BACHELOR'S THESIS 2021

A study of development collaboration in a water-gile-fall organization

Astrid Jansson

ISSN 1650-2884
LU-CS-EX: 2023-79

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

KANDIDATARBETE
Datavetenskap

LU-CS-EX: 2023-79

A study of development collaboration in a water-gile-fall organization

En undersökning av programvaruutvecklingskoordinering i en water-gile-fallorganisation

Astrid Jansson

A study of development collaboration in a water-gile-fall organization

Astrid Jansson
nat15aja@student.lth.se

January 17, 2021

Bachelor's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Lars Bendix, lars.bendix@cs.lth.se
Examiner: Emelie Engström, emelie.engstrom@cs.lth.se

Abstract

In this empirical study, we investigate root causes for loss of adaptiveness and efficiency along with solutions to eliminate these in an agile middle-sized project with several teams highly affected by the traditional waterfall structures that infuse other parts of the company.

Based on interviews with employees from varying roles and teams, 60 issues are identified and grouped into nine areas. Root cause analysis is performed on three of these areas. Efficiency and adaptivity are found to be negatively affected by unclear code ownership between teams and faulty implementation of Scrum. This drives the codebase to become more complex and more expensive to develop and maintain.

Solutions recommended include introducing clear ownership between the teams, improve the implementation of Scrum, focus on process improvement, as well as enforcing and supporting team self-organization.

Keywords: Scrum, agile, waterfall, efficiency, development, collaboration

Contents

Chapter 1
Introduction

Multiple teams developing a product is a task that requires much coordination and collaboration. The strategies used, cultures, industry practices, and employees affect the outcome and success of a development project. The specific aim of the development can vary depending on the factors mentioned. However, something that many can agree on is that development done effectively, with an easily maintainable product, is more easily said than done.

The purpose of this Paper is to do an empirical study of an organization that is experiencing a decline in effectivity and adaptivity in development delivery. The organization is a part of a company operating in the financial sector. Process and performance will be explored in a multi-team setup developing a software product. The development teams are using Scrum, an agile development framework, where the highest possible value tasks are developed during time-boxed sprints.

These agile teams operate in a department of over three hundred fifty people in an environment affected by bureaucracy regarding certain procedures, many imposed by external factors. More teams will most likely be added to the development collaboration of the product that today consists of 7 teams.

Today the teams' development processes are considered to be working well and follow industry standards. Still, there are concerns regarding incoming requirements and changes being difficult to include in the already ongoing sprints. These requirements are imposed outside of the department's control and need to be addressed.

Teams do not have single responsibility in the product. Instead, some teams are responsible for certain domains while others are responsible for features spanning several domains. This creates confusion regarding who has precedence. When the current sprint of one team is changed, due to urgent incoming requirements, not only this team is affected. Then, other teams also need to rescope and change what they are working on in the sprints. This affects the collaboration on the codebase and the organization suspects that the code development is affected by the processes around how the product is developed and maintained.

The initiating problem is that the development teams are becoming less productive and adaptive in the change process. The goal is to find the root causes for this and explore pos-

sible solutions to improve the efficiency of the teams, in the context of constantly changing requirements.

Given the background and the initiating problem, the following research questions will be explored

- **RQ1:** What are the root causes of teams experiencing loss in productivity and adaptivity in the development delivery?
- **RQ2:** What solutions could help to eliminate some of these root causes?

Investigation findings would provide insight into how the development is affected by factors as collaboration, culture, strategies, and other practices. In answering RQ1 we investigate issues occurring and their root causes. These insights will, in turn, lay the foundation for RQ2 where solutions will be explored, and recommendations made. The overarching goal is to understand what can be done to improve and support collaboration between teams and employees, and improve the outcome of the development result providing business value.

Semi-structured interviews [7] are the main source in the investigative part of the study where the organizational structure is documented along with what kind of issues that the teams and the support around them are experiencing. In the analysis, root causes are investigated answering RQ1. From the results of the analysis, possible solution designs are explored answering RQ2. Recommendations, statements, and arguments for certain directions are based on recommendations from related research applied to fit the context of the organization observed. Literature studies are performed throughout the preparations of the study, preparations of the interviews, during the interviews, data processing, analysis, and design solution composition.

In the following chapters, we first present the context of the organization under study. Then, the methods used in this work that include preparations, data gathering, data processing, validation, literature search, rescoping. In chapter 3 and chapter 4 we move into the investigative part of the study resulting in an organizational description along with a list of 60 issues grouped into nine areas. From these, we rescope the area of investigation focusing on three of these nine areas and then analyze root causes addressing RQ1. After that, we rescope again and explore solution designs addressing RQ2. Finally, we enter the discussion and reflection of our findings, validity, related work, and future work before we draw our conclusions.

Chapter 2
Background

The goal of this chapter is to give an overview of the context of the organization and the method used in researching, analyzing, and framing this thesis. This will provide the reader with an understanding of how the study is performed and what the overall environment that affects the teams collaborating on the solution looks like.

2.1 Context

The company is working in the financial sector and this part of the organization under study is heavily regulated and have stakeholders that rely on the Software development team to provide business value. As of today, 7 teams are collaborating on the product. The product has been merged from separate solutions over a period of 2 years. The separate solutions are in the same area and had functionalities crossing the solutions. The aim was to make it easier to do cross-functional changes over several solutions and be able to reuse functionality that was found very similar. This as the different solutions were operating in the same area, doing slightly different things.

The team setup is differing depending on the team. However, in general, the teams consist of a product owner, a tech lead, scrum master, business analyst, and multiple developers. The team setup aims at bringing business and IT employees together and the team then consists of people from both sides of the organization. The business side consists of the users and stakeholders of the product. In general team members reports to the manager on either business or IT side depending on function. The average team size is around 11 members with the largest team being 13 members and the smallest an outlier of 6. Teams are supported by a cross-functional architect team as well as other support teams.

Collaborations between the teams are structurally done through the respective roles. Tech leads align the development and product owners together with their stream lead on the business side set and prioritize the task.

The product is supporting the business side and an example of business value provided

in development is the automatization of manual tasks. The IT solution in general allows the business to work faster.

Two more teams are being onboarded to the product and more might be onboarded in the future. The current aim seems to be a total of 10-12 teams.

Clarification

Below we clarify the intended interpretation of terms in this study.

- Organization: refers to a part of the company that is limited to the teams' department, in other words, the teams themselves and their close collaborators, support teams, product users (business), product stakeholders (risk, etc), and department management.
- Business: refers to the users of the solution.
- Mono-repo: refers to the repository that is the home to the entire solution or product that the teams are developing.
- Solution/product: refers to the content in the mono-repo.

2.2 Method

The goal of this section is to present the method used in researching, analyzing, and framing this thesis. This by giving the reader an understanding of the decisions made in the process and showing transparency in how different aspects of the work have been performed.

The structure mostly follows the different stages of the working progress. Firstly, the preparation of scoping the thesis. Secondly, the investigation phase includes stages of preparing and doing interviews, how the data process is done, and how the result is presented. Finally, before moving on to the analysis phase there is the validation of results and rescoping performed due to limitations in the thesis timeframe. The analysis includes root cause analysis, rescopes, and solution analysis.

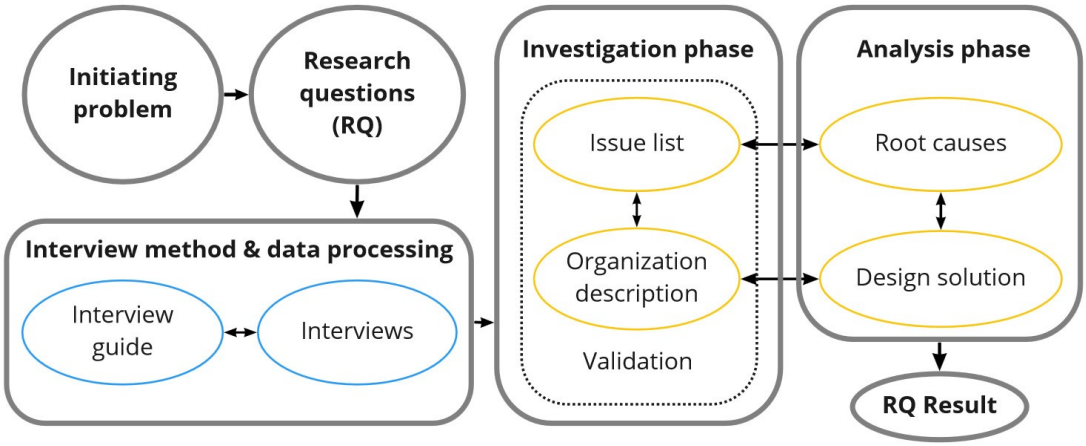

Figure 2.1: Overview of the process for this study.

2.2.1 Preparation

Given the "initiating problem" (from the Introduction) we did a pre-analysis to define the research questions, see Figure 2.1. This by consulting with two employees on problems occurring and aligning with what the organization perceives as the most important. The research questions are openly phrased in such a way that the possible answers can be varied depending on the information provided during the investigation. A more focused question to be investigated could be too specific for the benefit of the organization intended. This as the initiating problem domain is very broad and possibly affected by many areas.

- **RQ1:** What are the root causes of teams experiencing loss in productivity and adaptivity in the development delivery?
- **RQ2:** What solutions could help to eliminate some of these root causes?

There is a need of understanding the organization at a more detailed level along with issues that are occurring. The overview of the organization has been created from a walkthrough of the organization with company representatives and details are included based on the interviews and internal documentation. Simultaneously, issues have been explored using semi-structured interviews.

2.2.2 Interview method and scope

The goal of the interviews is to provide insight into how the organization function from different points of view and also bring light to possible issues that are occurring both directly and indirectly. Therefore, key individuals should be chosen from different kinds of teams and with different roles to cover many team cultures, knowledge and experience areas, and possible backgrounds.

The people that have been interviewed cover large parts of the organization, and besides management, employees with different roles have been interviewed covering six out of the seven teams (all seven teams have been represented when follow-up talks are included). Furthermore, support teams that are directly collaborating with and affect the team's development are included. The people that were interviewed have the following roles: developer, tech lead, stream lead, and product owner. It would be beneficial to include a Scrum master as well, however, the scope of the thesis did not allow this at the time because of the investigation timeframe and the availability of employees.

The goal in preparing and planning for the interview was to create a strategy for the interviews. This to have relevant questions for the different roles of the interviewees, consistency between the interviews, and issues raised by one person confirmed and further discussed with others. Interview type is chosen to be semi-structured as it provides the possibility of leading the direction of the interview easily while allowing the interviewers to speak more freely of what they consider important: like their working environment, processes, ways to do things, and possible issues. This makes the interaction more discussion-like with follow up questions and provides the possibility to dwell on a specific subject.

Interview Guide

We use the method presented in Kallio et al (2016) [7] to create our interview guide. The prerequisites of doing a semi-interview are identified as we are studying people's perceptions and opinions along with complex issues related to the initiating problem and research questions. Retrieving and using previous knowledge (on agile development, configuration management, collaboration, and software development) is done through researching the organization's Confluence and Jira pages, discussion and sparring with company mentors along with researching best practices on the tools and strategies used in the company.

A preliminary interview guide is formulated with a focus on clear wording, and focus on the participant, and as few leading questions as possible. The interview guide essentially contains the main questions with the expectation of the follow-up questions being improvised on the form of what, when, how, who, where depending on the answer and context. The questions are categorized as general or role-specific allowing for adaptation depending on the role and background of the interviewee.

The preliminary interview guide is pilot tested using a mix of Expert assessment and Field-testing where the interviewee answered as a study participant and then provided feedback and discussion concerning structure, wording, context, relation, and form. The test pilot interviewee is specialized outside of the team working on close collaboration with them and therefore has a unique overview of the organization.

The interview guide is updated after every interview with new questions from concepts and discussion point that has come up and been found interesting, regarding possible issues or organizational structure, during the interview to be discussed further or confirmed by others.

2.2.3 Data processing

Each interview requires work in processing the data to create the organizational description and the issue list. A summary of every interview is made, from the recording, noting all that has been said by the interviewee. This as we want to have hands-on data and be able to compare what different interviewees have said on similar subjects. The summary consists of direct quotes or the information phrasing the discussion as closely to original wording as possible to better display the intent of the interviewee and the discussion.

Investigation Phase 1 aims to portray the organization's information to provide the reader with sufficient information of the organization to be able to understand the setting the organization works within and the regular processes that are performed regularly. Many of the interviewees have very different roles and experiences. This will allow for an organizational description, Investigation Phase 1, that is more detailed and diverse than what could be provided by one person.

The results of the interviews, the talks, and discussions on issues and problems are presented in Investigation phase 2 (chapter 4). We group the issues to make it easier to have an overview of the issue topics.

Each issue follows a template with a description, a motivation, and a comment (optional). The description presents what the problem is, the consequence of it, and possibly an example. The motivation proves that it is an issue and presents statements and topics from interviews. We add a comment on issues that can be connected further or concluded in some way from

other issues or topics that holistically is derived from the interviews. The optional comment adds context to the issue if when the descriptional and motivational part of the issue needs further explanation.

For purposes of anonymity, issues are motivated by role only and not what stream or team the interviewee belongs to. For the same reason, support team members and sometimes business management employees are generalized, not mentioning a specific role, apart from architecture support and infrastructure as their roles are very specific in terms of how it directly affects development. This essentially means that referring to a specific role could be referring to any of the interviewees with that role. If it is relevant for the issue, then info on the team context is given as well.

2.2.4 Data gathering other than the interview format

It is noticed in parts of the interview data that there is a potential to further develop and investigate the mentioned topics and issues. We want to gather this information and make sure is part of the problem domain or has sufficient level details to be added to the organization description.

Therefore, shorter interviews or talks are performed to get a specific team's or role's point of view on a specific issue. This way we can get comments that extend and cover more of the area without spending too much time as the timeframe is tight already. Topics that are initially only mentioned and not dwelled upon in one interview can further be developed with another employee.

Furthermore, the data document was commented on by support team members regularly, filling out some parts that were mentioned but not entirely explained by interviewees. This makes the data more complete regarding everything that comes to light in the interviews.

2.2.5 Validation

We want the results to be trusted and agreed to by employees of the organization. The validation focuses on making sure that nothing in the results go against an employee's view of the organization.

The validation of the investigation is done by 2 members of a support team that have an overview of the organization as their combined experience cover collaboration with teams directly as well as with the different levels of management. When these two members found content outside of their knowledge, the interview source was contacted directly to confirm validity.

As many of the issues overlap statements made by different roles with different perspectives there might be some disagreement on a counterpart's description. This is documented and can be used to further develop the issue's root cause. If the disagreements on statements are on an interviewee's own statement, this is fully recognized and changed accordingly.

2.2.6 Literature study

Literature studies are part of the whole process presented in Figure 2.1. Initially, a broad literature search was performed before the creation of the interview guide. During the inves-

tigation phase, new concepts introduced by interviewees as well as issue related topics were explored. Relevant keywords were brainstormed regularly during the study and used to find articles in research search engines. Papers relevant to this study are sorted out from titles, then the abstract, then the introduction and conclusions. The further the study progressed, the more focused searches could be performed. ResearchGate has been used to sort relevant papers through citations.

2.2.7 Rescoping

A rescope has been performed before the start of the analysis and solution design. To be able to go more in-depth, specific areas are chosen for analysis of root cause. This so that the rescope ensures both the organizational and the academic perspective.

The aim is for the rescoping is to reflect the organization's needs in terms of what issues are selected for analysis. To reach a spread of opinion, different employees with varied work responsibilities are asked to partake. The employees have the following roles: developer, architect support (tech lead in an earlier role), stream lead, and agile coach (experience working with all the teams).

Employees were asked to rank the three most important issues in order. The Point system uses the ranking from these answers (3 points to the most important, 2 points to the next in line, and 1 to the last). From this, the areas have been ranked, 6 out of the nine areas have been given points. From this, we initially choose the 4 highest ranked to focus on further, see Figure 5.1. The Prioritization area is taken out due to the time-frame, benefiting deeper analysis in the other three areas. Two areas have the same ranking and the reason Agility and Scrum is chosen over Prioritization is the effect it has on the organizational structure and therefore the collaboration.

Issue Areas with points	Code Ownership Creating Issues	Collaboration Between IT and Business	Agility and scrum	Prioritization	Standardization	Architecture and state of codebase
Points given	1 3 3	3 2	2 2	1 3	2 1	1
Total points	7	5	4	4	3	1

Figure 2.2: Employees' ranking of the most important areas.

The analysis of the now three highest-ranking issue areas is scoped thinner to contain a specific academic point of view focusing on the theory and research in these areas. The area Agility and Scrum is "limited" to Agility focusing on identifying root causes where the organization is not following agile principles. We focus on collaborative ownership around the organizational structure as well as the area Code Ownership Creating Issues. Collaboration Between IT and Business is investigated from its relation to development strategies.

Issues from other areas than the three that are the focus are included in the analysis when appropriate.

Rescoping was furthermore done between the analysis and the solution design. The focus of the solution design is on the teams' environment and root causes connected to code ownership issues and Scrum implementation.

Chapter 3

Result From Investigation Phase I - Organizational Structure

This chapter aims at presenting how the organization functions, is structured, and what processes are used in creating and setting the direction of the development of the solution in question. The goal of the investigation phase is to identify issues leading to the development being less effective. Before addressing the issue, we need to understand more about how the organization works.

In the following, we have a general structure that we follow. First, the steps an idea takes to get released into production are presented. Second, we will investigate the department's structure and overall process to get an overview of the environment that the teams work within. Lastly, we focus on how one of the teams operate and collaborate with other parts of the organization. Here we furthermore compare with other teams and point out when there is something specific to that team. The information that is presented here has been obtained from processing the interview data, see 2.2.3.

3.1 Idea to Production

From idea to developed task used in production, there are a few steps that the collaboration goes through that can be viewed in Figure 3.1.

The pre-backlog phase is where input comes from stakeholders where ideas are discussed and reflected upon. These stakeholders have much influence, consisting of management above the area, risk area, solution stakeholders, etc. From this, stakeholder requirements are collected and presented more clearly. Then stakeholder requirements are handed over to Business Analysts and product owners for researching prerequisites and objectives on the tasks. Based on this, an overall requirement specification is created and when selected for development it is added to the backlog by the product owner, starting the work done by the team in the pre-sprint phase.

The pre-sprint phase includes extensive assessment before it can be approved for development and release. The backlog is then refined within the teams, meaning that tasks are discussed within the teams in an iterative process until they are considered ready for development by the team and Scrum master, that has the last say on the subject. Then requirement is picked for the sprint.

During an ongoing sprint, there is an iterative process of tasks being developed, regression tested, and system tested in SYST. The team is using a Definition of Done to assess if they consider the backlog item complete. The product owner has the last word on when the task can be considered ready.

Post-sprint, there is a demo and the tech lead is leading the technical release and tasks go live into production. After the sprint retrospectives are performed. The release is done once every 4 weeks.

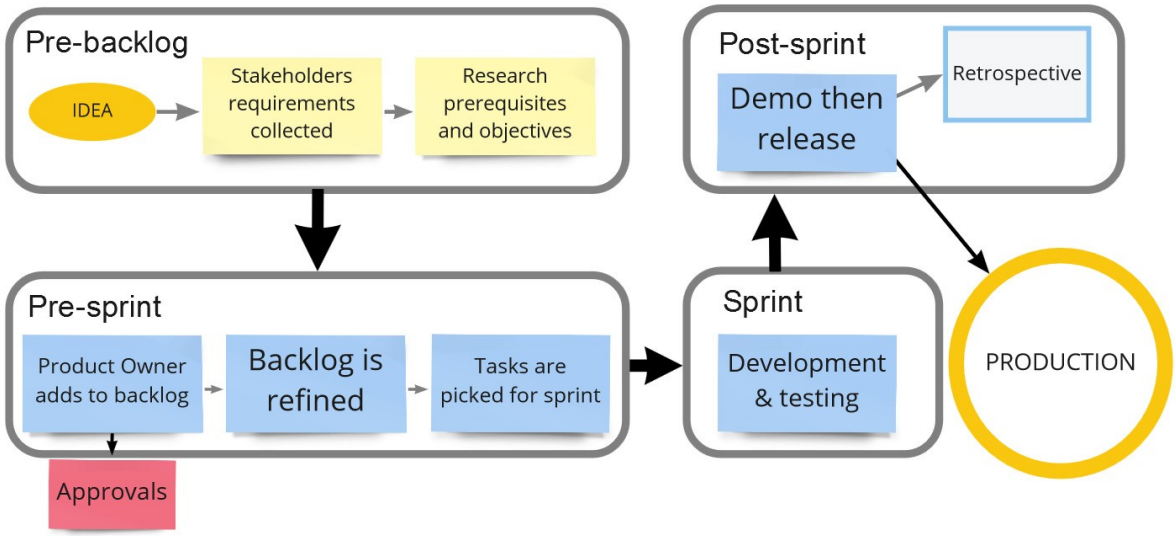

Figure 3.1: The steps that an idea go through to end up running in production. Yellow presents steps before tasks are added to the backlog. Blue boxes represent the team's work. Approvals are outside of the team's control and can take a few months.

3.2 Organization

The organization directly affects the teams and the development of the solution. The environment and organizational decisions of processes that affect the team make it possible to later understand why things are done a certain way or lead to a certain result. The description will focus on what the organization looks like along with team-spanning processes and development coordination.

The seven development teams are operating in an agile environment in terms of sprints and development but above that work lies a waterfall model in terms of deadlines from the business side where the annual goals and budget is set, the so-called water-gile-fall model. The water-gile-fall model is a spectrum where one can work more or less agile. The first one

is strived for in the department. At the same time demands on many tasks makes the teams less agile.

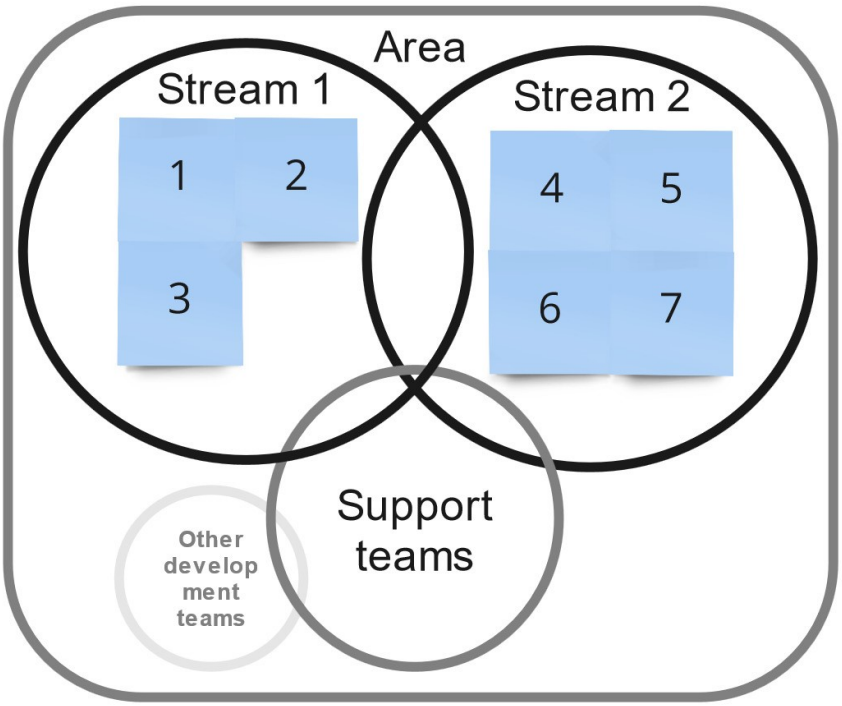

Figure 3.2: Overview of the area with focus on the two streams. Support teams are supporting all developer teams not only the 7 teams that are the object of this study.

The seven development teams are split up into two focus areas, so-called streams. This is shown in Figure 3.2. Stream 1 contains three teams and stream 2 contain four teams. Team sizes are the following in order: 10,13,6,12,12,12,10.

The setup of the teams aims at having the business and IT working closely together. Therefore, all teams have some people with a technical background as developers, tech lead, and Scrum masters. Moreover, the teams have product owners, Business Analysts, and Scrum masters with a business background.

The roles and their connections are illustrated in Figure 3.3 where the black arrows pointing to the Lead that is being reported to. Scrum masters are recruited from both business and IT sides. Depending on what faction the people have experience from, that's where their manager is. Business report to the Stream leads while IT report to IT leads. These types of managers have an human resource (HR) relationship. Managing the product is on the Product owners' plate.

Stream lead together with IT leads makes sure that the direction of the solution encompasses with some strategic elements on the IT leadership side. The manager's job in the different levels of the organization includes filtering the noise so that only the important and relevant things are essentially lifted and spent time on within the teams. This is not necessarily part of the role but is considered good to work on. Managers' roles also include mandatory excessive reporting, assessments, and bureaucracy guiding.

Each stream has a Lead on the business side. Their main source of information regarding

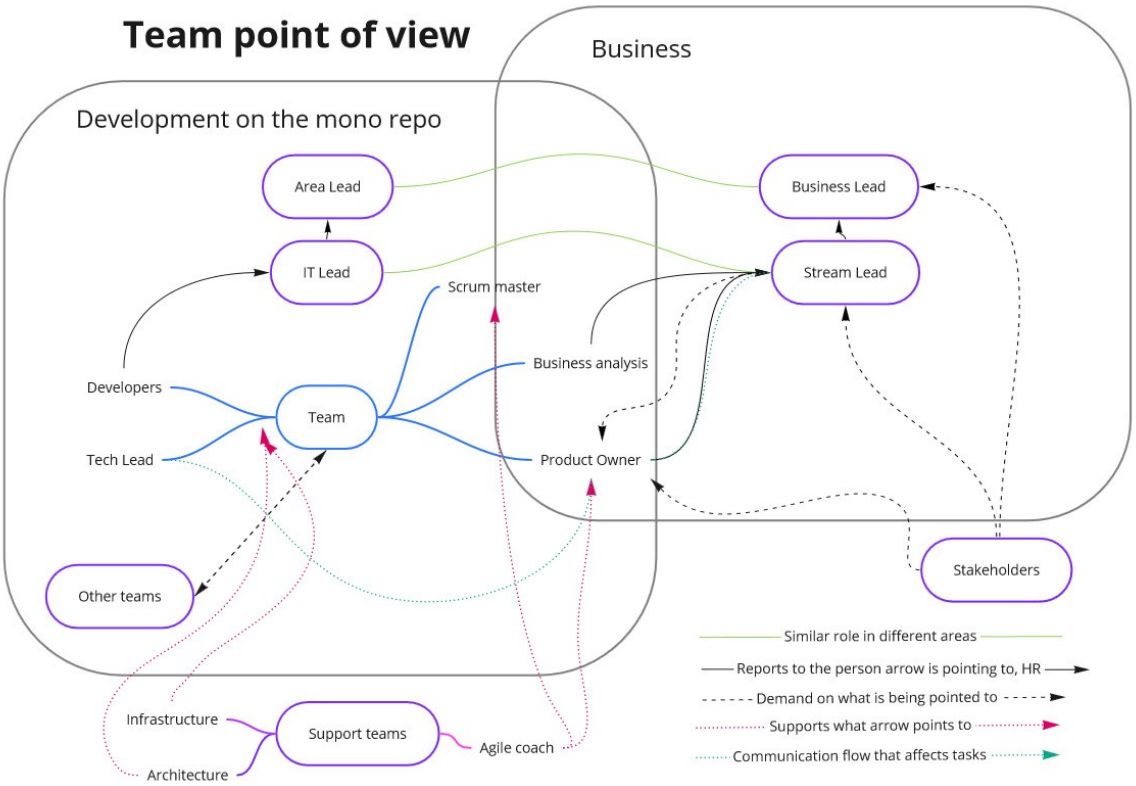

Figure 3.3: The organization from the development point of view. Roles report to IT or business side leads.

what happens in the teams is the product owners. Besides that, the stream lead also receives input from stakeholders. With this information, they can provide product owners with information that can help them set and prioritize tasks.

Each team has a Product Owner that is responsible for delivering as much business value as possible. Product Owners get input from stakeholders and collaborate with the tech leads in the teams, as shown by Figure 3.3. Stream Lead relies on Product Owner to receive information on the IT perspective from the tech lead, as they themselves do not communicate directly with tech leads.

The stream leads report to the business area lead that is involved only in the most important issues on the IT side. Stream lead communicates with stakeholder leaders and the product owner with the stakeholders that have detailed knowledge.

Product owners are the ones prioritizing the tasks given to teams. However, the Stream Lead has a say in prioritization and gets input on requirements from stakeholders mentioned. Priorities are also influenced by Business Area Lead and Architecture Support. Product Owners in the team along with Business Analysts have the business knowledge while developers and tech leads have technological knowledge.

The IT part of the teams usually is mostly involved in the refinement of the tasks, not the prioritization itself. Product owners motivate the priority within the team up in the hierarchy, to Stream Lead, using input from tech leads regarding technical requirements and tasks. Then the priority decision is on a general basis made on whatever seems to bring the most business value.

There are four waves (big room planning) of planning per year. Before the wave, the stream lead has a pre-wave planning session with the stakeholders. Business area lead (that stream leads report to) and IT lead participate in that talk. Stream lead then asks teams what they plan on bringing to the wave planning, teams talk to their stakeholders. The plans made for the wave should not be considered as fixed as unexpected things can happen during the sprints. The aim is to be clear on the deliverables on the first sprint. As the wave progresses, uncertainty in the deliverables usually goes up.

Tasks and prioritization of tasks are done by product owners in collaboration with stream leads. The plans for the sprints build on task estimations, using story points, done by the development teams themselves. Furthermore, estimations are done differently depending on the team. They all have different views and context on what is the full expectancy of the role is as a Scrum master, product owner, tech lead, etc. It is not set in stone how to do estimations, rather it is up to each team to decide. Here Agile Coache and Scrum Master have the job of guiding the team and give input from observation on what they can improve to get closer to agile ways of working.

Support to the teams is given from support teams outside of the area. There is a collaboration with Scrum masters on improving the agile ways of working. Infrastructure support regarding configuration and infrastructure is more effectively handled by one support team than having each team acquire this knowledge themselves. Architect support is mainly concerned that the area and the solution are coherent with the rest of the organization's guidelines and architecture. The architecture support is not concerned with the detailed architecture in the domains, this in some way falls on tech leads. However, architect support is available for larger things that need input, development of new features, or if there is a conflict between teams that cannot be resolved among themselves.

3.3 Team Perspective

The perspective of different teams varies as they operate in different ways and are differently exposed to external pressure. Below is an example from of one the team's perspectives and at the end a few comments comparing to other teams will be made. Team 5 operating in stream 2 is chosen and is generally considered to function well. They also have in their favor that they are more isolated in terms of how much they collaborate with other teams in terms of shared ownership.

Team 5 has frontend, backend, and full-stack developers, as many of the other teams do. In planning for the sprints, the developers are given user stories or epics. These come from the Product Owner and the Business Analyst that has researched prerequisites and objectives. Initially, the product owner receives a business need from the stream lead that is then discussed with the team, estimations are done and then tasks are prioritized together with the stream lead.

The refinement sessions are attended by all developers and take place after the Business Analyst has analyzed what should be done. In other teams, developers sometimes do not have time to join these, or only a few do. The refinement session can be done before and also during the sprint. In team 5 during the refinement session, there are discussions and inputs given. If there are unclarities and things that can be improved, iterations are done, the Business Analyst finds the details and then a new refinement session will be scheduled. In

the sprint planning session, we expect most things to be evaluated at the refinement session before and considered to be ready for development.

The procedure of the estimation of tasks is left to the Scrum master to figure out. This team's procedure is to do estimation in days, where the daily capacity of the team is considered, and tasks are given an interval size with best- and worst-case scenarios. Estimations are done in the planning session and the sizes are rough and could be described by s, m, l, and xl. This as hours are not worth to estimate in as it cannot be reliable, intervals are more reasonable from a development perspective. This also creates less stress on the team. The Scrum master in the team is responsible to estimate team capacity. To make it possible for different developers to work on the same thing, without disturbing each other, there is an effort put into splitting up larger stories into smaller pieces.

After the sprints there are retrospectives. If there are issues in the development these are raised here. According to the developer in team 5, they have good communication during these and are not afraid of talking about problems. Depending on the type of issue it gets raised either by tech lead if it is development-related or Scrum master of more related to communication as an example.

When working on a task in the sprint the procedure is to finds out how to do it. If there are uncertainties within the team tech lead is contacted and from there other teams can join in the investigation if it concerns them. If there are larger things or mainly that the teams cannot solve or agree on it between themselves then the architect support team is contacted for input.

Team 5 has ownership in 3 domains and contributes to 5 domains owned by other teams. In terms of traceability of the software and the development Team 5 uses the Azure DevOps tool. They consider it clear and nice in terms of overviewing commits and the different domains they work in. Other teams have expresses issues regarding the traceability regarding getting an overview of and go through the history as all teams are committing to the same repository. Teams are in general able to choose any tool they wish to use.

In the case of team 5 there is no documentation other than task management and commit information. Old versions of the diagram are referred to in confluence. One member giving support to the teams has expressed that new features are being documented over the solution worked on by all teams. In Team 5 they rely on experience to know what they need when maintaining the solution and it is usually very difficult for newcomers in the beginning.

Most of the tasks worked upon is providing business value, additionally, there is an aim to include technical tasks into the sprints. However, technical tasks usually get down prioritized because of the pressure or need to provide business value. Sometimes technical sprints are added so that the team can catch up on dealing with technical depth, about once every quarter. This is specific for team 5, other teams have wanted to do something similar but have not come through.

Sometimes in the area, there are hard deadlines even though the aim of the organization is to work in accordance with the agile framework. The hard deadlines are dealt with by managing the scope, feature within is spread out over different sprints and then things must move out from the original sprint planning. In many cases, not enough is moved out and it happens that the team is pressured. The scope of the hard deadline keeps on getting larger as the initial promises on the delivery are not clear and can be reinterpreted. There is usually a huge underestimation of the target that has been promised. These promises are made outside of the department and development teams are not consulted on the size and effort of the

target.

There is a lot of pressure coming in from the business side to have new functionality and features, besides the legal requirements that usually have a hard deadline. This is a blend of legal requirements imposed outside of the organization as well as a need for new or extended features that improve the effectiveness or the functionality of how the business side uses the system.

However, sometimes it appears that the business side in regard to features do not know what they ask for and the consequences it has for other tasks being pushed out of the sprints. The team's Business Analyst investigates the requests coming in and find out if it is urgent. Business analyst communicates with the stakeholders regarding the requirements of the issues, collects input from UX-support and risk-support teams. Then the story is prepared, and refinements are done with support-teams and then the team, iterative refinement processes. Here the development has some say on the relevance of the technical part of the solution. The product owner and the business then move tasks considered prioritized into the sprints and move out tasks accordingly. When it comes to taking in new critical assignments something must go. The priority here is key.

The collaboration with teams works well in most cases. There are rarely merge conflicts between developers in different teams. Communication between developers is mainly around the pull requests. These must be approved by the owners of the features and owners of domains. When there is friction it happens usually regarding disagreement on a solution. One team decides to do things a certain way and the other team doesn't agree. There is an unclarity around ownership where one team owns the feature, and the other team owns the domain where part of the feature resides.

The branching strategy used by the teams is release branching. Before the pull-requests are created, the solution unit-tests have been performed along with the code review. When the pull request has been approved the pipeline moves the branch and build the code in the test environment. If it passes, then it can be moved manually to the SYST environment. All tasks are merged into master as soon as they are completed so far and are then tested in the SYST environment. There is a freeze period during regression testing before release. Releases to production are done once every four weeks. The move of the branch can be done by anyone, however, the procedure is that it should be approved by tech leads and management.

Feature toggles are used. In the case that something is caught last minute that can be toggled of and still go to release.

As business requests, in terms of improved and added functionality, are prioritized in the prioritization of tasks it happens that quick fix-solutions are implemented. According to the developer in team 5, the consequence of this depends on how bad the quick fix solution is. For a developer in team 5, a quick fix happens about 2 times every 6 months. In team 5 the procedure is that after a quick-fix is implemented you should add a task to back-log a task to fix it with analysis of root cause and there is a lot of collaboration to find the right solution. This happens, but not every time, or it takes a while. This is a larger problem in some of the other teams where quick-fix solutions even are maintained and functionality added over a long time.

Chapter 4
Investigation Phase 2 - Results

In the second investigation phase, we focus on the issues that lead to the development being less effective and adaptive. From understanding what the issues are, we will in the next chapters investigate the root causes and reasons for why many of these problems occur. We present the issue list consisting of 9 areas and other results from the investigation. Below is a summary of the results of the investigation, for more details consult Appendix A with its 11 sections (A-1 -A.11).

4.1 Summary

The problem being addressed is why there is a loss of efficiency and adaptivity of the development teams from the perspective of the employees. Different roles experience several issues and this is reflected in the interviews. The issues found relate to development practices, processes, organizational structures, communication, and collaboration. The investigation has resulted in 60 motivated issues affecting development. They have been grouped into the following 9 areas (section A.1 - A.9):

- Standardization
- Architecture and State of Codebase
- Traceability
- Collaboration Between IT and Business
- Code Ownership Creating Issues
- Code Collaboration and Development Teams
- Development and Shortcuts
- Prioritization
- Agility and Scrum

Due to the limited timeframe of these thesis and the extensive nature of our issue list, we leave the treatment of several recorded issues and statements as future work. They are presented in Section A.10. Furthermore, all interviewees were asked to make a statement on the one thing they considered would improve efficiency and adaptivity. This collection of statements is presented in the section Wishes From Employees (A.11).

4.2 Results

The loss of effectivity and adaptivity experienced by the teams links to the issues employees encounter. How these are intertwined and the specific root causes for these will be explored in the analysis of the next chapter. Three of the issue areas will be the focus of the analysis. These are the result of a rescope performed, described in section 2.2.7. A similar analysis for other issue areas would provide an interesting direction to extend the work of this thesis.

The three areas (A.4, A5, A.9) affect the efficiency and adaptivity in more than one way and the direct consequences are not always clear due to the complexity of the issues.

The area Collaboration Between IT and Business (A.4) is important due to the close collaboration between the two sides of the organization and is particularly interesting due to the differences in processes and structures between them. IT is using Scrum and agile methodologies while the business side is working more closely to traditional waterfall strategies.

Code Ownership Creating Issues (A.5) affects the teams' ability to make improvements to the code and processes. Issues become larger and more difficult to handle along with rising costs of maintenance and development.

Agility and Scrum (A.9) is essential for the development strategy to function as intended. Issues in this area affect the development directly and connect to other issue areas amplifying the consequences of these.

Conclusion

From the result of this investigation of organization and issues, found in Appendix A, we conclude that there is room for improvements in aspects around collaboration and process in the organization. Specifically concerning investigating consequences of organizational structure, unclear responsibilities between roles, and lack of commitment to strategic processes from the organization. We now turn to address some of these issues in the next chapter.

Chapter 5
Analysis and Design

In this chapter, we aim to analyze root causes and design solutions to these. The analysis is to focus on problem areas that have been found in the investigation phase. In analyzing root causes we gain a deeper understanding of how some of these issues occur and why. Furthermore, we can by investigating possible designs to solve the root causes find solutions best fitted to the environment and the condition of this organization that has been studied.

Firstly, a rescoping of the results from the investigation phase is performed, see 2.2.7. Three issue areas are analyzed. Each issue has root causes examined with results presented and design solutions examined with preferable solutions proposed and motivated.

5.1 Agility and Scrum

In this chapter, we analyse and discuss design solutions regarding where the organization is not following agile principles. This will contribute to answering the research questions. The structure contains analysis of root causes, where agile principles are matched against the issue list in Appendix A. The analysis result is followed by a rescope then solution design, see 2.2.6 regarding literature study and 2.2.7 for rescope method.

5.1.1 Analysis

The agile method is principle-based and the agile team is to be guided by these 12 principles [14]. Agile standards are not fully implemented, see Agility and Scrum (2). Furthermore, from the result found in this study, we find that at least 8 principles (of 12) are being broken in different ways.

The organization is committed to using Scrum as a method. Scrum teams should be self-organizing and self-directed [14]. This part of the Scrum Methodology is not followed as many decisions are done in higher management and other parts of the organization have a high impact on the teams' everyday work. This points to an organizational issue in terms

of how management is expecting the teams to work and how they actually do in reality. This by committing to working per a methodology without adapting the structure of the organization to follow through.

The teams are part of a large organization and many structures are inevitably inherited into the department. The supposedly agile part of the organization is differently managed compared to the business side of the organization. Introducing Scrum into an organization is similar to entering an organizational change, with great opportunities and also risks.

Decisions should not be made by an authority, rather by people who are affected by the issue at hand [5]. In this case, we find that the business side and management have much effect on the outcome of decisions being made, see 5.4. This creates a difficult environment to implement and improve the Scrum method and agile principles. There is lacking a systematic reflective approach to effectivity where issues regarding this do not have a place in retrospectives or other forums along with Scrum masters not being given the authority or tools to solve issues that occur. Here we again find the Double ownership as a root cause as it moves decisions away from the teams.

The agile principles regard the practices that collaborating agile teams should strive for [14]. Broken principles contain both isolated incidents that point to issues in the collaboration as well as structures that are not strived for because of the business and management organizational structure.

From the interviews, we have found that strive for continuous delivery is lacking. The consequence of the issues is partially described in Double ownership and is noticeably a management decision where it points towards mistrust and misunderstanding on how agile development should be done. Here we are refereeing to the rigid approvals that need to be done before releasing into production.

Furthermore, the high pressure is not maintainable in the teams as this affects the solution while prioritization is performed without the technical necessities considered along with top-steering decisions on what the teams should spend their time on. The opposite of what the agile team using Scrum should do. This unfortunately is leading towards down prioritized technical excellence and good design followed solution becoming more complex and difficult to maintain.

Result

The 12 agile principles are matched against the issue list in Appendix A. Below are the eight principles that are being broken. They are motivated by issues presented in the appendix.

- Principle 1 (early and continuous delivery) is broken as the teams are not delivering early and continuously, see Agility and Scrum (5) in Appendix A.

- Principle 5 (give support and a good environment, trust the people, build a project around motivated individuals) is broken when team is not given an agile-friendly environment and support, they as well are not being trusted “getting the job done”. This is seen as a consequence of ownership structure in issues under Code Ownership Creating Issues (1-6). The creation of a dysfunctional environment in Code Collaboration and Development Teams (1), lacking trust and overruling team decisions regarding development in Development and Shortcuts (4).

- Principle 7 (working software is the measure of success) is broken when working software isn't the primary measure of success. The teams are enforced to work with a bureaucratic approvement process, see Code Collaboration and Development Teams (12)

- Principle 8 (sustainable development, stakeholders and developers shall maintain a constant working pace) is broken when sponsors, developers, and users are not maintaining a constant pace indefinitely. Teams working overtime is a symptom of this rule being broken. Blockers found can be in the approval process (Future work), when testing cannot be done as stated in Code Collaboration and Development Teams (13). Part of the organization not even aware of how it should work and constantly will break it, see Collaboration Between IT and Business (2, 5).

- Principle 9 (attention to technical excellence and good design) is broken as technical efforts are overshadowed by the endless stream of business requests. This issue appears in Development and Shortcuts (1, 2, 3). Teams are not dedicating enough time for refactoring, see Collaboration between IT and business (4, 7), see Code Collaboration and Development Teams (11).

- Principle 10 states that simplicity is essential. Being broken when system complexity rises, see Architecture and State of Codebase (3) and Collaboration Between IT and Business (7).

- Principle 11 implies that self-organizing teams should be strived for as this allows better architecture, requirements, and designs. Self-organization is lacking as teams do not operate in isolation, are managed from above and decisions made in the team can be overruled. Please see issues Code Ownership Creating Issues (6), Prioritization (2), Agility and Scrum (4), Collaboration Between IT and Business (2). For misunderstanding on what teams are expected to deliver and at what pace, see Collaboration Between IT and Business (5).

- Principle 12 regards regular reflections on process improvements, action to optimize and adjust process and behavior accordingly. For issues regarding regular reflection and improvements on effectivity and adjustments accordingly, see Agility and Scrum (3, 4).

5.1.2 Solution Design

From the results of the analysis, we find that many of the agile principles have been broken from the issues found in Investigation 2 and many are related to how Scrum and agile principles are implemented into the organization. In the design, we explore possible improvements to the Scrum process to improve some of these with the focus of empowerment of the employees. The focus that is chosen to discuss is solutions for the violation of principle 5 and principle 11 as these revolve around the team and its environment and is connected to code and collective ownership issues.

The use of Scrum does not directly introduce productivity, unlike what many corporations seem to believe. Rather the method used properly makes problems transparent so that they are solvable and once solved productivity may rise. Self-organizing teams are one

of the core necessities in using the Scrum method. To derive the benefit from Scrum self-organization is a necessity[5] and in agile development is reflected by principle 11. Several improvements can be introduced into the organization and the teams that operate around the mono-repo (solution). In the structural support and the environment, the team operates in, as well as the trust it receives from stakeholders and its management as mentioned in principle 5.

Team members having very clear responsibilities and the power to enforce their decision is an important part and should be included in management responsibilities in supporting the teams. To be self-organized there is a need for clear structure and responsibilities. A single product owner needs to have the singular power to make decisions concerning the product in the team [5].

It is reasonable that the product owner base decisions on objective criteria and the stakeholder wishes to benefit the stakeholders and business side of the organization: area lead, stream lead to name a few. The product owner is making the decisions based on external and internal input from the team regarding technically related, business-related, and other tasks. In no case would someone else (including management) make decisions overruling the product owner on decisions regarding the product [5].

From the results of the analysis, we find that managerial and business-related functions, individuals, or teams want to control the teams in an inappropriate way. This works against the agile principle 11 for enforcing self-organizing teams or shows a lack of trust working against principle 5. This is something that happens in other companies as well. Examples of corporate functions that can contribute to this usually include concept development, software architecture, quality assurance, and process management [5].

Other parties acting to solve or influence the team's decisions regarding tasks and issues, is not acceptable when using Scrum. Issues and tasks should be solved by the team itself. If other parties want to contribute, they need to join, as regular members, the development team[5]. If there are any doubts as to why this would be the case, or someone would like to not abide by the rules "just this time" one should keep in mind that there is a difference between what is good for one individual or team and what is good for overall corporate success. Scrum works well when decisions are being made, on a well-coordinated and clear product vision, by a product owner and not by someone that has a perspective from outside of the team.

The teams and the organization that is being studied are under much pressure in terms of what needs to be done. What has come forth during the interviews is mistrust between the development teams and other parties. Teams are not being trusted to do their job and are therefore being top-steered.

The pressure will not move away, and it might be overwhelming for all parties and might feel impossible to change. Even if it is not a possibility to change the pressure, it is not an excuse to move away from the decided upon development strategies, we here refer to Scrum and agile principles that are being broken. Moving away from these and cutting corners in this area does not solve the problem, rather makes it worse as the benefits that an organization wants in using Scrum and agile development will not appear.

When trust is lacking organizations try to build up control systems that are close to impenetrable [5]. This has been shown in this organization as well, for example in how decisions are being made outside of the team. Even decisions regarding development and solutions around it have on occasions been overruled by upper management in specific cases.

Trust is something that is a core requirement for the Scrum method to work and it goes

both ways. Then decision-making authority can go to the appropriate level: the product owner. To make this fair for all parties that have a stake in the development teams everything should pass through the product owner. Management or business should not have direct access to team members. Scrum demands that all requirements are submitted to the product owner that is the only person to sequence and prioritize those requirements [5]. By following this, a vicious cycle circumventing the process by communicating directly with developers can be broken. If not, then the benefits of Scrum are lost. Therefore, enforcing trust between all parties should be of the highest priority.

A possible way to improve in that matter is to move the responsibilities of the decision to the product owner, along with solving the double ownership as this is a pre-requisite for it to work effectively, see 5.3. Without single ownership, there would be more than one Product Owner. Teams need to be given the trust to work agile and be protected from all parties that might prevent them from doing so. If not, there might be a reason to consider other strategies of development than Scrum and the agile approach. However, similar issues will most likely still appear and the reason for using Scrum is that if properly implemented issues become visible and consequentially approachable.

Summary

- Improving the implementation of Scrum is necessary for the indirect benefits of increasing productivity. Therefore, management's focus on encouraging self-organizing teams is recommended to be focused on and continuously encouraged.

- Team members with clear responsibilities and the authority to introduce change in the process can improve the self-organization of the team. To successfully implement self-organizing teams mutual trust in the organization is essential.

- The product owner is always the decisionmaker on everything regarding the product: technically or business-related and makes decisions based on information from team management and stakeholders.

- All parties that are outside of the team, including stakeholders and management, are recommended to not be allowed to influence or communicate ideas directly to the team outside of proper channels. The product owner is the link that all requests should go through. If one wishes to contribute one can join the teams as a regular member during sprint events.

5.2 Collective and Double Ownership

In this section, we discuss root causes and possible design solutions to the area Double Ownership and the organizational solution setup from a collective ownership perspective. This will contribute to answering the research questions. Each part of the subtitles in the analysis section contains an analysis of root causes ending with a summary. Lastly, we reflect on design solution.

5.2.1 Analysis

Collective Ownership

Collective ownership is having one team as an entity responsible for the code, allowing all team members to change in the team's code. Results from research have highlighted the benefits of collective ownership [8]. Collective ownership is practiced properly within the teams in the organization as all members take collective ownership with their respective ownership responsibility, see 3.

However, we can conclude from the results of this study that the collective ownership does not have boundaries as a team can own code in other teams' domains as well as other teams having code ownership in their domain, as seen in the organizational description in Investigation Phase 1. This has a consequence of the setup and coordination between teams in regard to that collaboration between teams simulates what is expected of individuals in a team, as seen in Code Ownership Creating Issues (1, 2, 3, 5, 6).

In practice, there are clear boundaries of the solution's development teams on paper. However, we find that in practice the teams are operating as a larger entity. Essentially, the borders of the teams are not fully set as the current setting requires much cross-team collaboration. Therefore, the organization cannot control how "large" the teams are in day to day work.

Team size is ideally 5-10 individuals according to most experts. Larger teams are considered ineffective in delivering as well as unwieldy [1]. In the organization being studied the size of most teams is closer to 10 or more. Therefore, in terms of size, it is already on the upper scale.

Hence, a root cause for the teams being inefficient, that stems from double ownership, is either the misinterpretation of collective ownership or the size of the team. If the teams were smaller or the teams were more isolated the issue would be less severe. About 20 percent of developers' productivity is lost for every additional project that is added [5]. Something that affects the effectiveness of the teams when developers have responsibilities outside the team.

Summary

- Most team sizes are too large.
- Collective ownership is misunderstood and team structure does not have boundaries. This can be seen as the catalysts, or root cause, that keep the teams from working in isolation. About 20 percent of a developer's productivity is lost for every additional project they work on. Teams do not have collaborative boundaries that allow them to work efficiently and be adaptive. Unclear boundaries affect the "real" team size that appears in practice.
- Double ownership is a catalyst that keeps teams from working without disturbances from other teams.

Double Ownership

The advantages of having clear code ownership are in the establishment of a clear and single-minded vision for the area under ownership [14]. In using double ownership, as described

in the organizational description in Investigation phase 1, this will not be the case as will be shown below. What will be shown is that it creates issues in collaboration, code development, and incorrect ownership. Furthermore, there is a deviation from optimal pressure on teams, team adaptiveness being reduced, increased work overhead for teams, and solution affected regarding complexity and maintainability.

The design of a system (solution) may be defined as the intellectual activity that creates a useful whole from its parts [3]. The teams that are working on the mono-repo are essentially the parts that create the whole and the intellectual activity is the team's development process, collaboration, and communication.

The collaboration of the mono-repo being investigated has an organizational structure as described in Investigation phase 1. The double ownership issues that are described in Code Ownership Creating Issues (2, 3, 5, 6) is shown to affect both the collaboration between the teams and in the organization as a whole as seen in Code Collaboration and Development Teams (1, 3, 4, 5) and Development and Shortcuts (4, 5, 6). Furthermore, it in extension affects the solution that is being developed as seen in Code Collaboration and Development Teams (7). The ownership is split between domains and features. Domains contain features that are owned by another team other than the one owning the domain.

Unclear boundaries on who has ownership will be reflected in the code design and collaboration around it. The organization chosen strategy to have the double ownership structure is therefore far from optimal. It leads to the unintended consequence that is described in Collective Ownership above. Furthermore, it leads to many issues that take much time from all people involved in the development. Mention of this is found in Code Collaboration and Development Teams (3).

The reason to have smaller sizes of teams is to have individuals or smaller groups working in isolation as this is an advantage in terms of producing results effectively as it limits the overhead work compared to organizing and collaboration in larger teams.

The right team do not always have correct ownership, see Code Ownership Creating Issues (1). This stems from the fact that there is a high continuous demand for maintenance and development. If something is considered prioritized along with many other issues and most of these are within the same team domain, then it happened that another team is given the task to develop. Then another team will own the feature in someone else's domain. Whenever a new task regarding that feature will come in then it would be redirected to the feature owner, not the domain owner. From this, we consider the double ownership a root cause for the situation as the coordination required has consequences for future collaboration in the solution as seen above.

The consequences of this however are not limited to double ownership. The lack of team boundaries mentioned in Collective ownership points towards an unstructured organization. Moreover, as mentioned by [3] the solution and collaboration around it will reflect this. We argue that the issues presented in the collaborative ownership are a symptom of a root cause connected to how the organization is organized as a whole (This will be discussed more in Collaboration between IT and Business). About 20 percent of a developer's productivity is lost for every additional project added[5].

The pressure is related to double ownership and the consequences relate to the team's ability to effectively maintain the solution in the long haul. [10] mentions an optimal point where the amount of pressure on a development team is as effective as possible. The further away from this point, the effort of development will growingly increase, and when the pres-

sure is enforced developers will eventually burn out and end up taking more time to finish their work. Pressure in relation to this is viewed in Code Ownership Creating Issues (3). The consequences of these are dire for the development and maintenance of the solution. Therefore, too much external pressure on teams becomes a root cause for teams to be less effective over time. Other issues related to pressure can be seen in Code Collaboration and Development Teams (6, 7), Development and Shortcuts (2).

Structure wise there a large amount of overhead work that the teams are experiencing. Some of these derive from the pressure described above. Even if there is a strive to uniform the way teams are doing certain things and create a component for it many of the domains have similar functionalities that are upheld and maintained separately. Some teams do their own thing and sometimes do not follow technical standards, some of the teams have the knowledge and others don't.

Furthermore, on the topic of overhead work, many of the domain have similar functionalities that are upheld and maintained instead of improvements being made, here we find that external pressure is a root cause for this as well as technical tasks are being down prioritized, see Code Collaboration and Development Teams (11). Along with double ownership making it difficult to change it as described in Code Collaboration and Development Teams (3). Therefore it is a problem that this team in Architecture and State of Codebase (2), and most likely others, avoid dealing with team-crossing technical architecture issues due to complexity. What we derive from this is signs of lack of adaptiveness as changes seem to be difficult to do. Examples of this we can find in Code Collaboration and Development Teams (14) where it is found that the solution has a bad foundation for scaling. For more on adaptiveness, we refer to Agility and Scrum.

Work overhead can furthermore be found in Development and Shortcuts (3, 4, 5, 6) regarding maintenance of development shortcuts, pull requests, and collaboration issues that are a direct consequence of the double code ownership. Developments shortcuts cause maintenance to be more expensive and add complexity to the solution. Pull requests are taking unnecessary time from development as teams do not have single ownership to code and need to involve other teams in changes and affects adaptiveness. When changes are made multiple parties need to be involved, agree, and approve. Collaboration wise this, then, creates the risk of friction between teams, as seen in issues Code Collaboration and Development Teams (4, 5).

The success of software projects relies mostly on people. Having highly skilled people that coordinate effectively is more important than tools and methods [1]. Therefore, the coordination issues created by the increasing work overhead regarding coordination are affecting the resulted outcome heavily.

The double ownership strategy is a root cause for many of the issues found in this study and covers multiple areas. Furthermore, the impact of this as mentioned above from these issues is affecting the team's ability to coordinate effectively as well as creating a need for much overhead work from individuals, leading to organizational issues and affects the code in terms of solution maintainability, complexity, and scalability. Thus, closely related the adaptiveness and effectivity of the teams as the results today will affect the results being done tomorrow.

Summary

- Double ownership leads to an increase in overhead work for employees.
- Double ownership as well as incorrect team ownership affect the teams' collaboration and the maintainability of the solution.
- An unstructured organizational team structure with lacking boundaries is a root cause for issues regarding adaptiveness and effectivity and is reflected in the solution (in terms of scalability and complexity) and the collaboration between the teams.
- High complexity along with the double ownership that even could be triple or more when refactoring in the solution leads to teams avoiding working on improving the solution.
- Double ownership along with too much pressure on teams is a root cause for teams increasing work overhead that affects the team's effectiveness.
- As the solution has a bad foundation for scaling adding more teams to the solution will further introduce more complexity and work overhead to the teams if the root causes described above are present.
- Effective coordination (and communication), one of the most important parts of a software development project, is lacking and therefore affects the solution maintainability, complexity, and scalability.

5.2.2 Solution Design

The team sizes are initially too large and the collaborations going on with other teams is increasing the "efficient" team size in practice. This a root cause that has been found to be closely connected to the root cause of double ownership between teams. Improvements made to these root causes would improve the efficiency and the adaptivity of the organization as having been mentioned in the results of the analysis.

From the organization perspective, it has been difficult to enforce strategic decisions that have been mentioned in other parts of this paper, see 5.4.1. In improving the implementation of Scrum many of these issues that are mentioned in this study can be solved. To do this there is a need for team structure and size to move towards what is recommended in Agility and Scrum. For this to be possible the culture around communication and structure needs to change along with the Ownership of the teams clarified.

A challenge for the company is transforming and removing the double ownership is the complexity of the solution where we from the interviews draw the conclusion that the cohesion could be higher, and the coupling could be lower. The feature and domain ownership needs to be split and changed in the code itself, and the encouragement of cleaning up the solution is recommended to be prioritized.

Below we present modularization as a possible solution to change the current structure to one that will allow changes to be made without jeopardizing future maintenance and changes. Furthermore, we discuss the challenges of development in an environment heavily affected by traditional waterfall structures and how this was dealt with in another company operating in a similar context as the organization in this study.

Modularization can be used to improve flexibility and comprehensibility of a solution and at the same time improve the effectiveness of development. This is however much dependent on the conditions that modularization is performed, aka the criteria. Furthermore, it is depending on how the modules transfer control between each other on how negatively affected a solution is if the criteria are not fitting for the solution [11].

We have from interviews found that the requirements in the code domains are constantly changing and that the solution has many domains and features that create a complex and intertwined system. [11] proposes that the initiation point in structuring modules begins with finding where the difficult design decisions are as well as where the design decision is likely to change. The modules should be designed to hide this complexity from other modules.

To create an efficient implementation, one should allow features and subroutines to be assembled code from different modules, rather than thinking that a module is one or more subroutines. In the solution kernel there has been an effort of creating a module that can be used by multiple domains, although the criteria for the creation is unknown, we find that there has been a strive to modalize certain functionality. The first step to be done is to give the teams the time to implement it.

Furthermore, we find that the changes to the requirements resulting in new features and existing ones that are being changed should be modularized in accordance with the criteria around changing designs.

In [4] we find a study about an organization operating in a similar external environment to the traditional waterfall approach model. The teams are geographically distributed as is similar to the current organization where developers and teams are in different countries and most development is done online due to the ongoing covid pandemic.

[4] takes a stance where one of the issues is related to flooding of change requests to the teams due to weak project management or shared responsibility between 2 project managers. The projects in [4] are trying to keep up with documentation causing work efficiency to go down and experiencing difficulty keeping documentation up to date. Furthermore, there were issues in lack of or constantly changing requirements, described as a barrier in capturing business requirements easily and effectively.

In one of the projects lack of management maturity and understanding of agile practices affect the development life cycle where immature decisions caused rework and overtime from the whole team leading to defects impacting software quality [4]. In the ability to successfully do the retrospective exercises the teams found ways to improve current processes. This making it possible to improve the team maturity and implementation of agile principles and Scrum practices [4].

We find that many of the issues found in [4] are similar to issues found in this study on collaboration, lack of requirements documentation, and management (or business) not adapted to the use of agile practices. Thus, affecting the development efficiency.

The organization must be more aware of the reasons for adopting agile principles. The teams need support, from management and surrounding organization, to self-organize. In the ability to successfully do the retrospective exercises, the teams could improve and streamline their working process. We find that proper retrospectives are an important prerequisite for the teams to be able to improve their way of working. Therefore, measures should be taken to make sure that retrospectives are performed and facilitated in accordance with the Scrum method.

The teams, when given clear ownership over domains and modules, then have the pos-

sibility of improving their agile practices using the retrospectives. Once these are found the teams additionally need to have the tools to be able to implement these changes into their own work processes, as they can be if the Scrum method and agile principles are adopted and structurally followed by both management and the teams being encouraged to self-organize.

Summary

- We recommend the use of modularization with proper criteria to hide complexity. Furthermore, identify and deal with often changing code and changing requirements. This will require much effort looking into the code but would be an improvement for the development efficiency long-term.
- Encourage teams to deal with technical issues before issues that are difficult to identify appear. Thus enforcing a mindset of working in a preventative manner, focusing on the essential underlying challenges rather than the visible issues.
- Retrospectives are key to the implementation and improvement of Scrum processes and self-organization, supported by management, allows teams to implement changes and becoming more adaptable.

5.3 Collaboration Between IT and Business

In this chapter, we will analyze and discuss design solutions related to issues in collaboration between IT and business. This will contribute to answering the research questions. The structure contains an analysis of root causes ending with a summary followed by a solution design that relates to the area and follows up on earlier solutions in this chapter.

5.3.1 Analysis

That there exist collaboration issues between business and IT has come up in most interviews from the investigation phase. The issue is viewed differently depending on the perspective of the employee. What most of them agree upon is that some things need to improve. What we have found is that there seem to be complex issues that have much to do with communication and background experience between teams or employees from both IT and Businesses. These issues are allowed because of the organizational structure set in place that is both strictly hierarchical as well as unclear on responsibilities and therefore fluent. Many issues are being fixed on an ad hoc-basis where priority is unclear.

Conway already in 1968 in [3] stated and showed how a solution is heavily dependent on the organization and its structures, processes, etc. If a solution is then to be dependent on best practices within software development, the structure around and within the creation must be tailored to it.

What has been shown during many of the interviews is that this is far from the reality of the 7 teams that are working and collaborating on one single product. This happens even as the organization has an agile development strategy in the form of Scrum. In many cases, the managerial and organizational structure is ad hoc based and evolves around the stakeholders and business side first-hand.

Soft factors highly affect the performance of development teams [12]. As the organization evolves around the business and the stakeholders these have been given much power micro-managing teams using the managerial hierarchies that exist in the organization. When the business side is not happy with a decision made on the IT side, the conflict is escalated up the ladder of management.

Moreover, the fate of the product in terms of the prioritization of technical and other tasks lies outside of the IT domains control, leading to increasing technical depth as well as no room and time to improve existing processes so that teams can work more effective long term, instead, they are on a road where tasks will take longer and longer to perform [6]. This is a consequence of the issues that are mentioned in IP2: Collaboration with IT and business, Code Collaboration and Development Teams (9), and Development Shortcuts. This undermines the organization's strategic decision of using the Scrum method in developing software. We refer to Agile and Scrum for more on how this affects the organization and the product it develops.

In the organization, there is a department objective to have IT and business work closely together and part of that is to incorporate having both business and IT employees within the same team. From the interviews, there has been some differentiation in the answers from business leads and leads on the IT side. When the team structure was implemented there initially where some friction and effort have been made to bring people together and see each other's point of view. This was done through communication and informal education.

There is a clear difference traditionally between both sides and how they operate. In many ways, the issue is considered to have been overcome using communication, and what there now is in place is a larger understanding between them. A stream lead has mentioned, "I can regarding a feature ask both IT- and business-people and get almost the same answer". However, similar conflicts still appear when people come in, either from the IT or the business side, who are not used to working this way.

We find here that the issue will continue to be a problem as employment rotation is inevitable. The same issues will continue to take more and more time from all people involved as the symptoms are being treated, rather than the root causes. The goal of having IT and business in close collaboration is noble, this is handled by the Scrum method in terms of Business Analysts and product owners in the team. Why is there a need for the whole team to be in contact with the business side in other circumstances than specific follow up on tasks where the business knowledge is not in the team already?

One thing to notice is that the prioritization is being made not only on a business level but also on a technical level and we find that the communication of the importance of technical issues is failing. This relating to who is doing what and the hierarchy structure. The organization structure leaves open for decisions being made by employees that are not in the scope of the knowledge needed for those decisions, while the people with the knowledge to do so doesn't have jurisdiction to overrule or not even given the chance to raise their concern on a decision before it is solidified. The consequences of this can be seen in all areas of this chapter as we find that the pressure, that is allowed to be created as a result of IT not being able to "push back", affects the solution to its core in terms of development, processes, quality and the team's effectiveness.

Something that has come up in the interviews is that demands are being pushed from business and that IT lacks push back on the technical side. This is considered to be a reason for the undermining of the solution being technically developed in a sustainable matter following

technical standards and practices. The pushback seems to be expected so that decisions can be made with both IT and Business needs in mind.

A conclusion that has been drawn by people in the organization is that the empowerment of people is low, especially on the IT side, and that the structures of the organization and how the solution is developed in terms of prioritization is not supporting the IT side as they do not have a channel to provide this information to stream lead.

The stream leads are the main decision maker for the full product and mostly communicate with the product owner. It is a problem that the IT side does not have a channel directly to this decision-maker with the power to make decisions spanning the entire solution. Tech leads can of course within the team, work with the product owner on these matters, but if the product owner is not empowered to decide on this then it falls on the stream lead on the business side. Product owners that have taken part in the interviews have mentioned that it is difficult to fully motivate the value of technical tasks. This suggests being a root cause for the view that IT is not pushing back enough. The organization and how it is structured does not allow IT to do it.

We have found that important decisions can be made without the teams that have ownership over the solution. As have been pointed out in Agility and Scrum all communication should go through the Product owner, otherwise, it affects the teams in terms of effectiveness and the benefits of using Scrum. In this, we find that the structure of the organization and the working culture seem to be out of control as any kind of informal communication with the team is accepted and might even be encouraged.

Teams should not need to push back on the IT side as the Scrum method is used and all decisions then are to be made in the team by the product owner that has both the technical, business and other stakeholder knowledge from entities outside of the team. The root cause then seems to be that the Scrum method is not implemented properly in that teams are not self-organized and too dependent on others along with the product owner lacking empowerment on decisions.

Not only the product owners suffer from this. We have further found from the many procedures and issues discussed that certain responsibilities seem to be falling between chairs of different people, one root cause being the management structures and roles that are not tailored after the product development needs.

What we find in terms of structure and how decisions are being made is a way of working that is unique for the organization and is not based on the methods (Scrum and agile) that it markets itself to be using. The deviation from strategic decisions is a root cause for the conflicts that appear. This might briefly be beneficial for the business side in that they get more functionalities done faster in the short-term however is devastating in terms of maintenance and the future delivery of the development teams.

The consequence of the issues found is that the organization does not prioritize technical excellence or quality. High-quality delivery is a requisite for long-term productivity [12], and therefore the organization's efficiency will continue to decline. The consequences of not acting and allowing issues like these to be unsolved, the problems slowly creep in more and more, affecting the development and the organization. This will continue until a decision is taken and then much effort will have to be spent in aligning a larger department, compared to if the decision had been taken from the start. Just the same as with most other technical reasons and decisions to develop IT solutions a certain way. The later an issue or a problem is corrected the more money has to be spent to correct it [6].

We can further relate this to issues in Development Shortcuts when development shortcuts are implemented commonly over the entire technical solution. A root cause for this would further then appear to be the lack of action in aligning the department in the strategic decision of using Scrum. This can however in part also be explained by the rapid growth of the department.

However, if Scrum was properly implemented then this would not be a problem as the teams would be responsible for their own work. Inappropriate organizational structures might still exist, and practices would be happening on a less complex scale, be more visible and easier to fix, as this is the reason for using Scrum [5]. Self-organizing teams that have clear responsibilities (that they do not share with others) are not as affected by organizational structures as much and this would be less of a problem. In that case, we come into the possibility of the root cause being that the teams do not have clear ownership and not being self-organized, something that would allow the teams the freedom to fully take their own decisions. This leading to minimal need for overseeing the practical parts of the team with less imposed decisions from the top.

Summary

- The managerial and organizational structure is ad hoc based, and decisions revolve around the stakeholders and business side first-hand with less consideration of technical tasks.
- The organization's strategic use of Scrum is undermined by the organizational structure and how decisions are made.
- Conflicts that come from the differentiation between IT and business will continue to diverge, even if efforts are done to bring them together, due to employee rotation.
- The organization structure leaves open for decisions being made by employees that are not in the scope of the knowledge needed for those decisions. And the people with the knowledge can be overruled or even unknowing of the decision being made.
- IT does not have the means to push back the way that the business side does. IT does not have a communication channel to decision-makers in practice.
- As Scrum is used IT should not need to push back higher up in the managerial structure as teams by default should be self-organized and any team product decision is done by the team product owner. The main root cause then is that Scrum is not implemented properly.
- The deviation from strategic decisions is a root cause for conflicts that appears between IT and Business. This might briefly be beneficial for the business side in that they get more functionalities done faster short-term however is devastating in terms of maintenance of the product and the future efficiency of the development teams.
- Many other issues appear as well as a consequence for the lack of alignment on how Scrum should be implemented. Organizational structure issues affect the organization more as a consequence of Scrum not being implemented properly. This ties well into

double ownership being a root cause as well as clear ownership work well with product and code ownership and would make top steering of the teams superfluous.

5.3.2 Solution Design

An organization designing a solution will produce designs that are copies of the communication structures of the organization itself [3]. How the organization organizes itself and how the communication structures look like is therefore essential in term of how the solution turn out. This structure is a root cause for issues in terms of both the solution itself and friction between teams.

From the analysis, we have found that one main root cause, for many of the structural issues that lead to conflicts, is the faulty implementation of Scrum that ties in with unclear ownership. Solution design is discussed in Agility and Scrum as well as Collective and Double Ownership on this topic. For the focus of this design part, we look into what can be done to increase trust in the area. This is beneficial for the organization as a whole and as mentioned in Agility and Trust necessary to be able to improve the implementation of Scrum.

For different reasons, it happens that people work against each other, no matter if it is conscious or not [5]. People have a different point of view on how things are to be done. This can be seen in the conflicts between IT and Business as well. Furthermore, there are signs of a lack of trust between IT and other parts of the organization like the business and other stakeholders enforcing control systems and top steering that works against the Scrum method.

From situations that have been expressed from multiple interviewees with different perspectives, both management on IT as well as business, shows the empowerment of the individuals within the teams and their close management to not be empowered enough to make important decisions to the development.

Occasionally it happens that situations, where there is disagreement on something, get escalated to management not only one step up in the hierarchy but more. In the end, the decision might be done by someone who does not have a detailed overview and this person does most likely not have the time to get that detailed understanding for a technical solution or the decision affecting it. Here we refer to the decisions that must be done but for some reason are not done by the appropriate management level. Reasons for this are as mentioned before that the imperfect implementation of Scrum and the double ownership leads to multiple decision-makers both on the IT and the business side.

Teams communicate with their stakeholders regarding what is and what they might need. From a few of the interviewees, there has been discussed that stakeholders when they perceive they do not get the answer they would like, sometimes escalate issues to their managers. Then the managers on the business take it up with the managers on the IT side. Then the managers are dealing with an issue that has already been dealt with, where the detailed knowledge is, and then unnecessarily need to be dealt with again.

No matter the outcome, of the situation above, the company will lose time and effort from the employees. There is most likely a reason that the request was rejected the first time, the request either did not present the business need correctly or it is not a priority compared to other tasks. What happens here is that stakeholders on the business side can raise the issue again and force a solution from upper management. If the product owner already has weighted the business value compared with other tasks then that should be final, as it is in

the product owner's job description to prioritize.

Trust is a foundation that an organization needs in order to implement Scrum properly. One way to enforce trust is to put effort into a common vision that everyone can fall behind. If the vision can be trusted so will then the people and the teams who embody it[5]. Product owners should be trusted to do what is best for their product using both external input from business or other stakeholders and internal input from the team.

In general, an IT team should not need to defend itself in the manner we have observed happening. Business needs to trust IT to do what they are there for. Business value is dependent on the product working well and being adaptable in the future for further changes. How things are done is just as important as why. Therefore, there is business value in having a healthy IT solution and trust within the organization for others to do their part. The longer time goes before something like this is handled, the worse it will get, as distrust feeds more distrust and becomes part of the culture.

Not addressing the influence the business currently has in IT development decisions will highly affect the future of the organization. Misconduct in the teams will be a consequence. For examples, of how external influence affects the teams see Code Collaboration and Development Teams (1, 6, 7, 9, 11) and Development Shortcuts (2, 4). This also applies to the misunderstanding of the purpose of a wave plane.

To overcome distrust is a difficult task. A common thing that is mentioned in companies when there is mistrust is that employees need to be empowered. The empowerment of employees has been a topic of discussion with many of the interviewees. Empowerment is something commonly mentioned in other companies when the issue is distrust[5]. To only empower people in the organization is however not enough as the attitude is on a higher scale than the team level. One part is to improve the skill of the development teams' employees to make sure that they have what is necessary to develop the product in the strategic manner that is decided upon, allowing the business side to trust IT to be skilled in developing the product using Scrum. It can involve Scrum education both on a technical level as well as seminars to change people's attitudes. To be on the safe side one can train all employees in the core topics. A cost-benefit analysis is helpful to find out what the training should contain [5].

Among the IT teams, we find that especially retrospectives can improve as these generally do not follow the Scrum guidelines causing the process improvement to be neglected. The organization is recommended to expect an understanding of the common vision from the business side, and if members from the business are to be closely connected to IT it should be expected that they do not concern themselves with influencing what the team does or how they work. If this is found to still be the case then training is one possible way to deal with it.

Rather than focusing on general Scrum training for business employees one should focus on why it is used and how it should work from the business perspective: how business is expected to act in order to be compliant with the Scrum method that the product and collaboration depend upon. This way IT can trust the business side to be clear and know how to communicate what they need.

Another way to improve the attitude is open communication and convincing people of what is important for the area and not only from their perspective. In having a clear product vision and trust among the people in the area the established processes, models, systems, and practices that are normalized but contradict the product vision can then one by one be solved

and changed [5].

Summary

- Decisions made on the team-level should be respected and management should not allow requests being pushed up the management hierarchy. The product owner is trusted to have weighted the business value against other requests or tasks.
- Trust is a needed foundation for Scrum and should be a high priority. Overcoming distrust solutions include improving the skill of the employees, Scrum education, and seminars to improve attitude within the area. Cost-benefit analysis can be used to find out what to include in educational efforts.
- Within the development teams, retrospectives should be the focus of education as it is the process for how the teams improve their way of working, thus improving all development processes. The business side can focus on the common vision and education regarding why Scrum is used and how business can act to be in compliance with the Scrum method that the collaboration and specifically the solution that provides business value long-term depends on.
- To change the attitudes from mistrust to trust the recommendation is to communicate openly and convince people of the goals and common vision. Furthermore, enforce structures that do not encourage individual performances over the success of the area.

Chapter 6
Discussion and Related Work

In this chapter, we aim to describe the significance of our findings, how these stand in terms of validity, the application, and other results in the research field. Furthermore, in interpreting the processes and results along with exploring future work the study behind this paper is reflected upon. This will bring understanding and an overview of the study as a whole and how it presents itself in this field. The general and specific application of this study is discussed along with validity. Furthermore, processes and results are reflected upon along with related and future work.

6.1 Application and Validity

What has been included in the data collection are issues and problems in the area of the organization being studied. The spread of the interviewees covers the entire area, where all teams are represented at least once. Furthermore, employee roles of all kinds are interviewed, both management, team members, and support teams. The roles that were not interviewed include Scrum masters and Business Analysts. Interviewing these might have brought a deeper understanding to some of the issues but as product owners, tech leads, and agile coach are in close collaboration with these much of the information from many of the interviews broadly covers issues experienced by these roles as well. Therefore, the intended area has been covered as all the teams along with representatives for collaborators have been interviewed.

The content in the interviews concerns possible issues around effectivity and adaptivity. Most of the issues found focus on relation to effectivity and includes many areas that are affecting this. Most of the issues and statements mentioned in the interviews have been followed up and expanded with comments from other roles and teams to increase accuracy, dept, and possible variations in answers. These results, organization description and issue list in chapter 3 and 4, have been written and then validated by employees from the organization.

Rescoping performed before analysis limits the scope to the areas Agility and Scrum, Collective and Double Ownership, Collaboration between IT and Business. Root cause analysis

is focusing on these three areas. From the root causes found the focus of the design suggestions essentially relates to agile implementation and clear code ownership and how they can be applied from this organization's prerequisites specifically. Analysis and design results are in a resonate form where the root causes of issues from the investigative phase are discussed and analyzed along with possible design solutions. Validity lies in the data collection along with the validity of the sources used to compare with the organization, specifically [5], around implementation and the use of Scrum.

The application of the result is connected to the organizational settings and the culture that exists in the organization. There might be applications that are similar in companies that similarly are not satisfactory implementing Scrum or a company using multiple incompatible code ownership strategies, which might lead to some of the communicational and collaboration issues found in this study. These are explained in detail in chapter 4 and by comparing with the organizational description in chapter 3 the application in another context can be determined.

The identification of similar issues that are related to the same root causes as the ones found in chapter 6 allows the application of possible solutions as well, although it is much tailored to the organization in question in this study. With that said, by understanding the organizational setting and why the issues are appearing one can determine the possible application of this study in another context. The issues related to the main root cause of Scrum being imperfectly implemented can be seen in other studies and especially is common when Scrum is being implemented in an organization heavily affected by traditional waterfall-structures.

6.2 Process Reflection

In the preparation of the study, the initiation problem of teams becoming less effective and less adaptive led us to research questions RQ1 and RQ2, see 1. The first research question being answered in Investigation phase 1 and 2 eventually leads us to an organizational description and 60 issues group into 9 areas. There are too many to research in the given scope and therefore a rescope is performed at the beginning of the analysis and design phase. After the rescope four areas are chosen and later become three due to the aim of focusing more and going deeper in analysis and design to achieve more depth.

The investigation phase result was reached through data extraction from interviews and talks with employees. The interview guide used to prepare and perform interviews were initially very close to the actual interview questions asked during the interviews. For every iteration, the more detailed the follow-up questions around the questions became due to our growing understanding and knowledge of both the organizations and the issues that are being found.

Many interviewees received questions that follow-up on what other people have said, either they then confirmed with the same thing or gave a new point of view on the matter improving the context of the problem. To be able to cover the whole collaboration of the solution as much as possible it was from start focus on the importance of issues and workings of the organization being discussed with many kinds of people in different roles. Therefore, the initial nine formal interviews are covering different management and different roles within the teams and also aiming for a spread of representatives over the seven teams and the two

streams that they belong to.

Some of the topics brought up in the interviews were still not fully confirmed by multiple people and whenever it was found that an issue is related to a certain team or role follow-up talks were performed. Then specific questions were asked around these and then the focus was not on following the interview guide. In the end, 15 people contributed to the data presented as the 60 issues and the organizational description. All teams have had a representation of some form along with management outside of the teams and representatives for architecture support and infrastructure chapter that collaborate closely with the teams.

This way of aligning how the organization is represented has been necessary to receive a good overview. On the other hand, the amount of work it contributes to in retrospect makes it a little too ambitious for the scope of a bachelor's degree thesis. The more data that comes out of the investigation the more work goes into data extraction and combining the data from different interviews. Furthermore, extra time was needed for RQ2 to be answered, this motivating rescoping several times both at the beginning of the analysis phase and before the design parts of the different areas.

In retrospect the research questions could have been more focused, however as the scope would have been smaller less varied results would have been found, perhaps less valuable for the organization as many of the issues found are over-arching the entire solution collaboration. The positive part of performing it this way has been that the spread of data broadly covers the entire area (solution).

Another possibility would have been to limit the research questions to a team or a stream. This would most likely not have led to overarching collaboration and product-spanning structural issues. This as the data would depend on the specific part of the solution collaboration the investigation would have been limited to. As is now the results of the study can inspire possible solutions and further work on analysis and improvements on the solution collaboration as an entity rather than a specific team or stream.

6.3 Result Reflection

The results of the study and answer to RQ1 and RQ2 is essentially the issue list that is understandable through the organizational description and the analysis of some of the root causes of these issues along with solution suggestion to handle the root causes to solve issues found in the list.

From the preparation for the interview and from what I learned from my mentor at the company there were some expectations on certain issues. Even as the questions initially were very open and broad, we find that many of the interviewees brought up some of these issues and also added more details. Furthermore, I found that interviews sometimes gave me completely different answers taking me in entirely new directions. This is most likely as a consequence that the broad spectrum of both different teams, roles, and management includes people with very different ways of perceiving the issues. We were initially expecting more issues in the development practices as originally, the understanding was that that the development was becoming more inefficient.

Something that early on was a surprise is how well many parts of the development in terms of detailed technical practices that are done by the team. Many of the engineers and programmer finds that the technical part of the development is working very well. It is

not until collaboration between teams must be done or other external things like pressure from business, overhead communicational work, waiting to be allowed to do testing on the shared test servers and another bottleneck is the approval process to release into production that can take months. Therefore, after a few interviews, it became clear most of the issues came from communication, collaboration, and the organizational structure around the teams. This affecting how development is done in practice. This is a good thing as by enforcing the solutions presented in this study a large part of the issues that the area has will most likely be dealt with.

In the general sense, we find by looking at the analysis and design that can be perceived as the main root causes are the imperfect implementation of Scrum along with the code ownership not being clear. The two main root causes connect to other root causes leading to issues that have been found in investigation phase 1. These affect many parts of the organization, for example in communication, collaboration and code development processed: affecting efficiency, adaptivity, and the relationship between development teams as well as between Business and IT.

In focusing solutions on the main root causes there is the possibility of dealing with many of the issues that exist in the area. In working with three analysis and design parts everything is intertwined and essentially leads back to the main root causes. This was not expected initially as the effort was to analyze and design solutions separately in the three areas. Instead, the areas can be read in parallel and they contribute to each other in that the analysis and design of one area contribute to the next due to the intertwinement of issues and that the root causes found eventually lead to the main two: imperfect Scrum implementation and the unclear ownership found in all of the three areas eventually is affected by the other as the combination of issues effect on the area.

As rescoping was performed the designs in the three areas are much dependent on each other as they are combining a solution that deals with the two main root causes. In the design part in Agility Scrum mistrust is a recurring theme that leads to many agile principles being broken and affects the Scrum implementation within the teams. Therefore, the design in Collaboration Between IT and Business focuses on handling this by suggesting possible ways to enforce trust in the area in both directions.

This study focuses much on the development efficiency and therefore what is not working for the development will eventually affect the business side and the organization in terms of what is produced and expensive it will be to maintain. As is a common theme in development is that whenever development maintenance becomes too difficult the management chooses to redo the solution from scratch will cost the organization to spend even more money [3].

The underlying hope of this study is to clarify the issues that are leading the project in this direction of becoming more and more complex, expensive, and difficult to develop. In understanding the issue from a development perspective, it can be dealt with. Traditional companies that move into IT are many times strangers to how the way development should work, how fragile the development becomes as soon as the traditional ways of working start controlling the development.

Developing using Scrum, or other development strategies is a long-term commitment that requires continuity, discipline, and in the case of a Scrum constant improvement mindset in the retrospectives to receive the benefits from the strategy. Done right the company can create a better solution, work environment, and improve the collaboration that benefits the area, teams, and the company as an entity.

6.4 Related Work

In setting the context of this study in relation to the field and related works two papers have been chosen to be presented and reviewed. These two papers are specifically chosen as the research generally relates to the content of this report and the organization that the results relate to. The first paper by Masood et al (2020) is describing the variations in Scrum implementation and when they are acceptable or not. Chosen as the classifications of issues are related to and possibly applicable to issues found in this report. The second paper by Conboy and Carroll (2019) has a focus on challenges and how to address these in large scale agile framework transformations. Chosen for the implications of an agile-framework change. We have in this report found the current implementation faulty and a framework change is being prepared within the organization. Both paper reviews include a summary of the paper, an evaluation, and the relation they have to this study.

6.4.1 Paper Review 1

Summary

The research problem the paper [9] explores is the lack of clarification on variations in Scrum implementations both by the book and in practice. The differences in how Scrum is to be implemented "by the book" and how it is implemented in practice is something that is recorded in research, in other words, variations.

However, these findings are in the context of aligning research and not usually the focus itself. The contribution to the field includes investigating what the variations look like, when they occur and how they vary from the expected use of Scrum. Five teams in different areas and their Scrum practices are observed and 45 semi-structured interviews are performed with members of these. From the variations found from the investigation compared to the expected implementation of Scrum "by the book", the study presents classifications to determine variation type. This to find out whether a variation is motivated by context or misuse of Scrum, for example, if certain variations stem from a clear misunderstanding of the method.

Variations to be accepted are the ones explicitly described in the Scrum book for reference and necessary deviations that come from the book being vague or ambiguous. Another variation that should be considered accepted is contextual that allows temporary contradiction to the Scrum method. Whenever there are clear deviations, in the form of frequent or ongoing variation these are to be considered an excuse for poor implementation and not acceptable. These are labeled as abuse or misuse depending on whether the deviation is intended or not.

Evaluation

The differences in how Scrum is to be implemented "by the book" and how it is implemented in practice is something that is recorded in research, in other words, variations. The paper investigates a relevant area. The result with the classification of variations deepens the understanding of types of variations. It appears throughout the paper as these classifications could also be used in practice however are explicitly stated in the conclusions to only be used to understand them. This is surprising as the authors' aim is to lay the foundation for future

work and the classifications are specific and approachable. If the classifications are not to be used, then this is misleading for the reader.

As the focus in the article is on roles, specific Scrum practices, and reasons for why they are working a certain way it is surprising that some Scrum variations, for example, retrospective are considered out of scope. This as retrospectives are essential for the transparency and possible improvement on processes and essential to the areas of variations investigated in the organizations themselves. Moreover, the teams that were investigated did not have significate variations in retrospectives, quality assurance, and design and implementation. This makes the study of these teams homogenous. The study data would have benefited from a broader and more diverse representation. This might be an explanation for the authors' reluctance on committing to the classification's broad use in practice.

Relation to this study

The paper is reflecting on a classification that potentially could be used to motivate some of the variations from the standard way of using Scrum that has been found in this study. How [9] investigate the teams in question is semi-structured. It is a similar approach that has been used in this study with the difference being the smaller scope and observation on teams in their everyday work environment.

The classifications can help in determining whether issues in the implementation of Scrum should be accepted or not as future work from this study. The classification of roles in the Scrum team is applicable in our study on issues related to responsibilities on roles and issues that come from these, for example, product owner responsibilities split between Product Owner and Stream Lead. We then find that as they have not been temporary, they fall into the clear deviation classification, should not be accepted and instead dealt with.

Reported reasons from the interviews and observations in the papers furthermore confirm that many deviations that appear are a result of habits from traditional ways of working as well as perceived efficiency. In relation to our study, we have similarly found that when Scrum is not working that traditional process lies behind and works against the Scrum method. Furthermore, when issues are moving up the management ladder there might be perceived that efficiency in certain areas is improving as specific things are escalated. However, this is not the case for the area as the Scrum method is undermined and other (more?) important tasks will be done later.

Furthermore, the results of the paper contain variations around the breaking down of user stories, work-assignment, back-log refinement, and prioritization. This aligns with some of the issues and discussions with interviewees in this study. It would be beneficial for the teams to look more into an effort of streamlining and improving the teams' internal Scrum processes. These very much depend on the individual teams and therefore have been touched upon in the interviews but not investigated further in this study, rather left for future work.

6.4.2 Paper Review 2

Summary

The problem the paper [2] addresses the empirical evidence the how the adoption of a large-scale agile framework is used along with the effectiveness and the challenges of its implemen-

tations. The paper introduces nine challenges and recommendations to handle these when implementing large-scale agile frameworks.

The research includes 15 years of observing 13 organizations going through all parts of adopting agile frameworks, some more successful than others. The results of the paper are based on long-term observations, interviews and internal data, documentation, and tools. The nine challenges identified and recommendations to reflect and handle these are encouraged to specifically be used by organizations considering implementing or are in the process of implementing a new framework. The application of the recommendation is encouraged to be used in a context that combines different employee and stakeholder groups.

In applying the result, the reflection can expose challenges however, the complete removal of some of them might be difficult. A limitation of the recommendations is that not all might be applicable for every individual case, so an evaluation of what applies to the organization side is needed.

Evaluation

The paper is addressing an important part of the research area. This when empirical evidence on challenges of the implementation of large-scale frameworks is much scarcer than more general investigations and detailed inquiries into specific problems. The authors use well-known companies in different industries for data extraction over a 15 year period that provides a significant contribution to the empirical study.

The methods for how the results are provided are generally explained and therefore difficult to evaluate specifically however generally seem to provide a good base for the intended use. Claims made by the paper is clearly stated and exemplified, results are presented through context and citations from interviews. The paper is (appreciatively) very short, and concise. Personally, we would have liked to see a more detailed method description along with more details in how the companies using this paper should rank and exclude challenges.

Relation to this study

While this study focuses on issues, root causes, and solutions the paper focuses on challenges around the transformation into a large-scale agile framework. This is very much applicable to the organization in question that shortly is going through a reform into a large-scale agile framework. The new framework is adapted from one of the frameworks of this study.

This paper investigates companies in a similar context. Specifically, regarding the framework (that is being adapted) used in a heavily regulated context. It is a mention of this framework being difficult to merge with traditional organizational structures, where the development perhaps is not as adaptive as in more generic IT companies. Results in the paper are also relevant for example on how progress is measured, the difficulty of maintaining developer autonomy as well as the difficulty of seamlessly blend software development processes with customer (business) processes. In relation, we can compare this with what has been shown in this study.

When the development in relation to the traditional process has created many issues that are undermining strategies and the work being done on the code base affecting both communication and collaboration. Therefore, many of the issues that are found in the organization today will most likely not be dealt with as a result of the framework change unless an effort

is made to deal with these challenges mentioned in the related work along with solutions for dealing with the two main root causes that are presented in this study. Furthermore, The paper can be used to evaluate the challenges the framework change will impose on the organization.

6.5 Future Work

The area is currently going through a reorganization that will take place after the turn of the years. In the process, certain roles are being empowered, for example, the product owner. This is one of the issues that move the teams away from the implementation of self-organized teams in the old structure. Some of the issues might be solved by the restructuring of the area. This reorganizational process would benefit from being closely observed to make sure that the Scrum implementation is further being enforced in the process.

The reorganization is also in some way addressing the double ownership as new teams are being formed and some things will change. This however is at this point not resolved. How this will turn out will very much affect the development as it connected to the team structure and therefore Scrum as well. There is a risk here that the new ownerships are difficult to establish as part of the issue still will be that features are spanning several domains no matter how the ownership is arranged. These features that require ownership need to be dealt with through for example modulization. Further investigation should include follow-up on if the issues that have been found that remains after the reorganization as well as possible new issues that potentially will be introduced.

The issues in the implementation that have been pointed out in this paper are framed around the issues that have been come forth during the investigation and the focus has been the collaboration, double ownership, and agility. Therefore the analysis and designs are reflecting this. More things that are related to for example the product vision could be interesting to investigate to see if it is as clear as it needs to be along with more investigation into how the stakeholder chapters and approval process framework is affecting the implementation of Scrum. Furthermore, continuous delivery is something that would be very beneficial for the efficiency and adaptivity of the product, how can the area introduce it while the rigid approval processes are in place.

Chapter 7
Conclusions

In investigating root causes, for the teams experiencing loss in productivity and adaptivity (RQ1), we identified 60 issues affecting the development of the solution. The issues have been grouped into nine areas where the three most important ones are Agility and Scrum, Collaboration Between business and IT, and Code Ownership Creating Issues. From the issues found the root causes of these can be traced to two main root causes: Scrum not being implemented properly and unclear code ownership. The codebase becomes expensive to maintain and complex due to communicational and collaborative issues that stem from Scrum guidelines not being followed. The imperfect code ownership furthermore affects the overhead work of the employees along with the solution scalability and complexity to a difficult future (and present) maintenance and expansion.

In investigating solutions, to eliminate root causes for the teams experiencing loss in productivity and adaptivity (RQ2), the solutions dealing with the root causes for the problem revolves around solutions to improve the efficiency and agility of the teams. The solutions include enforcing clear ownership between the teams, improve the implementation of Scrum, process improvements, as well as enforcing and supporting team self-organization.

Based on the results from the study answering the two research questions, we recommend the organization to separate the code ownership through modularization along with improving the implementation of Scrum from the business value providers perspective, to allow for long-term efficient and adaptive development implementation. The reason for this is that the better the technical solution, the easier maintenance and scalability, and the more business value the developers can produce. Therefore we recommend the focus of the organization should be on prioritizing technical excellence, and constant effort to not only follow the Scrum structure but also embracing the mindset and principles of the strategy.

References

[1] Olalekan Akinola and Babatunde Ayinla. An empirical study of the optimum team size requirement in a collaborative computer programming/learning environment. *Journal of Software Engineering and Applications*, 07:1008–1018, 01 2014.

[2] K. Conboy and N. Carroll. Implementing large-scale agile frameworks: Challenges and recommendations. *IEEE Software*, 36(2):44–50, 2019.

[3] M.E. Conway. How do committees invent, 1968.

[4] M. Cristal, D. Wildt, and R. Prikladnicki. Usage of scrum practices within a global company. In *2008 IEEE International Conference on Global Software Engineering*, pages 222–226, 2008.

[5] Maximini Dominik. *The Scrum Culture: Introducing Agile Methods in Organizations*. Springer Nature, 2018.

[6] Bill Haskins, Jonette Stecklein, Brandon Dick, Gregory Moroney, Randy Lovell, and James Dabney. 8.4.2 error cost escalation through the project life cycle. *INCOSE International Symposium*, 14(1):1723–1737, 2004.

[7] Hanna Kallio, Anna-Maija Pietilä, Martin Johnson, and Mari Kangasniemi. Systematic methodological review: developing a framework for a qualitative semi-structured interview guide. *Journal of Advanced Nursing*, 72(12):2954–2965, 2016.

[8] Likoebe Maruping, Xiaojun Zhang, and Viswanath Venkatesh. Role of collective ownership and coding standards in coordinating expertise in software project teams. *EJIS*, 18:355–371, 08 2009.

[9] Zainab Masood, Rashina Hoda, and Kelly Blincoe. Real world scrum a grounded theory of variations in practice. *IEEE Transactions on Software Engineering*, 09 2020.

[10] N. Nan, Donald E. Harter, and Tara Thomas. The impact of schedule pressure on software development: A behavioral perspective. 2003.

[11] David Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15:1053–, 12 1972.

[12] Goparaju Sudhakar, Ayesha Farooq, and Sanghamitra Patnaik. Soft factors affecting the performance of software development teams. *Team Performance Management*, 17:187–205, 06 2011.

[13] Adam Trendowicz and Jürgen Münch. Factors influencing software development productivity - state-of-the-art and industrial experiences. *Advances in Computers*, 77:185–241, 12 2009.

[14] Laurie Williams. Agile software development methodologies and practices. *Advances in Computers*, 80:1–44, 12 2010.

Appendices

Appendix A
Issue List

In the second investigation phase, we focus on the issues that lead to the development being less effective and adaptive. By understanding what the issues are, we will in analysis be able to investigate the root causes and reasons for why many of these problems occur.

The issues are drawn from the interviews and statements or data related to these, see data processing method in 2.2.3 and validation in 2.2.5. Some issues are raised by the interviewees themselves. Other issues are the result of discussions and points made by multiple interviews. The issues are described and then motivated from the source directly or through resonate motivation that initiates from the source. Lastly, we present the wishes the interviewees made regarding the solution.

A.1 Standardization

1. Lacking standardization in many areas

Standardization is lacking across the 7 teams. They are developing a single solution and many of the teams have the freedom to use any processes or tools they prefer. This becomes more of an issue the bigger the organization is. A larger organization needs to organize itself more in terms of procedures to not lose efficiency. Here we are not referring to hierarchal structures, rather taking decisions on strategic ways of doing certain things, from the business value providers' perspective. If this does not happen then the development moves to be more chaotic, affecting both results and the work environment.

The issue has been mentioned by one of the tech leads and has been followed up upon in interviews with architect support and agile support. Lack of standards can be seen at different levels. It regards both technical standards and how things are done. Architect support´s take on the matter at hand: "Giving freedom to so many people and hoping they will establish a common standard is utopic. Imposing something from the top with a certain disconnection to reality could have an even worse effect. A consequence of the reason for many of the

decisions on standards to have been left out is the rapid growth of the department. This would have taken its toll on people and all the decisions that should have been made weren't, as many new people kept on coming in, management structures changing and it just became too much to deal with along with the other pressing matters. Teams are very free to do and work how they want: regarding libraries, the latest and greatest tools, etc. A two-edged sword, too much freedom or too much structure is bad."

2. No common backlog and no common way to present the sprint development

Even though the teams are working on the same solution there is no common backlog for all teams and all teams have different ways of communicating about their sprint. This makes it very difficult for anyone outside the team to understand the state of development.

The issue is discussed with a support team member: "If it is hard for me as support having an overview and close collaboration, one can imagine how difficult it is for outsiders not working directly with the teams."

We can here compare this with the lack of isolation of the team's work. If teams were isolated and only working on their own things, then it would make more sense to have separate ways of doing things. However, this is not the case as the solution is one product, and the code is very coupled and seems to require extensive communication and collaboration. It makes the job of coordinating the development over these 7 teams harder and inevitably will make the development to be ineffective.

3. Product owners have no common way of prioritizing and displaying tasks

Product owners can prioritize and use any method they like in tracking the team's issues, generally, they don't share this outside the team. As there is no common standard of doing this, there is no transparency to other teams and stakeholders that have an interest in the progress but might also have a say on priorities.

The source mentioning this is a product owner. This person works on his own team's prioritization and chooses to have it numbered and the most important going on top. This is a personal working document that in general is not available. We have spoken with the support team that mentioned that the prioritization depending on team might not be what is expected of an agile development team. The transparency especially is lacking as they, as support, do not know how the prioritization works in the teams.

We can connect this with that it is coherent with IT-development and makes it easy for developers to pick the most important issue on the top. However, that strategy is not implemented in all the teams and adds to the difficulty of having an overview of the development of the solution.

4. Teams breaking standard using infrastructure support for infrastructure related implementation and configuration

Some teams do not follow the set standard in the area of having infrastructure and configurations handled by the infrastructure support. The reason this decision was made to have

the support team doing this is that they have specific knowledge of how it is implemented correctly. Here, there has been a clear decision that is expected to be followed. When a set standard is broken, this leads to the expected implementation to move from what is really being implemented, affecting the teams involved and requires extra work if the implementation was not done correctly the first time.

The issue is raised by the infrastructure support mentioning incidents where teams develop these parts themselves and when issues, in this part of the solution, are picked up by the support, support finds that they have to redo it as the implementation was not done right the first time. It is unnecessary to implement something twice and go through the steps more than once.

5. Teams breaking communication standards set for the benefit of Infrastructure support

One team has been breaking standards when directly contacting infrastructure support instead of sending emails. It follows that the issue at hand might not be prioritized along with other tasks on the infrastructure team side and instead done before other things that have been prioritized.

This issue is discussed with developer of one of the teams. According to him, many prefer talking to the infrastructure team directly rather than emailing, that is the practice, and they are generally able to help. Infrastructure support confirms that this happens regularly.

We argue that something else is being done besides the set standard and that it might undermine the planning and efficiency of the support team. There might however be a reason for things being a done a certain way, diverging from the official communication channel, and it would require investigation into why this is preferred, perhaps the current standard could be improved. It might be that more teams are having the same practice.

6. Teams not following communication standards set

It happened that teams are rushing what they are doing and miss to communicate this to the other teams. The consequence of this may differ, however, essentially undermines the teamwork being done around the development of the product, making it more difficult for everyone involved.

An example mentioned by Tech lead: “sometimes teams are not informed of important decisions, for example, a last-minute change made by another team after regression testing before going to production. There are channels for this kind of information.” The result in these cases was that someone found issues in the added parts of the code that had already been regression tested before. This added unnecessary work in finding out what had happened.

A.2 Architecture and State of Codebase

1. State of the codebase

The codebase is tightly coupled, change history is bad. Sometimes merges can be difficult and time-consuming. Motivated by tech lead and architecture support.

There is too much spaghetti-architecture in the detailed part of the architecture, as code has many development shortcuts, coupled code, and technical debt. This has been discussed with three tech leads and comes from them.

A consequence of many other issues in this document. We argue that all of this is a big loss in business value as it makes the solution very expensive, inefficient to maintain and develop as teams. A further consequence will be the inability to develop it efficiently.

2. Difficulties in cleaning up the solution code-wise

Teams have been added to the solution over time. This to adjoin how the different functionalities are implemented and to make sure that functionality is not implemented more than once and do not maintain the same functionality in different parts of the organization. As this is one of the goals, surprisingly, we find that specifically, this seems very difficult to do. Efforts to unify the way certain things are implemented across the teams is very difficult and takes too long. This unfortunately has the consequence that it will not be done.

This issue is raised by the infrastructure support team member. Earlier efforts have been made on their part to aid the teams in this. A minor effort became major. This as the pull request had to be reviewed by many teams and took too long. In the time the pull request had been approved, major rework had been done, approved, and committed by others. Furthermore, as these were error-prone made it very difficult to get a pull request in time. Merging the changes, spanning multiple domains, before more changes were committed was difficult needing approval. The team spent time preparing the merge over and over again. This led to the infrastructure deciding on never doing something similar again.

We would like to argue here that the teams, that are pushed to provide business value rather than taking care of the existing solution, are not given enough time to focus on these issues. And if the reason for having the teams in the same solution, namely that things do not have to be redone in different parts of the solution, why is this strived for? If this goal is strived for, then the developers should be given a chance to incorporate it properly. This will cost. On the other hand, the cost of fixing an issue rises exponentially with time [6], so doing it later will cost more.

We would like to further explore the possibility of micro-services to solve the issue with the solution being so intertwined and difficult to improve structure-wise. The goal should be to separate everything as much as possible.

3. Architecture responsibilities unclear (A)

No one individual or team has an overview responsibility for the solution architecture, not even architects as they are considered a support and mostly concerned with architecture decisions made on a higher corporate level. The consequences of this can be seen as there is no one driving decisions from an architect's point of view directly in the team. This leading to the solution gaining complexity while at the same time becoming harder to maintain.

Raised by a product owner: "There is no clear big picture of how the solution should look like as a whole. More clear and communicated architecture would be nice". From the developer's perspective, they only contact Architect support for things that are not solved among the teams themselves. As a developer mentioned: "Architect support is contacted for larger things and mainly when we can't solve it between the teams".

Our interpretation of this is that compared to how the organization structure looks like today, there is confusion on who is responsible for what. If everybody has responsibility, is there really anyone responsible? This can be seen in other areas as well with responsibilities between Product Owners and stream leads for example.

4. Architect responsibilities unclear (B)

Architect and team lead responsibilities on architecture are unclear. The architect support does not have the possibility to be part of all the detailed architecture in all the team and it varies how proactive tech leads are in taking lead on the detailed architecture.

Discussed as a follow up with a manager higher up in the organization: "The idea is that they work together, but I see that it is too unclear where one role stops their work and the other one takes over. It is also a big difference between the tech leads and how skilled and interested they are in architecture so that is another challenge"

A follow-up question on this is: is it reasonable that tech leads have the responsibility of architecture when the ownership strategy makes no one fully responsible?

5. Confusion and contradiction on Architecture responsibilities

Coordination of the architecture between the teams seems to be lacking in that they are not aligned. The consequence of this is that the solution will not be able to incorporate the common direction for the solution that is expected and needed. It also seems to be some confusion on the matter. There are contradictions on what has been said from tech leads and others on who is responsible for driving the architecture. One says tech leads are part of taking responsibility while another says the architect support is driving the decisions.

Architecture support has mentioned: "Not all tech leads have understood that they also are architects". Stream lead has also mentioned: "The tech leads need to coordinate across the squads that what they are building actually fit with the architecture". Another stream lead expects the architect support to be the one that should set the direction.

This aligns with issues of unclear responsibilities, as well as consequences for common directions across the organization not being set. Other people than team leads expect those team leads to take action even though it is not fully included in the tech lead responsibilities. This is an example of when the expectation of management is not aligning with how the teams actually work.

A.3 Traceability

1. Technical documentation lacking

Lack of technical documentation means that all aspects of an application are not available for employees and many of the benefits of having documentation are lost. The knowledge of how the application is supposed to work will either not be transferred fully (or even at all) or will be done so in an inefficient way. This can for example make it difficult for new employees to get up to speed in their new job or when an employee starts working in another team's domain, as it frequently occurs in this organization.

The lack of documentation of the technical solutions has been discussed with a developer in one of the teams and a member in one of the support teams. Most of the solution has no documentation. However, new features are being documented by Business Analysts. These, unfortunately, lack technical details consequently. Outdated diagrams are referred to in some parts of Confluence.

2. Missing code history and overview difficulties

If it is difficult to get an overview of the code and its history. This essentially affects all work that employees do in understanding the code, solving issues, improving code structure, etc. It affects both product quality and the time employees spend on these aspects.

This issue has been discussed with three tech leads and one developer. The history is difficult to go through as there are many commits pushed to the repository. Getting an overview is hard because of the size and the number of solutions within the repository. One tech lead has mentioned that some history is missing as Team Foundation at one point was used alongside Jira. Another Tech Lead has commented that information is missing only in certain tools used and that all the information in their team is viewable in Git. Moreover, he comments that a reason for some information not being found is dependent on the tools that the developer uses. Many use embedded tools instead of Git.

3. Lacking traceability between Jira task and the code changed

There is no traceability between a given Jira task and the code changed other than the commit message. The effect of this is connected with the issue above. It means it will take developers a long time to solve the issue at hand, because of the time spent on finding the code that has been changed.

This has been mentioned by a member of the support team and a tech lead. To find the code that has been changed, one must go into the repository and browse through the commit history. There is a plugin that instantly presents the code changed in the Jira task. The plugin is used in other parts of the organization but has not been paid for by the Atlassian team for this department, so it is not used by the 7 teams developing the solution.

A.4 Collaboration Between IT and Business

1. Collaboration issue between IT and business (A)

The communication on the stakeholder/business level is not clear or even faulty on issues that they want to implement. It happens that they change their mind after efforts have been put into developing a solution. This affects the development, in the aspect that unnecessary time is put into something that might have to be redone later. Developers need to know the requirement, which should be checked for accuracy beforehand so that they can direct their resources properly and develop effectively.

The issue has been raised by a tech lead and presented as follows: “Sometimes stakeholders (business) do not know what they ask for. This is a huge thing for the team where they really try to make stakeholders understand beforehand. It has become essential to put it in writing”. We have confirmed this in other interviews as well. How severe the issue is and whether it

is a problem seems to depend on the team, its stakeholders, and the environment the team operates in.

2. Collaboration issue between IT and business (B)

There is a misalignment in collaboration between business management and IT where there is a lack of understanding of what the other parts need and how it works. This affects the overall development efficiency in that the whole organization is not working towards the same goal. Can they collaborate better, if they do not understand each other's work and what is needed for them to succeed?

This is mentioned by one of the tech leads: "Seems to be miscommunication between how management thinks the teams work and how they actually work." The management on the business side also mentions that sometimes "IT goes in another direction (on tasks) than business has set. This ends up being something that the business can't use".

What we derive from the business manager's comment is that it has a dependency with (A), in terms that members of IT have raised that requirements aren't clear where the result can be shown here as well. From what can be assessed from the overall interviews, is that this seems to be a consequence of miscommunication on how things are to progress and that something needs to be done here in making both sides come together as one and work in the same direction or perhaps separate their relation when that is not possible. As business is setting the direction for and managing IT, this becomes an organizational decision that negatively affects the development (see other issues on collaboration that can be found in this document). One might think that the development decisions should be left to IT. Business is their client and should prioritize the deliverables they want and then perhaps stay away and leave the development teams to do their work, to keep the development as effective as possible.

3. Lack of discipline in commiting to requirements leads to requirements changing

There is a lack of discipline concerning requirements as they are allowed to change, even during development. Starting to develop something and at the same time changing requirements is wasting the time of the developers, affecting their efficiency. Changing requirements that are not done properly from start undermines all the hard work that has gone into planning and it makes the estimation less accurate.

It is an issue raised by one of the tech leads. The effect of this issue, as he sees, depends on the progress reached. If the development has not started, then it is usually not a problem. When requirements change during the development phase, it is frustrating for the team, things take longer, and it interrupts the sprint. Other interviewees have been asked to give their point of view on this issue. One of the support members with much experience of working in the area and now supporting mentioned the following: "Business wants so much to be done. Therefore, they make proposals. They usually do not know the consequences of what they ask for." This can be compared to issues where IT is not pushing for their point of view to be heard.

Team success example

Another team mentioned that it is not allowed within their team to change tasks within sprints, if a change is required then it is pushed to the next sprint. From what is interpreted from the interviews, it seems that the team that does not allow for changes has team members, including the Product Owner, that are setting boundaries for this. This is good, as it handles the problem within this team. However, what is the causing issue is that business can deviate and if not handled, then there is a direct problem for the development team that will require resources where it should not, either on effect on development, most costly for the solution, or extra effort to handle it.

4. Collaboration issue between IT and business (C)

There is inequality between IT and business on how much both sides are heard in terms of communication regarding issues management, prioritization, and development. This affects the product in the aspect of important technical tasks getting down prioritized.

One of the stream leads formulates that the voice from IT is not heard while Business is pushing on and on. “More push back from the IT side to keep working on technical issues and solution, as well as the tasks that business wants to implement, so that we minimize technical debt and keep good practices”

5. Collaboration issue between IT and business (D)

Stakeholders directly collaborate with development teams while not understanding the approaches that are used in Agile complex technical environments. The consequence is the strategic element of using Scrum and Agile work methods are undermined in its core. This affects relations between IT and stakeholders and results in time being taken from development in trying to communicate to someone who is not used to or don’t understand this way of working. This affects the team’s environment, and therefore their work when stakeholders do not understand why agile practices and Scrum is used in IT. This is an issue that seems to be spanning across the whole solution, affecting many of the teams.

This issue was firstly discussed with a tech lead that mentions: “stakeholders sometimes think that landmarks in wave-plan or even sprint plan are hard deadlines”. This is further discussed with and motivated by a stream lead that adds that Business Analysts then think that the sprint deadlines are hard and promised results. This is done without consideration to more critical tasks might have to be done instead, and therefore, other tasks get pushed aside. He adds that this becomes a very stressful environment for the teams. A Product Owner from another stream agrees that there are times where the stakeholders do not understand the uncertainty of the plan on when tasks are to be developed and released. Instead, they took it as something certain.

Stakeholders sometimes have expectations on IT in terms of delivery, which is not reflected in the reality of how development work. How much IT can perform can only be compared to how IT has performed earlier. One of the Product Owners describes it as “Business (stakeholders) want things to be done faster than they are. They want so much done. Therefore, they make proposals. We argue that it should not be on the teams to educate on this or should need to do the explanation on certain tasks in communication with stakeholders. If there is a priority system, then this should be determining the order a task is developed. Preferably this could be communicated or educated by someone else so that the team’s effort

could be spent on their tasks. The less disturbance the teams experience the more effective they will be in producing results and business value in the solution.

6. Collaboration issue between IT and business (E)

IT and Stakeholder sides are misaligned in the collaboration and lack mutual understanding for the other's situation and working environment. IT appears from the business side to not understand that some things have to be done or it will have consequences for the organization.

This was raised by a Product Owner and a member of the support teams, they added that many of the challenges that IT struggles with are really a problem and that some of these processes that halter the development progress are self-imposed on both IT and business-side. Issue closely related to (B).

7. Technical tasks value and priority

The IT capability in communicating the value of certain tasks is lacking, while Product Owners and Stakeholders seem to not understand the long-term consequences of systematically down prioritizing technical tasks and technical business value.

The architecture support frames it as follows. "When technical tasks are down prioritized, then agility and the expansion of the system will suffer. An expansion will be more expensive. We will have a more rigid system, more complex and we all know complexity kill. Possible solution: it is all about understanding each other. Certain things are easy to understand while some are not. Better language and explanations from the IT side to make other parts of the organization understand. And stakeholders need to listen!"

8. Misalignment between IT and business

There is a difference between business and IT goals. It shouldn't be as such, as they are both collaborating on the same solution. What harms the IT solution will eventually be noticeable for business in terms of how expensive the development and maintenance will become, as well as the stability of the system.

One Stream lead was asked to comment on this issue (Development shortcuts: examples): "We expect the tech leads and architects to give info on possible solutions so that we can make an educated decision. As a Stream lead, I am not strict about how things are built. If I only know the time perspective, I will take the fastest solution possible." The member of the support team commented, that this shows the duality between the Business who want the product in production as fast as possible IT who wants the most robust and maintainable solution". Another tech lead adds that if the tech lead is not speaking up then that is not considered in the process.

From this, we can conclude that in a case where stream leads are taking decisions on how something gets developed it fails if they are not given the proper information. At the same time, Product Owners have stated that it is difficult to motivate why technical tasks are important over business tasks. This is a contributing factor that makes the issue more severe as there isn't, in general, an established collaboration between Stream leads and tech leads. It generally goes through a middleman, the Product Owner in this case.

9. What is maximizing business value?

It is not clear for the stakeholders what business value can be found on the technical side. The architecture support mentioned that, in another role, it was very difficult to take out time in a sprint to do technical tasks. This as business wants to maximize "business value" only and have to ability to do so.

We would like to argue that having a system that is cheaper to maintain when making changes as it has good architecture and less technical depth, is a business value as well. In the long haul, it will lead to more traditional business value being added faster as the development will be more effective.

10. IT is missing communication channels, and technical information is not reaching decision-makers.

Managers on the business side seem to not have enough information when making decisions when they are above Product Owners in the hierarchy of the organization. This will lead to decisions being made without the technical perspective in mind, affecting the solution and unintendedly make it more expensive to maintain in the long run.

One stream lead mentions that if tech leads and architects' info on possible solutions is not provided, it is hard to make an educated decision. If only the time perspective is known, then the fastest solution will be chosen. IT experts are expected to speak up on technical issues without having a proper channel to decision-makers, here referring to the Stream leads that mostly communicate with Product Owners that are expected to give the teams technical perspective without having a technical background. Product Owners have in turn mentioned the difficulty of presenting arguments for technical tasks, due to lack of understanding.

This is regarding decisions being taken above Product Owners in the hierarchy. We derive that, if decisions are being made on a higher level than Product Owners on the business management side, then it becomes very difficult for the IT to speak up on decisions that are being made already and going in a bad direction.

We argue that overall, it seems like there is a division between IT business, being two separate organizations and lacking transparency in the process. This while there is an ongoing effort to have them work together. It could be worse, probably, but it also could be much better.

A.5 Code Ownership Creating Issues

1. Double ownership (A)

Wrong ownership of features is given to teams because some teams are under more pressure than others. When this happens the team, that picks up the task instead, suddenly has a feature to maintain in someone else's domain. This affects the team's efficiency in terms of producing results in a long term, as the work overhead needed increases.

This has been raised by a tech lead and confirmed by tech leads in other teams. Following is claimed: "It happens that the feature is not owned by the right squad as new development features are given out and prioritized in the sprints. So, if one team knows the codebase but is busy another team might have to pick it up". Another tech lead mentions that it has also

happened that features ended up in the wrong domain, but the team should still take part in the review of the pull request. A Product Owner added that there also is no clear structure on what team should pick up which the task. The work overhead consequence is motivated in (B).

From this, we can conclude that the structure and the ownership are most likely causing the teams to spend unnecessary efforts in working around these, either because the ownership falls on a team without the knowledge of that area or the team has to deal with that someone else owns something in their domain that shouldn't even be there. In other words, the original team that should have gotten the issue either spend time on introducing the other team to the code base and both teams will forever after, spend time on reviewing and aligning decisions on this part of the codebase along with any refactoring that might be wished to be performed, for example cleaning up technical depth.

2. Double ownership (B)

As some teams have large features on other teams' domains, there is no control over how this affects the team's ability to maintain the product. It is more difficult and takes more resources to make changes to code that is co-owned with others.

Teach lead brought up the issue as follows: "Many large features in our domain are owned by other teams. This is not considered or stated anywhere". There are features and code in the domain where the owner does not have an overview of it. Therefore, it is difficult to maintain the solution properly as it requires a lot of collaboration and efforts from other teams as well. A developer in one of the teams considered to communicate well has pointed out that if there is friction, it usually happens regarding something that is partially domain-owned by one team and partially feature-owned by another. Someone decides to things a certain way and then others do not necessarily agree.

3. Double ownership (C)

Feature and domain ownership strategy creates issues in the sense that pull requests containing code that do not live up to standards of what it should look like. This has the consequence of culminating technical debt and other work overhead, causing the teams to be less efficient and adaptive to changes.

When another team forces this, it is left to the domain owner to deal with the added cost of maintenance. The issue is discussed in one of the teams where it constantly occurs that other teams put bad code into their domain, as the teams are under pressure themselves from their stakeholders. The tech lead concludes that the double ownership can only be handled if pull requests live up to the expectations, in terms of where the feature belongs, that the code is done well, and there is no pressure in merging features that do not live up to these standards for whatever reason.

What is concluded from this and what can be synthesized from the interview discussions, is that the double ownership fails as teams are being pressured, development misconducts are forced, and regularly has become expected in consensus as a way of working by managers and teams.

4. Double ownership (D)

The ownership strategy is making the development inefficient and making the scaling difficult as there is no clear ownership.

This is a consequence of the issues above (A, B, C) mentioned by one of the tech leads that "if more than one (team) have ownership then essentially the result is that none of the teams really own the solution".

5. How issues are handled from double ownership and very coupled code

As code is very coupled and ownership is unclear, developers within a team and outside teams are constantly in need to "bother" each other making changes in each other's domains.

The architect support mentions that change history is bad and when merges occur, it is a painful process, and it is the individual responsibility to handle it. There have been issues and it has happened that people, not knowing the code very well, tried to solve it and this is very dangerous according to architecture support. As a developer, it feels as if you are working in a very large repo even when you are doing small changes.

6. Double ownership should not be part of the strategy for the organization

Teams are not working in isolation. This highly affects the efficiency of the development, as the more people are involved, the more over-head work will have to be performed on a daily basis in terms of effort regarding organizing, communication, etc.

We can conclude this from the interviews and the effect that it has on the teams. A Product Owner has mentioned that it happens that one team gets affected by another without knowing about it and that it needs to be communicated before. We would like to argue that it is unnecessary from start, that so much time is going into this handling of an issue, that could be avoided if teams worked more in isolation from each other and had fully separated ownership responsibilities.

A.6 Code Collaboration and Development Teams

1. Collaboration issue between Teams (A)

Collaboration between teams in the different streams is sometimes lacking in the sense that if teams are helping others with things that are not within their sprint, it is not accounted for. This makes some not willing to help and other teams then take longer to solve the problem or end up solving it improperly.

Stream lead describes it: "Sometimes difficult in the collaboration between teams in the different streams. We have succeeded in creating visions for the streams, however, we perhaps lacked to articulate that we are not a success if only one of the streams is completing the deliverables. Especially when time pressure is up, we see that barricades come up".

2. Collaboration issue between teams (B)

Communication between teams regarding preparation for the sprints can improve and processes for this are lacking. The consequences of this affect the benefits of agile ways of working and the Scrum system in place is undermined making teams less effective. This is related to (C) below.

Product owner comment on this is that teams can improve in knowing what other teams are working on early on. The later it is discovered that something that someone else is working on is affecting you the more difficult to take it into account.

We argue here that one way to solve the issue is as the Product Owner suggests. However, this can also be a way of treating the symptoms rather than the underlying issue, referring to issues on Double ownership, or it might another reason altogether. The consequences of handling the symptoms are expensive in overhead work required by employees and therefore worth looking into.

3. Collaboration issue between teams (C)

The effectiveness goes down the more teams are doing development in other domains, as learning time and extra coordination must be put aside by the developers. This is a consequence of how the collaboration structure looks like.

The issue raised by a Product Owner points out that, when all teams have one solution they are working on, as it looks like today, you do things in domains other than your own. This Product Owner gives an example of this, where the planning was not close enough to reality, and much more time than expected had to be put in by the team, that was given a task in someone else's domain.

Here we observe that the time of the team looking into the issue is spent along with the team owning the domain.

4. Friction between teams (A)

Friction between teams occurs on multiple levels. For example, when some teams are guarding their code harder than others, then friction can occur between many of the counterparts as the issue is escalated up the ladder. This kind of thing happens all over the solution.

Raised by a tech lead, who says: "Some squads are protective about their code and do not want others to make changes in their domain". According to a stream lead, this happens across all teams, even those who are part of the same stream. If something needs to be pushed out and isn't fixed immediately, that can cause friction between two developers, tech leads, or eventually product owners.

5. Friction between teams (B)

Teams have overhead work that is not considered in the plans and estimation or the team's performance. The consequence of this can be that some teams are reluctant in helping each other out and collaborate.

A Tech lead describes multi-team tasks in the backlog: "Many of the tasks in the backlog require collaboration and coordination between the squads. And if one is doing 80 % and another 20%, the one that isn't taking lead on the task, might have to postpone other tasks that

are not as critical". A Tech lead has mentioned that some teams are not always collaborating as well as others would wish, leading to that their team has to put a lot of extra code into their domain as they are more open to it., see (A).

We would here like to argue that this happens as a consequence of the code being coupled and the code ownership does not belong to one team, see issues on double ownership. This will take more of the teams' time to collaborate, will be harder, and require more effort to plan the development along, with requiring more flexibility than stakeholders seem comfortable with, as there are efforts on the IT part to make sure that the plans are not considered deadlines by external stakeholders. Furthermore, we would like to add that the teams that are not as collaborative, do not do this without reason. Rather, it becomes a way to protect the integrity of their teams, plans, estimations, code base, and sustainability. This as the teams operate in an environment that has many issues structure-wise (where some can be found in this very document) and that leads to the teams becoming more inefficient the more they collaborate with others. This relates to all issues under Collaboration Between Teams and Collaboration between IT and Business.

6. Pressure and development shortcuts

Teams want to go with shortcut solutions whenever there is time pressure.

This is raised by the Architecture support who adds that sustainable solutions should be advocated for as much as possible.

7. Development shortcuts

Some teams are being pressured by hard deadlines promised to business. This causes them to put code of poor quality into theirs and other team domains according to Tech Lead. Development shortcuts also happen when teams disagree on a solution.

A Tech lead gives two examples where more than one team was involved, and the collaboration has faltered. As a result, a decision was made by the management to do the solution that took the shortest time: "In one case (within one team) the architects drew a solution while the squad presented a quick solution for the same problem, then the business decided that they needed it now, and chose the quick solution. In another case (two teams) the squad was ordered to do it the wrong way due to a disagreement between the architects"

A developer mentioned on another topic that it has happened that sometimes the architected solution does not solve the problem, that sometimes architects become detached from reality. This might be related to why the result in the situation above was handled as it was.

8. Plan milestones are mistaken for hard deadline

There is confusion regarding plans on scope of delivery, being plans, and not hard deadlines. The department is operating in an agile environment in terms of sprints and development but is surrounded by a waterfall model in terms of deadlines from the business side. A Waterfall-model is used when setting the budget and annual goals.

According to a Tech Lead, this, mixed with Agile sprints, seems apparently to create confusion on why tasks haven't been finished. Usually, the case is that after sprints are finished, some of the original tasks in the sprint are not finished as other more critical tasks that were

not scheduled, comes up. Stream lead comments as follows: “We do what we can to mitigate in the quarter (wave) that the plans are not fixed as impediments occur can happen during the sprints. We should be clear on the first deliverables but as we progress further in the wave (plan timeline), the uncertainty goes up.”

We argue that the issue happens in a way that is difficult to control and that more should be done to make sure it is contained. See Issue on Hard deadlines and pressure on teams.

9. Hard deadlines and pressure on teams

Putting on hard deadlines happens commonly, and pressure makes teams less Agile.

A Tech lead says the following: “Demands on many tasks makes us less agile. It might be that too much is promised to the business by the management. We should not say it can be done at a certain date while working Agile.” A Stream Lead has been asked about the issue as well and adds that certain things have to be done and agrees that sometimes there are hard deadlines: “We are working in sprints and we do have fixed deadlines, we also have fixed requirements. These do not necessarily mix. It’s hard to combine the Agile ways of working with the “real” world and many times Agile principles are compromised. If it’s a regulatory requirement, then there is a hard deadline “. A Product Owner mentions that there aren’t often hard deadlines on tasks within sprints, rather it is a larger milestone. There is one coming up in a few months.

10. Code collaboration

Teams are communicating ad hoc regarding the development. This affects the risk for mistakes, miscommunication, and misunderstanding. It will also affect newcomers and how fast they will get up to speed into their work as they will learn by doing, including making mistakes, rather than being introduced into a structured way of working.

A Tech lead has mentioned it as “Not all teams agreed on a conduct regarding communication”. A Developer, when asked about communication, mentions that people rely on experience to know whom to communicate with regarding some issues.

11. Kernel not used

The existing kernel that has been developed is not used! A kernel is a shared codebase with features that are generic and can be used by multiple teams that need a certain functionality. The kernel is supposed to provide functionality in one place so that solutions of the same functionality are not maintained in different places. This would lower maintenance and development costs.

Raised by a tech lead who adds that the reason for the kernel not being used, is the rapid flow of issues the teams take on. An effort hasn’t been done in moving and allowing the teams to take time to implement the kernel into their code.

We argue that this issue exists as technical tasks systematically are down prioritized.

12. The way from development to production is too long

The approval process is taking too long and the way it is done is inefficient, as everything is treated generally the same, no matter if it is a simple or a more critical change. This does not

concern reviews rather bureaucracy and the extensive process of code going into production.

A Tech lead states that it takes too long to get something from development into production: “A semicolon takes weeks to get into prod”. This has to do with the extensive review and approval process that is many times slower than the actual development of the product.

13. SYST-environments

There are too few environments, only one SYST environment used by 7 teams. This halters development and testing.

SYST is under freeze the days before release and according to the tech lead, some teams often have prioritized things to test, meaning that often it cannot be used. Another tech lead agrees and adds that in the last year the freeze periods, when regression testing was done, had more than doubled. A third tech lead mentions that regression testing is done manually, that it takes too long, and most likely it is not very fun for the ones doing it. The consequence of this is, according to the third tech lead, that teams cannot test and move forward in integrating with other solutions. Comment: There are plans in motion on adding more SYST environments.

14. Is communication a solution for everything?

There is a bad foundation for scaling the solution, affecting the solution in the long term and the organization’s intention of bringing more teams into the solution repository as well.

A developer has mentioned that the solution to avoid issues (that stem from double ownership, big pull requests, and code dependencies) is communication. We can then observe that this is very common in the sense that everything that might be considered an issue is handled by communicating more between the teams. This will lead to, that the bigger the solution the more of the employees’ time has to be put into communicating with each other, taking time from developing the solution. In other words, extinguish fires instead of working on preventing them.

15. No filter in production release

Anyone in any of the teams can release code to production from master before regression testing is done. Then there is a risk of a mistake being made, or malignant action performed that end up in production.

Raised by tech lead: “Technically it is possible to move a branch to release after a feature is merged to master before regression testing is done. This is not procedure however there is nothing in place stopping it from happening.” Support team member adds that freedom has a cost and that more rigidity of the process might slow the process of getting things from development to production even more.

A.7 Development and Shortcuts

1. Temporary shortcuts (technical issue) tends not to be fixed

A lot of code is tightly coupled as a result of shortcuts taken when a lot of requirements are coming in. It is then hard to fix the structure and the architectural issues as these tend to be moved lower down on the backlog. There seems not to be a clear incentive to keep up with technical tasks. The consequences of this are that the solution will become more complex, will cost more to maintain, and fixing it afterward will be more costly than doing it right the first time. This as the cost of errors escalates over time in a project [6].

This is an issue raised by the tech lead. Another tech lead agrees that unfortunately a proper solution is in general not enforced to replace the quick-fixed (development shortcut) one. The decision lies on the Product Owner; if the tech part of the team is not disagreeing and if it involves other teams and the issue is not solved on the Product Owner level, then the issue moves up to the Stream lead. A Product Owner adds that an effort is made to weigh how poor the quick fix is against proper solution and also consider the time: "We have redone quick fixes before. We went with a quick-fixes and used them until we figured out how to do it properly (sounds like prototyping and should not be a permanent solution). However, in general, putting in place the correct solution is down prioritized." This is probably a consequence of the business setting the tasks and prioritizing those which gives customer value, according to a Product Owner: "If a feature has to go into another domain and is too big, then there is a re-evaluation going on. If product owners cannot solve it, the issue moves a step up to the Stream Lead. In wave planning (big room planning), we try to find out who is dependent on who, so that time is set aside for helping others."

A stream lead was asked about this and added that many times there are conversations about making a better solution later, but that is not followed through with: "We say that we do the quick-fix solution in one month and then do the proper one after that during 3 months. However, what usually happens is that we have the first one working and then move on to the next task".

Architects support comments that: "it is always a problem catching up with technical issues. As soon as the business value is received, then the stakeholders are happy. Iterations need to set aside time for technical tasks".

We argue here that there should exist incentives and reasons for keeping up with technical tasks. However, this needs to be clarified in terms of costs, of not doing those, for the decision-makers. Every time a shortcut is implemented, the development takes up future costs of maintenance and also for further developments. Somehow, a mentality of pay now or more pay later should be encouraged. This should be considered in the decision and prioritization process.

2. Development shortcuts are common

Developments shortcuts are common, and it gets worse when teams are under pressure. This affects the solution integrity and maintenance cost in the long haul.

Too often, it happens that shortcut solutions are pushed rather than a good solution codewise. This was raised by a tech lead who adds that some squads have been really pressured and then have made some bad solutions or taken on debt as a consequence. According to this

tech lead, it should not be pressure on merging features that do not live up to standards.

We observe here that there is a lot of future costs that have been taken on when allowing development shortcuts to be common and not fixed. The cost will show in terms of maintenance, effectiveness in the teams, and functionalities of the system. Essentially the mentality of pay now or pay more later.

3. Temporary shortcuts are maintained

Developments shortcuts that are supposed to be temporary are maintained and even extended as new requirements are coming in. Because of this, there is extra work in maintenance as it is not the intended way of loading the information and managing the information is done manually as a consequence of this. It is a never-ending story that drains the team's effort.

This issue is raised and discussed with tech leads and architect support. Some teams are more heavily affected than others.

4. Encouragement to approve bad pull requests

Teams have been pressured by the management to approve pull requests that the team does not consider good enough for approval. This undermined the point of having review filters, as the goal of these is to protect the codebase from bad code. It connects to the issue of coupled code. The system allows this in that when stakeholders are not happy with the answer they get from IT, they can escalate the problem to the closest management and then the next management level, until someone decides to overrule the initial position that came from the development team.

The issue has been discussed with the support team and the tech leads. A tech lead makes an example, mentioning one of the team's domains, where other teams have implemented features with a lot of shortcuts taken, and pull requests forced to be accepted that have a lot of poor quality code in it. This, as management, say: "we need this feature now". As a consequence, this team has a lot of poor quality code and features implemented. Some features are very large, and they aren't owned by that team but others, making it even more difficult to fix it afterward. For more on this, we refer to issues in Double Ownership.

5. Pull request (A)

The review of pull requests is sometimes not done by the person who knows the code the best. The owner of a domain or feature must approve and any person in the team can be an approver, according to the tech lead. The review of pull requests usually takes too long, although that has improved.

The issue is raised by a tech lead. The Member of the support team has added that it might be very common for tech leads, as they are leading the team.

We can observe that review filters are there to handle code that is faulty or not being up to standards and protect the codebase from careless changes. This is undermined when reviewers don't know the code, they then approve code changes they don't know.

6. Pull request (B)

Reviews take too long. This has been improved in terms of new channels of communication where things regarding review can be communicated. This is an example of handling the issue with communication.

7. Pull request (C)

Pull requests are too large. This adds to the overhead work the development teams have as branches become more long-lived. This is not following the Agile principles, which is part of the area strategy.

This is raised by a tech lead and discussed with others. This could be a reason for (B). According to a developer, there have been efforts put into splitting them up as much as possible, but in many cases, a review contains only one feature and is difficult to split into smaller pieces.

A.8 Prioritization

1. Too many prioritized tasks

Too many tasks are prioritized. This essentially undermines the point of prioritizing issues.

The issue is raised by tech lead that adds that this can be viewed in some of the team's sprints on confluence as well.

We argue that prioritization works best if there is a clear prioritization list, as this makes it very clear what should be worked on.

2. Product Owners' job of prioritization is not clear and therefore more difficult than necessary

It is difficult for a business person to prioritize technical tasks and often, they get down prioritized. This leads to uninformed decisions when a Stream Lead or higher management is taking decisions, as their source is only the Product Owner, that is the person they communicate most with, and not teach leads on the IT side.

This is raised by a Product Owner: "It is difficult to prioritize tech things for a businessperson. I rarely completely understand the technical improvements. And it is difficult to motivate tech stuff over business requirements when this is done 'upwards'. It is easier to convince when the tech lead says we must do this now otherwise the system will break down. We try to fit it in here and there, but it will also be down prioritized if there is other stuff to do."

We observe here that it happens that decisions are raised upwards and the technical knowledge might not reach a decision-maker higher up as tech leads communicate mostly with their Product Owner.

We argue that Product Owners should have the last say on prioritization in the team's area, not people above in the hierarchy. This, as they are close to both their Stream lead and Tech leads, getting both the business and technical input. However, this works more efficiently if the domains are isolated from other teams' influence.

3. No solution spanning standard on prioritization on value against cost on technical issues

There are no guidelines for prioritization in terms of weighing cost against value. Product owners, and other decision-makers, are not provided with information on the costs of down prioritizing technical tasks. Therefore, they are not considering this enough in terms of prioritization. The prioritization is mainly regarding benefits for the business or the case of severe technical tasks.

Following is a Product Owner's own reasoning in weighing requirements: "Legal requirements go on top. Other business tasks underneath that and then the tech. It is difficult to convince the stakeholders on the business side. The stream lead is in close collaboration with them." Example of a case: "If there is a new feature against a technical task, I identify business needs so the new feature will go on top". Another case: "Development added functionality to an already implemented tasks compared to a technical task then the last one usually wins. The closer we are to a severe technical dept the more we prioritize it". This Product Owner points out that it is about the benefits as he sees it being the highest. When the new task is added then, things to be processed are identified, and the alternative is to do it manually and costly for the organization. That is why it is a higher priority; the benefit is high. When a functionality already exists then the benefit is lower if the task is only a tweak of the already existing functionality.

A.9 Agility and Scrum

1. Incorrect implementation of help processes

Estimations, sprints, and plans are not used properly. Too much is expected to be solved by the teams without having laid the groundwork. The consequence of this is that the expectation of the teams do not reflect the reality of the team's capacity, resulting in pressure on the delivery, leading to shortcuts being taken. Closely related to issues in Collaboration Between IT And Business.

A Product Owner points out that the team works a lot and has a lot on their plate. The team is part of the planning, estimations, and is part of this process in the discussion of what we can do. It is suggested that estimations could be done more conservatively.

What we observe from the interviews is that it is a common theme, that estimations do not seem to have the desired effect on either the teams or stakeholders. Are estimation points done properly and measured? Then an interval can be viewed on the history of how many points the team has finished in historical sprints and from there, a more reasonable measurement of the team's capacity could be used.

2. Agile standard not fully implemented

Agile principles are not implemented properly.

A Product Owner mentions that some teams are not fully involved in following the Agile principles and some are perhaps missing some of the planning. Business analysts sometimes think that sprints are hard deadlines. This adds to the issue as well, when part of the organization is not on board on the way the teams should work. A Stream Lead comments on this

as follows: “We need to stop to work with fixed scope and deadlines. Go back to a discovery phase. We try to make sure that everything we deliver is split into small parts. How can we slice it even thinner, so that we deliver something of value as soon as possible?”

We argue here that the team’s environment, in some cases, does not allow or require teams to operate accordingly. Furthermore, this is also pushing back on the use of Agile principles, as the other parts of the organization, using waterfall methodology, are not used to this way of working and perhaps do not understand the value or the strategic reason for it. It is possible that the environment the teams are in does not allow them to follow those principles.

3. Issues are either not being raised or clarified enough

Issues seem to be raised mainly in retrospectives and then only in regards to the work itself, as it seems. There isn’t naturally an awareness or an ongoing improvement of processes that are not related to product delivery. This becomes an issue if the structure does not allow for employees to be open-minded and bring out-of-the-box suggestions. Furthermore, if the team collaboration fails then issues will not be raised if they are not in the direct responsibility of the team or closely related to tasks.

This is something that has come out, as questions have not been fully answered by the interviewees, on the subject: either because of its absence or that it hasn’t been investigated enough. It might be worth to be mindful of when there are signs of lack of innovation regarding what could improve.

4. Empowerment of Scrum Masters

Scrum Masters are considered by others to deal with improvements that are not of a technical character. However, they are not empowered enough to actually be able to solve these issues and implement solutions in the teams. Therefore, even if issues are raised, most likely things will not improve.

Scrum Masters are not empowered to deal with things that could be improved. A Product Owner, when asked whom the team relies on for improvement or ideas, said it should be the Scrum master that then can take it up with other team’s Scrum masters. Member of support team adds that they are not empowered enough from his observation to live up to that fully.

We argue here from what has been seen in the investigation phase, that there might be a similar problem as that of the Product Owners, where there are managers outside of the team that has influence and therefore the boundaries on responsibilities become unclear. Another possible explanation could be rooted in that, the teams are not isolated and therefore do not have full control over their working process.

5. No continuous delivery

The production release is done only once per month. This goes against the principle of continuous delivery and the benefits of it.

A Tech lead adds that it would be nice to have an AB-solution for releases: two versions in prod, one that is currently used and accessed by users along with one containing new features. Easy switch. This change is a management decision. A member of the support team adds that it would be possible and that he does not see impediments to implement this infrastructure

change. Teams could then prepare releases during working hours, test those, and then flip the switch when allowed.

A.10 Other Issues and Statements

Some issues came up in the interviews, which cover many statements already discussed in other issues or there has not been time to investigate further. These can be found below. Sources are tech leads and stream leads.

- The solution will become too big at some point. Not as scalable as it could be.
- Bad Tools: Too many ad hoc tools are used in procedures behind the release. Se Investigation phase 1. The tools and methods used by developers are vital to the productivity of software development processes [13].
- Not easy to reproduce the PROD/SYST-environment locally (front-end developer).
- Tools could be better organization-wide: tool for internal service requests, release overview tools, change advisory board that approves and has processes with redundant information being over and over. Some of these have become better.
- Stressful environments for developers and some teams who are pressured too much.
- Handling the fact that some things need to be done and have hard deadlines. Hard requirements accepted and solved by scoping
- Too many have access to all the code! Mono-repo contains the code of the whole solution.
- Some teams are better than others when it comes to the product owner prioritizing and making sure the team is not overworked. This needs to be improved.
- "One big issue is the misalignment in regards to the architecture, where we set out to create a feature that gets the job done, but also introduces technical depth and is not reusable, not fit to the architecture, developed into a domain owned by a team that does not think this code should be there. This causes friction."
- Automated testing is lacking. If we had more automation, issues could be identified earlier.
- The backlog is too large and seems to contain things that might never be done. Many of those issues which seem to never be done are technical tasks.
- A good thing about the mono-repo (teams share one repository) is the overview of the entire solution and the induced ownership. A tech lead mentions on the inducement ownership goal (internal document: tech forum presentation based on data analysis of mono-repo survey answered by the teams) that it is terrible to strive for development-wise. "If everybody owns, then nobody owns"

- There is more communication going on where there were silos before. Improvements have been made.

- "We get requests all over the place regarding features and things to be developed. The more teams we onboard, the more requests we get".

A.11 Wishes From Employees

Every participant in both interviews and shorter talks, in the end, was asked an out of the box question: "If they would be imagining the best improvement for the solution, to make adaptivity and efficiency as good as possible, from their point of view; what would that look like?". Find their interesting answers below.

Tech Lead

All the pains addressed, especially communication issues, that we have discussed in the interview. Automated testing in place so that issues can be identified earlier. More than one SYST environment for testing. It happens that external dependencies don't work. Others are doing work that is prioritized or even during freeze when regression testing is ongoing. As there is only one SYST environment today, it is sometimes not possible to use it and it makes the development take longer time. The Test environment is not sufficient sometimes? SYST does not contain all the data that prod. It has happened that something that wasn't caught in SYST, appeared in PROD.

Tech Lead

The ability to do continuous integration, no living branches. We need to complete automated testing.

Each team is the consumer or provider of APIs. All teams are working in isolation.

Tech Lead

"If we had automatic testing (regression today), more SYST environments, and AB-solution in prod, then I think we would have a pretty good workplace. This would just ease our day and could get us to work very fast".

Product Owner

IT and business seem not to be on the same page. It is not clear on their side. A wish would be for them to be more pro-active and take ownership in their area.

Product Owner

We should be better at including chapters: Risk, UX, etc. It happens that we think we can solve it ourselves. We come up with something UX related on our own. The consequence is that we have to start over and redo the work. We should be better at delegating things like this. UX support is part of the refinement session and is also available, we should use them.

Stream Lead

Improve the way we organize. Remove the silos and have one organization where IT and business are close collaborating colleagues. For example, having the same leaders.

Stream Lead

We need a better channel to push things to production. It means that we cannot push features out and toggling them off. A better release channel and automated testing. We need to improve this if we want to do continuous integration and development.

Support Team Member

More frequent releases, with close involvement and collaboration with stakeholders.

Infrastructure support

In the future, all infrastructure stuff should go through the infrastructure support. Sometimes teams decide to do it on their own, there have been issues related to this and afterward, they come to us and ask why it is not working. There used to be errors that then proves costly as it wasn't done right from start.

Developer

Fewer people working on the codebase (less collaboration and communication between teams). It is hard to talk to so many people. It becomes much easier with fewer people when making decisions (the reason for the tech forum). We have experienced while dealing with many teams, that we are adding more but not a big deal as we already know what to do, in terms of communication. Less mess with only 2-3 people. There is a mess also because of the monorepo. Too many commits when so many people are working on the same. Many merge conflicts, many changes. It is not possible to checkout services from different branches for testing purposes before PR on local. We do not want them to change or we want to isolate our work.

Developer

An easier way to see what the customer is seeing. Better ways to reproduce issues. It is too difficult to set up PROD/SYST-like environments on the workstations.

Architect support

Smaller scopes and more iterations. Business analysts and product owners want too much at once. If we would be satisfied with less and do more iterations on the solution, we could increase our cadence, improve our velocity, and become more effective. Do less at the same time. More iterations and smaller tasks. More frequent deployments, reducing working progress (Idea to PROD).

What kills large organizations is not the blocker, it is the backpressure. When communicating on tasks, when things stop and you are not progressing, you are generating the backpressure as new tasks are always coming from above. We always should make sure that things are flowing. To continuously deliver keeps the flow going and minimizing backpressure.

You can see as we are growing in our area: things are cluttering up, become more rigid, and things are slowing down. This is due to complexity, but also backpressure. Everything around the teams is slowing down the development in this sense. For example, with the communication in upper levels of management: when things are discussed and then conflicts are pushed up to the manager next in line. People have a lot to do and this slows everyone, less back pressure should make everyone happy. Work clutter-up and it also comes from teams growing.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2020-12-16

EXAMENSARBETE A study of development collaboration in a water-gile-fall organization
STUDENT Astrid Jansson
HANDLEDARE Lars Bendix (LTH)
EXAMINATOR Emelie Engstöm (LTH)

Challenges of development teams in a water-gile-fall organization

POPULÄRVETENSKAPLIG SAMMANFATTNING **Astrid Jansson**

Root causes and solutions are investigated in a multi-team software development setup losing efficiency and adaptivity. The codebase is becoming more complex and expensive to maintain, due to faulty implementation of Scrum and unclear code ownership.

In the study processes and performance are investigated in a multi-team setup developing a software product. The teams are using Scrum and operate in an environment affected by bureaucracy regarding certain procedures, many imposed by external factors.

There are concerns in the department about how efficiency and adaptivity in teams are decreasing even as IT processes are considered to work well and follow industry standards. Incoming requirements, and changes being difficult to include in the already ongoing sprints, are common. These requirements are imposed outside of department control, and therefore need addressing.

Teams becoming less productive and adaptive in the change process is the initiating problem explored by the following research questions.

- **RQ1:** What are the root causes for the teams becoming less productive and less adaptive regarding the use of the mono-repo?
- **RQ2:** What solutions could help to eliminate some of these root causes?

In the analysis, root causes are investigated answering RQ1. From analysis results, possible solution designs are explored answering RQ2.

In investigating RQ1, we identified 60 issues affecting the development. The issues are grouped into nine areas where the three most important ones are Agility and Scrum, Collaboration Between business and IT, and Code Ownership Creating Issues. Two main root causes are discovered: Scrum not being implemented suitably and unclear code ownership. The codebase is expensive to maintain and complex due to communicational and collaborative issues that stem from Scrum guidelines not being followed. The imperfect code ownership furthermore affects the overhead work of the employees, the solution scalability, and complexity. Leading to a difficult future (and present) maintenance and expansion.

In investigating RQ2, the solutions include enforcing clear code ownership, improving processes and the Scrum implementation, as well as enforcing and supporting team self-organization.

Based on the results, we recommend the organization to separate the code ownership through modularization and implement Scrum from the business value providers' perspective, allowing for long-term efficient and adaptive development implementation. The organizational focus striving to prioritize technical excellence, with the effort to not only follow the Scrum structure but also embrace the mindset and strategy principles.