scrum event.	Difference in culture.	Inevitable
DDS4	Daily scrum takes more time than 15 minutes.	Number of developers in the team and so many raised issues.	Inevitable
DDS5	No fixed time for the daily scrum.	Waiting for developers to arrive	Inevitable
DSP1	Teams are not refining their stories on the product backlog.	No real PO within the team	Inevitable
DSP2	Teams are not calculating the number of available resources for the upcoming sprint.	Too much freedom in the team	Unclassified
DSP4	Teams are defining the sprint goal at the end of the planning meeting.	No product backlog and the team is young	Planned and Inherited from DSP1
DSP5	A team is not breaking down stories A team is not correctly breaking down stories	Difficulties of the product and no product backlog. To shorten planning meeting.	Inherited from DSP1 Planned
DSP7	Teams are not using agenda during their planning meeting	No real PO.	Inevitable
DSP8	Teams are having only title and description as fields for one story.	Teams trust the included fields	Unclassified
DSP9	Teams are not estimating their stories.	Not benefiting form estimation	Planned and Inherited from DDS2
DSD1	A team are not ending their sprint with a demo.		Planned and Inevitable
DSD2	A team is not presenting the right target.	No dependent team	Inevitable

Table 8.1: Deviations classification depends on root causes

8.2 Results

This section represent the answers of the research questions.

8.2.1 RQ1: How are companies practicing the scrum framework?

DeviationId	Deviation	Frequency
DDS1	A team does not answers all daily scrum key questions.	41.2%
DDS2	A team does not have any tool or use any method to trace the sprint status.	70.2%
DDS3	Not all team members are contributing in the daily scrum event.	41.2%
DDS4	Daily scrum takes more time than 15 minutes.	53%
DDS5	No fixed time for the daily scrum.	35.3%
DSP1	Teams are not having product backlog.	55.9%
DSP2	Teams are not calculating the number of available resources for the upcoming sprint.	47.1%
DSP4	Teams are defining the sprint goal at the end of the planning meeting.	44.1%
DSP5	A team is not breaking down stories A team is not correctly breaking down stories	55.9%
DSP7	Teams are not using agenda during their planning meeting	61.8%
DSP8	Teams are having only title and description as fields for one story.	35.3%
DSP9	Teams are not estimating their stories.	61.8%
DSD1	A team are not ending their sprint with a demo.	44.1%
DSD2	A team is not presenting the right target.	70%
DDR4	Teams are not doing retrospective.	35.3%

Table 8.2: Frequent deviations at teams

Companies are not applying the scrum framework by the guide. In fact, they are deviating from the guidelines. 34 respondents have confirmed the commonness of the five identified deviations in both teams for the daily scrum event. In addition, seven out of the nine deviations that were identified have confirmed the commonness by the same respondents for the sprint planning meeting. Furthermore, by the same respondents the two identified deviations for the sprint demonstration scrum event are also frequently confirmed. Finally, not all identified deviations seem to be common since only 5 of the respondents confirm both deviations DDR3 and DDR2 and 4 of them for the DDR1.

Additionally, the survey uncovered one more frequent deviation by the respondents of the survey, they confirmed the commonness of the deviation DDR4 by reporting that their teams are not doing the retrospective process.
Table 8.1 represents the confirmed deviations with its frequency, but with the exception of the unconfirmed deviations for the sprint planning event.

8.2.2 RQ2: In cases where teams deviate from a specific scrum event compared to literature, what are the factors that cause deviation?

Sections Identified Root Causes Of The Deviations in chapters 4, 5, 6, and 7 which represent the root causes of the resulted deviations.
For example, section 4.5 explains the root causes related to the daily scrum event deviations. When practicing the daily scrum, while waiting for other team members causes the deviation DSP5 where the team does not have a fixed time to practice this event. Further, developers believe that answering the daily scrum key questions is not the only pattern that can be used to synchronize the latest findings within the team. For example, they believe that they can do this by an open discussion instead, this cause deviations DDS3 and DDS1 as reported. In addition, not breaking down stories causes deviation DDS2, by which the team don't use any tool to trace the sprint status. Moreover, the number of raised issues and the high number of developers within the team cause deviation DDS4.
Thus, the reported root causes classify the deviations into: 2 inevitable, 1 inherited, and 2 unclassified for the daily scrum event. There are 4 inherited, 2 unclassified,3 planned, and one of the deviations is classified as planned and inherited for sprint planning event. One deviation is planned and inevitable, and one deviation inevitable for demonstration event. Finally, one unclassified deviation for retrospective event. It's obvious that there are some deviations that have multiple classifications and that's due to the difference in interviewers' opinions and the difference in the reported root causes. For example, the deviation DSP4 has two classification planned, and inherited, it is planned because the team decided not to do the demonstration at the end of the sprint. Beside, this deviation is also considered as inherited because its affected by DSP1.
In conclusion, the different root causes and opinions in interviews makes it difficult for us in deciding which deviations are totally planned, which ones are inherited, or which are totally inevitable. Beside the interviewers could not clarify or specify the reason of some deviations what makes it harder to precisely classify these deviations.

8.2.3 RQ3: What are the positive and negative consequences of the deviations reported by the developers?

One positive experience has been reported by interviewees from both interview rounds, while the remaining deviations were reported as negative or neutral.
The noticeable thing about these reported experiences is that interviewers have reported distinct experiences for one deviation. For example, one developer may report a deviation as neutral while the other may reported it as negative. For

example, the team have planned the deviation DSP9 and decided not to estimate stories, some developers reported it as a neutral deviation when they mention that they are not following the scrum guide by book; however, others have complained about the resulted drag on and on stories that might block required stories caused by the same deviation. In addition, one deviation was reported as positive for one team while as negative for another for the resulted deviation DSP5.
The second round interviews identifies the deviations' consequences and reveals how some of the negatively reported experiences have been negatively affecting the studied teams.
Therefore, this indicates that the team is loosing control of some deviations and cannot size it's consequences regardless whether the deviation is planned or not.

8.3 Threats of Validity

Three threats of validity are presented in this section: internal validity, external validity, and construct validity.

8.3.1 Internal Validity

In the observation step, the result is totally based on what I observed, I avoided asking developers during practicing their scrum event to do not affect their activity and their decisions and collect as much possible relevant measurement. However, the product owner in team A mentioned jokingly that they were shortening their daily scrum event so that I would believe that they are practicing the framework correctly.For this reason, more clarification for the PO and the team regarding this study was provided so they apply their practice as normal. However, I am still afraid if my presence with the team at the observation phase has effected their practice, this can lead to unrealistic measurements.

During the interviews, developers are selected randomly with different years of experience working with agile. One bias that I encountered is the fact that team members from team B tried to base their answers on "What is the correct practice to do". Hence, I tried to always redirect the questions so I get more relevant data on how they are practicing the scrum framework.

Also, I tried to keep the question as simple and short as possible to avoid confusion and make the question easy to read for the respondents. However, not providing enough information have it is disadvantage, it might lead to misunderstand the question, and thus, it might effect the result of the responses.

Finally, one more internal thread that I might encounter is that two developers from the same company have answered the questionnaire which effects the results. I can not say that 20 teams are deviating for example in not answering the daily scrum key question, the correct is 20 respondents.

8.3.2 External Validity

I conducted this work at two different big companies in Gothenburg plus the online survey from developers at different teams in different companies. Therefore, it is not clear whether this result can be generalized on all companies in Sweden because more data is required to investigate how companies at Sweden are applying the scrum framework today.
Not enough data is provided from interviewee from second round interviews, hence I cannot claim that all teams are having common consequences of the discovered deviations.

Also, I am afraid if teams behaved differently under my presence when they practice an agile scrum event. Therefore, I tried to stay away from the team in the corner and avoided ask them a question during the observation.

8.3.3 Construct Validity

In order to avoid a thread of construct validity in misinterpreting the questions of the interviews, I send the purpose of the study and the goal of the interviews of each developer but I was unsure if they get the idea behind what I am trying to study. Hence, to mitigate this risk, I tried to explain for each interviewer the goal of the interview and that I am not making an agile exam to him/her.

Both teams didn't agree to record their voice I mitigated the risk of losing important data to write a note of each answer directly in the template create for that, write it as a text and send it to the interviewed person for confirmation, they all agreed on what I wrote. All this happens directly after the interview of each developer to avoid missing data and not forgetting all important information that developers state.

Another threat that I encountered is the fact that teams were afraid from the observation process, I tried to ensure the team that the study is not going to mention any personal information about the developers or the company.

8.4 Discussion

ID	Deviation	Frequency	Consequences
DDS2	A team does not have any tool or use any method to trace the sprint status.	70.2%	Not achieving the sprint goal, and workload on the team.
DSD2	A team is not presenting the right target.	70%	Not reported a specific consequence
DSP7	Teams are not using agenda during their planning meeting	61.8%	Missing some key points, unfinished meeting, and unfocused and tired developers.
DSP9	Teams are not estimating their stories.	61.8%	Not achieving sprint goal.
DSP1	Teams are not having product backlog.	55.9%	Long meeting, unfocused team, difficulties in defining stories, missing important stories, and difficulties in defining sprint goal.
DSP5	A team is not breaking down stories A team is not correctly breaking down stories	55.9%	Positive for one team, However, negative for another: unifinished stories, and blocked stories.
DDS4	Daily scrum takes more time than 15 minutes.	53%	Fatigue, cut of the DS, and Not give the turn for all developers to speak.

Table 8.3: Frequent deviations and it is consequences

Obviously, it is concluded from table 8.2 that some deviations are frequently happening among the teams. The teams are facing the same problem during which more than 50% deviate in each deviation. However, we cannot affirm from this table if the teams have the same consequences as teams A and B. In addition, from all the selected deviations only one is considered important and positive for one team, in which the team has planned. However, the same deviation that is classified as inherited have negative consequences on the other team. Thus deviations have varying consequences from one team to another no matter if it is negative or positive. All the teams who reported frequent deviations are threatened by several consequences which were common among them. This is explained by the fact that the negative consequences affect the team seriously as reported. To expand, the developers' work was affected negatively, they were put under workload, affected the team's productivity, and failed to achieve sprint goal.

Excluding the remaining deviations do not mean that they are not important, each deviation has its own importance and can be differentiated from the others according

to the reported experiences and consequences. However, since there frequency is less than 50%, we did not include them.
In conclusion, the teams must take care from the severity of these deviations, since the reported consequences are very dangerous which caused many problems such as the developers leaving the team. However, if the team already knows the deviation, and it's consequences after reporting, then the question is: What sets back and prevents the teams from planning the deviation and change it from a negative one to a positive deviation? And why they can't make a plan for this deviation to fit the team's work? These questions open the door for more studies that we should investigate.

8.5 Future Work

Some of the discovered deviations during the interviews and observation are not confirmed by the online survey therefore, these deviations are not common in all teams. However, it is important to study these deviations to understand the causes and consequences of it. There is a reason to believe that developers that are occupied with another meeting while practicing an agile event will increase the probability of having a team member that is not aligned with the latest updates.
Also, from the online survey, it is reported that teams are not doing their retrospective (DDR4), where some guides see retrospective as the most important event in the scrum framework. It is there where all improvement should happen. Not practicing the retrospective event is harmful for the team since a teamwork is based on communication, the handling between members, identifying the problem within the team, identify the issues in the work-flow, and others. These reasons might lead to unhappy and unfocused developers within the team. Therefore, more studies and research is required to investigate this deviation and understand its causes and consequences.

Bibliography

[1] Peggy Gregory, Leonor Barroca, Katie Taylor, Dina Salah and Helen Sharp. Agile Challenges in Practice: A Thematic Analysis. *Conference Paper in Lecture Notes in Business Information Processing*, May 2015.

[2] Eva-Maria Schön, Maria J. Escalona and Jörg Thomaschewski. Agile values and their implementation in practice. *International Journal of Artificial Intelligence and Interactive Multimedia, Vol. 3, Nº5*, 2015

[3] Zahid Masood, Farooq S, "The Benefits and key Challenges of Agile Project Man-agement under recent research opportunities", *International Research Journal of Management Sciences. Vol. 5 (1) 20-28*, 2017

[4] Juyun Cho. ISSUES AND CHALLENGES OF AGILE SOFTWARE DEVELOPMENT WITH SCRUM *pp. 188- 195 VOL IX, No. 2*, 2008

[5] Miller Gloria J. Agile problems, challenges, and failures. *Global Congress North America, New Orleans, LA Newtown Square*, 2013

[6] Imran Ghani, Naghmeh Niknejad, Dayang Abang Jawani, Murad Khan.*A survey of agile transition models, pp.141-164*, 2016

[7] Kai Petersen, Claes Wohlin. A Comparison of Issues and Advantages in Agile and Incremental Development between State of the Art and an Industrial Case. *Preprint submitted to Journal of Systems and Softwar*el, 2009

[8] Barry Boehm, Richard Turner. Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *IEEE SOFTWARE Published by the IEEE Computer Society*, 2015.

[9] M. R. R. Lazwanthi, Abeer Alsadoon, P.W.C. Prasad, S. Sager, Amr Elchouemi. Cultural Impact on Agile Projects: Universal Agile Culture Model (UACM). *7th International Conference on Information and Communication System*, 2016.

[10] Henrik Kniberg, Scrum and XP from The Trenches: How we do Scrum. *2nd Edition ISBN:978-1-392-22427-8*, C4Media 2015

[11] Ken Schwaber, Jeff Sutherland. The definitive Guide to Scrum: The rules of the Game. www.scrumguides.org, 2017

[12] Nonaka and Takeuchi. The New New Product Development Game. https://www.scaledagileframework.com/scrumxp/. 2019. [Online; accessed 09-04-2019]

[13] Mike Kuniavsky. Observing the user experience: a practitioner's guide to user research. Elsevier, 2003.

[14] Paul T. Robinson, Sarah Beeeham. TWINS - This Workflow Is Not Scrum: Agile process adaptation for Open Source Software Projects. *ICSSP 2019 International Conference on Software and Systems Process*, May 25-26, 2019

[15] Cugola, G., Di Nitto, E., Ghezzi, C., Mantione, M. (1995, April). How to deal with deviations during process model enactment. In 1995 17th International Conference on Software Engineering (pp. 265-265). IEEE.

[16] da Silva, M. A. A., Bendraou, R., Robin, J., Blanc, X. (2011, August). Flexible deviation handling during software process enactment. In 2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops (pp. 34-41). IEEE.

[17] da Silva, M. A. A., Blanc, X., Bendraou, R. (2011, November). Deviation management during process execution. In Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering (pp. 528-531). IEEE Computer Society

[18] Bendraou, R., da Silva, M. A. A., Gervais, M. P., Blanc, X. (2012, June). Support for Deviation Detections in the Context of Multi-Viewpoint-Based Development Processes. In CAiSE Forum (pp. 23-31).

[19] Kuhrmann, M., Diebold, P., Münch, J., Tell, P., Garousi, V., Felderer, M., Prause, C. R. (2017, July). Hybrid software and system development in practice: waterfall, scrum, and beyond. In Proceedings of the 2017 International Conference on Software and System Process (pp. 30-39). ACM.

[20] West, D., Gilpin, M., Grant, T., Anderson, A. (2011). Water-scrum-fall is the reality of agile for most organizations today. Forrester Research, 26.

[21] Silva, F. S., Soares, F. S. F., Peres, A. L., de Azevedo, I. M., Vasconcelos, A. P. L., Kamei, F. K., de Lemos Meira, S. R. (2015). Using CMMI together with agile software development: A systematic review. Information and Software Technology, 58, 20-43.

[22] Jil Klünder, Regina Hebig, Paolo Tell, Marco Kuhrmann, Joyce Nakatumba-Nabende, Rogardt Heldal, Stephan Krusche, Masud Fazal-Baqaie, Michael Felderer, Marcela Fabiana Genero Bocco, Steffen Küpper, Sherlock A. Licorish, Gustavo López, Fergal Mc Caffery, Özden Özcan Top, Christian R. Prause, Rafael Prikladnicki, Eray Tüzün, Dietmar Pfahl, Kurt Schneider, Stephen MacDonell. Catching up with Method and Process Practice: An Industry-Informed Baseline for Researchers. *IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSESEIP)*, 2019

[23] Georgios Theocharis, Marco Kuhrmann, Jurgen Munch, and Philipp Diebold. 2015. Is Water-Scrum-Fall Reality? On the Use of Agile and Traditional Development Practices. In International Conference on Product Focused Software Development and Process Improvement (Lecture Notes in Computer Science), Vol. 9459. Springer, Cham, 149–166

[24] F. Lanubile and G. Visaggio, “Evaluating defect detection techniques for software requirements inspections,” 2000.

[25] B. I. Simidchieva, L. J. Osterweil, and A. E. Wise, “Structural considerations in defining executable process models,” in ICSP, ser. Lecture Notes in Computer Science, Q. Wang, V. Garousi, R. J. Madachy, and D. Pfahl, Eds., vol. 5543. Springer, 2009, pp. 366–376.

[26] Todd Sedano, Paul Ralph, Cecile Peraire, "The product backlog", Conference paper, May 2019

A

Appendix 1

A.1 Interviews

A.1.1 Company A

A.1.1.1 Developer D1

This developer has 2 years' experience working agile, he likes the fact of how working agile improves the team and enhance team communication between members. Practicing daily's helps them to stay updated with their commitments and findings.

This developer does not care so much about not having the daily scrum at a fixed time, it has been working fine for them to meet once all developers are in place. However, he reported his annoyance of waiting for the others to arrive to practice this event. He confirmed that each developer updates the sprint backlog once he/she is done with his/her task. In some cases, they update the sprint backlog during the daily's if the developer has forgotten to do it.

Doing planning as a team is good, it helps the team to plan for the next upcoming 2 weeks. However, he reported that sometime their planning can be a waste of time because it takes so long time and never come to an agreement on what to include for the next sprint. During observation their meeting toke around one and a half hour so I asked about the observed time and what he claimed. He said that their meeting variate, it happens that their meeting can be short or too long depends on what they decide to implement for the upcoming sprint.
He reported a scenario where developers have selected a feature to implement and discussed it for a long time, however, the discussion ends with dropping this feature which is time-consuming.

AA confirmed that they do not have an agenda for their planning meeting, he does not care that meetings are not structured.

They always try to keep the number of included stories in the sprint between 10 into 15 stories maximum. They follow this methodology after experimenting with a different number of stories through multiple sprints. They found that 15 stories are their maximum limit within one sprint. Also, he confirmed that they are not estimating stories because they are not strict with determining when a task should be done.

During the observation, some unfinished tasks were migrated into the newsprint backlog, he reasoned that to the low priority these tasks have.

AA is not happy with how his voice is not heard in the team, the PO has the most power and experience in the team what makes him to the decision maker. AA felt that he is not engaged in all decisions because the PO drops all his and other suggestions.

I asked if they are used to define a sprint goal because of not observing this activity during the meeting. He confirmed that they always have their sprint goal attached to the whiteboard in the room which is always defined at the end of their meeting. He reported that they forgot to define the sprint goal during my observation, however, the team had a small meeting after their daily scrum the next day and defined the goal of the sprint.

Regarding retrospective, developer AA was happy with the way they do their retrospective and discover issues as a team. However, he is not satisfied with the result since the same issue can be raised multiple times through multiple retrospectives which cause repetitions. He dislikes the fact that some raised issues have nothing with the work-flow to do as raising the issue of having more juice in the fridge, for example, he felt like the time of this discussion should be used for something more valuable.
Sometimes, developer AA is not confident to tell his thought to the team because he does not trust his teammate, he felt that his voice is not going to make any changes. When I asked him why he has such a feeling?, he only said *"I do not want them to understand me wrong"*.

Regarding the sprint demo, this developer thinks that they must have longer sprint length than 2 weeks because of their demos are meaningless. Not many developers from internal teams attend their demo because their team is independent of the others. He suggested it but no further actions were taken from the team. The team has demonstrated only 2 times per year and never got any feedback.

He reported that all team members are committing in the team, however, there is always one developer in the team who is doing more than the others. They decided to start use mob programming to distribute the workload on all members within the team to get everyone involved and equally distribute the work within the team.

He reported that practicing agile needs more sociability and can be a bit complicated when team members are from a different culture. Also, he thinks that agile is a little bit complicated sometime since the developer needs to attend in all agile events and put extra effort on these events instead of completely focus on coding.

A.1.1.2 Developer D2

Developer AB has approximately six years of experience of working agile, he reported that following the scrum framework distributes the workload on all team members. It makes it easier for them as a team to adapt their development depends on customer and stakeholder feedback.

Regarding daily scrum, AB stated that they try to keep the range of their daily's within 15 minutes but not always this works for them. Sometimes their daily's has been longer than 15 minutes. Their raised problems during the daily scrum were discussed it through another short meeting. He liked how the team communicates, the reason is the good working area the team has set up lately. He reported that they are not strict with answering the key question of the daily scrum. He claimed that raising faced issues during the daily's and reporting the latest commitment is more important than focusing on answering the scrum key questions. *The team has to be free and do it through what the developer believes is correct to do instead of following the rules of the scrum framework.*
More, he said that they made sure that all developers are contributing during the daily.
He confirmed that lately, they were not running their daily at a fixed time. They ran it once all team members were in place. He did not have any special feelings about it but he more likes organized meetings.
He felt that daily scrum is a good opportunity for each developer to talk and give his input about the latest update, he likes the way they do it.

AB was not happy about their planning meeting that takes so much time (a half day sometimes), he reported that he is a structured person and liked when everything is planned and structured. He confirmed that their meetings are not planned, unstructured and has no agenda. Therefore, he dislikes being in those meetings. Also, he stated that their team does not have a real PO that prepares for the planning and prepares the stories on the product backlog, they are used to create all stories together and add it into the sprint backlog. AB clarified for me that the backlog they use on the wall works as both product and sprint backlog, they divide the backlog into two sections, all stories at the top part are included in the sprint and everything bellow remains for the next sprint. They used to call the bellow part "product backlog".
He also confirmed that they used to determine stories together and attach them to the wall. Each developer has the responsability of breaking down the story when he moves it into Azure-DevOps. He Believes by doing this, the team saves time and makes their planning shorter, he hates long planning meetings what made him tired and unfocused in many cases.
He also said that they are not estimating their stories what made him believe that not estimating stories makes it difficult to trace the sprint status.
AB told that all members in the team have the power, however, since the PO has more experience than the other, it happened in many cases that the PO became the decision maker. AB continued and said, this can be good and bad, it is good because the more experienced the PO is the more solution he can find in a short time. The

bad thing about it is that the PO does not listen to our opinions, sometimes the PO just adds his thoughts even if ours are correct.

They are not strict with defining the sprint goal it happened often that they missed to add a sprint goal and he does not know why they did it frequently. He reported that at many different situations the team got confused during the planning meeting and came to the point where they wonder about what the team should do for the upcoming sprint. He said that it happened often that their sprint fails even if they have defined a goal for the sprint. He confirmed that they were used to add the goal at the end of the meeting, however, he dislikes that because he believes that the goal must be defined first to allow the team to select the stories upon the goal.

Regarding the demo, they do not have any stakeholder, they only made two demonstrations for their customers within one and a half years. He felt that their demonstrations are meaningless and believed that their demo should be more to get feedback and not just to show what they have implemented.
They were used to start planning for their demo three days before the demo day to confirm that everything was integrated together to avoid a late problem during the demo day.

They always tried to variate while doing a retrospective, they did not use the same approach to avoid duplication and try to improve their retrospective. Not always their taken action points are implemented, it happens that sometimes same issue can be raised multiple time during the retro. The reason for that as he reveals is that the team is not taking any action to solve these issues which are bad. It made him feel that they must solve this and be better in that. He like the atmosphere in the team and he trusts his teammate, he feels that he is free to say whatever he feels, and nothing hinders him. Their retro is time boxed and not more than 1 hour, but it happens that they call for another meeting if they did not agree about one issue during the retrospective to discuss it further.

A.1.1.3 Agile Coach A3

Asking the Agile Coach who has 10 years experience working with agile at different countries about the effects of not having the daily scrum at the same time (DDS5) clarifies that team members are not considering the daily scrum as the main event that should be practiced every day. Also, the agile coach reported that this behavior has led to skipping daily scrum at some scenarios.
Regarding DDS4, the answer was not so clear but he noticed sometimes that the consequences of having a long daily scrum are having fatigue, bored, and unfocused members. He mentioned that sometimes he can see how developers start to look into their mobile or watch to check the time. He also speaks about how to solve this which is outside of the scope.
When asking about deviation DDS1, AC reported that it made him think that the developer has nothing to do or he/she not committing in the team. This led to losing the trust in the developer, also, it made him wonder about how the reported time by this developer was consumed. AC reported that he noticed at some situations

that a not contributing member became not aligned with other members. It made the developer asks about some information that has been already explained during a meeting or the daily scrum.

AC answered shortly regarding the deviation DDS2, he said: *not tracing the sprint status simply have led to sprint failure many times* because they are not identifying the hinders at early stages. When asking about the definition of sprint failure, he said: *"It is like the team has wasted 2 weeks for nothing."*

Furthermore, Not organizing the planning meeting has led to missing important key points during the sprint plannings, for example, it happened that a discussion regarding one story drog on and on without being aware of the time. Also, the unstructured meeting made him notice how developers got tired and not focused (developers start to look into their mobile, start a walk in the room, ask for a coffee break...). Not setting an agenda (DSP7) has led to an unfinished planning meeting. They are used to plan their sprint directly after the demonstration on a Friday after lunch time. He continued: at Friday the team has consumed his maximum energy and some developer may be unfocused and tired, it has led in many cases to speed up discussions to finish things or maybe cut the meeting.

DSP2: the agile coach reported that the team has been in situations where they had unfinished stories at many sprints because the workload set on team members. It happened many times that the team included stories without being aware of the available number of developers for the next upcoming sprint and its capacity.

Asking about the deviation DSP3 where the PO affects the team decisions during sprint planning made the agile coach point on some consequences to that which he had met earlier. He said: *it makes the team not happy with what he is doing*, he continues: *"Never let a developer do something he dislikes."*. The coach mentioned the following but he didn't confirm if the team have met this situation before: when the PO stress the team to speed up their implementation process through deciding the stories and trying to reduce the time of implementing one story, the PO is touching the internal quality of the product without being aware of that.

Defining the goal at the end of their planning meeting DSP4 have no effect on the team, AC sees it as a positive deviation because the entire organization has recently started using agile and they still not aware of what is the goal for the next sprint, hence they try to figure out what stories to can include so they can decide a goal upon it.

Also, AC sees DSP5 as a positive deviation since it shortens the time of the planning meeting and as he mentioned earlier that the planning might happen at a day where everybody was tired and just wants to get things done. The team decides the main stories and they break it down in another day. However, AC mention earlier that breaking down the stories separately might cause that not all team members are aligned with the information related to each story and that what he noticed within the team.

The agile coach didn't add any comment on DPS6 and DSP8 deviations, he only mentioned that the name and description are good enough to understand what each story is about (DSP8).

Regarding DSP9, AC stated that the teams won't put more effort into estimating stories because they don't trust estimation, no team in the organization is doing story

estimation. The agile coach continued and said, it's not strange that sometimes the sprint fails when nobody wants to try story estimation.
The agile coach agreed that the deviation DDR4 is relevant and confirmed that sometime their retrospective's action points are not implemented which causes reputation for some issues that might occur at multiple sprints. AC stated that not improving and listening to developers opinions has a huge effect on the team, some developers have left the team because of that and other reasons.
Not doing the demo for a customer or stakeholder has made the team wonder sometimes behind the reason of following agile scrum framework, they don't see any benefits from it: no feedback from customer neither from internal teams at the organization. The team has a problem to understand the importance of their sprint demonstration.

A.1.2 Company B

A.1.2.1 Developer D4

Developer BA has 2 years' experience of working agile in Sweden, he likes the fact of how agile events try to keep the developer updated, however, practicing it may require additional effort.

Regarding daily key questions, the developer stated that it is not important to follow the agile guide rules, they are not strict with answering these questions, he responded that the reason of daily is for team synchronization to make all team members updated with the latest issues or findings.

Also, he confirmed the problem of having long daily's that was observed, however, he does not care so much about it since the meeting is meant to have an open discussion between team members and share their experiences.

The way how they communicate the hinder of achieving the goal of the sprint is to raise the issues during the daily, it made him feel that all team members are brave to talk about their blocked tasks instead of hiding their progress.

Furthermore, tracing the sprint is meaningless for them since some tasks can take more than 2 or 3 weeks because of the difficulty or missing information about the task. It can happen that a story can be migrated into multiple sprints until it is done. When asking about why not splitting the story into multiple tasks, BA reasoned that to the difficulties of the stories: *"sometimes one story needs time to investigate how to implement and test it due to the heaviness of the product we are working on".*
He also confirmed that it happened that a sprint failed when they had too many blocked items, it happened frequently. He said, the word "fail" is not too dangerous because they are aware that a task could take more than expected.

He felt that all developers in the team have the same power and that the work is distributed equally. It may happen that one developer is more loaded than the

others especially when one team member is on vacation, someone else needs to step up and provide support for the other teams and do his own task. It has led in many situations to have unfinished stories since the team got work loaded.

During the observation the team complained about the number of meetings they have per day. Developer BA revealed that the need of synchronizing with 16 other geographically distributed teams across the world is the reason: *"It require around one and half hour daily meeting to synchronize or give support to the other teams"*. He stated *"We cannot avoid these meetings and we have to live with it"*. He felt that these meetings are still better than reading so many emails per day. When asking about if he felt that they are adding more activity to the agile framework, he answers that he feels that they are not technically working agile and support his claim with that they are not delivering anything by the end of each sprint. They need minimum 5 sprints to deliver something for stakeholder.

Regarding sprint planning, they do not prepare an agenda and he have no problem with unstructured meeting. They also do not prepare a list of the available developers for the upcoming sprint, he felt that they need to be aware of that, they better distribute the workload on team members. Regarding the goal, he confirmed that they always define the goal at the end of the sprint, but he never understood the meaning of it because it is not giving them any benefit. Asking about product backlog, he confirmed that they do not have a product backlog, the team members create the stories and add it to their sprint backlog that is hosted on Jira, he is not happy with the fact of waiting for the PO until he is done typing: *"it makes the meeting so quiet and boring"*. He said, not having a product backlog has made it hard for the team to figure out the stories to include within a sprint. Sometimes, we miss implementing important features that are required for another.

Regarding team atmosphere he felt that the Scrum Master need to be more engaged and do his role more, he became irritated over some decisions in the team as for example, the "fin system" that they use when talking about another language than English inside the team. he felt that they need to together try out things to find a solution instead of having military decisions within the team. Also, he mentioned that he does not have trust in the team when asking about team trust. He said, *"Sometimes I do not say things because it is not going to help even if I say it"*.

He also said that developers in the team need to manage their time to avoid being occupied with something else during practicing an agile event. However, this behavior cannot be avoided because of the support they have to provide for other teams at the same time where they practice a scrum event.

A.1.2.2 Developer S1

Team member S1 is three years experienced Scrum Master and practiced agile since 2013. He is aware that the team is not applying the scrum framework perfectly. He reported his unhappiness regarding, however, he cannot do so much about it. For example, he is aware that each team member must speak during daily's scrum

and can solve this by passing a pen to each member of the team to force him to speak. However, sometimes he feels that it is "OK" to let the members have an open discussion if it is information. He takes into consideration that some team members have another cultural background and experience with agile. He is also aware that daily are taking more time than it needs but again he welcomed the discussing where its information is needed, he cannot interrupt it just to follow the agile rules. He also said, that not answering the key questions do not give a report of what the developers are occupied with or doing or even the impediments to achieving the sprint goal.

S1 explained the difficulty of tracing the sprint status and caused it to the following: the number of dependent teams (16 teams), the difficulties of some stories, and the difficulties of estimating these stories. Sometimes, one story can take up to one or two months depends on the available resources or the difficulties of the issues they face during implementation. Also, in some situations, the tools they have in the teams do not provide so much efficiency that helps them to speed up the process of implementation and testing.

Asking about how they communicate their work to the other 16 teams, BB responded to that through the synchronization meetings. Furthermore, he was aware that increasing the number of meetings (around 1 and half hours per day) has led to adding more activity on the team, but these meetings cannot be avoided because they need daily synchronization with the other dependent teams.

During planning, D5 confirmed that they are used to define a sprint goal at the end of the planning meeting and did not see anything wrong with this behavior. He dislikes the fact that the team does not achieve the sprint goal which happened many times. He tried to solve this by pushing the team, but he is careful with that because of the cultural background for the other developers within the team.
All members within the team have the power, they decide the stories of the next sprint without any interruption from the PO. However, one person in the team has the most knowledge and experience in the product, hence, it led to has one developer in the team as the decision maker but no one was offended by that. Also, He stated that they have a document where they list all the available developers for the next sprint, each team member must fill the document when he is planning a vacation.

Asking about the estimation method they use and the purpose of it, he reported that the estimation they use is not related to time, it is used for comparing the size and complexity of other stories. Because of the previously said reason (dependency and difficulty of the product), some stories are hard to estimate they may take up to one month until a story is done. They used the cut feel method to include stories in the sprint backlog. He did not like the fact that the team becomes so quiet while waiting for the PO to write stories and attach them into Jira.

When asking about breaking down stories, BB said that the difficulty of some stories hinders them to break down these stories. He said that not doing this, has led to having blocked stories and unfinished stories that migrate through multiple sprints.

Regarding his role as Scrum Master he was aware that at some places he was not

practicing his role because of the following:

- Jira rights: he does not have rights from the mother company for Jira, therefore, he does not run the planning meetings. He was not happy about it, but it is the company rules.
- cultural hinders: the team is so oriented, so he must avoid some rules just to keep the OK atmosphere in the team.
- Break the rules is not a bad thing when it feels good in the team and the discussions are informative.

A.1.2.3 Developer D6 and D7

During the interview the two developers were shy, it was a bit hard to get information from them. They both agree that the team is not technically practicing the Scrum framework. Developer D6 (have 2 years of agile experience) said: *"it is good to practice it if we follow it properly"* while D7 (have 5 years of agile experience) mentioned that doing agile is good for customer and stakeholder's feedback but they are not doing this in their team.

Regarding daily scrum, D6 reported that a developer does need to speak if he does not have any comment to add. He continued if a developer has an issue while implementing his task it can be raised during the daily's and have a technical discussion about it, it is important to solve this issue. Developer D7 sees daily scrum as an important station to keep all team members aligned. He also, agreed that not everyone needs to contribute if not necessary.

Furthermore, they mentioned that during daily scrum each speaker tries to update the sprint backlog, for example, if a developer is done with his task, he moves the tasks from doing it into done and so on.
BC stated *"I don't think about the length of the daily's I just want to solve the issue I have"* while BD said: *"we are eight developers in the team so it's ideal to have 20 minutes as daily scrum, we try to keep it short"*. They both agreed about that.

They both confirmed that they did not use any tool to trace their sprint status, but they are aware that the Burndown chart should be used. They said that they have to estimate their stories to be able to use it. It happened often that they do not achieve the goal of the sprint and that they migrate some tasks from the previous to the new upcoming sprint. They both did not claim so much about their feeling regarding not tracing their sprints.

All developers in the team have the power and tell the PO the stories to include in the next sprint. They were not sure about the product backlog if they have it one or not. They reported the difficulties they faced multiple times to define some stories during the sprint planning because sometimes, they were not aware of what they needed to do in the upcoming sprint. Also, they have missed implementing some features that are required for other stories, therefore it has let to block some stories. They also mentioned that it happens that some stories are too large, so they break it into smaller tasks on a glass board they have it in the room. They are both aware

that stories should be selected from the product backlog and include it in the sprint backlog instead of defining the stories during the planning meeting. They both did not care about the quiet moments they have during the sprint planning; developer D6 said *it's around one and half hour we stay focused in this meeting so it's good to have some quiet moments to relax.*
They also confirmed that they are used to define their stories first then determine the goal of the upcoming sprint at the end of the planning meeting.

They also mentioned that they as a team try to minimize the number of meetings during the sprint, but it did not work because of their dependency with other teams that are working on the same product. They are fine with that because it made it easier for them to communicate their outcome to other teams instead of documenting it, however, they both don't like meetings.
Asking about not calculating the number of available developers, both developers reported that the team got loaded in many situations and has some stories that were unfinished which led to blocked stories.
In many cases, it happened that one developer was occupied with supporting other teams during their meetings, however, recently they are working on avoiding such conflicts. They also reported that this behavior has led to missing some important raised information during the planning which could be difficult sometimes to find it if the responsible developer is occupied or not present when searching this information.

Regarding retrospective, they both agree that they are not getting any benefits from retrospective despite all the action points they create at the end of each sprint. BE felt that sometimes they are not doing any action to improve these action points while BE stated that same mistakes could happen during multiple sprints. They both felt that there were always duplicated issues raised during their retrospective. They trust their teammate however sometimes they don't say their feeling during the retrospective because it will change nothing and nothing is going to happen.

The last question was about the reason for not doing a demo and they both did not have a direct answer for this question.

A.1.3 Other Companies

A.1.3.1 Interview With S2

The discovered deviations were discussed with another scrum master with 8 years of experience with working agile. He stated that he is being with all kind of teams: some who dislike agile, some who failed to practice agile, some who wants to practice agile and not succeeding and many other kinds of teams.
When discussing the discovered deviations of the daily event, the scrum master stated that it happened frequently that the team skips the daily when they do not practice the event at a decided time. The reason is that some developers are involved with other things when all members arrived, the team avoids to interrupt the occupied developers so simply they skip the meeting. Also, It happens, that some members miss the daily scrum because they are involved with another activity

as another meeting that they already planned before, the member has to prioritize on the daily scrum.
Having long daily scrums has resulted that some team members do not get the chance to contribute and share their outcomes. Sometimes, the team does not realize the consumed time and suddenly the team recognizes that they have to cut the meeting because of another activity the team has to do, the daily scrum is then interrupted without giving the turn to the remaining developers/members to contribute/speak.
In all teams he has worked in, they never used a tool or a method as the burn-down chart to trace the sprint status so he is not so much aware of the effects of this deviation. However, the scrum master reported that it happened many times that just some days before the sprint demonstration they recognized that their sprint is not doing well because of the number of remaining stories. That put the team under pressure to finish the stories before the sprint demonstration. Speeding up the implementation has also led to release some features without been tested, which led to bugs in the product that hanged through multiple sprints to solve. Also, it happened many times that one developer raised a late issue just a few days before the sprint demonstration. This forced all members to focus on solving the issue because of its importance or dependency. The scrum master reported this while discussing the DDS1 deviation.

Regarding the sprint planning deviations, X stated that he does not understand the reason for using agile if a team does not have a product backlog. He has been in such a situation once and confirmed that not having a product backlog has made the team unfocused and made planning drog on and on. Also, he said without having a product backlog it was hard for the team to identify their sprint's goal.
He never used an agenda for their planning meeting but this because they follow the same pattern all the time. They already have their product backlog so the only thing they do is just include their stories of the next sprint. He does not see anything wrong with that and never noticed any effects on the team.
He mentioned the importance of breaking down stories to be able to estimate each. He reported that before starting breaking down stories, they had a lot of unfinished stories that drag on and on for multiple sprints. Also, it made the team member wonder about the definition of done of this story? how to finish such a story, it happened that a member asked: *"we have been working on this story too long now how to finish it?"*. They start breaking down stories lately but not estimating stories. The reason for not estimation is because developers do not believe in estimation they want to have their freedom during implementing. He reported that he never used estimation before and things were working fine for them.
The consequences of having some team members occupied during sprint planning as he noticed that these members miss some important information that he might need later. He noticed that in many cases.
It happened sometimes that the description on one story is not so informative and the team member who has all the information about it is busy or not attending. The other member who missed the information during the planning meeting is now struggling to go further. X reported that this happened to them many times.
They always tried to keep their planning so short and never been with a meeting

for more than 3 hours.
They also, define the sprint goal at the end of their sprint because it is hard for them from the beginning to identify it. Upon the selected stories they decide the sprint goal.

Regarding retrospective, the scrum master X reported that not improving the team has led to having an unstable atmosphere within the team where some developers were uncomfortable. He got so many complains from developers and some have decided to change the team or leave the organization because of that.
Also, sometimes, he felt that developers do not take retrospective seriously. Some developers have reported this for him, the wanted to skip the retrospective because they are not seeing any benefits of it. He said, he been involved in a long discussion regarding that with developers.
He has been in teams where they did not demonstrate. He said not demonstrating is like you are building a house without the roof. He reported that teams were not getting feedback from customers or stakeholders at early stages what made it difficult for them to adopt customers changes. Sometimes, one feature is dependent on another, it is a huge process to adopt customer requirements at late stages. He continued and said, "What is the difference of not using the waterfall model then?"