Measuring feature team characteristics of software development teams

MAJA GIDLUND
Measuring feature team characteristics of software development teams

MAJA GIDLUND
MAJAGI@KTH.SE

Master’s Thesis in Computer Science (30 ECTS credits)
at the School of Computer Science and Engineering
Royal Institute of Technology
Supervisor at CSC: Åke Walldius
Examiner: Jan Gulliksen
Employer: Scania IT

August 2016
Abstract

This report evaluates the team-structure of three software maintenance teams in order to decide their level of featureness (a term that defines to what extent a team has the quality (the set of characteristics) of being a feature team). Simulations of changes that are expressed as beneficial in an agile environment and that could increase the teams’ level of featureness within the team structure are performed. The results show that each team’s level of featureness is affected differently by each change. Partly, this underlines the importance of understanding the current team-structure before implementing changes that aim to increase the level of featureness. And secondly, within the scope of the study, the change where a user expert is declared a team member is concluded as the change that increases the teams’ level of featureness the most. Based on the results the report also concludes that it is essential to implement changes that affect different areas, which in combination can increase the level of featureness.
Referat

I detta examensarbete utvärderas graden av featureness (en term som definierar i vilken utsträckning ett team har snarlika egenskaper som går att finna hos ett feature team). Detta genom att undersöka team strukturen av tre styck- ken team inom mjukvaruutveckling som alla arbetar med underhåll av bestående system. För att senare utföra olika simuleringar av förändringar som alla är uttryckt passande inom en agile miljö och därmed ska leda till en ökad graden av featureness.

Resultet visar att teamens grad av featureness påverkas olika av förändringarna. Delvis visar detta att det är viktigt att förstå ett teams struktur innan man utför olika förändringar vars syfte är att öka graden featureness. Sedan visar resultet också att en förändring som innebär att inkludera användar expertis i teamet är den mest passande när man vill öka graden featureness. Baserat på resultatet kunde det också konstateras att det är fördelaktigt att implementera olika förändringar som i kombination berör olika områden för att i större utsträckning öka graden featureness.
ABSTRACT

Discussions about in what ways a software development company should be structured in order to create customer value and increase efficiency exist in a large extend. Among researches, some express a belief that working in teams is an efficient way of structuring software development [2, 3, 22]. In this report, a concept feature team that describes a team-structure within an agile environment is discussed. The aim of the study is to evaluate the team-structure of three software maintenance teams and to decide their level of featureness (a term that defines to what extent a team has the quality (the set of characteristics) of being a feature team). Changes that are expressed as beneficial in an agile environment and that could increase the teams’ level of featureness are investigated and discussed by simulating changes within the team structure.

The study is done by gathering data regarding the teams’ communication with external parties. The data is gathered by using an issue tracking program, sending out a survey to the team members and holding individual qualitative interviews with the team members.

It is discussed how communication with external parties affect the teams’ level of featureness and the teams’ waiting time in their working processes. Changes, such as declaring new team members or using new media to communicate with external parties, are simulated with purpose to find out how each change affects the level of featureness of each team.

In conclusion, it is found that each team's level of featureness is affected differently by each change, which makes it difficult to find one change that would increase all teams' level of featureness. Further, this underlines the importance of understanding the current team-structure before implementing changes that aim to increase the level of featureness. Within the scope of the study, the change where a user expert is declared a team member is concluded as the change that increases the teams' level of featureness the most. Also, it is essential to implement changes that affect different areas, which in combination can increase the level of featureness.

Keywords

Feature team; Software maintenance; Agile Software Development; Cross-Functional; Self-managed;

1. INTRODUCTION

To create and maintain an efficient team has been discussed in terms of what qualities a team should hold and what elements a team relies on [2, 22]. In this report, three diverse teams are studied within software maintenance. The goal of software maintenance is to keep a software product usable in a changing environment after it has been delivered [1]. The investigation aimed to compare the teams team-structure with a specific team-structure that is expressed as beneficial in an agile environment. The study is performed at Scania IT, which is the IT solution provider for Scania CV - one of the world’s leading manufacturers of heavy trucks, buses and engines. Scania IT’s software maintenance department is characterized by their use of agile methodologies, such as Scrum, Kanban and Lean software development. Their departments within the mentioned context are divided into teams, an approach based on the idea that working in teams is more efficient [4][2].

The software maintenance teams at Scania IT commonly consist of one maintenance manager, developer(s), one architect and tester(s). The working process of the teams usually starts with a modification request (MR) which in this thesis is defined by:

MR is used to identify proposed modification to a software product which is maintained by a team. It involves modifications of a correction or enhancement that causes corrections, preventions, adoptions or perfections of the maintain software product. [1]

A MR is usually created by the product owner who owns the feature the teams are maintaining and it describes what changes the teams have to do in order to keep the software product usable. The working process of a MR of the teams can be generalized and demonstrated with 8 steps as shown in Figure 1. In the third step, Customer Feedback, the customer is considered as the PO. The customer, i.e. PO, is also involved in the seventh step, Acceptance test, since the customer is the one that performs the acceptance tests.

1.1 Scania IT strives for a good team-structure

Scania IT has not found an agreement of how software maintenance teams should be structured to achieve maximized efficiency in an agile software environment. Consequently, the company is investigating which kind of team-
structure that would result in minimum waiting time and maximum value creation in the working process of a MR, which is considered as an optimal efficiency and throughput. Based on the company’s current situation a research question was created:

- How may changes measured affect a software maintenance team’s dependency of external parties with respect to the team’s level of featureness? 

In order to answer the research question it is appropriate to find a method that can validate the current team-structure of the investigated teams. Since the company also has requested suitable improvements that can be done to increase team efficiency and throughput, a high understanding of the team-structure is required which will be achieved by measuring featureness of the teams.

2. THEORY

2.1 Agile software development

The light-weight methodology that was introduced in software development to encourage a higher rate of changes is nowadays categorized as agile [9]. It aims to increase customer satisfaction and quality and it addresses the problem of rapid changes [9]. It is focused on the talents and skills of individuals; hence the processes are customized to adapt to specific people or teams [2]. Agility in practice requires that teams have mutual trust, focus and respect.

Several methodologies can be classified as agile software development, such as Scrum, Extreme Programming (XP) or Lean Development. All of these emphasize design quality, which is specifically addressed in each method [2], and argue that the agile teams should be self-managed [16]. A key idea in agile development is that teams always should aim to reduce the elapsed time between a made decision until the plan, execute, monitor and manage the team process and production in each agile iteration. Existing research also proposes that this kind of team-structure breaks the barrier between development and testing by putting them in the same team [16].

A team that is self-managed, has the authority to design, plan, execute, monitor and manage the team process and progress at an operational problem level [14] [12]. The results obtained in [12] discovered that a self-managed team that has this type of authority level acquires an ability of accuracy in problem solving [12]. Research also suggests other qualities that need to be held by a self-managed team to successfully meet challenges as they arise. These challenges are the ability to reorganize when needed and that the team needs to consist of one or more generalists [2][12]. The authors also found shared resources, organizational control and specialist culture as the main issues when a company wants to make a transformation from traditional command-and-control management to collaborative self-management [12]. Shared resources, from a team point of view, can result in inaccessible resources during an agile iteration.

2.3 Team performance and Team efficiency

The advantages of implementing cross-functional and self-managed teams are well acknowledged where the implementation seeks to increase team performance. Nevertheless, team performance is dependent on several factors which not necessarily need to be correlated with the team. These factors are addressed in [12] where the authors discuss how a team can be certain that it is surrounded by everything it requires in order to become a successful self-managed team. In the end, the authors found that an understanding of the organizational context, such as training, resources and organizational structure, has an affect on team functioning and is significant when one wants to determine effectiveness.

Models of team effectiveness can be found in several disciplines of science [4]. One model is used in [12] where factors such as team leadership and mutual performance monitoring are assumed to have an influence on team effectiveness. Other found factors that affect team productivity are the ability to delegate to other team members and make decisions within the team [22].

2.4 What is a feature team?

The concept of feature teams is not a new phenomenon in agile. It could rather be described as a refinement of a cross-functional team. Research on feature teams shows a variety of descriptions. One argues that a feature team is about five things: empowerment, accountability, identity, consensus and balance [18]. While others state that a feature team is characterized by a long-lived, cross-functional and cross-component team [16]. It possesses primary skills and holds the necessary competence and knowledge to complete an end-to-end customer-centric feature [16]. After all, a feature team is stated as the key to managing accelerating...
time-to-market and to adapt to a scaling agile development [16]. In short, a feature team is a team that has complete ability to build something and is independent of others.

2.5 Differences between traditional component teams and feature teams

There are several qualities that separate a feature team from a traditional component team (both concepts are described in section 2.5.1 and 2.5.2). The main differences essential for this study are the following:

- A feature team has the responsibility to complete a specified customer-centric feature request without any handoffs, compared to a component team where the team has responsibility to complete only a part of a customer-centric request with handoffs [16].
- A feature team has minimized dependency between teams which increases flexibility, compared to a component team where its dependency between teams causes additional planning [16].

From another perspective, the differences can be described in how the two teams are structured regarding the component of systems (or architect modules that has the same significance) and features. The component systems can be for example front end code, back end code or a database. The feature, on the other hand, can be for example the functions of a website or a smartphone application. This means that the feature is built upon components such as front end code, back end code and a database. For example, both the smartphone application and the website can have separated components of front end code and back end code, but share component of the database.

2.5.1 Component teams

Assuming that a company has six different component teams, where each team holds a specific knowledge of a component. One team consists of front end developers, one of back-end developers, one of database specialists, one of architects, one of user experience specialists and last team consists of testers. The teams are handed one MR that concerns feature X. The MR implies corrections of front end code, back end code, the database and the architectural interactions between front end and back end. Therefore, every team has to perform their specific task one by one, which results in a working process of the task that looks like Figure 2. This means that the task has five handovers from one team to another before the MR can be finished. A handover can involve more than just giving the next coming team the task. It can be that the team needs to explain facts regarding the team’s component, ask questions concerning other components, or ask for access or decisions of other components.

2.5.2 Feature team

Assuming that the same company, as in the example of component teams, instead has one feature team. In that case, the team holds a solid responsibility for feature X, which means that the team needs to hold the knowledge of each component that constitutes the feature. Therefore, the team consists of a front end developer, a back end developer, a database specialist, an architect, a user experience specialist and a tester. The working process of the task is illustrated in Figure 3.

This means that the task has no handovers to another team since the team holds competence, has access and has the authority to make decisions of every component that constitutes feature X.

2.6 Benefits of a feature team

In [11] the authors have explored how daily build and feature oriented development can decrease lead time and increase productivity and development quality. The authors mentioned several guiding principles to achieve these goals, one being cross-functional teams that hold sustainable responsibility, which in the study are referred to as feature teams. The authors conclude that feature teams are suitable for the complex project environment daily builds create since the teams need to cover every phase of the development process, from customer contact to system test. The study discovered that the advantages of a feature team are increased motivation, that change management is absorbed more easily, that there is more efficient coordination between associated projects and a more efficient interface coordination.

2.7 Transformation to feature team

It was found in [11] that moving from a component team to a feature team (especially within an organization with a strong (traditional) development process) is a difficult task. The main reason is that in a traditional development process features are built upon components, which commonly are dependent on each other. Therefore, by changing or updating a feature, other features will probably be affected since the changed feature consists of one or several components that constitute other features. This creates a need for coordination and planning when updating or changing a feature [11].
In addition, [16] state two tactics to make a transition to a feature team: reorganize into a broad cross-component feature team or gradually expand team responsibility. The first tactic implies that specialists in a component team are divided into new feature teams where each specialist is unique with their knowledge. By expanding responsibility of each team, which implies that no one team has the sole responsibility for a specific component, the team has the authority to make necessary changes. In combination, if the team is broad cross-component, the team also has the knowledge that it needs to have to be able to perform changes. Therefore, the team can work self-sufficiently which causes less need for coordination and planning when the team wants to make changes or updating a feature.

As mentioned above, a feature team should also be a long-lived team. To create a long-lived teamwork, processes need to be regrouped to fit the teams, i.e. the team is considered as the unit of organizational work [16].

2.8 Strategies to increase efficiency

To work efficiently could in one sentence be described as avoiding unnecessary activities, like asking one question several times, performing the same tasks, or removing waiting time that arises from handoffs or is caused by misunderstanding.

A common strategy that can minimize the coupling between external systems and unit-code for a team is the use of acknowledged design principles or design patterns. A design pattern supplies a set of syntactic notations and a set of rules for how and when to use each notation [10]. Some claim patterns as a good vehicle for knowledge sharing within context [5].

One way to manage design patterns within an organization is to use standards for work flow, which can be designed and created in different ways. Among several approaches is one presented in [17]. The purpose of the created standards was to encourage same design for similar problems and to increase consistency and usability across the company. The authors gained an understanding of the work process and accomplished to design a suitable pattern library that was accessible for all employees. The solution was reviewed and contributed by an expert staff, and from an authoritative perspective the authors concluded that it would have been impossible for a small group to create the same kind of standards.

An alternative way to communicate within a company that in some cases are considered being more efficient than traditional communication tools such as email is software that can provide different open channels for discussions. One example of these recent applications is SLACK. It provides accessible historical archives of centralized information that are beneficial if a new person enters a project that lacks awareness of made decisions, performed implementations or given requirements. So, instead of having to ask other project members for a recap, the new person can go through historical work itself [15].

2.9 Research methods in software engineering

Since software engineering stretches over different kinds of issues - technical, linguistic, social and psychological - it is significant to understand the methods that are available in software engineering in order to do a scientific research [8]. Basili (1993) has presented four research methods in the software engineering context: scientific, engineering, empirical and mathematical [16].

Of these four, the empirical method is the most relevant in this study. It is traditionally used when it is difficult to state any laws of nature, as in social science, but are also suitable to use in human-intensive software engineering [8]. The empirical strategies in software engineering are experiments, case studies and surveys [8].

Among the three different empirical strategies, I found a case study to be the most appropriate in this study, which in software engineering can be defined as “an empirical enquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified” [16].

2.10 Method to present a visualized result

Studies of team-structure and team-performance show that social network analysis, an approach to visualize team structure, is useful when the goal is to understand interaction processes within a team. This approach has also been beneficial in software engineering research when the objective is to compare team performance based on gathered data [13]. Hence, social network analysis is found appropriate to use when visualizing the results of this study.

A social network consists of a finite set or sets of agents, represented by nodes, which are connected by one or more specific types of interdependence, represented by edges [21]. In social network analysis the social relationship is analyzed in terms of network theory [20]. This kind of analysis is appropriate when investigating community structure or relationship patterns [20].

3. SCOPE OF THE STUDY

Even though studies have shown that a feature team is beneficial in an agile environment, the report has not found research that presents a validated method for deciding if a team is a feature team or a component team. And when briefly scanning through documentations regarding Scania IT’s software maintenance teams it seems like the company has feature teams since their teams consist of different roles, i.e. hold knowledge in different areas. If this is true or not cannot be concluded without an investigation. Therefore, the study will create a method that both evaluates the structure of Scania IT’s software maintenance teams and decides the level of featureness of the investigated team. Also, by gaining an understanding of the team-structure, the study will aim to further propose changes that will increase the teams’ featureness.

3.1 Measuring the level of featureness

Based on the theory, the study classified three elements that affect independence; the ability to make decisions within the team, the ability to hold the right authority level and the ability to have the necessary knowledge. These abilities affect the number of handoffs a team has to other external parties which in terms affect the level of featureness. The report also stated that the higher number of external parties a team has the lower is the level of featureness. This means that the main element that determines the level of featureness of the teams is the number of external parties each team has.
4. METHOD

Three teams were included in the study, which are referred to as team A, B and C. The teams were all selected by the convenience in discussion with the supervisor at Scania IT. The purpose of the method was to gather data and information about how the team worked with external parties. The application of the method was divided into three steps, where the first two base steps serve to provide material for the main third step.

Figure 4: Working process of the method

4.1 Base step 1: Collect performance data

In the first step performance data of specific selected MRs was gathered from the company's issue tracking system. For each team, five MRs were selected in discussion with each of the team’s maintenance managers in order to ensure that the MRs represented work that concerned a feature the team maintained. The issue tracking system that provides the performance data makes it possible to track which team members had been involved in the work process of a MR. In some cases, dependent on to what extend each team member had documented his/her work, is it possible to understand what each person has done and with whom it had external contact. The purpose of the performance data was to collect information about the working process and understand which team members that worked with the selected MRs.

This first step was based on [6] which had a similar evaluation method, i.e. the authors gathered historical change requests, trouble reports and source code changes. The authors were, with the chosen evaluation method, able to find which form of development that would result in high performance software processes, which was the aim of the study.

4.2 Base step 2: Survey

The survey was based on the gathered performance data and the purpose was to get a more precise picture over the team’s contact with external parties. The survey was first sent to the supervisor at Scania IT for evaluation, and later on tested by the maintenance manager in team A to verify that the survey was understandable. After received feedback from the team member in team A some changes were made to make the survey clearer before sending out the survey to the rest of team A. The survey sent to team A was anonymous and consisted of ten questions that were divided into two questions per page. Each page had the same two questions but handled a specific MR that was specified in the top of the page. The first question was about if the team member had any external contact during his/her work for the specified MR and with whom. In the second multiple choice question the team member was asked, for each stated external contact, to give the reason(s) for it.

The survey that was sent to team B and C was handled a specific MR that was specified in the top of the page. The reason for adding the question was because it made it easier for the interviewer to prepare the interviews, and it aimed to add some reflection for the respondents. The added question was then followed by the same two questions in the survey for team A.

4.3 Main step 3: Qualitative interviews

The main step in the method, qualitative interviews, were individually prepared based on the interview object’s answers of the survey. The purpose of the interviews was to find errors that might have occurred from misunderstanding in the survey, verify the external parties and further discuss the team member’s external contact. Each team member was asked the same questions.

Two questions were asked concerning in what media the contact was taken and how long it took to receive an answer. The purpose of these two questions was to gain an understanding of the working process of each team and its time span. Two rating questions were asked regarding how intense the external contact was and how effectively the team member was able to use the waiting time for receiving an answer. These questions aimed to understand how the team member worked and on what level of dependency the relationship was with the external party. One rating question combined with an open question was asked in the end which was about on what level the team member and the external contact understood each other and if they used any references documentations to understand each other. The meaning of reference documentations is any information that are documented in words, pictures or numbers, or documented principles or guidelines.

5. RESULTS

All of the team members answered the survey and all of the team members except one were interviewed. The gender split was equal between women and men and the middle age of the team members was between 30-40. The results are categorized to make it easier to view the result such that team competence is separated from team member competence. Hence, an interview object that answered a team member as external party and another interview object that answered a team member as external party where both external parties belonged to the same team are merged and categorized as Another team. An interview object that answered a team member as external party that belonged to a team which no other external party belonged to is categorized as Another team member. Also, the charts of team A, B and C correspond to different teams and team members, that are found in the y-axis.

5.1 Party dependencies visualized in Social Network Diagram

A social network diagram has been created for each of the teams, which is a merged result of the five chosen MRs. An edge from the team node to an unfilled node means that a team member had contact with an external party regarding one specific reason. This means that a team member who had contact with an external party regarding two different things, e.g. didn’t have access themselves and wanted to ask a question, results in two different edges from the team to two separated unfilled nodes. The three different social network diagrams, Figure 5, show that team A had 13 external
parties, team B had 33 external parties and team C had 24 external parties regarding the chosen MRs.

Figure 5: Social networks diagram

5.2 Reasons for taking contact with external party

In Tables 1-3 the result of each team is presented regarding the reasons for contacting each external party. The y-axis corresponds to each external party the team had contact with and the staples on the x-axis corresponds to numbers of contacts the team had with the external party.

Table 1: Reasons for taking contact, Team A

Table 2: Reasons for taking contact, Team B

Table 3: Reasons for taking contact, Team C

From Tables 1-3 it can be seen that both team A and team C had the most contact with one of its PO. Which differs from team B who had most contact with another team.

The reason other corresponds to a reason that could not be translated into the chosen categorizes. Other in Table 2 for Another team and Integration team represents a requested integration, and other for Another team member 2 represents a collaboration with the team member.

5.2.1 Waiting time to receive answers from external parties

In the interviews the interview objects were asked to estimate the waiting time before it received the favor or answer that they asked for. The estimated times are converted into days and correspond to the total estimated time of all team members from a team of all of the investigated MRs, which is found in Tables 4-6.

Table 4: Waiting time, Team A

As seen in Table 4 for team A, the external party who had the longest waiting is the external party the team had the most contact with. For team B and team C, the external
Table 5: Waiting time, Team B

Table 6: Waiting time, Team C

The party who had the longest waiting time is the external party the team had second most contact with.

5.3 Used reference documentation

Team A used reference documentations when communicating with two external parties. In both of the cases the level of understanding of each other was 5 out of 5. For team B reference documentations were used for eight external parties. In five cases the level of understanding of each other was 3 out of 5, in one case the level was rated 4 out of 5 and in the last case the level was rated 5 out of 5. For team C reference documentations were used for four external parties. In three of the cases the level of understanding of each other was 5 out of 5, and in the last case the level was rated 3 out of 5. The average level of how well the team member and the external party understood each other relative well.

5.3.1 Media used for communication

Something interesting that was found in the results but can be considered as a curiosity are the chosen media that were used to communicate with the external party. These are represented in the Tables 7-9.

Table 7: Used media, Team A

Table 8: Used media, Team B

The media email, phone and skype business are closed and private ways to communicate, compared to the issue tracking system Jira or having a meeting. In a majority of the cases of all the teams private media were used when communicating with the external party.

5.4 Validity of the method

One problem that occurred in the method was that one team member from team B was unable to answer the survey or participate in the interview. As a consequence, if the team member would have had any external contact these external parties are not included in the results. Hence, the level of featureness of team B can appear higher than it actually should be. Other elements that can affect the validity of the method are if the MRs do not correspond to the teams actual working tasks, or if the team members forgot some
Table 9: Used media, Team C

external parties. If the team members forgot some external parties the level of featureness appears higher than it actually should be.

To minimize the risks of having a team member that forgot some external party the issue tracking system served to the team member to remember its involvement in the chosen MRs. If the team member was tracked in a MR it was brought up and discussed in the interview.

6. DISCUSSION

A team with zero edges from the team node to an unfilled node has a maximum level of featureness, which means that the team has no dependency to an external party and is hundred percent self-sufficient. When going through the social network diagrams of each team in Figure 5 it can be seen that team A had a higher level of featureness than team C, and team C had a higher level of featureness than team B. In order to increase featurelessness, edges from the team node to unfilled nodes need to be removed, which would mirror changes taken for increase the independence of the team.

6.1 Treat each team individually

When comparing the teams’ results with each other it is noticeable that their results differ, both in terms of numbers of edges in the social network diagrams and what kind of roles the team needed to contact. In addition, this can imply that it will be difficult to find one solution that suits every team and increases each team’s featurelessness. In other words, finding a standardized solution that increases featurelessness may not exist. This hypothesis will further be elaborated by presenting different simulations of improvements that aim to increase featurelessness, to see if the same changes work for all of the teams.

6.1.1 Change One: Considered PO as a team member

As mentioned above, a user expert is suggested to have close contact with the team in order to decrease elapsed time between decision making [2]. In this case, the PO has the closest contact with the users of the features which the teams are maintaining. In addition, in two of three teams the PO was the most contacted external party, and for team B the PO was the second most contacted external party. The reasons why the PO was contacted were because the teams needed to ask a question or wanted the PO to make a decision. This indicates that the PO holds necessary expertise or has a level of authority which the teams are lacking. Therefore, based on the result and earlier studies the PO would be elaborated to be a team member. This can be considered as another way to measure the teams’ featurelessness. How this affects the social network diagrams for each team are illustrated in Figure 6.

Figure 6: PO as an external party to the left and PO as a team member to the right

As seen in Figure 6 the change affects team A and team C significantly since approximately half of the edges from the team node disappear. For team B less than a fifth of the edges disappear. Though the removed/decreased waiting time is most significant for team B, 20 days compared to team A who has 16 days and team C who has ten days. If these days were considered as a blocker all of the teams would save a lot of waiting time by declaring and treating the PO as a team member (which means that the PO is available in relation to the team all the time).

It can be discussed if treating the PO as a team member, i.e. make the PO available in relation to the team all the time, would be a positive improvement or not. As mentioned earlier, the PO is involved both in an early and a late stage in the working process of the teams. If that is the only involvement that affects the teams’ dependency to its PO is something this case study can’t answer. However, considering the resulted increased level of featurelessness and removed waiting time, it might be a good idea to analyze how high accessibility the PO should have in relation to the team. Also, an awareness of to what extent the team experienced a dependency of the role might be important to have in mind when making future transformations. It would also be a good idea to explore if the dependency of the PO only is correlated with the working process of the teams, or if the dependency continuously exists during the whole working process of a MR, which would make the problem more complex.

6.1.2 Change Two: Considered the second most contacted external party as team member

By declaring the PO as team member a big difference was observed. To understand if that is the only role that
should be included in the team the most contacted external party, which is not a PO, will be named as a team member. In the case when two external parties were contacted equal times, the external party that had the longest waiting time was named team member. In this simulation, the role that is declared team member and is included in the teams are; *Integration team* for team A, *Another team* for team B and *Web tech team* for team C.

![Figure 7: External party as an external party to the left and External party as a team member to the right](image)

As seen in Figure 7 the change affects all of the teams. Approximately half of the edges disappear from team B, and approximately a fifth of the edges disappear from team A and team C. The waiting time decreased by five days for team A, 20 days for team B and 13 days for team C. Compared to changes one, this change had a larger effect on the featureness on team B but not on team A and C. Regarding the waiting time that was removed this change had a larger effect on team C but not on team A and B. This indicates that change one has a higher effect and is more applicable to implement than change two. Also, it confirms the PO as the most dependent external party.

### 6.1.3 Change Three: Open communication channels

Based on the assumption that open channels in some cases are more efficient than traditional channels, I would like to elaborate with the use of such environment in order to remove asked questions to external parties [15]. This is considered a good change, since in the majority of the cases the teams used a private media when they asked a question. By having this kind of channel, the report is assuming that questions can be removed because the team can found the answers in an alternative way that would remove the waiting time to receive an answer. The asked questions are removed from the diagrams and is demonstrated in Figure 8.

![Figure 8: Asked questions are included to the left, and asked questions excluded to the right](image)

As seen in Figure 8 roughly half of the edges disappear from the teams, which indicates that this change is better than change two. Whether a collaborative open environment would have these effects is difficult to answer, but it encourages to find a solution that would tackle the problem. A problem that might occur when implementing a collaborative open environment, where questions and their answers are visible, is classification of information within a project or a team. Another problem when using these kinds of software is that employees can feel uncomfortable asking questions that are visible for other departments.

Since the interview objects were asked to estimate the waiting time for each external party it had contact with, it makes it impossible to summarize the waiting time only for receiving an answer for an asked question. On the other hand, it can be assumed that the change would decrease the waiting time since roughly half of the edges would be removed. Therefore, it would be a good idea to try new ways to communicate if that would tackle this problem.

### 6.1.4 Change Four: Increase authority

As mentioned earlier, the meaning of being self-managed is that a team should hold an authority level to manage their working process. If the investigated teams would be self-managed it would not exist a need to ask an external party to make a decision. The results show that an external party was contacted in a third of the cases to make a decision. Therefore, it can be further discussed if the teams hold the right authority. What would happen if the teams’ authority level increased, will that tackle the problem? This is an interesting question, but too complex to answer in short. Given that a decision was asked to be made by a PO in roughly half of the cases it can be assumed that increasing the teams’ authority level would not help, since the decision has to be made by a person with user expertise.

### 6.2 The method in retrospect

The two base steps gathered similar data, therefore it would have been enough to use performance data or survey. Also, in order to get more precise answers in the interviews, I could have prepared the interview objects better, e.g. encouraging them to go through historical data in their email before the interview.

### 6.3 Possible to generalize the method and the results?

The method measured featureness of three different teams. All of the teams were working with software maintenance, hence hypothetically the method can only be generalized to measure featureness of teams with the same working focus. The results are based on five MRs of three teams gathered over a nearby period of four to five months, which is crucial when determining if the results can be generalized. It cannot be assumed that the three teams can represent all kinds of software maintenance teams, since they work with...
different systems and are using different programming languages. Also, it cannot be concluded that the five MRs used to measure featureness of the teams were representative to correspond the teams actual work. The fact that some team members were not working with any of the MRs indicates that more MRs should have been included in the study to make them representative. On the other hand, by only using MRs that were performed in a recent time period decreased the possibility of adding MRs that corresponded to a team structure that had been changed over time.

6.4 Future research

Further research is needed to validate the changes. Hence, it would be a good idea to perform a similar research in a larger context with more teams and more MRs, since my conclusion indicate but not confirm the proposed changes.

6.5 Ethical, social and sustainability perspectives

During the writing of the report the gathered data from the company was continuously discussed with the supervisors, such that essential guidelines were followed. When performing the interviews, each interview object was informed about the purpose of the study and was asked if they felt comfortable being recorded. Each interview object was also handled anonymously with respect to the participants if they wanted to share private opinions. This to avoid possible ethical implications. When analyzing the data, the three teams were treated anonymously since the purpose was not to compare the teams with each other. Before the report was published, the company decided to not be confidential in the report which strengthen the validity of the report.

Generic, the report is not directly connected to sustainability from an environmental perspective since it is focused on how to evaluate changes that affects a software maintenance team in terms of the level of featureness. The report performs different simulations that aims to decrease the teams’ dependency to external party, i.e. the report wants to understand how a software maintenance team should be structured in order to be as efficient as possible. From a broader perspective, this understanding is essential for companies that operates in evolving markets. Because, it is likely to assume that efficient teams have a higher ability to realize new products in the same phase as customer’s needs changes. In the long run, companies have to be adaptable to changes otherwise it will be hard to develop products that are requested by the market. And creating products that are based on customers needs in a larger extend is more sustainable from a society perspective because it decreases the need to replace systems, services and products.

7. CONCLUSION

By using a method that measured the level of featureness and team qualities, simulations were performed that showed how different changes affect the team-structure and level of featureness of three software maintenance teams. Also, it was noticeable that the teams corresponded to the changes differently.

Change one simulated the results of having a PO available in relation to the teams. It was found for team A and C that the level of featureness could be increased with fifty percent and for team A the waiting time could be decreased by 20 days (if the waiting time was assumed to block the progress). Change two simulated the affect of including the second most contacted external party in the team. This change had a larger affect on team B’s level of featureness and waiting time than change one. In combination, these changes showed that one change that is considered applicable for one team does not have to be preferable for another team. In other words, having a PO available would not always result in the highest effect of increased level of featureness and decreased waiting time of different software maintenance teams.

Change three encouraged to explore new ways to communicate with external parties in order to increase the level of featureness and change four made it comprehensible that raising authority level might not tackle the problem of having a team that cannot make all decisions within the team. Based on the results and the use of logical reasoning, it can be assumed that authority and knowledge are dependent on each other if the goal is to increase featureness. For example, a solution needs to consider changes that combines different elements in order to create independence of others and ensure that the team has complete ability to finish a feature.

To summarize, it was found that the teams were affected by simulations differently, which indicates how important it is to measure the team structure when aiming to perform changes that will make the team more independent and complete. Also, changes that aim to create teams that are independent of others and have a complete ability to finish an end-to-end feature need to contribute with different elements. Further, this study confirms by simulations existing research that states a close collaboration with user experts as a tactic to remove elapsed time caused by decision making.

8. REFERENCES