Tampereen yliopisto

Lassi Mäkelä

# IMPROVING FEATURE ESTIMATION USING SCRUM HISTORY DATA

Faculty of Engineering and Natural Sciences
Master of Science Thesis
November 2019

# ABSTRACT

Lassi Mäkelä: Improving feature estimation using scrum history data
Master of Science Thesis
Tampere University
Master's Degree Programme in Information and Knowledge Management
Examiners Assistant Professor Henri Pirkkalainen and Professor Samuli Pekkola
November 2019

---

Agile software development companies react to the changes in their volatile business environment to satisfy customers to the best of their ability. Planning the releases of software versions, making promises to customers and deciding where to put resources are difficult tasks which require information about the current state of features and their estimated time of completion. Making schedules takes resources and is difficult because of the complex and changing environment.

This research aims to identify problems of feature estimation in agile software development and discover solutions to these problems. The study was conducted as a qualitative case study for a Finnish agile software development company and the DSRM framework was used to provide guiding lines for the research. Information was gathered by reviewing the academic literature and interviewing the employees of the case company. An artifact which used the case company's scrum history data as an input was created to improve feature estimation. The artifact was created in a form of a dashboard which displayed general information about the features currently being developed to the software product of the case company. The general information included for example name, responsible team, estimated completion date, completed percentage and the number of active developers. The estimated completion dates were calculated by using the company's scrum history data and more specifically the user stories' median cycle times.

The results of the research provided identified problems regarding feature estimation in agile software development and a description how an artifact was made to solve these problems in the case company's context. The performance of the artifact was evaluated comparing it to its requirements and by calculating average error of estimates. The accuracy of the estimates was on an acceptable level according to the case company and the artifact was seen beneficial. The transferability of the research and the generalization of the results is difficult because of the unique environment of the case company.

Keywords: Agile software development, Scrum, Effort estimation, Feature estimation, Agile planning, Release planning, User story, Story points, design science research

# TIIVISTELMÄ

Lassi Mäkelä: Toiminnallisuuksien valmistumisarvioiden kehittäminen scrum historiadatalla
Diplomityö
Tampereen yliopisto
Tietojohtamisen DI-tutkinto-ohjelma
Tarkastajat apulaisprofessori Henri Pirkkalainen ja professori Samuli Pekkola
Marraskuu 2019

---

Ketteriä menetelmiä käyttävät ohjelmistoyritykset pyrkivät reagoimaan ailahtelevassa liiketoimintaympäristössä tapahtuviin muutoksiin voidakseen palvella asikaittaan parhaansa mukaan. Ohjelmistoversioiden julkaisemisen suunnittelu, asiakaslupausten pitäminen ja resurssien jakaminen ovat haastavia tehtäviä, jotka vaativat informaatiota toiminnallisuuksien tämänhetkisestä tilanteesta ja niiden arvioiduista valmistumisajankohdista. Aikataulujen tekeminen on myös haastavaa moniulotteisen ja muuttuvan liiketoimintaympäristön johdosta.

Tämä tutkimus pyrkii tunnistamaan ongelmia, joita esiintyy ketterän ohjelmistokehityksen toiminnallisuuksien valmistumisarvioiden tekemisessä ja löytämään ratkaisuja näihin ongelmiin. Tutkimus toteutettiin kvalitatiivisena tapaustutkimuksena suomalaiseen ketteriä menetelmiä käyttävään ohjelmistoyritykseen. DSRM-viitekehystä käytettiin tutkimusprosessin vaiheiden määrittämiseen. Aineistoa kerättiin perehtymällä ketterään ohjelmistokehitykseen liittyviin akateemisten kirjoituksiin ja tekemällä teemahaastatteluita kohdeyrityksen henkilöstölle. Valmistumisarvioihin liittyvien ongelmien ratkaisemiseksi luotiin artefakti, joka hyödynsi kohdeyrityksen scrum historiadataa. Artefakti luotiin automaattisesti päivittyvän kojelaudan muotoon, joka esitti yleistä tietoa kohdeyrityksessä sillä hetkellä kehityksessä olevista toiminnallisuuksista. Esitettäviin tietoihin sisältyi esimerkiksi toiminnallisuuden nimi, kehitystiimi, arvioitu valmistumispäivä, valmiusprosentti ja aktiivisten kehittäjien määrä. Arvioidut valmistumispäivät laskettiin käyttämällä hyväksi kohdeyrityksen scrum historiadatasta löytyviä käyttäjätarinoiden mediaani läpimenoaikoja.

Tutkimuksen päätelmiin sisältyi löydetyt ketterän ohjelmistokehityksen valmistumisarvioihin liittyvät ongelmat ja selvitys minkälainen artefakti rakennettiin korjaamaan näitä ongelmia kohdeyrityksen tapauksessa ja millä perustein. Artefaktin suorituskykyä mitattiin vertaamalla sen toiminnallisuutta kehitysvaiheessa luotuihin tavoitteisiin ja laskemalla valmistumisarvioiden keskimääräinen virhe. Keskimääräinen virhe oli kohdeyrityksen mielestä sopivalla tasolla, tavoitteet ja toiminnallisuus olivat linjassa ja kohdeyritys näki artefaktin hyödyllisenä heidän toiminnalleen. Tutkimuksen siirrettävyys toiseen kontekstiin ja tulosten yleistettävyys voi olla haastavaa johtuen kohdeyrityksen ominaisuuksista, jotka mahdollistivat tutkimuksen onnistumisen.

Avainsanat: ketterä ohjelmistokehitys, ketterät menetelmät, käyttäjätarina, valmistumisarvio, design science research

# PREFACE

Studying to be a Master of Science has been the most joyful and interesting time in my life. During the last five and a half years I have met wonderful new people, made a great number of friends and learned more than I could ever have imagined. Unfortunately, this phase in my life is soon coming to an end as graduation draws near.

I want to thank Minna and Jari from the case company for providing me with an interesting topic and constant feedback for my master's thesis. I'm also grateful for all the people from the case company who participated to the interviews and made the whole research possible. In addition to that I want to thank Assistant Professor Henri Pirkkalainen and Professor Samuli Pekkola. Without their practical guidance and constructive feedback this research would not have been completed at all. Finally, thanks to my family and friends for supporting me throughout my whole academic journey.

Tampere, 19.11.2019

Lassi Mäkelä

# TABLE OF CONTENTS

# LIST OF FIGURES

# 1. INTRODUCTION

The idea of agile software development is to develop software incrementally in short periods of time, which helps the developers and other personnel to react to the unexpected situations in the volatile business environment and accumulates rapid value for customers (Boehm & Turner 2003; Beck et al. 2001). According to the agile manifesto created by Beck et al. (2001) detailed plans and documentation are not focused on when performing agile software development. Instead a working software and reacting to change are valued much higher in agile projects. Estimating effort and creating schedules quickly based on estimates are essential, but difficult parts of the process in agile software development. User stories are a common way of describing the functionality of developed software and story points are often used in estimating the effort needed to complete the user story. The story points of a user story are usually estimated using expert opinion rather than any technological tools or algorithms. Release planning is also an important part of the process, where the goal is to choose which functionality is released in which software version. The success of release planning is highly affected by the schedules' accuracy. (Cohn 2005; Coelho & Basu 2012)

Cohn (2005) and Boehm & Turner (2003) argue that all the decision making in agile software development should include analysis of costs and business value. Without business value the decisions should not be executed. In agile software development processes this should be considered in most situations. For example, by prioritizing which user stories and functionality would accumulate most return on investment. In this research the business value aspect is left out of the scope and the improvements and benefits are studied from the aspect of planning and executing agile software development.

McDaid et al. (2006) and Cohn (2005) agree that problems in creating schedules are usually the unexpected elements in the environment. The effort needed for completing the user story could be estimated accurately, but all the meetings, problems in the environment, difficulties with the technologies, testing and other matters which prolong the completion are hard to always take into account. Schedules need to be done again after these unexpected issues push the completion dates forward. Agile software development teams

do not want to use too much resources on planning and making schedules. They want to concentrate on the developing.

Usman et al. (2014) state that there is a great need for more research on the effort estimation topic in the context of agile software development. Accurate approaches or tools for effort estimation have not been widely accepted. Buglione & Ebert (2011) present different commercial software estimation tools, which are used to estimate the effort of a user story. All these tools use the metric lines of code, which is said to be unfitting for estimation purposes. Coelho & Basu (2012) agree and state that lines of code cannot be accurately used to estimate the size of functionality.

This research is done to investigate possibilities of improving agile planning and feature estimation by creating an artifact which is using scrum history data of an agile software development company. As the expert opinion has been considered the most popular and accurate way of effort estimation, it is used in this research instead of creating a tool which would generate the story point values. The story point values are included in the scrum history data which is analyzed in this research.

To find answers to the previously mentioned problems the main research question is:

- How to improve feature estimation and agile planning using scrum history data?

This main research question can be dissected to a smaller sub research questions which need to be answered to answer the main research question.

- What kind of problems can be identified related to agile planning, feature estimation and scrum history data?
- What kind of artifact could help solving problems in feature estimation?

This research is done as a case study to a Finnish software development company which develops and sells an information management system product all over the world. The research is conducted because the case company has noticed possible problems in feature estimation and release planning. The case company gathers considerable amount of data weekly about themselves but does not use the data as effectively as they could. This research aims to solve problems of feature estimating by creating an artifact, which provides useful information about the features, their progression and estimated completed dates. The artifact is in a form of a dashboard, which is automatically updated several times a

day. Some processes related to agile software development are discussed and how they affect the accuracy of the estimates provided by the artifact.

The research was executed as a qualitative case study, and it uses the design science research method framework provided by Peffers et al. (2007). The DSRM framework is used to create an artifact to solve identified organizational problems. The solving of the problems must yield value to the organization and the artifact must be evaluated thoroughly after the design and development. The unique environment of the company pushes the research towards case study and the DSRM framework provides a solution in a form of an artifact which fits well with the research question and the needs of the case company. The subjects related to the research question are complex and qualitative data needs to be gathered. Semi-structured interviews were chosen for this purpose and 14 employees from various positions of the case company were interviewed.

The paper is constructed of theory chapter, research methods chapter, research results chapter, discussion chapter and a conclusion chapter. It begins by first presenting the relevant agile software development methods and concepts using the academic literature. This information is needed for understanding the context of the results chapter. Secondly, the selected research methods are presented and justified by academic literature, followed by the results provided when the methods were used in practice. The next chapter is a discussion chapter which compares the results of the research and the academic literature. Finally, a conclusion chapter which summarizes the generalizable answers to the research questions, presents the theoretical and practical contributions of the research and discusses about the limitations of the research as well as suggestions for continuing the research from other aspects.

# 2. AGILE SOFTWARE DEVELOPMENT

In this chapter the most important methods, concepts and models of software development are explained and a brief history of the evolution of the methodologies used in the field presented. Scrum methodology is explained more thoroughly than other alternatives. Agile planning, effort estimation and release planning are the key concepts which are explained. These facts are needed for understanding the environment and context where the research was conducted.

Agile software development can be explained by first going through what it is not. The quite opposite of agile methods is the old waterfall model. Petersen et al. (2009) state that the first publications of the waterfall model are from 1970 and it has been used in software development to this day. The waterfall model consists of several steps, which are done in this order:

1. Gathering requirements for the system
2. Designing the system
3. Implementing the system
4. Testing the system
5. Maintaining the system

According to Sommerville (2011) the model states that before moving to the next step, the previous step must be finished. Completion of a step is usually done by creating and reviewing a great number of documents regarding the step. This rarely happens in practice since the information from other steps is needed and problems might arise only in the other phases of the model. For example, some problems in the design could only be realized in the implementation phase and a slight overlapping of these phases would be beneficial.

In some small projects with exceptionally clear requirements this model might work well, but if the project is sizeable and takes considerable amount of time to complete, some common issues have been noticed to emerge during the project. Sommerville (2011) and McBreen (2002) agree that the creating and approving of documents takes effort, money and time which could be spent better. This also slows down the whole process and stiffens

it. Other common issue with waterfall model is that the customer usually changes their requirements, because they do not know what they want, and they might not even know what is possible to implement. The model does not take this into consideration. Jones (1996) mentions that in this process model the customer cannot provide feedback on the system and in a worst-case scenario they receive a complete system, which does not do what they want. He also points out that since testing is done after the development it is usually the place to excise resources if the project goes over budget or is delayed, which is not rare in software development projects. Neglecting testing will build up more costs later when the customer discovers bugs which need to be fixed.

After realizing the issues in the waterfall model, new methods and practices were developed. In 2001 Beck et al. (2001) presented a collection of values and principles for better software development. This collection is called Agile Manifesto and it is the base for all the most popular methods used in agile software development. Agile Manifesto states that they value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

Agile Manifesto clearly responds to the shortcomings of the waterfall model. Cockburn (2002) explains that agility means the ability to adapt and respond to the changes happening in the environment. In practice the agile methods usually include breaking the project to smaller subprojects or increments, which incorporate working and tested parts of the software to the customer. It is important that the customer can provide feedback to the developers when these smaller parts of the software are delivered. This ables the software companies to react to the changes in customer's requirements and in the environment. Doing a project in smaller pieces and collaborating with the customer as often as needed ensures that the customer receives what they want, and problems are noticed and fixed as soon as possible.

According to Miller (2001) and Beck et al. (2001) agile software processes usually use same iteration time throughout a project to keep deliveries consistent and ease up the planning of the project. Iteration times are usually short and vary in different projects

from one week to six weeks. Reflection regarding own work and processes should also be done repeatedly after every iteration.

Beck et al. (2001) mentioned **individuals and interactions** as important part of agile software development. In practice this means that individuals should be provided with proper tools and environment where the work is done to increase productivity. They should be supported, trusted and not harassed with deadlines or constant monitoring. Communication is essential part of the interactions and it should be bolstered by appropriate tools and processes.

Many different agile software development methods have been used in the software development industry which have taken the agile manifesto's principles and values to use. Some of the most well-known methods are Extreme programming (Beck 2000), Scrum (Schwaber & Beedle 2002), feature driven development (Palmer & Felsing 2001), test driven development (Astels 2003) and KANBAN (Ahmad et al. 2013). These methods have similarities, but some of the methods are whole processes which can be used to develop software, and others are just approaches which can be used inside processes. Only Scrum will be presented more thoroughly, since it is the process model which is used in the case company.

## 2.1 Scrum

Takeuchi & Nonaka (1986) were one of the first to use the methods that later was called Scrum. The term is from sport of rugby and it means a mechanism in the game when ball is unplayable, and it is taken back in to play. Ken Schwaber (1995) published the first papers regarding Scrum and software development and since then he has been actively contributing to the literature. According to Rubin (2013) Scrum is agile process model for developing innovative products and services. It is used most when developing software products, but the principles of scrum can also be used in other fields as well.

Schwaber (2004) and Rubin (2013) both begin the explaining the scrum model from the product backlog, which is a prioritized list of requirements that the product must be able to fulfill. The development work is done in iterations called sprints and these are usually from one to four weeks long. During this time the team should be able to complete some small parts of the whole product that can be presented to the stakeholders. It is desirable to deliver new functionality to the customer in every sprint. In the beginning of a sprint a

meeting is held where the sprint is planned and at the end of a sprint a meeting where the completed work of the current sprint is reviewed with the stakeholders. These two meetings can usually be done one after another, since after ending a sprint a new one begins. In the sprint planning meeting the team estimates what can be done during the next sprint and they move those requirements from the product backlog to the sprint backlog. During the sprint it is not allowed to add more requirements from the product backlog to the sprint backlog. Daily scrum meetings are also part of the process, which are usually held in the morning before anything else. In these short 15-minute meetings all members of the team report what they have done yesterday and what are they planning to do today. This helps with the communication, since time is reserved every day for informing others what has been done and what is planned to do next. Possible problems are also discussed and tried to solve in these meetings. In the next figure 1 the scrum process is demonstrated with a picture.

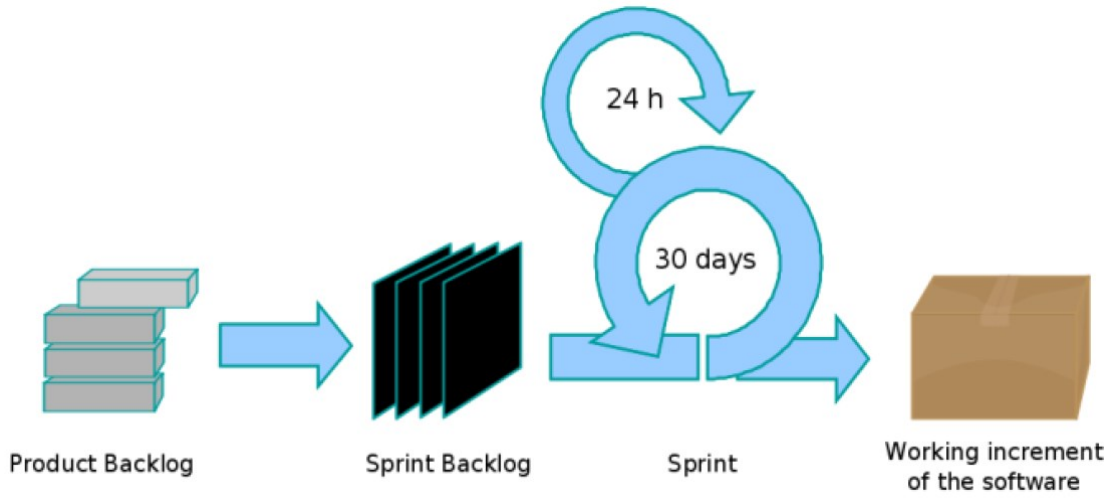



***Figure 1.*** *SCRUM Process Model (Way et al. 2009)*

Schwaber & Beedle (2002) argue that a Scrum team should ideally consist of about five to ten people. If more people are working on the project, multiple teams should be created. Schwaber (2004) identifies three roles in the scrum team: Product Owner, Scrum Master and the Scrum Team. Product Owner is responsible for the project, managing it and keeping the product's backlog updated. Scrum Master is responsible for the scrum side of the project, which means that the whole team follows Scrum's principles, rules and methods. The Scrum Team is the team which is responsible for turning the backlog into working functionality of the product.

## 2.2 Agile planning and effort estimation

Agile planning and previously mentioned effort estimation are important parts of agile software development. Cohn (2005) argues that the purpose of planning is to find answers to questions like:

- What are we developing?
- How are we developing?
- When are we developing?

Answering to these questions provides the team a schedule, available resources and a goal to strive for. Planning and forecasting the future is not easy and it is much about managing the risks of the unknown. Cohn (2005) states that agile planning is more about the planning rather than creating a plan, which suggests that changes are welcome and reacting to them is essential, rather than trying to follow the originally created plan. Following initially made plan strictly to the end is not agile and the information gathered during the project is not used as effectively as it could be. It is rare to have all the relevant information in the beginning of a project when the initial plan is formed. More information should be gathered during the project by for example saving data about the processes and by monitoring the work done. This consumes more resources, but it provides better understanding regarding the estimates and risks. One interesting and accurate model related to agile planning is called the cone of uncertainty, which is presented in the figure 2 below.

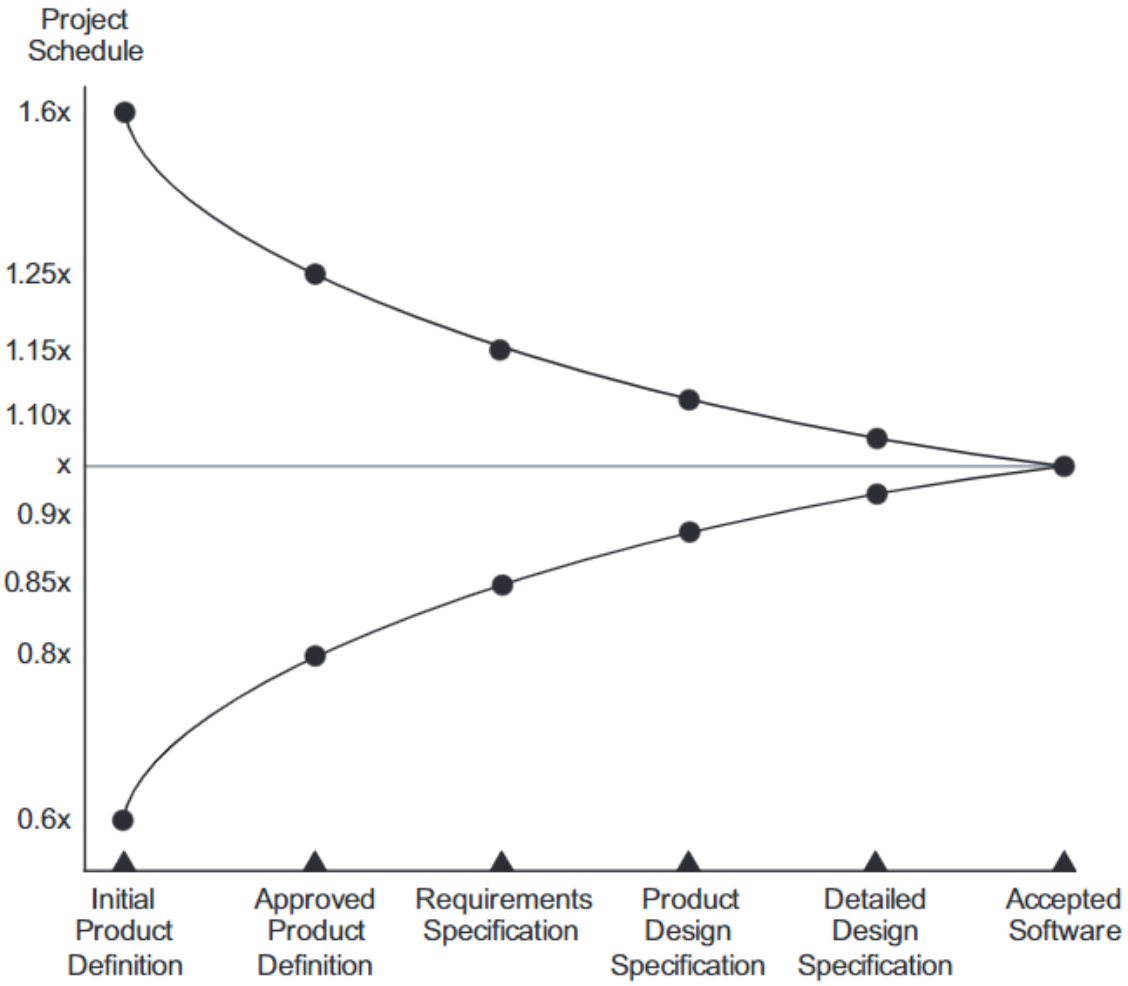



***Figure 2.*** *The cone of uncertainty (Cohn 2005)*

This figure can be interpreted that as the project progresses and more information is gathered, the estimation will also become more accurate. The steps in the figure's x-axis are just examples what could be included in a project, but the main point is that when something happens it can be reacted to, which leads to better understanding of the whole project and its estimation. As the workload is diminishing, less variables and fewer possible unbidden problems are expected to occur. If the initial plan is followed, relevant information is left unused and risks and estimates are not as accurate compared to the way of agile planning.

Schwaber & Beedle (2002) bring up an important part of scrum process which is called effort estimation. Effort estimation is an iterative process where the backlog items are evaluated for their size, meaning how long should it take to complete such item. Usman et al. (2014) suggest that the most used metrics in effort estimation are story points and use case points. Both metrics measure the size of a feature or requirement to be build. For example, the traditional metric lines of code is not commonly used in agile projects, since

it does not provide much useful information. The most used effort estimation techniques were Expert judgement, planning poker and Use Case Points -method. All these methods include experts who have experience on the subject, discussing the estimates together in a group. Cohn (2005) states that expert judgement as a technique is what it sounds like. An expert on the field will offer an estimate based on intuition and knowledge possessed by the expert. In practice the developer or the team which is developing a user story will come up with the points. Expert judgement might include some other techniques for example, estimation by analogy. Estimating by analogy means comparing a story to other stories which are already estimated. Stories should not be compared to one single sample, but rather to several other stories which provides more information to base decisions on. Johnson et al. (2000) state that after testing many methods the expert judgement was the most accurate. Even though several other methods which included analytical tools and software were tested. Other estimation tools included for example linear, exponential and logarithmic regressions.

Buglione & Ebert (2011) state that estimation and measurement is used insufficiently across majority of organizations and they have a few suggestions which can help making estimation better. First of all, an organization should always gather data about their own work and the data should be on the right level. It should not be too detailed, because the unnecessary data will fill up the hard drives and be in the way when trying to analyze the important part of the data. The data should not be too high level either, because the data might then be too simplistic and cannot be used for anything. The data usage processes should be planned well, which includes for example that the data is gathered to working databases and some verifications are done to confirm that the data is accurate, and it presents the right metrics. After the data is gathered it can be analyzed and turned in to information which brings value to decision making, reporting and communication.

### 2.2.1 User stories

According to Cohn (2005) and Lucassen et al. (2016) user stories are one of the best ways to split and describe requirements and features. They are one sentence long descriptions of the functionality from the user's or customer's perspective. For example: "*As a user I want to be able to search for songs by their artist's name*". User story is a high-level description including only the most crucial parts of the requirement and it can usually be created quickly. This is useful when discussing requirements with the stakeholders and it is not necessary to plan functionalities on a technical level.

The size of a user story is estimated using the previously mentioned story points. Cohn (2005) and Coelho & Basu (2012) agrees that story points try to estimate the complexity, effort and risk when developing a user story. The points are abstract, but relative to each other, which means that two-points story should be two times more effort consuming, risky and complex than a one-point story. A new concept called velocity is also introduced, which reports how fast a team is developing user stories. In simplicity it presents how many points a team is able to complete on average in one iteration. This number can be used to estimate when the whole project is completed, if all the wanted user stories are already on the project backlog.

Cohn (2005) states that he has had success with story point scales of 1,2,3,5,8 and the same scale without the 5. With the 5 included in the scale, it is the beginning of the Fibonacci sequence where the last number is added to the current to generate the next one. The logic for using the Fibonacci sequence is that the gaps are growing as the size grows and this is also true in the estimation. It is harder to estimate the size of a user story as it grows, since more uncertain elements are present.

Estimating effort accurately is hard and uncertainty is always present. McDaid et al. (2006) state that many factors affect the accuracy of estimates. Software development is usually rather complex matter and it is hard to estimate accurately what is needed for the development of different functionalities. New technologies can take surprisingly great time to learn and the experience of the developer has a great impact on the estimations. Story point values are usually difficult to compare to calendar days, because various other tasks like fixing bugs, attending to meetings and communication with stakeholders need to be handled also. Furthermore, sick leaves, problems in internal systems or other unexpected situations affect the time to complete a user story. Ktata & Lévesque (2010) state that some Scrum Experts estimate the average error of planning poker estimates to be about 20%. This is considered acceptable error and it is mostly affected by the experience of the developer who is estimating.

Ktata & Lévesque (2010) have gathered some problems related to effort estimation and data collection. It is common to not change the story point values, if the estimation was wrong. Updating the story point values would help improving future estimates and if the history data is used in any way, benefit of better presentation of reality exists. The work items should be defined distinctively to not have any vagueness when trying to measure

them and metrics related to them. Re-opening a user story after it has been closed is also known to make the data more inconsistent and harder to analyze.

Coelho & Basu (2012) and Cohn (2005) agrees that using the history data of story points and velocity in estimation requires few attributes from the environment. The environment should be like the environment from where the data is gathered. The elements in the environment include processes, developer teams, technologies, tools and the domain. If any major changes occur in these factors, the usage of history data might provide less accurate results. In the figure 3 below a process model is presented which demonstrates how a schedule can be made using the scrum data.

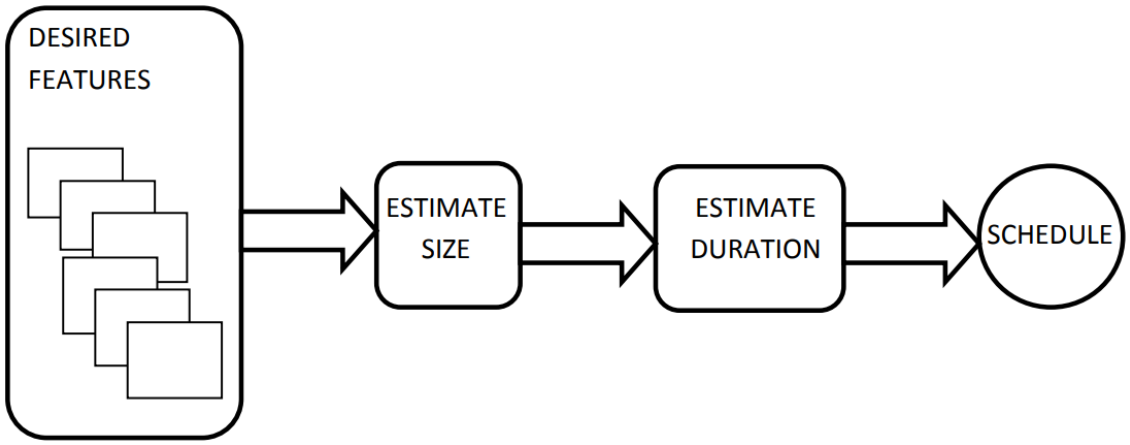



***Figure 3.*** *How to create a schedule with scrum data. (Coelho & Basu 2012)*

First the user stories are made from the required functionality, followed by the assigning of the story point estimates of the effort needed complete the user stories. After that a duration can be estimated by adding all the other activities, which take time out of the development and finally a schedule can be made from these estimates.

The purpose of this research is to find approaches to improve feature estimation using scrum history data. Scrum history data includes data about the user stories, which could be used in estimation. Different aspects regarding the truthfulness of the scrum history data are also considered in this research. Additional possibility to improve the feature estimation is to have the estimation take less resources by for example proper presentation of the data.

### 2.2.2 Release planning

Greer & Ruhe (2004) state that incremental development and delivery are done because it provides more flexibility and enhances customer satisfactory. This is the point where release planning becomes relevant. The developed features should be incrementally delivered to the customers, in order to receive feedback frequently. According to McDaid et al. (2006) and Cohn (2005) release planning is the process of deciding what functionality will be included in each software product version and when are these versions released for the customers. Fixed issues are also usually updated in the next release, unless the issues are critical and need to be fixed immediately. McDaid et al. (2006) argue that the most important question regarding release planning is what user stories and features should be prioritized. Prioritization should be done based on the business value and returns on investment which the story or feature provides. The current state of the features and estimated complete dates also affect the decisions of prioritization. Resources need to be managed differently if features are not being developed as fast as planned. Logue et al. (2007) state that release plan can be used in strategic planning activities like customer training and product promotion. When creating a release plan all the affected stakeholders should be considered because they might see the usefulness of the released functionality differently.

McDaid et al. (2006) and Cohn (2005) agree that release planning can be done in a few different ways. One option is to set fixed dates for releases and estimate which features are included in that release. Other option is to do it by first choosing the features which would be in one release and start estimating the release date afterwards. Dependencies between user stories and features are an important factor which must be taken into consideration in release planning. It is possible that in some cases certain functionality must be added before something else can be implemented. Effort estimation has an essential role in successful release planning. The more accurately the completion of stories and features can be estimated, the greater the probability of a successful release. If releases do not contain the functionality they are promised, customers will not think the organization is trustworthy and its reputation can be easily ruined. Furthermore, Cohn (2005) states that release planning helps the development, because short term goals create a clearer goal to aim for and the development does not feel like endless labor.

In this research improving release planning is linked to the improved effort estimation and data presentation. As the release planning is highly dependent on the estimates and

their accuracy it can be assumed that by improving estimates and by presenting information regarding them, the release planning can also be improved.

# 3. RESEARCH METHODS

This chapter explains how the research was conducted. Information about the case company and their processes are shed light on. The common traits and benefits of design science research, a case study and a qualitative study are introduced along with the process model of conducting design science research. Lastly the executed actions in each step of the DSRM model in this research are presented.

## 3.1 Qualitative case study in a finnish software company

This research is conducted as a qualitative single case study. Saunders et al. (2009) and Yin (2017) describe case study as a strategy of doing research on a specific phenomenon in its own context and using several sources of data. Case studies are exceptionally well suited for answering questions "why?", "what?" and "how?" in the context of the research. However, since the environment and context are usually almost unique, the generalization of the findings to the whole field is difficult. Case studies can be single case studies or multiple case studies and pros and cons can be found from both. When conducting a single case study, the researcher can focus on the one case more thoroughly, but the results are even less prone for generalization. Saunders et al. (2009) and Yin (2017) state that it is common to use multiple sources of data and triangulate them. The term triangulation refers to the action of combining information by using multiple data gathering methods to improve the trustworthiness of the findings.

Selecting case study as the research strategy came naturally, since the author worked in the case company and the case company asked for this research to be conducted. They wanted concrete solutions to problems they had discovered, and a master's thesis was an appropriate option to have someone focus on the subject. Problems were in the almost unique environment of the case company and a more general research on the topic might not have been as effective as a single case study. The less generalizable research results are not a problem to the case company, since their primary goal is to solve a real-life business problem. On the other hand, the contributions to the academic literature might not be on par with a multiple case study.

This research is conducted in a case company located in Tampere, Finland. The case company develops and sells an information management software and services around it.

The company has about 600 employees around the world. This research is done in the Product Development division of the company. The development, testing and releasing of the software is done in this division and the author's position is also in this division.

The software is developed using scrum practices, but since the company is developing their own main product, the process is not exactly the same as used in companies which develop custom software projects to specific customers. The product development division consists of several development teams which all have different fields they are specialized in and one Quality Assurance team which tests the user stories of all the teams. Teams have two-week sprints and they use user stories and story points in their processes. The Fibonacci sequence has been commonly used when the story points are assigned. The different development teams have the same processes but might focus on different kind of software development. For example, some teams are in contact with the customer and some are not. Others develop software for web usage and some for mobile devices. The teams within the company are fixed and the personnel do not change often. After assigned to a team, the employee usually stays in that team. Some work is done in virtual teams and over regular team boundaries, but most of the work is done inside a specific team. The fact that the employees stay in the same team for a long period of time and usually use the same technologies developing new features to the product establishes better possibilities for accurate effort estimation.

The product development teams use Azure DevOps information system to keep track of the past, current and planned epics, features, user stories and tasks. This system holds all the Scrum data and is the main tool for the software development project management. The tool is most often used in sprint retrospectives and sprint planning meetings, where the information about current and future user stories are discussed. More information about Azure DevOps is presented in the results chapter.

The company's main product has one-month release cycle, which means that every month a new release version is published, and it is available for download for the customers. The company also has a public roadmap for the customers, which informs the expected functionality in the releases for the following six months. Product management division is responsible for the roadmap and keeping it up to date. This creates difficulties, because the communication regarding the development of the features must be done well with the product development and product management departments. Product management has

different product managers assigned to different teams and they all have their own responsibilities. The case company is in need of a concrete solution to improve feature estimation which would also help release planning and communication between different departments.

## 3.2 Design science research methodology

Hevner et al. (2004) state that design science is a problem-solving paradigm which aims to reach its goal by producing innovations. These innovations must be designed and created after thorough planning and an analysis of the related problems. Design science research methodology was proposed by Peffers et al. (2007) after several years of design science research done in the field of information systems without a commonly accepted framework. Design science is valued highly in most of the engineering disciplines, because it provides actual artifacts which solve identified problems.

Hevner et al. (2004) provide practical rules on how a design science research should be conducted. The most critical part is to create an artifact which is targeted for an unsolved and important organizational problem. The artifact should be evaluated thoroughly on the aspects of quality, utility and efficacy. Existing theories and knowledge regarding the field and type of the artifact should be utilized in the development of the artifact. And finally, the artifact and the whole research should be properly communicated to the relevant audiences in the organization.

Peffers et al. (2007) introduce the DSRM Process Model, which consists of 6 activities identified important when conducting a design science research. These activities are:

1. Problem identification and motivation
2. Define the objectives for a solution
3. Design and development
4. Demonstration
5. Evaluation
6. Communication

These six activities, which are also displayed in the following figure 4, were used as guidelines to perform this research.

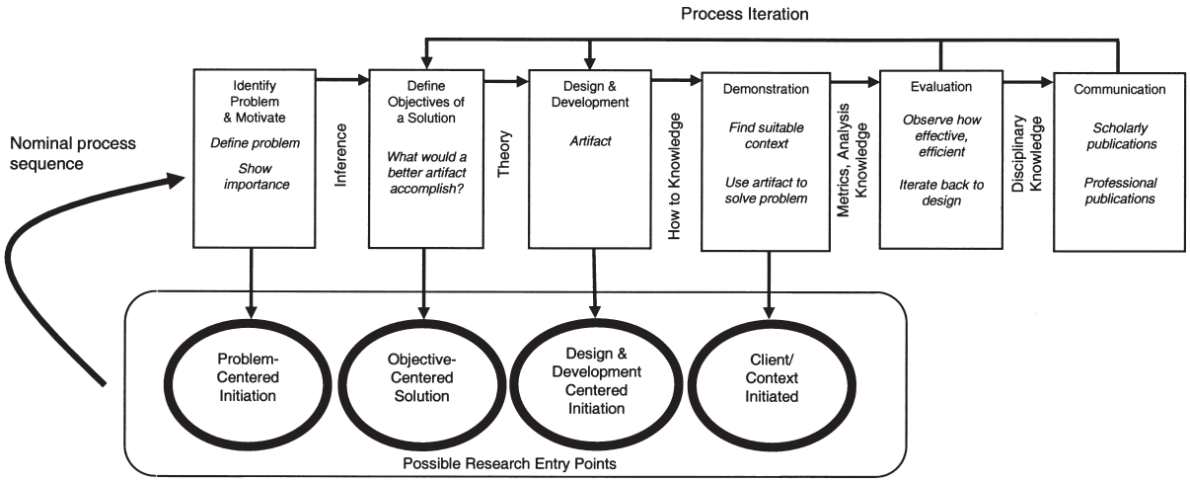



***Figure 4.*** *DSRM Process Model (Peffers et al. 2007)*

Hevner & Chatterjee (2010) argue that many different methodologies and frameworks exists for design science research, which have valuable guidelines. It should be noted that these frameworks and methodologies are useful only if they are suitable for the context and the environment of the research. Next the previously mentioned six steps are explained more thoroughly by presenting examples.

Peffers et al. (2007) state that in the first step the research problem is identified, and motivation sought to justify the solving of the problem. This is important step for the whole process, since it provides the goal for the artifact and it also analyses the value what would be achieved by solving the problem. Only important problems should be solved. It has also been noticed that splitting the problem into several smaller problems might help understanding the complexity of the problem. Several methods exist which can be used to identify the problems and their relevance. According to Becker et al. (2009) interviewing relevant people is an advisable scientific method for gathering the needed qualitative data for a research.

According to Peffers et al. (2007) the second step in the process is to use the identified problems to define the objectives for a solution. The objectives should be analyzed well on the aspects of feasibility and sensibility. The objectives can for example answer questions like: How does this artifact outperform the former solutions? What new benefits does this artifact accumulate? Walls et al. (1992) propose that it is beneficial to split the objectives to smaller pieces which they call meta-requirements. Splitting complex objectives to more simpler bits could help in the designing phase. The objectives do not go into technical detail in this step, they just state the goal what the solution should be able to do.

Hevner & Chatterjee (2010) state that the next step consists of deriving the needed functionality from the objectives, coming up with a design for the artifact and finally developing it by combining the design and functionality. The artifact can be in almost any form. It can be a model, method, software or some other tool. The important part is that the design is done by taking the objectives and research acknowledgements in consideration. (Peffers et al. 2007)

Peffers et al. (2007) describe the actions in demonstration step as the use of artifact to solve one or more instances of the problem. It can be done by using an imaginary case as an example to test the artifact. The purpose of this, is to inspect how the artifact solves the problems which were identified, and does it fulfill the requirements which were set to it. This step is just to demonstrate the artifact and the more thorough evaluation is in the next step.

Evaluation is the more thorough and rigorous step where the artifact is tested and evaluated in the aspects of quality, utility and efficacy. The evaluation can be done in many ways and methods depend on the artifact's type and its environment. For example, by comparing the real results the artifact produces to the objectives and requirements, doing a satisfaction survey, asking feedback from relevant stakeholders or by using some quantifiable measures and metrics. (Peffers et al. 2007; Hevner & Chatterjee 2010)

The sixth and final step is introduced by Peffers et al. (2007) as a step where the problem, its importance and the solution in the form of an artifact is presented to the relevant audiences. These audiences can include for example researchers in the field and all the relevant people in the organization where the research takes place. The artifact's design, effectiveness and novelty are important factors which should be shed light on.

### 3.2.1 Collecting data with a qualitative approach

Saunders et al. (2009) points out that research is often divided to qualitative and quantitative research. Dey (1993) and Robson (2002) describe the difference of these two by saying that quantitative data is "thin" and qualitative data "thick" or "thorough". The quantitative data comes in masses and the results are analyzed in numbers and statistics. On the other hand, qualitative data is more complete and deeper. Silverman (2013) describes the data sources in qualitative research to have their own unique voice and they are distinguishable from the masses. He also states that qualitative data usually answers better to questions like "how?" and quantitative data to "how many?".

Interviews are one possible method for gathering data for a research and according to Saunders et al. (2009) interviews can be categorized to structured, semi-structured and unstructured interviews. Healey (1991) categorizes interviews to standardized and non-standardized interviews. Semi-structured and unstructured interviews are these non-standardized interviews and they are considered to be better suited for gathering information to qualitative researches. Structured interviews are used more often in quantitative researches. (King 2004)

Drever (1995) and Saunders et al. (2009) state that when conducting a semi-structured interview, the researcher chooses general topics and themes which can be in a form of several open-ended questions. The researcher should follow the "flow" of the interview and try to gather information from the interviewees by asking thought-provoking follow-up questions and trying to keep the conversation ongoing. It is important to inform the interviewee about the context of the interview, why is it conducted and how is the gathered data used. It can help the interviewee understand the interviewers point of views better and figure out what kind of information the interviewer is after.

According to Saunders et al. (2009) and Silverman (2007) in some cases qualitative interviews are a desirable way to gather data. One of the situations is that the researcher needs to know the attitudes, opinions and the reasoning behind the decisions of the interviewed people. This is hard to uncover by using a questionnaire for example, since it is not possible to ask more questions regarding the topic. The open-ended questions might also lead the conversation to paths which were not anticipated and thus gathering information and produce new ideas which would be left undiscovered otherwise. Healey (1991) and Saunders et al. (2009) also agrees that arranging an interview is sometimes a prerequisite for receiving any data out of relevant people, since attending an interview is seen more interesting than answering to a simple survey or questionnaire. It also affects the quality of the data collected, because people are often more likely to discuss about sensitive and confidential information when a personal contact has been established.

Using semi-structured interviews in this research was obvious choice after considering the options to gather data. The questions to be asked seemed more open-ended and most of them were beginning with a "how", which indicates that semi-structured interviews would be the most beneficial for gathering this type of data. The needed data included people's actions, opinions, attitudes and reasoning for them. Semi-structured interviews enable the gathering of this in-depth and more personal information which was needed in

the context of the research. The subjects discussed in the interviews were complex and somewhat sensitive, hence creating a superficial survey or questionnaire for the whole product development would have been less fruitful. The group of potential interviewees was not excessively large either, therefore it was not a problem to hold interviews for all the relevant people.

Neuman & Robson (2007) state that the sampling should be considered when conducting interviews for a research. Qualitative and quantitative studies usually use different kinds of sampling, since the focus of the interviews is different. In qualitative researches it is common to use sampling to better understand events, cases and environments in specific contexts. These samplings are called nonprobability samplings and common samples in this category are for example quota sampling, haphazard sampling, purposive sampling and snowball sampling. Saunders et al (2009) and Neuman & Robson (2007) describe purposive sampling as a principle where the researcher chooses all the possible cases using specific criteria. Criteria should be set to help answering the research question as much as possible.

The used sampling in this research was purposive sampling. The author selected all the interviewees by asking his supervisors and the first interviewees for their opinion who should be selected for interviews. The criteria were that people who contribute to the generating of the used data, are involved in the processes or would use the created tool should be interviewed. In addition to that, people from every hierarchy level and from different positions should be interviewed to probe opinions and attitudes from different aspects. Positions interviewed and the reasoning for choosing them are presented in the table 1 below.

Table 1. *Positions of interviewed employees and reasoning for choosing them*

| Position | Reasoning |
|---|---|
| Program Manager | Has knowledge of the processes and uses the Azure DevOps -system which provides the data for research. Also is one of the main users of the solution that the research would provide. |

| | |
|---|---|
| Software Developer | Is part of the software development team and has opinions about the processes and systems. |
| Team Manager / Scrum Master | Contributes most to generating the data to the Azure DevOps -system and knows how the team follows the processes. |
| Product Manager<br>(Director of Product Management is also in this category.) | Is interested when the features will be completed and would use the solution which is to be created. |
| Director of Quality Assurance | Different perspective from QA point of view. Has knowledge about the processes and all the teams. |
| Vice president of product development | High level position and has information about processes and how they should be used in the case company. |
| Director of Software Development | Is next in the chain of command after developer and a scrum master for several teams. Is making decisions regarding resourcing and the actions of a team. |

In some cases, more than one person was interviewed from the same position. Different teams are used to different methods and processes are not always followed exactly as in the other teams. Three different teams were selected, and a developer and the scrum master were chosen to be interviewed. Two of those teams' directors were also interviewed to gather information from the whole chain of command. Two product managers and their director were interviewed.

## 3.3 Research design using DSRM

After the clarification of the topic and the research questions the DSRM-model was chosen for this research. It fitted well to the context, since it was already decided that an artifact was to be created to solve these organizational problems. The artifact would be a

tool or a dashboard which used the data of one of the case company's information systems. Next the actions executed in this research are presented using the steps of DSRM framework. In the previous figure 4 different research entry points were presented. The problems were not previously documented in the case company; thus, this research starts from the problem identification and motivation step, which suggests that it uses the problem centered initiation as the research entry point.

### 3.3.1 Problem identification and motivation

Problem identification occurred partly before conducting this research and it gave the spark to initiate the further researching of the topic. Semi-structured interviews were held to identify the problems in the case company and to estimate what value would the case company gain by resolving the problems. In addition to that the possible problems which would make the research more difficult were discussed and investigated.

14 members from different positions of the case company were interviewed to this purpose. The compiled interview questions were about the used processes, concepts, systems and attitudes towards them. Questions were also to probe what kind of benefits would solving these problems bring forth to the case company. The outcome of this research was known to be an artifact, which could be a dashboard, therefore some general questions regarding those topics were also asked, which could help designing the solution. The interviews were usually one hour long and had the same 9-11 questions or themes for all the interviewees. Different positions had difficulties answering to different questions, but this was intended, since the author wanted to also gather information about how those interviewees assumed tasks were executed in the parts of the case company which they were not familiar with.

The interviews were recorded, and some notes were written down during the interviews. The recordings were listened thoroughly afterwards and all the relevant information documented in the same document. All the answers to the specific question were gathered under the question and the questions were categorized for what purpose they were. This made the analyzing of the answers easier, since all the opinions on one subject were in the same location. The themes of the semi-structured interviews can be found from the appendix A.

The sampling was considered sufficient for the research, because most of the answers to the questions were occurring repetitively. Gathering the sample was relatively easy, since

all the employees work in the same building and the product development division is not excessively large. At least someone was interviewed from all the relevant positions in the case company. Some demographic information about the interviewees and details about the interviews is presented in a table in appendix B.

Most of the people in the case company were already aware of the problems, but a proper identification was never done in the form of interviews and documentation. The semi-structured interviews worked well, since more problems and ideas for solutions were discovered in these interviews. It was known that the feature estimation was not done by using any information systems and the scrum history data was not effectively used for this purpose. This information provided suitable topics for the interview questions. The questions investigated for example, how is the feature estimation done in the case company, could it be done better and is it accurate at the moment. Another question examined the benefits if these problems were solved by creating an artifact which would work perfectly. The questions also probed how the scrum data is inserted into the system and how does the attitudes and processes affect the quality of the data.

### 3.3.2 Define the objectives for a solution

The previously mentioned interviews yielded information about the problems regarding the topic of the research. The interviewees also had ideas for the solutions to these problems, which were used when identifying the objectives for the solution in this research. Using the identified problems, the author and his supervisors compiled the objectives for the solution, which would be the requirements for the artifact. The objectives were split to one sentence long requirements. These were documented and used in the designing and developing the artifact.

### 3.3.3 Design and development

The designing was done by utilizing the defined objectives and the information gathered from the interviews and the academic literature. Designing and the development of the artifact took place in the case company's premises and in its IT environment. The author did the designing and developing of the artifact and demonstrated the progress at least monthly to the two supervisors guiding the research. The supervisors gave feedback and their opinions on the artifact, which helped to improve it iteration after iteration. The artifact was designed and created based on the requirements gathered in the previous step. It was also improved and modified as the development went on and better ideas emerged.

### 3.3.4 Demonstration

After the first version was considered ready, the artifact was tested by the Program Manager, since he would likely use it the most. He tested it and provided feedback about the artifact. The first meeting where the artifact was presented to a wider audience could also be regarded as demonstration. Several people of relevance were invited to a meeting where the artifact was thoroughly presented. The attendees had questions and provided feedback regarding the artifact.

### 3.3.5 Evaluation

Many of the artifact's functionalities are easily evaluated, but one of the artifact's objectives is providing estimates when a feature is ready and that is a more complex subject. The logic behind these estimates can be tested with history data of completed features and how accurately the artifact would have estimated their complete date. The results are to be analyzed using the principles of mathematical statistics.

Error statistics must be calculated to investigate the possible errors in the estimates. Willmott & Matsuura (2005) present two common statistics for calculating the average error in model evaluation studies. These two statistics are the root-mean-square error, which is here on called the RMSE and the mean absolute error, which is MAE in short. Willmott & Matsuura (2005) compare these two and state that MAE is the most natural way to present average error magnitude and should always be used. In their opinion the RMSE is unambiguous and provides much higher values because the errors are squared. Chai & Draxler (2014) take the opposing side and present that RMSE is better to be used when the model errors follow normal distribution. They also illustrated that with over 100 samples the error distribution can be closely re-constructed, but with 10 or less samples neither of the methods are particularly accurate.

MAE is chosen to this research, since the currently completed features which are used as the number of samples for evaluation is under 50 and the RMSE was proven to be more accurate only when over 100 samples are used. MAE is also used and recommended by Shepperd & MacDonell (2012) and Choetkiertikul et al. (2019) in their studies related to software development effort estimation and measuring its error. The formula for calculating MAE is:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |e_i|$$

Where $n$ is the number of samples and $e$ is the difference between the actual durations of how long it took to complete a feature and from what would the artifact would have estimated for the duration. ($e_i,\ i = 1,2, \dots, n$). The errors are absolute; therefore, it does not matter if the duration was estimated shorter or longer and it is not biased to overestimating or underestimating.

In this step it is also decided what is sufficient for the case company in terms of fulfilling the requirements. The thought of using the error in percentages also was considered but decided to not be used since 50% error on a 20-day feature does not feel as significant as in 200-day feature. The days metric is more ambiguous and easier to understand.

### 3.3.6 Communication

This research paper is part of the communication of this study, but in addition to that, presentations and meetings will be held inside the case company to inform all the relevant personnel how the study was conducted and what were the findings. Also, the finished artifact is to be presented and training established for its use. The research will be publicly available in the university's database.

# 4. CREATED ARTIFACT AND RESULTS

In this chapter the work towards the artifact and results of the research are presented using the DSRM steps as guidelines. First starting with the problem identification, then to the definition of the objectives of the solution and further to the design and development of the solution. Finally, the artifact is evaluated by comparing the results to the objectives of the solution.

## 4.1 Identifying the problems regarding feature estimation

Feature estimation, release planning and making schedules is known to be hard in the volatile environment of agile software development. The interviewees commonly agreed that feature estimation can be improved. At the moment no tool exist which could be used to examine all the currently developed features fast and effortlessly. The feature estimation is currently done manually by checking the remaining points and approximating the team's velocity. Related to this, one of the scrum masters said:

*"The completion of big features is not estimated at all in the beginning. The feature is split to stories and they are given story points. In the later stages when the feature is close to the completion, we start thinking when this feature could be released. One thing affecting the estimates which is sometimes overlooked is the effect which the existing source code has to the new implementation. We want to develop high quality software and not just some quick fixes to get things working. The size of the software has been growing fast lately and it makes this problem even more difficult."*

Estimating features' completions has been considered hard, because many unpredictable variables exist in the case company's environment. Even though the effort estimations would be accurate, tasks of higher priority might emerge, which interrupts the current work and needs to be dealt with first. In addition to this, the employees have hard time estimating the time spent on source code merge reviews and user story testing. One of the developers commented how they currently estimate features:

*"By doing an effort estimation and a schedule. It is important to understand that effort estimating and making schedules are different things. Sometimes the effort estimations are done well, but the schedule does not hold, because something else must be done first.*

*Factors affecting the accuracy of an effort estimation can be for example bugs or unexpected challenges in the technology. Customer can also change their requirements in the middle of developing, which slows things down."*

Related to the same subject, one of the scrum masters thought that:

*"Some people might think story points as a time estimate and not effort estimate. They don't realize that if it takes longer to finish because of some external factors, it does not always increase the size of the effort needed completing the story. This should also be considered when talking about story points."*

The program manager explained more about the factors affecting feature estimation:

*"When estimating the features, the current story points, and the velocity is examined with the responsible scrum team. Things which affect the estimation's accuracy are for example something of higher priority which cut in the queue and must be completed first. High priority cases can be top 5 customers' requests, extremely harmful bugs and problems in the organization's systems. These must be handled fast. Priority has a big impact on the time when a feature is completed."*

The interviewees had some concerns about the accuracy of the effort estimations, which are a relevant part of estimating the completion of features. It turned out that the size of story points might vary between teams, which sounds reasonable, since story points are an abstract measurement unit. The teams might have slightly different methods for coming up with the story points, but all of them used expert opinion, which means that the developers who have knowledge of the field are present to offer their views. Teams determined the story points with the whole team, which is beneficial, since they are provided with opinions from different aspects. The program manager thinks that:

*"The values of the story points might have different sizes between teams and even between developers. Two points for someone is different size than two points for someone else, since it's an abstract unit of measure for the effort. It is also the best estimate at that given time with the information available."*

One of the scrum masters explain how their team comes up with the story points:

*"We determine the story points value for the story with the whole team and everyone can give their own opinion. This happens using post-it notes and comparing the new stories and what size do they feel like. We usually don't compare the estimates to the data of the*

*completed user stories. "We have a feeling for example how big is two points story and go with that. Method is planning poker or something similar."*

One of the product managers presented important observation regarding the accuracy of story points:

*"It is important that the developers which will do the development are included in the meeting where the story points for the user stories are decided. They know the best what they are capable of doing and how fast. They might also have more knowledge on that specific field."*

To improve the accuracy of the story points' effort estimations and comparability between teams, a common and standardized way for coming up with the story points should be decided. The vice president of product development stated that:

*"There should be better standardized guidelines for determining the story points. It doesn't matter what the method is, but all the teams should use the same method to make the story points more comparable between teams. One possible method is the planning poker which some teams use."*

The use of the Azure DevOps -system which has all the scrum data stored was also one point of interest. Possible problem regarding the system was that employees did not fill the data properly and that they thought that using the system was unnecessary bureaucracy. These problems would affect the quality of the data in the system. Employees' attitudes towards the system were positive and they knew the purpose of the system. The accuracy was also on the required level, since the user story data is filled properly, and more detailed information is not needed in this research. The program manager, who is one of the most active users of the system stated:

*"Teams understand the importance of the Azure DevOps and they fill in the asked bits of information. I believe that if something more is asked to be filled the teams will do so, if we have something to motivate them. A dashboard using the data would be great concrete motivator to fill in more detailed information."*

One of the scrum masters agrees and says:

*"Story points are filled strictly, but tasks under them are done by the developers. Some developers mark tasks and hours to them with detail, but some do not."*

One of the developers stated:

*"There has not been any signal that we should mark the tasks with more detail. We don't want to waste time documenting something that isn't used anywhere. Concrete evidence of the use of data would motivate to fill in information in better detail. For example, if user story is once set active, it will not be changed even if it is not developed at the moment."*

Previous findings only identify how accurately the data is written to the system, but it does not concern the accuracy of the content of the data. Another possible problem was that sometimes the story points might be estimated wrong. In these cases, it might be beneficial to update the story points value to the system. The interviewees had many different opinions regarding this kind of process. The majority was on the side of keeping the old values for the story points for various reasons. They also proposed better alternatives for the solving the problem. One of the developers stated:

*"The initial story points should not be changed afterwards, because it would affect the velocity negatively. Stories can be scoped again or split to several stories if they feel too big. If the story points would be changed there should be different columns in the data for the initial effort estimate and the realized effort."*

Some employees had different opinion and even saw benefits in changing the values of untruthful story points. One of the product managers said:

*"It is a bad habit that the story points are not changed if they didn't match the original estimate. It is also in the guidelines. If the points would be reviewed by the scrum master for example in the sprint review it could help the developers to estimate better next time. The scrum master is responsible for changing the points for the right reasons, for example if the story is prolonged because of vacation, the estimated effort might not have been wrong."*

One of the scrum masters usually changes the story points if they are wrongly estimated, but had mixed feelings about it:

*"When we are going through the stories in sprint reviews, I ask about the truthfulness of the estimate and change it if it didn't match the actual size. I'm not sure if it has any benefit in the big picture, if other teams don't do the same. If this would be included into the process, everyone should do the review to every story, not just few people to some stories."*

Intuitively it would sound smart to change the effort estimates afterwards to the actual sizes, if the data is used to estimate the completion time of future stories with the same size estimate. Splitting stories and creating new stories is also proper solution for providing more accurate story points values. The most important factor is that when untruthful story points are noticed they are somehow reacted to.

## 4.2 Benefits and value of improving feature estimation

Finding problems is important, but it is also essential to investigate what kind of value solving these problems would create for the case comapny. It is not wise to use resources in solving a problem, which does not generate any value. The interviewees saw benefits in an automated artifact which would present the currently developed features, their progress and estimate of completion. An idea of a dashboard was also presented. The benefits of the artifact included better communication, resource management, customer success, roadmapping and if the artifact would be automated, it would also save resources from the feature estimation. It was noted that since the case company has great amount of data, they should try to use it as effectively as we can.

The program manager's opinions on the benefits of the artifact were:

*"Everyone would benefit from that kind of tool. We don't have a tool which shows easily all the features and their status. That alone would be a great help even without the estimates. Automated tool would be great, since we don't want to use too much resources on making detailed plans. When something more important interrupts the development of a feature, the automated tool could then estimate the new completion date easily."*

One of the product managers stated that:

*"The biggest benefit would be that we can keep our promises to the customers, which would lead to better customer success. It is also hard to put features to the roadmap without any estimates. When we have estimates of the features' completed dates, we can better plan our resources and what to do next."*

Other product manager felt that:

*"It would help a lot with communicating, since there wouldn't be that big of a need for all the meetings which address the status of features and the current estimates."*

One of the scrum masters had an idea that:

*"It would also help with pricing in subcontracting, if we could better estimate how long it would take for us to do it.*

In the next table 2 the identified problems are gathered and value of solving them presented.

Table 2. *Identified problems and the value of solving them. Asterisk (*) marked problems are not the main problems of this research but might affect negatively it.*

| Identified problem | Value of solving the problem |
|---|---|
| Information of all the current features and their progress is not available in a convenient place. | Time and other resources would be saved if the information was communicated better. It would help in feature estimation and release planning. |
| Estimation done manually and no tools for estimating features. | Already gathered data would be used more effectively and the artifact would save resources. |
| Lack of automation in estimating features. | Automation would save resources, since the estimations would not be a need for updating the estimates manually. |
| *The story points are not generated using the same process. | More accurate and truthful story point sizes in the whole case company. |
| *The story points are not changed to their truthful value, if they are estimated wrong. | More accurate and truthful story point sizes in the whole case company. |

## 4.3 Objectives of a solution to improve feature estimation

After the problem identification, objectives for a solution must be defined. Using the identified problems, the requirements for the artifact can be made. The initial requirements for the artifact included:

- Display information about all the currently developed features.
- Refresh automatically.
- Provide the estimated complete dates for features as accurately as possible.
- Use scrum history data from Azure DevOps.

After investigating the possible solutions currently available the decision was made to design and develop the artifact by ourselves, since none of the found alternatives were viable. The artifact was to be created in a form of a *dashboard*. The dashboard's goal was to display the current state of all the developed features and estimate completion dates for them. It would use the scrum history data of the past features and user stories to calculate the estimates. All this data is stored in the Azure DevOps information system. Other interesting information which can be received from Azure DevOps is which teams are developing which feature and the planned release month. Another relevant bit of information is the progression of a feature, which can be reported by for example presenting the last date a user story was completed in this feature. If the teams are consistent with their work, it is expected that this history data could be used to estimate when the current features and user stories are completed. Unexpected elements exist in the environment which cannot be considered in the solution and some inaccuracy must be accepted. It is just the best guess using the data of past user stories.

As said before the effort estimation and schedule are two different matters, the solution must take this into consideration and adjust the estimates if no resources are put into a feature. The solution should be automated to save resources. As a result, only the designing, developing and maintaining the solution would consume resources. The iterative nature of the agile software development should be considered and as the feature progresses the estimates should also utilize the new information and update the estimates.

## 4.4 Design and development of the feature estimation dashboard

In the beginning of the research the author and his supervisors had some ideas of the design of the solution. The solution would be an artifact which would fulfill the previously mentioned objectives of the solution. The main objective of the solution was to provide essential information about the agile software development processes to the user, which included the current status of the feature, estimated completion date, name of the team and the title. Therefore, the form of the artifact was decided to be a dashboard, since dashboards display information and they can be automated to refresh and present up-to-date information. The case company had some dashboards already in use and they were created and published with the Microsoft's PowerBI software. The case company owned license for this software and the author already had some experience and knowledge of

the software, hence it was considered to be the best possible platform to create the artifact. Azure DevOps is also Microsoft's service and it could be easily connected to the PowerBI with a connector which gave access to the data in the Azure DevOps. This could be tested in the early stages of the designing and development and after perceiving how effortlessly all the needed data was imported to PowerBI the decision was made to use this software. To sum it up, the dashboard was made using Microsoft PowerBI Desktop and the scrum history data was imported from Microsoft Azure DevOps.

### 4.4.1 Design

One of the interview questions was about dashboards. It investigated what sort of opinions employees had regarding dashboards and what type of information should the possible upcoming dashboards have, if such artifacts were developed. Many interesting ideas emerged, but not everything was related to this research. Few people wanted to see a similar dashboard which one of the product managers described like this:

*"A dashboard which could have a list of what the product development is developing at the moment and an estimate when would it be completed. It is important to also inform that the information is just an estimate and not deadline or promised date of completion."*

This description is rather accurate compared to the artifact which was designed and developed as a result of this research.

The design of the artifact was decided early in the development phase and it was to be a dashboard which was created by using Microsoft PowerBI. The dashboard would also be published in the web service of PowerBI where anyone with the case company's license could see the dashboard. The dashboard was designed to have columns and rows just like an SQL table. In the first iteration the dashboard had a single row for each feature and some basic information in the columns. The columns included:

- Team assigned to the feature
- The feature's title
- Planned release month
- Completed story points and total story points
- The completed percentage in a form of a progress bar
- Estimated days left to completion

- Estimated completion date
- Estimated days left to completion using the feature specific algorithm
- Number of days since a user story was completed to this feature

Special columns which needs more explanations are Feature done percent which was in a form of a progress bar and the two separated estimated days left columns. The progress bar was created by dividing the completed story points with the total story points. The progress bar indicated the percentage the feature was completed. The length of the bar would be derived from the percentage. At 100% the bar would be full and at 50% the bar would be at the halfway. The two different estimates were done by different algorithms, which are explained more thoroughly in the development chapter. The other one used average cycle times and other one feature specific velocity. Two estimates were provided because it was not clear how accurate the estimates were at that time and it was only assumed that if the two estimates were close to each other, they had higher chance of being accurate. The two estimates were shown as estimated days left to completion, because it was easier to compare days than dates. The first iteration of the dashboard which was published in the case company's PowerBI service is presented in appendix C.

In the first iteration a warning text with large red font instructing to not use the estimates anywhere was placed on the top of the dashboard and under it some notes about the information on it. As mentioned in the interviews, it is important to communicate that these are just estimates and not decided information, especially when the artifact is incomplete. The design changed during the development when feedback was received after each iteration. More of the final design will be in the Development chapter 4.3.3.

The automatic refresh and adapting to the changes were also part of the design and it could be established from the PowerBI web service. The refresh could be scheduled, and the timestamp of the latest refresh is presented in every dataset. At first the dataset was set to refresh 8 times during the normal working day between 8AM to 5PM. Every time this refresh happens the dashboard changes its values if the data has been changed in the Azure DevOps. The data usually changes during the sprint reviews and sprint plannings of a team, since that is the time when a team uses Azure DevOps the most. The information about the features' recent progress was wanted and it was implemented in a column which informed how many days ago the last user story was completed to that feature. A color coding was also added, which would paint the background of the column with

green if the value was under 14 days and started to turn redder after passing the 14 days mark.

### 4.4.2 Data in Microsoft Azure DevOps

When connecting to the Azure DevOps database, PowerBI asks what kind of data the user would like to see. The data is in SQL tables. All the accessible SQL tables' names and descriptions are presented in the figure 5 below.

| Name | Description |
|---|---|
| Bugs - Today | All Bugs for the entire team project. No history. |
| Stories - Last 30 days | All Stories for the entire team project. Last 30 days of history with dail... |
| Tasks - Today | All Tasks for the entire team project. No history. |
| Active User Stories | |
| Tasks - All history by month | All Tasks for the entire team project. All history with monthly intervals. |
| Tasks - Last 26 weeks | All Tasks for the entire team project. Last 26 weeks of history with we... |
| Bugs - All history by month | All Bugs for the entire team project. All history with monthly intervals. |
| Work Items - Last 30 days | All work items for the entire team project. Last 30 days of history with ... |
| Bugs - Last 30 days | All Bugs for the entire team project. Last 30 days of history with daily i... |
| Work Items - Last 26 weeks | All work items for the entire team project. Last 26 weeks of history wit... |
| Bugs - Last 26 weeks | All Bugs for the entire team project. Last 26 weeks of history with wee... |
| Work Items - All history by month | All work items for the entire team project. All history with monthly int... |
| Stories - Last 26 weeks | All Stories for the entire team project. Last 26 weeks of history with w... |
| Tasks - Last 30 days | All Tasks for the entire team project. Last 30 days of history with daily i... |
| Stories - Today | All Stories for the entire team project. No history. |
| Work Items - Today | All work items for the entire team project. No history. |
| Stories - All history by month | All Stories for the entire team project. All history with monthly intervals. |

***Figure 5.*** *Names and descriptions of all the SQL tables in Azure DevOps*

Many of the tables have same data but differently filtered and with different columns. The "Work Items" -table includes everything. For example, bugs, user stories, tasks, features, epics are all included in the same table and they are just separated with a column "Work Item Type". The tables with a specified work item type like "Stories" are just subsets of the Work Items -table. The Work Items -table has 109 columns and the Stories-table only 45, therefore it can be assumed that also the unnecessary columns are removed in the filtration. The different Work Items have a hierarchy, which clarifies the role of each work item. The most important work items related to this research are the features and the stories. Features are larger entities created by a combination of user stories. The Azure DevOps also allows to add user stories under Epics without the Features in the

middle. By not using the features, the process becomes less intuitive and it is not advisable. The hierarchy is demonstrated in the figure 6 below.

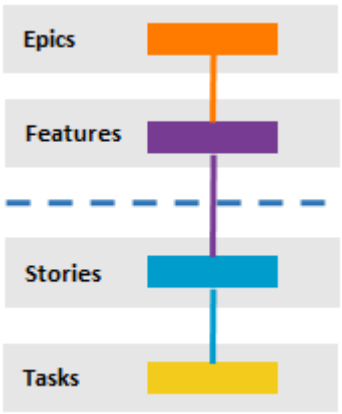



***Figure 6.*** *Azure DevOps hierarchy of work items (Microsoft 2018)*

One major difference between the tables is that some tables are history tables and others are "Today" tables, which means that the "Today" tables have the snapshot of the information today and the history tables have the history of every work item from every day since it was created. In "Today" tables each object has only one row for each object and in the history tables each object has a separate row for each day it has existed. Both types of tables are needed for the artifact.

For fulfilling the objectives of the artifact, the Azure DevOps data from the daily updated today tables of stories and features are needed. Also, the history of stories is needed for the evaluation of the accuracy of the estimates. The data is in the same SQL-style table format in PowerBI as the imported Azure DevOps data. The needed tables in PowerBI include Stories, Features, History of Stories, Calendar and Story points table.

All the tables in PowerBI are introduced in the appendix D with a short description of each column they have. The columns changed during the development, but these are the final columns. Some of the columns are made by the author after the original table was imported from Azure DevOps. These columns are marked with an asterisk (*) and some of them are explained better in the development chapter and different appendices.

The stories table is needed for the basic information of user stories. Using this table, it is possible to calculate the average cycle times for each user story size. The stories can also be linked to the feature by using the parent work item id. The next table is the Features tables. Most of the information in the artifact is taken from this table. It has the basic information of the feature and after combining it with the calculations from other tables

the relevant information can be presented in the dashboard with each feature on their own row. The Story Points -table was used to store the average cycle times of each story point size group. The story points column had all the distinct values of story points used in user stories. And these rows made all the different story point size groups. For example, one row had all the calculated information of user stories which had 2 story points assigned to them. The Story history table was used to calculate how many developers were developing each feature in each sprint. This table has distinct values of the combination of iteration path, assigned to and parent work item id. Which means that only one row is presented with the same values in each of these three columns.

The calendar table has just one Date column which has every date from 1.1.2018 to 31.12.2019 on their own row. This helps visualizing the data with dates. Usually data only has the specific date or timestamp and without the calendar table only those dates would exist. It is impossible to visualize the data related to time properly, if all the dates are not included in the data. Calendar table was mostly used in the evaluation.

PowerBI allows the tables to be linked to each other by using relationships. This allows tables to use data from other tables. These tables must have at least one column which has the same values to establish the connection. Without linking the tables, it would have not been possible to know which feature had which user stories under them. Calculations would not have been possible either, because the story point size groups' information about the averages could not have been transferred to the user stories data.

All other tables except the calendar table were linked to the Stories-table. The Story Points -table was linked to the Stories-table with the Story Points text -column. The Features-table's Work Item Id was linked to the Stories-tables Parent Work Item Id and the Story history -table and Stories-table were linked with Work Item Id -column. The PowerBI project's data model and the relationships are demonstrated in the figure 7 below.

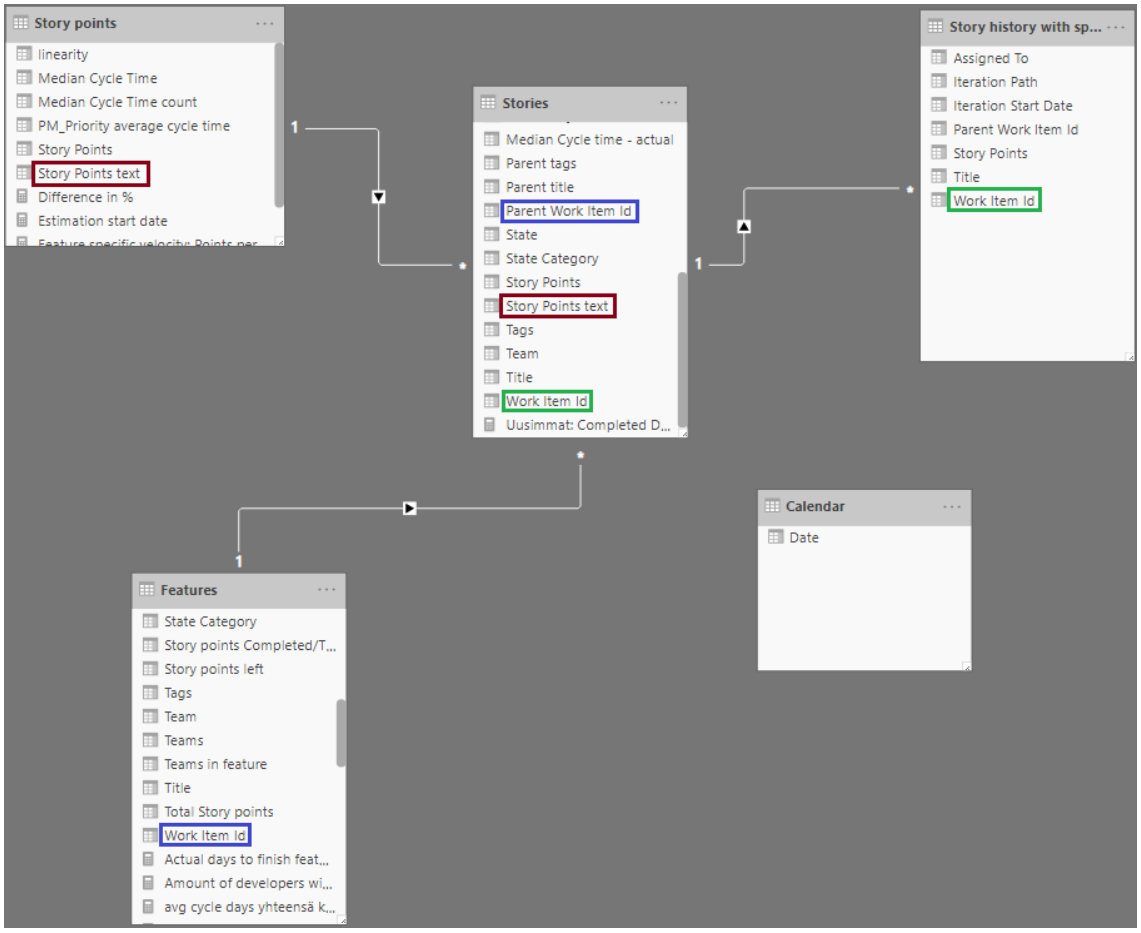



***Figure 7.*** *Data model of the PowerBI project and the relationships marked with different colors.*

### 4.4.3 Development

The development started after the Azure DevOps data was successfully imported to the PowerBI software. First the data was inspected, and each column reviewed if it had any relevant information for the artifact. After the first inspection, the data needed some "cleaning" and the unnecessary columns and rows were filtered to ease up the development. The first iteration, which was mentioned in the design chapter, was relatively quickly put together, but all the wanted information was not in the needed form initially. The tables needed some new columns which were calculated by using the other columns' data. These columns were marked with asterisk in the last chapter's tables which are found in the appendix D. The self-made columns which were not explained well are explained more thoroughly in the appendix E. In addition to the self-made calculated columns PowerBI had a feature which allowed creating Measures. Measure is like an invisible calculated column. It can be used in the equations, reports and dashboard's same as the value

on a column. Many Measures were also needed for the calculations in the project and these are also presented in the appendix E.

The most challenging part of the calculated columns and measures was deciding how the estimates were to be calculated. After reviewing the answers from the interviews and analyzing the data available, two possible methods for calculating the estimated completion dates were invented. Methods had different pros and cons. In this context the cycle time and completion of story is defined by the state data of Azure DevOps. Story is started when it is changed from New state to Active and it is finished after it has been tested and accepted in a sprint review which means the state is changed to Closed. Therefore, the testing time is also automatically included in the stories' completion times. In conclusion the cycle time is calculated by counting the days between the state change to Active state and state change to the Closed state.

First method was to use the average cycle times of a specific user story size group in the estimation and summing up the average cycle times of all the user stories related to one feature. This method is called AVG-method in the following text. Initially the AVG-method was used by selecting only the most important user stories as data, which were marked as PM_Priority in the data. Later, it was noticed that the priority of the user story did not have any effect on the cycle times. It only changed the order which user story was started next. It was mentioned in the interviews that priority has great impact when the user story is completed, but it does not affect the speed of the development.

The second method is to use the average velocity of a team and divide it with the amount of developers in the team. This results in the average points per sprint per developer -number, which can be used to estimate how many sprints it would take to complete certain user stories and features. The calculations could be done for example by taking all the story points completed from the last six months and dividing it with the number of sprints and finally dividing it with the number of developers in the team. This speed would be used only on that team's user stories. After calculating this number, it would be easy to divide the amount of points in the user story with the speed to calculate the number of sprints it would take to complete the story. This method is hereon called the V-method.

In the interviews it was mentioned that story points are abstract measures and they are usually assigned based on the feeling of the effort to complete the user story. The V-method assumes all points equal and in a linear scale since they are summed and divided

in the calculations. Abstract measures generated by a feeling does not guarantee linear points which is one problem that could impact the estimates negatively. The AVG-method is not affected the fact that story points might not be on a linear scale, since the story point size groups are separate.

Scenario where same developer is developing many user stories at the same time has negative influence on the AVG-method's accuracy. For example, if a person starts developing three 5-point stories at the same time and finishes them simultaneously, the developer completed 15 points during that timeframe. In the data all the separate 5-point stories only add most likely higher than average cycle times to the data making it less accurate. This could be fixed by only doing one user story at the same time, but it could also slow down the development. In the V-method this is not a problem, since all the story points are taken into consideration.

In the AVG-method it is also harder to generate estimates if many developers are developing the same feature. For this reason, the estimates in AVG-method were only for one developer developing the feature at first. As the V-method assumes the points linear, it is possible to just multiply the velocity if more developers develop the same feature. It is also possible to just divide the whole estimation of AVG-method with the amount of active developers in the feature, which could be better than nothing. But for example, in case of a small story and large story the estimation would divide the sum with two. The additional developer in the smaller story will most likely not decrease the cycle time of the larger story. Better solution would be to just not include the smaller story to the estimation.

One benefit in V-method over the AVG-method is that the data is team specific. In the interviews some people thought that teams have different size estimates and velocities. By having data separated per team it would benefit the accuracy of the estimates. This cannot be done with the AVG-method because after separating the user stories to the size groups and then by the teams, the sample size is too small for generating reliable estimates. Having team specific velocities is not always the best decision, since some teams fix bugs more than other teams. Bugs do not have story points assigned to them; therefore, it might affect the team's velocity negatively in the data. One possible solution to this is to use only the number of developers with active user stories, but this data is not always correct. The stories might be left active even though they are not developed at that moment as it was said in the interviews. Calculating the number of developers in a team is

also problematic, since teams change, and people might be working on tasks which are not directly assigned to their team.

The AVG-method was chosen to be used over the V-method, since it was easier to implement. It was hard to come up with the number of developers in a team which would be accurate, and the non-linearity of story points felt like an important factor in the beginning. Moreover, the fact that in the V-method all the most important user stories are lost in the mass seemed problematic. Some user stories were found in the data, which were not real user stories, but were created to remind some people what to do and random points assigned to them. Initially only the PM_Priority user stories were used as the data for the estimates. The total amount of these stories was not high, hence outliers had to be deleted by hand to receive more accurate estimates. For example, some user stories were over 5 times the average length which affected the average greatly.

In the design chapter it was mentioned that at some point a feature specific speed estimate was used, which is from here on called FSS-method. FSS-method used a rough feature specific estimation of the completed date by using the feature's starting date, last completed user story date, completed story points and total story points. The estimates were crafted by first calculating how many story points were averagely completed per day and then the remaining story points were divided by that number. The calculation provided the number of days left to complete the remaining points. This method was noticed to be more accurate especially on features which had many user stories and many points completed already. At that moment the AVG-method could not take multiple developers into consideration, for this reason this method would most of the time yield better results if more than one developer was developing the feature. Also, if the speed was greatly faster or slower than the average, it would obviously offer more accurate results. The flaw in this method was that it needed several user stories completed under the feature to provide accurate estimates. Feature specific speed cannot be calculated, if none of the points are completed. If only few points are completed the speed fluctuates greatly when more points are completed, or days added to the equation. A logic was then developed to use the averages first and after over 40% of the total story points were completed, the estimations would use the FSS-method. This would only happen if the 40% of the total story points was over 15 points, thus the FSS-method would not be used for small features.

After analyzing the data more thoroughly a realization was made that the PM_Priority user stories were not done faster than other stories, and the use of the tag was stopped.

The average of all stories was close to the numbers of PM_Priority averages after the outliers were removed. Later on, it was also realized that median should be used instead of averages. This removes the outliers, since the amount of user stories is sizeable, and few outliers do not affect the median as much as the average. At this point also the amount of developers with active user stories was taken into the equation. The estimate was just divided by the number of active developers and it was significantly better than not doing it. The AVG-method was now more often the most reliable one, therefore it was the only method used, and the FSS-method was dropped for now. The evaluation of the accuracy of the estimates is more thoroughly explained in the evaluation chapter.

One additional feature was requested to the artifact during the meetings where the progress was presented. The users of the dashboard wanted a possibility to add resources to the current features and see how it would affect to the estimated completed date. This was implemented by adding a radio button on the right side of the dashboard which had numbers from 0 to 10 and by choosing a number, the dashboard added that number to the active developers -value. As it was mentioned earlier the estimate was divided by the number of active developers and by adding more developers to the feature, the estimated date would come closer to today's date. In addition to that, a feature specific velocity was one of the metrics which were asked. It simply calculates how many points have been made averagely per sprint in this feature.

The user story data used in the estimates was decided to include only the user stories completed in the last six months. The reasoning for this is that the case company changes, and the estimates should be created using up-to-date data. New developers are recruited to the case company and current developers may leave which leads to the fluctuation of story points sizes in time. The time windows should not be too short either because for example the vacation periods in summer and during Christmas will affect the stories' completion speeds. The six months is an adequate middle ground for having enough time that the holidays will not affect the estimates too much and that the data is not excessively old. Other filters were also made to the user story data. If the user story did not have a parent work item id assigned, it would be excluded from the data. This would filter off some of the irrelevant objects marked as user stories. An Epic or a Feature object should exist one step higher in the hierarchy when user stories are created. Additionally, stories with less than one day cycle time were filtered out. It should not be possible to complete user stories that fast and several stories of this type was found in the data. These were

probably forgotten from the system first and added later or they might have been transferred from the previous information system. They should not be included in the data, since they will affect the estimations negatively.

The final dashboard is presented in the appendix F. Compared to the first version of the dashboard some changes have been made. The team column has been changed to teams column. First it presented only the team which the feature was assigned to, but now it shows all the teams which have user stories assigned to, which are related to the feature. This offers more detailed information, since some features are developed by several teams. The value of the completed percent has been added to the progress bar to offer more detailed information about the progression. Both estimated days left columns have been removed and only the estimated completed date has been left to the dashboard. This clarifies the dashboard and it is easier to understand. Feature specific velocity which is presented in the form of points per sprint has been added to inform about the velocity of each of the features. It can be used to compare different features' velocities. One column was added to show the number of developers who have active user stories related to the feature. In addition to this column a radio button was added, which can be used to add more developers to the features to estimate how much less time would be spend if more resources were added to the feature. The title, release and last user story completed days ago -columns have stayed like in the first version. The warning text is still present in the top, since the dashboard had not been taken into use officially. Next an example case of the artifact's estimation process is presented to sum everything up.

First the artifact takes the basic information of the feature which is the name, release and work item id of which the first two are presented in the dashboard. By using the work item id, the related user stories are linked to the feature and their data can be accessed. When a user story is created it must be assigned to a team. The teams column has all the different team names of all the user stories related to the feature. The total story points and completed story points are calculated by checking the states of the user stories. It is also presented in a progress bar with the percentage of completion. The latest user story completed is calculated from the difference in today's date and the latest story completed date. The feature specific velocity is calculated by dividing the completed story points with the number of days the development has taken and then multiplying it by 14, which is two weeks like the sprint length. The amount of developers with active stories is calculated by counting the different developer names who have been assigned to the stories

which are related to the feature. The estimated completed date is calculated by summing up the not completed user stories size specific average cycling times together and adding those days to the date when the sprint started. The summed-up days is divided by the amount of developers with active user stories and it also takes into consideration the radio button which can be used to add developers.

For example, if 2-point and 3-point stories are currently in development and the average cycle time for 2-point stories is 20 days and 3-point stories 30 days. These two numbers are combined which is 50 days and then divided by the number of different developers, which is 2. The final number 25 is the estimated days that it will take to complete both of the stories. The answer 25 is less than the average of 3-point stories cycle time, which sounds irrational, but this technique yields more accurate results than without using it. The 25 is closer to the 30 than 50 which would be the result without dividing the summed up estimated days with the developers. Possible improvements are discussed after the evaluation chapter.

## 4.5 Demonstration of the benefits of the dashboard

The first demonstration with the program manager went well and the artifact had the features which were needed. The program manager is responsible for the release planning and the dashboard and the estimates helps in this task. The program manager felt that all the information in the dashboard was important and helpful, since this information was not available before in any easily accessible place. The automatic refresh of the dashboard was also demonstrated and approved by the program manager. The dashboard's data was confirmed to be mostly correct by comparing it to the information in Azure DevOps. Overall feedback was positive, but some features could be developed further. For example, the only one estimate was wanted on the dashboard and its accuracy evaluated. The idea of adding developers to features was also mentioned but was not mandatory if it was difficult to be developed. The possibilities for improvements were endless, since the estimation is a multidimensional subject and it has countless factors affecting the accuracy.

The meeting which was held with the relevant people in the case company also resulted in feedback and improvement suggestions. People asked questions and gave their opinions about the dashboard. The opinions were mostly about the dashboard's user interface. Many people said that it had too many numbers which was distracting, and they felt that would like it more, if it was simpler. For example, the information regarding the current

planned release and the estimation date was suggested to be in one column which would just state that will this feature be ready before the planned release. In the demonstration meeting some bugs were also found in the dashboard calculations after some people questioned the values of data at some columns. The meeting confirmed the importance of receiving feedback from as many sources as possible, since it helped to include many different aspects to the design. It also gave confirmation that it was right decision to progress with many iterations to find all the bugs before starting to use the artifact.

## 4.6 Evaluation of the objectives' success

In the evaluation part the actual results of the solution are compared to the objectives of the solution. Evaluation was a continuous process during the research and in this chapter all the findings are presented. Most of the evaluation is comparing the results to the objectives and requirements created previously. Calculations are done to discover the average error in the estimates. The previously identified objectives for the solution were:

- Display information about all the currently developed features.
- Refresh automatically.
- Provide the estimated complete dates for features as accurately as possible.
- Use scrum history data from Azure DevOps.

First objective to evaluate is the presentation of all the information of current features. Since this was not possible to do earlier, anything which helps the organization in this matter is an improvement. Currently the data of all the features, the teams which are developing them, and the status of the features is available in the same location, in the artifact. The main user of the artifact, the program manager has confirmed that all the information is valuable and well presented. This shall be enough evaluation for this objective and the objective can be expected to be completed. One other objective can be also linked to this, which is the usage of the Azure DevOps scrum history data. The data presented is taken from the mentioned information system and it is confirmed to be correct. The data is imported by using a Microsoft's own connector between PowerBI and Azure DevOps and the data is always up to date because it is imported straight from the Azure DevOps database.

Second objective is the automation of the dashboard, which means that the dashboard should be automatically updated when the data changes. This was hard to accomplish, but

the artifact is close to being updated in real time. By using the normal license to PowerBI the user is allowed to set 8 scheduled refreshes throughout the day. Therefore, it is almost possible to refresh it every hour of a workday. This time interval for refreshes has been considered sufficient for the organization. Even though a real time dashboard would be better, this hourly refreshing automation will be almost as useful, and it will reduce resources which would go to updating the dashboard manually. Automation objective can also be assumed completed.

Finally, the most important and hardest objective to evaluate is the accuracy of the estimated days for the completion of the features. It is already explained earlier how the estimates were calculated, but in this section the accuracy of estimates is evaluated. In this evaluation it should be remembered that the story points are an abstract measure of estimated effort, they are not strictly standardized, and the cycle times can fluctuate drastically. It was still surprising to see that story points were relatively close of being linear when comparing the sizes and the median cycle times as seen in the figure 8 and 9. In both of these figures only the most used user story points are included, which are 1,2,3 and 5. These point groups had sample sizes of over 50 during the last 180 days and the samples of other sizes were less than ten.

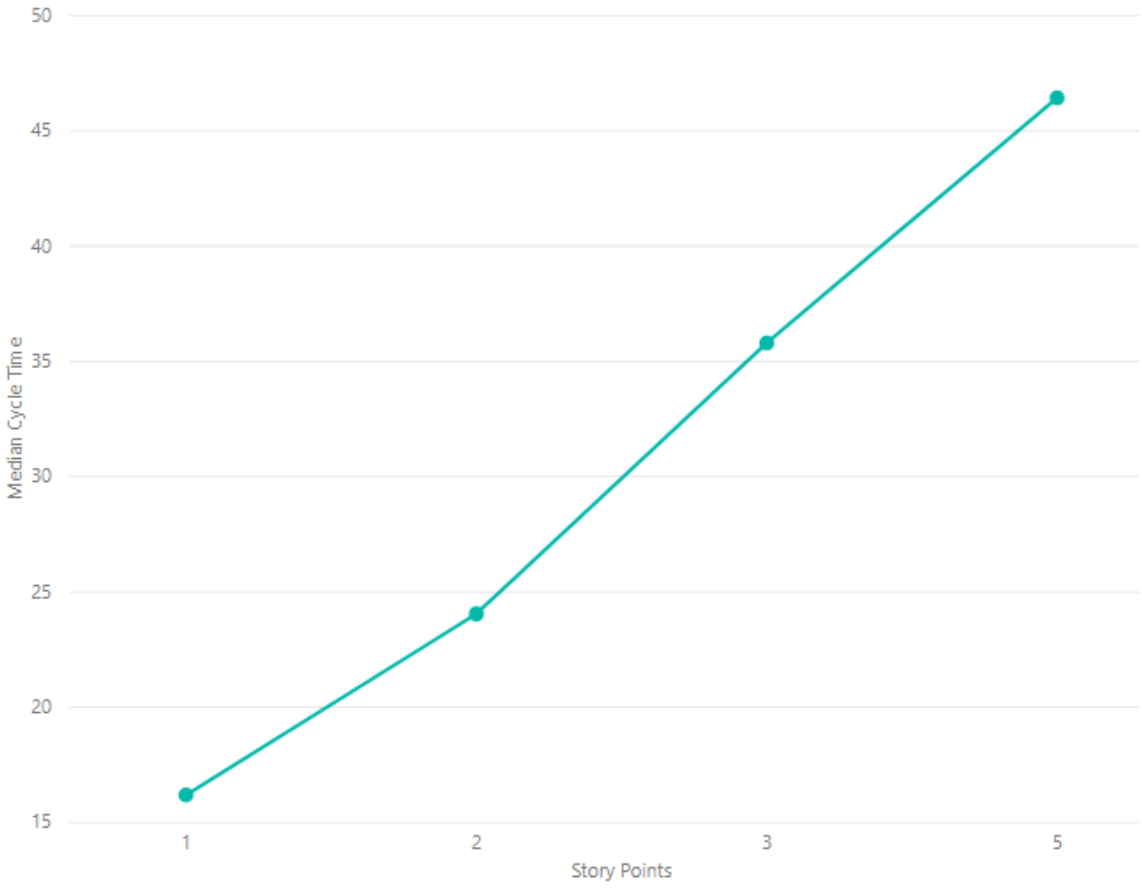



***Figure 8.*** *Median cycle times of each story point size group.*

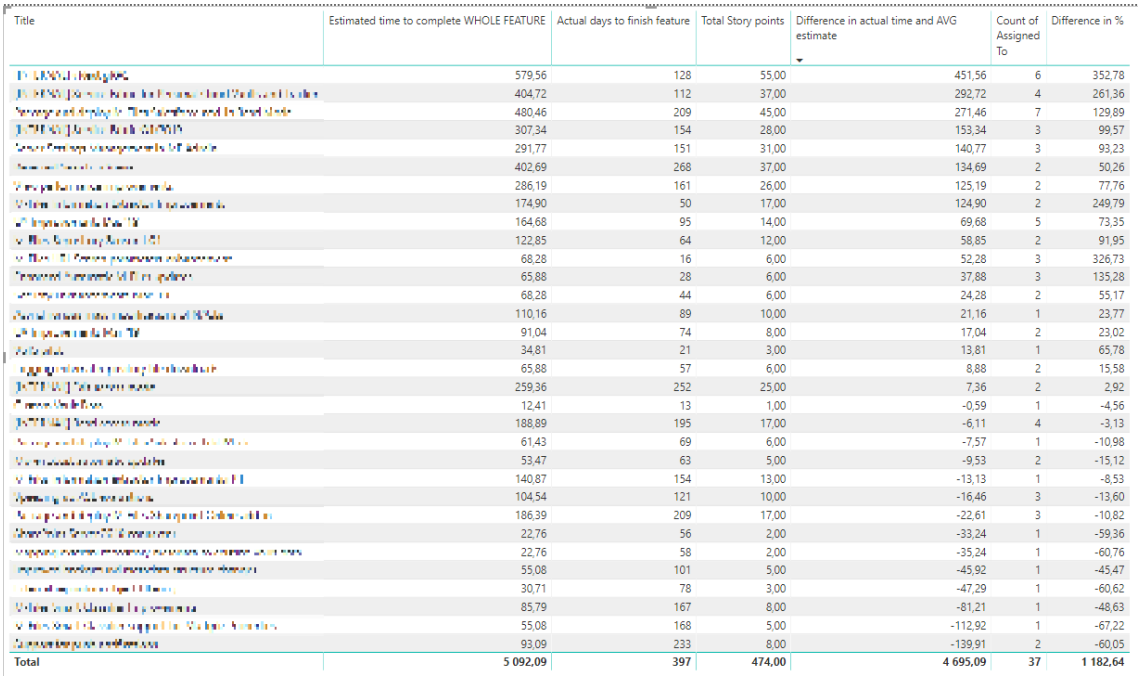
The time to complete a one point user story might actually be less than what this graph claims, since if a story is started in the beginning of a sprint and finished earlier than the end of the sprint, the minimum time to complete the story is 14 days in the data. If on the other hand the story is started in the middle of the sprint and finished right before the sprint review, the data might be more accurate. In the next figure 9 the line should be straight from left to right for the metric to be linear.


***Figure 9.*** *Median cycle times divided by amounts of story points per story point size group to present linearity.*

Due to this almost linear size scale of the story points, a mechanic for new story point size groups was made by calculating the average days per story point. This number was 11 and if a new story point size would appear it would be multiplied by 11 to calculate the estimated average cycle time. The story point groups which have less than 10 samples will be estimated by using this technique also, because a small sample size would have higher possibility to yield less accurate estimates. These are exceptions which should not happen often, and the most common story point sizes should always be used, but the mechanic is needed for the dashboard to work if a new story point group is made.

The chosen protocol for evaluating the accuracy of the estimates is to compare what the artifact would estimate for the time of completing all the user stories included in a feature and what was the actual time for completing the feature. The actual time was calculated from the first user story's starting date to the last user story's completion date. This evaluation can only be done to the AVG-method since the feature specific speed changes many times during the development of the feature. A PowerBI report was made, which

had all the completed features and their information. It included actual days to finish the feature, estimated days to finish the feature, total story points, amount of developers which had participated to the development and a column for the difference of estimated and actual finish time. At the first evaluation the artifact used only average cycle times and did not divide the estimates by the amount of average developers, and this affected the estimates substantially. After the data was analyzed it revealed that the features which had many developers and higher amount of story points, were usually estimated with the most error. In the next figure 10 the table of the data is presented.



| Title | Estimated time to complete WHOLE FEATURE | Actual days to finish feature | Total Story points | Difference in actual time and AVG estimate | Count of Assigned To | Difference in % |
|---|---|---|---|---|---|---|
| | 579,56 | 128 | 55,00 | 451,56 | 6 | 352,78 |
| | 404,72 | 112 | 37,00 | 292,72 | 4 | 261,36 |
| | 480,46 | 209 | 45,00 | 271,46 | 7 | 129,89 |
| | 307,34 | 154 | 28,00 | 153,34 | 3 | 99,57 |
| | 291,77 | 151 | 31,00 | 140,77 | 3 | 93,23 |
| | 402,69 | 268 | 37,00 | 134,69 | 2 | 50,26 |
| | 286,19 | 161 | 26,00 | 125,19 | 2 | 77,76 |
| | 174,90 | 50 | 17,00 | 124,90 | 2 | 249,79 |
| | 164,68 | 95 | 14,00 | 69,68 | 5 | 73,35 |
| | 122,85 | 64 | 12,00 | 58,85 | 2 | 91,95 |
| | 68,28 | 16 | 6,00 | 52,28 | 3 | 326,73 |
| | 65,88 | 28 | 6,00 | 37,88 | 3 | 135,28 |
| | 68,28 | 44 | 6,00 | 24,28 | 2 | 55,17 |
| | 110,16 | 89 | 10,00 | 21,16 | 1 | 23,77 |
| | 91,04 | 74 | 8,00 | 17,04 | 2 | 23,02 |
| | 34,81 | 21 | 3,00 | 13,81 | 1 | 65,78 |
| | 65,88 | 57 | 6,00 | 8,88 | 2 | 15,58 |
| | 259,36 | 252 | 25,00 | 7,36 | 2 | 2,92 |
| | 12,41 | 13 | 1,00 | -0,59 | 1 | -4,56 |
| | 188,89 | 195 | 17,00 | -6,11 | 4 | -3,13 |
| | 61,43 | 69 | 6,00 | -7,57 | 1 | -10,98 |
| | 53,47 | 63 | 5,00 | -9,53 | 2 | -15,12 |
| | 140,87 | 154 | 13,00 | -13,13 | 1 | -8,53 |
| | 104,54 | 121 | 10,00 | -16,46 | 3 | -13,60 |
| | 186,39 | 209 | 17,00 | -22,61 | 3 | -10,82 |
| | 22,76 | 56 | 2,00 | -33,24 | 1 | -59,36 |
| | 22,76 | 58 | 2,00 | -35,24 | 1 | -60,76 |
| | 55,08 | 101 | 5,00 | -45,92 | 1 | -45,47 |
| | 30,71 | 78 | 3,00 | -47,29 | 1 | -60,62 |
| | 85,79 | 167 | 8,00 | -81,21 | 1 | -48,63 |
| | 55,08 | 168 | 5,00 | -112,92 | 1 | -67,22 |
| | 93,09 | 233 | 8,00 | -139,91 | 2 | -60,05 |
| **Total** | **5 092,09** | **397** | **474,00** | **4 695,09** | **37** | **1 182,64** |

***Figure 10.*** *Information about features' actual finish times and estimated finish times.*

This made it clear that the amount of developers must be included into the equation. Most of the estimate errors which were much over the actual days to finish the feature had many developers developing the user stories.

At this time the FSS-method was used for the larger features and it yielded more accurate results. FSS was taken into use at the previously mentioned 40% completed story points and the number was chosen after doing a test using a simulator. The simulator had all the features' AVG estimates and FSS estimates for a selected date. The date could be changed using a slider which had all the dates from the times the user stories were developed. Focusing on one row at a time the slider was moved from the start to the end and a point where the FSS method became more accurate than the AVG-method was looked for. It was then decided that 40% would be used, since almost all the features were estimated

more accurately after 40% when using FSS-method. The simulator tool is presented in the figure 11 below.

| Title | Estimated time to complete WHOLE FEATURE | Actual days to finish feature | FSS dynamic estimate | Total Story points | Completed points at date | Completed % at date | Velocity at date | Difference actual and AVG | Difference actual and FSS |
|---|---|---|---|---|---|---|---|---|---|
| [redacted] | 12,61 | 13 | 9 999,00 | 1,00 | | | | 0,39 | 9 986,00 |
| [redacted] | 31,44 | 28 | 9 999,00 | 3,00 | 3,00 | 1,00 | 0,08 | 3,44 | 9 971,00 |
| [redacted] | 190,71 | 195 | 261,38 | 17,00 | 8,00 | 0,47 | 0,07 | 4,29 | 66,38 |
| [redacted] | 23,11 | 17 | 9 999,00 | 2,00 | | | | 6,11 | 9 982,00 |
| [redacted] | 62,89 | 69 | 9 999,00 | 6,00 | | | | 6,11 | 9 930,00 |
| [redacted] | 54,56 | 63 | 9 999,00 | 5,00 | 5,00 | 1,00 | 0,07 | 8,44 | 9 936,00 |
| [redacted] | 23,11 | 14 | 9 999,00 | 2,00 | | | | 9,11 | 9 985,00 |
| [redacted] | 262,15 | 252 | 1 125,00 | 25,00 | 3,00 | 0,12 | 0,02 | 10,15 | 873,00 |
| [redacted] | 67,16 | 57 | 60,00 | 6,00 | 1,00 | 0,17 | 0,10 | 10,16 | 3,00 |
| [redacted] | 140,01 | 154 | Infinity | 13,00 | | | | 13,99 | Infinity |
| [redacted] | 106,94 | 121 | 246,67 | 10,00 | 3,00 | 0,30 | 0,04 | 14,06 | 125,67 |
| [redacted] | 35,21 | 21 | 9 999,00 | 3,00 | 3,00 | 1,00 | 0,01 | 14,21 | 9 978,00 |
| [redacted] | 92,44 | 74 | 76,00 | 8,00 | 2,00 | 0,25 | 0,11 | 18,49 | 2,00 |
| [redacted] | 108,57 | 89 | 9 999,00 | 10,00 | 10,00 | 1,00 | 0,07 | 19,57 | 9 910,00 |
| [redacted] | 186,24 | 209 | 1 139,00 | 17,00 | 2,00 | 0,12 | 0,01 | 22,76 | 930,00 |
| [redacted] | 69,33 | 44 | 51,00 | 6,00 | 2,00 | 0,33 | 0,12 | 25,33 | 7,00 |
| [redacted] | 23,11 | 56 | 9 999,00 | 2,00 | 2,00 | 1,00 | 0,01 | 32,89 | 9 943,00 |
| [redacted] | 23,11 | 58 | 9 999,00 | 2,00 | 2,00 | 1,00 | 0,02 | 34,89 | 9 941,00 |
| [redacted] | 123,89 | 87 | 9 999,00 | 11,00 | | | | 36,89 | 9 912,00 |
| [redacted] | 67,16 | 28 | 9 999,00 | 6,00 | 6,00 | 1,00 | 0,04 | 39,16 | 9 971,00 |
| [redacted] | 31,44 | 78 | 9 999,00 | 3,00 | | | | 46,56 | 9 921,00 |
| [redacted] | 54,28 | 101 | 9 999,00 | 5,00 | | | | 46,72 | 9 898,00 |
| [redacted] | 69,33 | 16 | 9 999,00 | 6,00 | 2,00 | 0,33 | -0,04 | 53,33 | 9 983,00 |
| [redacted] | 125,78 | 64 | 9 999,00 | 12,00 | 12,00 | 1,00 | 0,09 | 61,78 | 9 935,00 |
| [redacted] | 165,64 | 95 | 9 999,00 | 14,00 | 14,00 | 1,00 | 0,11 | 70,64 | 9 904,00 |
| **Total** | **5 322,27** | **406** | **501,39** | **492,00** | **262,00** | **262,00** | **0,98** | **393,39** | **95,39** |

Date

19.1.2019 31.12.2019

***Figure 11.*** *The simulator tool used to compare accuracy of AVG- and FSS-methods.*

In the beginning the AVG-method used average cycle times instead of median cycle times and that also made a slight error to the estimates. In the next figure 12, the dots are representing all the completed features, on the Y-axis their estimate's error is presented in days and on the X-axis the total amount of story points. As the amount of total story points grew, the amount of developers usually also grew, and it can be seen from the figure that the larger features have greater amount of error.

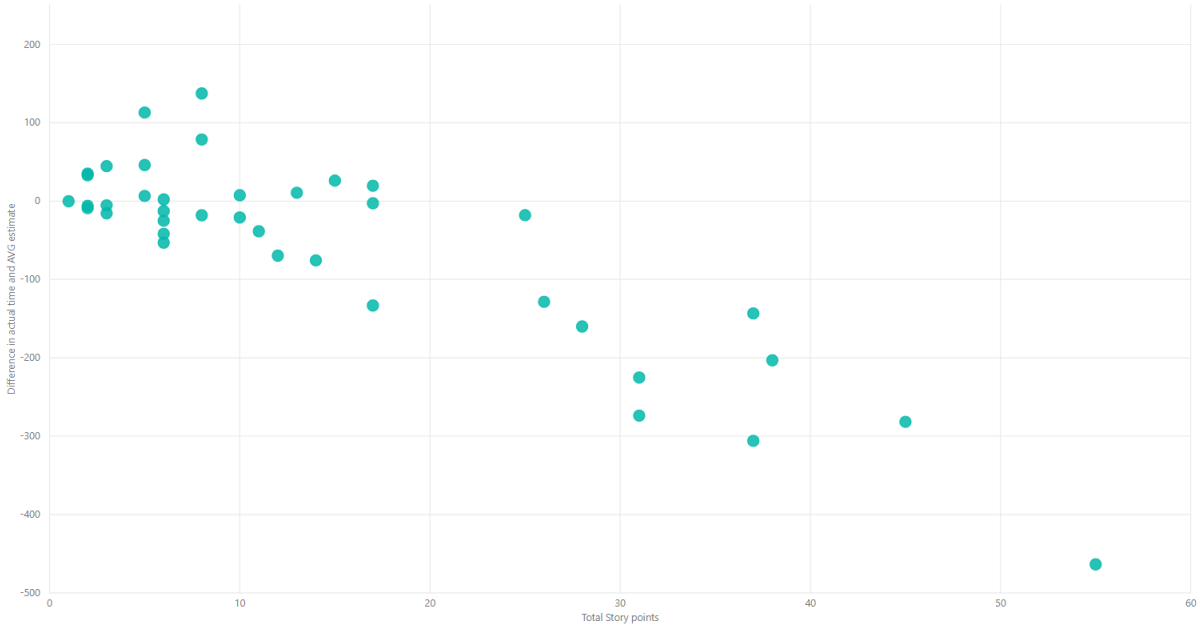



***Figure 12.*** *AVG-method estimate errors before adding median and amount of developers to the equation.*

After adding the average amount of developers to the equation and using median cycle times instead of the average cycle times, the same table looked considerably better as seen in the figure 13 below.

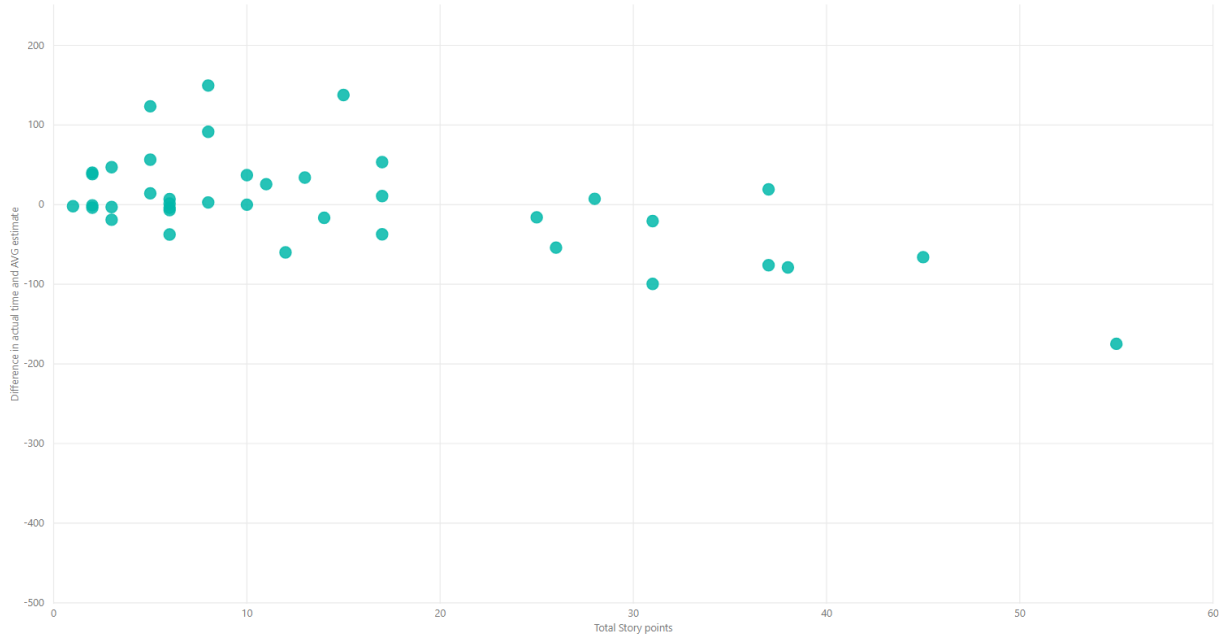



***Figure 13.*** *AVG-method estimate errors after adding median and amount of developers to the equation.*

After the notable changes to the AVG-method's algorithm it was compared again to the FSS-method and it was better every time, except in features which progressed with a steady velocity which diverged significantly from the average. FSS-method calculates the

average velocity of that one specific feature; therefore, it would make sense that it would calculate better estimate in these circumstances. It is common to complete many stories at the same time, since sometimes stories which are connected to each other are more efficient to test at the same time. This does not favor the FSS-method.

Most of the figure 13's estimates are less than 100-day absolute error, being it earlier or later. All the data is not in perfect form and after closer investigation of the larger errors, flaws were found in the data which led to situations the artifact was not able to react to. For example, one feature had 20-point user story which was completed quickly, and after investigation it was revealed that it was just a placeholder which was never used properly. This user story was then expected to take over 200 days to complete, because of the previously mentioned mechanic created for new story point size groups. In addition to this the data included features developed by external people, which cannot be estimated accurately, since they have different opinions about story points and their sizes. Some features have not had any active user stories for a few months which also makes the actual time longer than it would have been if the stories would have been done consecutively. This is only a problem in the evaluation, since the artifact dynamically updates the estimates. After excluding the data which had previously mentioned problems, a more realistic error calculation for the estimations could be made.

All the completed user stories are in the data which is used to calculate what the artifact would have estimated to the feature. This might not be the best practice, since it will push the estimate to a more favorable direction for that specific feature. After testing how it would affect the median cycle times if the user stories connected to a specific feature would be removed from the data for these calculations, the change was just one or two days at most. The amount of user stories in the data is large, thus it would not be impactful to remove just a few. Implementing an algorithm which would use only the other completed user stories data would have been hard and not worth the effort.

When discussing the acceptable error in estimates, many situations exist which can easily add days to the measured days to complete a story. For example, if a user story is even a day late from the end of a sprint, it will be postponed for the time of a full sprint, which is 14 days. Another example, which was already mentioned is that a 1-point story can sometimes be done in the first few days of the sprint, but it is completed after 14 days according to the data. In addition to this it has to be remembered that the story points are an abstract estimate of the effort and estimating over or under to a small extent is common.

Sometimes critical bugs interrupt the development of a user story and sometimes user stories can be developed without any disturbances. The realized scale is also relatively wide with the averagely 11-day steps. If the developer estimates the story's size wrong, the estimated complete time is automatically at least 11 days wrong. Taking all these factors into consideration the program manager came up with the acceptable error. The wanted accuracy when estimating the larger features further in the future would be the right quarter of the year, which is a time span of 3 months. If the estimate is in the middle of the quarter the acceptable error would be then 45 days and this number is used as the limit of acceptable error. The sample size after taking off the faulty data was only 24. It is not a large sample, but it will provide some information about the accuracy. The used metric for calculating error is the mean absolute error, which is calculated with the formula mentioned in the chapter 3.3.5:

$$MAE = \frac{1}{24}\sum_{i=1}^{24} |e_i| \approx 21\ days$$

This signifies that averagely the estimates were 21 days either over or under the actual days of how long it took to complete all the stories in the feature. The accuracy was better than expected and it was acceptable error for this artifact to be taken into use. The objective of giving the best possible estimate can be regarded successful, even though the estimates could always be improved by adding more elements to the algorithm. The research was decided to be scoped to the current state and improvements left for future work in the case company. Some possible improvements are presented here.

For example, the way that the algorithm takes many developers in equation of estimates. Currently the artifact just divides the summed up estimated days by the amount of developers. This was gone through earlier in this paper in the example of 2-point story and 3-point story which had 20 and 30 days expected cycle times and combined cycle time of 50 days. In future the artifact could be improved to remove the shorter cycle time from the estimate, in this case the 20 days and just leaving the 30 days. This would work well if only 2 stories were left, but for example with 6 user stories, it is more difficult to use this method. It is not possible to just remove the smaller stories in the hope of better estimations and the rough division with the number of developers will yield more accurate results in most cases.

One element which could improve the accuracy of the estimates would be including a mechanism, which takes the user stories' other states into consideration. For example, in testing, impeded, testing done states, which could be taken into the equation. It can be assumed that if the story is in testing done state, it will not take the whole cycle time to be completed. Average time spent in testing state per user story size group could be calculated also to figure out how long does the testing usually take.

The artifact currently uses only the AVG-method for calculating the estimated completion dates, but the mentioned FSS-method could be improved and added to the equation again with some modifications. It gave better results when the feature was developed with a steady pace, thus an algorithm could be developed which used this method if the pace was steady enough. Also, the V-method could be tested again, since the story points were found to be quite linear after all.

# 5. DISCUSSION

In this chapter the literature from the theory chapter and the results of the empirical research are compared. First the problems identified in the literature and through executing the interviews are discussed. It is also mentioned how they affect the feature estimation and what is the state of these problems in the case company. After that a created solution to the problem is presented which is in a form of an artifact. The artifact is a dashboard which has the functionalities which solve the demonstrated problems.

### 5.1.1 Problems identified in agile planning, feature estimation and scrum history data

Same problems were identified in both, the literature and in the interviews. One of them was the difficulty of making schedules based on the estimates generated. McDaid et al. (2006) and the interviewees pointed out that effort estimations and calendar days are not always comparable because many other tasks exist which need to be dealt with besides just developing the one user story. All the meetings, fixing the IT-environment, unexpected bugs, difficulties with the technologies and for example sicknesses can affect the schedule. Usman et al. (2014) and Johnson et al. (2000) both praised expert judgement as the most common and accurate method for coming up with the story point values. This included different variations like planning poker or similar techniques where experts gave their opinion on the effort of the user story. Expert judgement is also used in all the case company's teams and the results seem to be relatively accurate when inspecting the story points average cycle times and their linearity. Some interviewees still had concerns that inaccuracy could be present, because everyone is not using the same expert judgement method. Cohn (2005) stated that he has had satisfying results using the Fibonacci sequence as the scale of story points. The same scale is also commonly used in the case company.

Ktata & Lévesque (2010) and some of the interviewees identified the problem of not changing the story point values to more truthful values, if they have been estimated inaccurately. Some interviewees thought that changes should not be done, because it affects the velocity negatively and if the points are not allowed to be changed then developers will plan their work smarter and do the effort estimation more thoroughly. This problem can not be solved in the case company during this research, since the processes are not

going to be changed based on these findings and only the created artifact is used to improve estimation. Ktata & Lévesque (2010) mentioned that the work items like features, story points and tasks should be distinctively specified to avoid misunderstandings and to keep the processes and data coherent. This is understood and agreed in the case company and all the development teams use the features, user stories and tasks properly. Coelho & Basu (2012) and Cohn (2005) agreed that the development environment should not change much if the history data is used for estimating effort or creating schedules. This prerequisite is not a problem in the case company, since they are working on the same software product, with usually the same technologies and same teams. The processes and environments are also relatively stable; therefore, the chances are higher for the estimates crafted from the history data to be accurate.

Buglione & Ebert (2011) had some suggestions to improve estimation in an organization. They stated that data from the organizations processes should be properly gathered, stored and verified. After this it could be used as valuable information for decision making. The interviews revealed that in the case company the first steps are fulfilled, but the data is not used as efficiently as it could be for decision making. They have used the data in some cases, but the data is not in a form that it would be easy to use.

### 5.1.2 Solution to the problems

Buglione & Ebert (2011) present some possible commercial tools which could be used for the feature estimation, but they all use lines of code when trying to come up with the story points, which is not optimal. It was decided in the case company that the solution is going to be self-made. The artifact created in this research had requirements derived from the problems found in the interviews and the literature and it fulfilled most of them. The high-level description of the artifact is an automated dashboard which has information of the current features in development. Feature information includes for example the feature's name, responsible teams, completed and total story points, estimated completed date, planned release and amount of developers with active user stories. The dashboard also has a radio button selection for adding developers to the features, which can be used to probe what would be the estimated completion date if more developers were added to a feature. This already answers to the problem found in the interviews that the data of the current features is not available easily. This dashboard also establishes the better usage of the information for decision making which was one identified problem. It should be noted that the artifact alone does not do any decisions.

The estimates are calculated using the story points and their median cycle times, which means that it includes the whole time from start to finish of a user story. When the cycle time is used it includes all the other steps in the process to the time also, for example the previously mentioned meetings, bugs and testing. This was considered a problem in the estimation and if the artifact does the calculation, resources are saved from the employees who should come up with the schedules. The developers just have to continue using the same methods when generating the story points for this history data to be as accurate as possible. The automated dashboard means that the dashboard will update automatically, and this also saves resources from rescheduling as the dashboard does it automatically. If some feature is not progressed at all during a sprint the estimate changes further.

McDaid et al. (2006) and Cohn (2005) both agreed that the most important question of release planning is which features are chosen for which releases. This artifact helps with the planning and decision making, since it provides estimates when features are ready. Only completed features can be released and according to the interviewees having estimates would be a great help. This is also related to the customer success which was mentioned in the interview answers. By estimating the completions better, it is possible to keep promises to the customers more often and provide them with the functionality in agreed time.

The accuracy of the estimates should also be discussed. Ktata & Lévesque (2010) presented scrum experts' opinions on acceptable errors in effort estimations and it was as high as 20% percent. The effort estimations of user stories are used in the estimation of features completion dates; therefore, the same acceptable error can be used here to offer some insight. The worst acceptable scenario, where all user stories would be 20% overestimated, a 20% error would exist in the whole feature's estimate. A rough example: by using 11, the average days per story points calculated in section 4.6. and for example, a small feature with 10 story points the estimated days to completion would be 110 days total. Acceptable error in this feature's estimation would be 110 * 0,20 = 22 days. Compared to the MEA calculated in this research, which was 21 days, the acceptable error is already higher than the average error. As the feature's size grows, the acceptable error in the estimate increases as well. The majority of the features in the MEA calculations were larger than 10 story points, hence the current error of the artifact is on an acceptable level. One factor to be noted is that the feature estimation dashboard is mainly used on the large

features, since features with only few user stories left can be discussed with the developers. In the late stages of a feature, the developers usually estimate the completion better than the artifact.

# 6. CONCLUSION

This chapter provides summarized and generalizable answers to the research questions. Practical contributions to the case company are presented and theoretical contributions to the literature discussed. The whole research is also reviewed and limitations, which were part of the research introduced. Finally, some thoughts and suggestions about further research on the subject are presented.

## 6.1 How to improve feature estimation and agile planning using scrum history data

To answer to the main research question, it is first needed to answer to the two sub research questions:

- What kind of problems can be identified related to agile planning, feature estimation and scrum history data?
- What kind of artifact could help solving problems in feature estimation?

The research was done as a qualitative case study, which used only one case in the practical part of the research. This affects negatively the study's ability to provide generalizable answers, since the unique environment of the case company drives the results in a certain direction. Some insights can still be gathered for any software company struggling with problems related to feature estimation and agile planning.

### 6.1.1 What kind of problems can be identified related to agile planning, feature estimation and scrum history data?

The research revealed common problems identified in agile planning, feature estimation and scrum history data, which should be taken in consideration when performing agile software development. One identified problem was that the effort estimations and calendar days hardly ever match, because unexpected situations occur in the working environment and it is hard to add all the meetings, extra work and other activities when estimating. This makes it harder to come up with estimated complete dates and deadlines. The work items related to the processes should be unambiguously defined. These work items can include for example features, which are compiled of user stories and measured by story points which are an abstract measure of estimated effort. When using story points

in effort estimation, methods that include expert judgement have been considered to offer most accurate results. Reviewing story points values after sprint have been seen beneficial for learning to better estimate effort in the future and to keep the scrum history data more coherent. If the scrum history data is to be used in feature estimation, the environment, technologies, teams and tools should remain invariable for better accuracy. It has been noticed that some organizations are not using their own data efficiently. All organizations should have a plan for gathering data from their processes and work. It should also include some validations for the data's truthfulness. This data can be used in monitoring performance and progress, but also enabling decision making to yield better results.

### 6.1.2 What kind of artifact could help solving problems in feature estimation?

In this research it was discovered that an artifact in a form of dashboard which uses scrum history data can be used to solve problems related to feature estimation. The dashboard could display any basic information like for example name, total story points, completed story points and assigned teams about each feature. Important addition would be an estimate when the feature would be completed. The estimates can be calculated many ways, but one way is to calculate it by using the median cycle time of user stories which were of the same size in story points. The cycle time should include the time from starting to being accepted in a sprint review, which means it has testing and everything else calculated in the averages. Dashboards are appropriate at presenting customizable information and they can be scheduled to refresh automatically which helps saving resources. It should be noted that the estimates will always be just estimates and they should be used as such.

## 6.2 Practical contributions

The research was conducted in a case company and the practical contributions to this specific case company is the feature estimation dashboard which is in daily use. The program manager uses it often to check the statuses of the current features and if something looks out of place, he knows which team to contact and ask for more specific information. The dashboard helps with release planning, since some estimates are provided to each feature and it is possible to see general overview of all the work done in the product development department.

For other companies in the field this research provides information about problems in feature estimation and the benefits of a feature estimation dashboard which can help solving these problems. For it to be as accurate as in the case company, the environment should be similar, where the developers work with the same product, in stable environment and with same team members. For project-oriented software companies where team members and technologies might change after each project this research does not provide as much useful information.

## 6.3 Theoretical contributions to the literature

Buglione & Ebert (2011) have compared many commercial off-the-self technologies and tools which can be used to estimate user stories' and features' completions. All these tools and technologies were using the lines of code metric when trying to estimate function points or story points, which has been seen to provide inaccurate results. Coelho & Basu (2012) state that even though the amount of lines in the source code of a software is rather easy to monitor and measure, it does not yield any insightful information regarding the functionality. The same functionality can be developed with different amounts of lines in the source code depending on the used technologies and the skills of the developer.

Kusumoto et al. (2005) and Choetkiertikul (2019) have done research which provided effort estimation tools or systems which estimate story points or use case points automatically using different kind of models. In this research a different approach has been used and the developers do the effort estimation in the form of story points. Expert judgement has been considered to be the most accurate way of effort estimation; therefore, the average cycle times of the completed user stories are used when estimating the completion of features. Most of the literature in the field are about project-oriented software companies, and as this research focuses on product-oriented company, it contributes well to the field from this point of view. In a product-oriented company the teams, technologies and work environment are more stable and this kind of estimation is possible to use with acceptable error.

This research provides information how an estimation tool was created in this context using Microsoft PowerBI and scrum history data. Justifications on each decision made regarding the artifact is also offered. Estimation tool does not use lines of code, like most of the technologies available. The research also provides a model how to calculate the average error in the estimates and provides example of the error in this case.

Reviewing the work and limitations related to the research. In this research the goal was to go through the literature related to the agile software development, hold interviews to gather information from the case company and use these sources of information to build an artifact, which would help feature estimation. Cohn (2005) states that business value should be taken into consideration in every decision when making decisions related to agile software development. In this research the business aspect has not been considered and it is left out of the scope. Prioritizing user stories and features is also left out of scope, since it should always be done by considering the business value.

Korstjens & Moser (2018) argue that the trustworthiness of a research should be evaluated using credibility, transferability, dependability and confirmability as the criteria. In this research the credibility has been tackled by using a method of triangulation which means using data from several sources. The sources used were the academic literature and the interviews. The sample of the interviewees was limited to only 14 members of the case company, but at least the interviewees were selected from all the relevant positions from the case company. The generalizability of the interview answers is difficult with this small number of interviewees and by only choosing the relevant people, since the answers might be biased. Transferability means how well can the findings of the research be transferred to a different setting. This has been considered before in several chapters and the generalization of all the results is hard unless the other setting's case company and its environment are similar. The aspects of dependability and confirmability are ensured by using an audit trail. All the steps done in this research are reported in detail and the thought process of making the decisions explained in each part.

Korstjens & Moser (2018) also add reflexivity as the fifth criteria. In this part the author must evaluate the decisions made and the results received from the aspect of neutrality. The author's values, preconceptions and assumptions might affect the results and how decisions were made. The author in this research has been working in the company for over a year and was doing the work regarding this research mostly alone which might have caused the results and decisions to have some bias. For solving this issue, the author tried to include his supervisors and other people from the case company as much as possible to receive feedback and different opinions about the work. The artifact in this research was made to create estimations and it was evaluated for accuracy. When the goal is to have accurate estimates as results, it can be difficult to make decisions objectively.

Decisions made when evaluating the accuracy are explained in detail to be as transparent and neutral as possible.

## 6.4 Future research topics

Future research topics include the aspects of business value of the features and the prioritization. This artifact could be taken one step further by implementing data-analytics to calculate possible business value of feature in the same dashboard. The artifact could be used for making decisions which feature should be implemented next and how much value would it accumulate. The research would then be conducted more from the aspect of business value and cost estimation than from the estimation of features' completion.

In this research setting the processes of the case company where not altered to improve feature estimation, only the artifact was used. In some other research setting the altering of the processes could be taken as an approach to provide the best possible foundation for feature estimation. Another option is that the effects of the process changes for the accuracy of the feature estimation could be measured by first establishing the feature estimation environment and then changing the processes.

Furthermore, the same research could be done using a project-oriented software company as a case company, but it would doubtfully yield as accurate estimates. The part of creating an artifact which would display current features and would update automatically could still be useful for communicating the current situation of the features.

# REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). Agile software development methods: Review and analysis. arXiv preprint arXiv:1709.08439.

Ahmad, M. O., Markkula, J., & Oivo, M. (2013, September). Kanban in software development: A systematic literature review. In 2013 39th Euromicro conference on software engineering and advanced applications (pp. 9-16). IEEE.

Astels, D. (2003). Test driven development: A practical guide. Prentice Hall Professional Technical Reference.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). The agile manifesto.

Beck, K., & Gamma, E. (2000). Extreme programming explained: embrace change. addison-wesley professional.

Becker, J., Knackstedt, R., & Pöppelbuß, J. (2009). Developing maturity models for IT management. Business & Information Systems Engineering, 1(3), 213-222.

Boehm, B., & Turner, R. (2003). Balancing agility and discipline: A guide for the perplexed, portable documents. Addison-Wesley Professional.

Buglione, L., & Ebert, C. (2011). Estimation tools and techniques. IEEE software, 28(3), 91-94.

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. Geoscientific model development, 7(3), 1247-1250.

Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., & Menzies, T. (2019). A deep learning model for estimating story points. IEEE Transactions on Software Engineering, 45(7), 637-656. doi:10.1109/TSE.2018.2792473

Dey, I. (1993). Qualitative data analysis: A user-friendly guide for social scientists. London: Routledge.

Drever, E. (1995). Using Semi-Structured Interviews in Small-Scale Research. A Teacher's Guide.

Cockburn, A. (2002). Agile software development joins the" would-be" crowd. Cutter IT Journal, 15(1), 6-12.

Coelho, E., & Basu, A. (2012). Effort estimation in agile software development using story points. International Journal of Applied Information Systems (IJAIS), 3(7).

Cohn, M. (2005). Agile estimating and planning. Pearson Education.

Healey, M. J. (1991). Obtaining information from businesses (pp. 193-251). Harlow, Essex: Longman.

Hevner, A., & Chatterjee, S. (2010). Design research in information systems : Theory and practice. Boston: Springer.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS Quarterly, 28(1), 75-105. doi:10.2307/25148625

Johnson, P. M., Moore, C. A., Dane, J. A., & Brwer, R. S. (2000). Empirically guided software effort guesstimation. IEEE Software, 17(6), 51-56.

Jones, C. (1996). Patterns of software system failure and success. Itp New Media.

King, N. (2004). Using interviews in quatitative research. Essential guide to qualitative methods in organizational research, 2, 11-22.

Korstjens, I., & Moser, A. (2018). Series: Practical guidance to qualitative research. Part 4: trustworthiness and publishing. European Journal of General Practice, 24(1), 120-124.

Ktata, O., & Lévesque, G. (2010, May). Designing and Implementing a Measurement Program for Scrum Teams: What do agile developers really need and want?. In Proceedings of the Third C* Conference on Computer Science and Software Engineering (pp. 101-107). ACM.

Kusumoto, S., Matukawa, F., Inoue, K., Hanabusa, S., & Maegawa, Y. (2005). Effort estimation tool based on use case points method. Osaka University.

Logue, K., McDaid, K., & Greer, D. (2007, May). Allowing for task uncertainties and dependencies in agile release planning. In 4th Proceedings of the Software Measurement European Forum (pp. 275-284).

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E., & Brinkkemper, S. (2016). Improving agile requirements: the quality user story framework and tool. Requirements Engineering, 21(3), 383-403.

McBreen, P. (2002). Software craftsmanship: The new imperative. Addison-Wesley Professional.

McDaid, K., Greer, D., Keenan, F., Prior, P., Coleman, G., & Taylor, P. S. (2006). Managing Uncertainty in Agile Release Planning. In SEKE (pp. 138-143).

Microsoft. (2018). Microsoft Docs. Set up or configure hierarchical teams - Azure Boards. Available: https://docs.microsoft.com/en-us/azure/devops/boards/plans/configure-hierarchical-teams?view=azure-devops

Miller, G. G. (2001, July). The characteristics of agile software processes. In tools (p. 0385). IEEE.

Neuman, W. L., & Robson, K. (2007). Basics of social research: Qualitative and quantitative approaches. Power, 48, 48.

Palmer, S. R., & Felsing, M. (2001). A practical guide to feature-driven development. Pearson Education.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. Journal of management information systems, 24(3), 45-77.

Petersen, K., Wohlin, C., & Baca, D. (2009, June). The waterfall model in large-scale development. In International Conference on Product-Focused Software Process Improvement (pp. 386-400). Springer, Berlin, Heidelberg.

Robson, C. (2002). Real world research: A resource for social scientists and practitioner-researchers (Second ed.). Oxford: Blackwell Publishing.

Rubin, K. S. (2013). Essential scrum: A practical guide to the most popular agile process (1st ed.). Upper Saddle River, NJ: Addison-Wesley.

Saunders, M., Lewis, P., & Thornhill, A. (2009). Research methods for business students. Essex. Financial Times/Prentice Hall.

Schwaber, K. (1995). Scrum Development Process: Advanced Development Methods. In Proceedings of OOPSLA'95 Workshop on Business Object Design and Implementation, London, UK.

Schwaber, K. (2004). Agile project management with Scrum. Microsoft press.

Schwaber, K., & Beedle, M. (2002). Agile software development with Scrum (Vol. 1). Upper Saddle River: Prentice Hall.

Shepperd, M., & MacDonell, S. (2012). Evaluating prediction systems in software project estimation. Information and Software Technology, 54(8), 820-827. doi:10.1016/j.infsof.2011.12.008

Silverman, D. (2007) A Very Short, Fairly Interesting and Reasonably Cheap Book about Qualitative Research. London: Sage.

Silverman, D. (2013). Doing qualitative research: A practical handbook. SAGE publications limited.

Sommerville, I. (2011). Software engineering 9th Edition. ISBN-10, 137035152.

Takeuchi, H., & Nonaka, I. (1986). The new new product development game. Harvard business review, 64(1), 137-146.

Usman, M., Mendes, E., Weidt, F., & Britto, R. (2014, September). Effort estimation in agile software development: a systematic literature review. In Proceedings of the 10th international conference on predictive models in software engineering (pp. 82-91). ACM.

Walls, J. G., Widmeyer, G. R., & El Sawy, O. A. (1992). Building an information system design theory for vigilant EIS. Information systems research, 3(1), 36-59.

Way, T., Chandrasekhar, S., & Murthy, A. (2009). The Agile Research Penultimatum. In Software Engineering Research and Practice (pp. 530-536).

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Climate research, 30(1), 79-82.

Yin, R. K. (2017). Case study research and applications: Design and methods. Sage publications.

# APPENDIX A: THE THEMES OF THE SEMI-STRUCTURED INTERVIEWS

1. What position in case company? How long have you been working? Any previous positions in same company?
2. How is feature estimation done at the moment? Are the estimates inaccurate sometimes and if so, why? What unexpected things affect feature estimating?
3. What are the most important metrics for agile planning and feature estimation? Why?
4. How are features prioritized at the moment? How and where does it show in the processes?
5. What benefits would you see in improved estimation and planning? For example, if a tool existed which could always estimate correctly when a feature is completed.
6. Is the information recorded to Azure DevOps thoroughly? What feelings do employees have towards the information system? Do they feel it is meaningful or a boring and unnecessary part of the process?
7. How are user story points come up with? What methods do you use in the process? Any ideas to improve the process? How much time does bug fixes averagely take, since they are not estimated in story points?
8. Are the story point values updated if the estimations were done wrong? For example, if a story is significantly harder than expected. Should they be updated or not? Why?
9. How well can teams estimate their own velocity? What affects it?
10. How often the development teams communicate with other people than the members of the team? Who are interested in the status of the team?
11. If dashboards would be created for the company, what kind of information should be shown and who should be able to see it? Is some data missing currently which would be useful to be shown? Are team specific dashboards a good idea? Could dashboards help with communication?

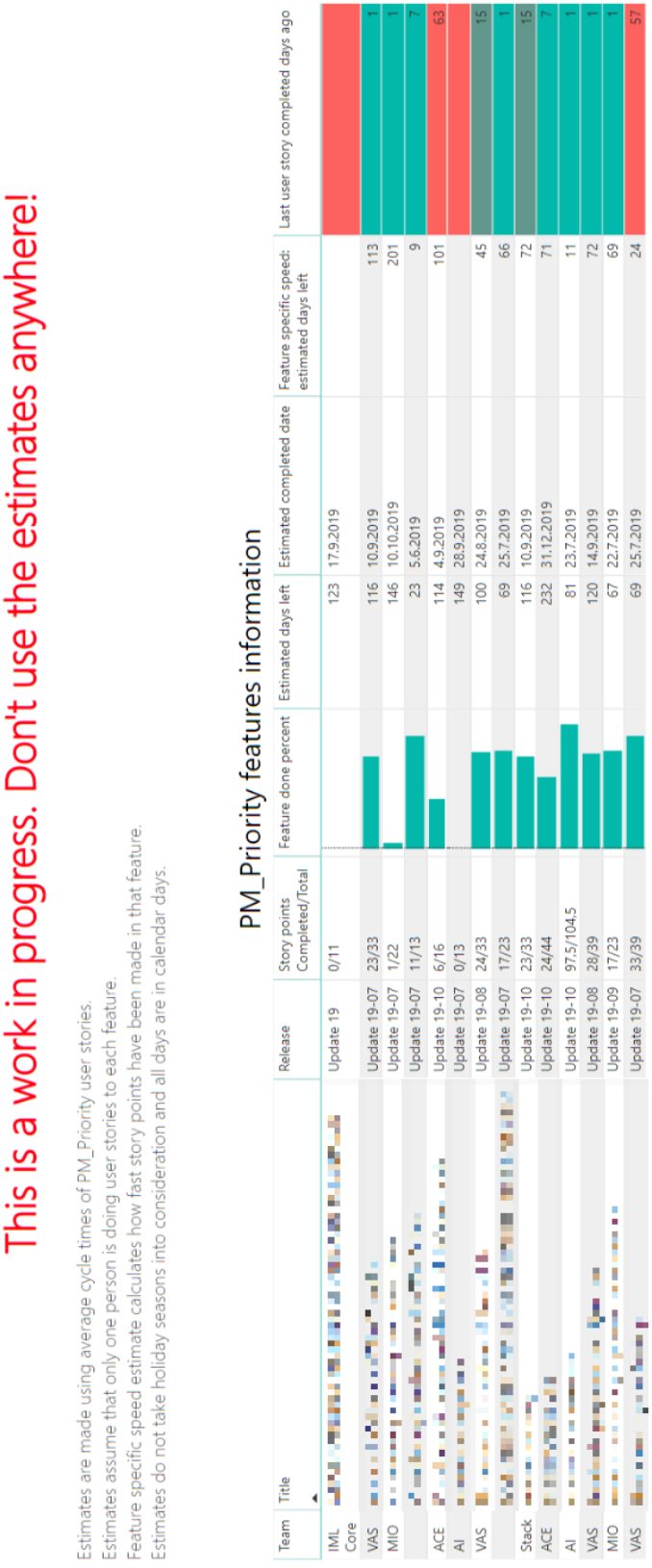
# APPENDIX B: DEMOGRAPHIC INFORMATION ABOUT THE INTERVIEWEES

*Information about all the interviewees and interviews.*

| # | Position (and former positions) | Length of the interview | Notes |
|---|---|---|---|
| 1 | Program Manager | 1:11:56 | |
| 2 | Software Developer 1 (QA Trainee) | 57:52 | |
| 3 | Software Developer 2 (QA Trainee) | 45:33 | Has been in the case company for 14 years. |
| 4 | Software Developer 3 | 1:14:53 | |
| 5 | Team Manager / Scrum Master 1 (Software Developer) | 1:12:21 | |
| 6 | Team Manager / Scrum Master 2 (QA Trainee, Software Developer) | 1:13:30 | |
| 7 | Team Manager / Scrum Master 3 (Lead Software Engineer) | 1:00:07 | Other current positions include: Chief research engineer and other team's Product Owner |
| 8 | Director of Product Management | 1:09:39 | |
| 9 | Product Manager 1 | 0:46:52 | |
| 10 | Product Manager 2 (QA trainee, Software Developer) | 0:59:14 | |
| 11 | Director of Quality Assurance | 1:08:36 | |

| | | | |
|---|---|---|---|
| 12 | Vice president of product development (Project Manager) | 0:28:51 | 10 years in the case company and information about all the teams. |
| 13 | Director of Software Development | 0:56:06 | |
| 14 | Director of Software Development | 0:42:38 | |

# APPENDIX C: FIRST ITERATION OF THE DASHBOARD



# APPENDIX D: TABLES FROM THE 4.4.2 DATA CHAPTER

*The names and descriptions of the data columns from Stories table.*

| Column name | Description of the content |
|---|---|
| Iteration End Date | The end date of the sprint in which the story data was recorded in the form of a timestamp. |
| Iteration Start Date | The start date of the sprint in which the story data was recorded in the form of a timestamp. |
| Iteration Path | The "Path" of the sprint in the system. It has the name of the project and the name of the sprint separated with a "/" -character. Usually the sprint name has the current year and the week number it was started. |
| Assigned To | The name of the person assigned to complete the user story. |
| Parent Work Item Id | The Work Item Id of the work item which is one step higher in the hierarchy if they are linked. |
| Activated Date | The date when the user story has been changed to active state in the form of a timestamp. |
| Work Item Id | Unique id for the work item. |
| State | The state of the work item. Includes New, Active, In Testing, Waiting for Merge, Impeded, Tested, Proposing Done and Closed. |
| Story Points | The effort estimation for the user story in the form of story points. Numeric value. |
| Tags | Optional tags, which can be added to any work item. |
| Title | Title/Name of the work item. |
| *Story Points text | Story Points value, but in text form instead of numeric value. |

| | |
|---|---|
| *Median Cycle Days | The median cycle time of user stories with the corresponding amount of story points. |
| Cycle Time Days | The time between Activated Date and Completed Date measured in days. |
| Completed Date | The date when the user story was completed in the form of a timestamp. Usually same as Closed Date. |
| Closed Date | The date when the used story was changed to closed state in the form of a timestamp. |
| Team | The name of the team which is responsible for the user story. |
| *Parent title | The title column of the parent work item. |
| *Parent tags | The tags set to the parent work item. |
| *Combined tags | The combined tags of the user story and the parent work item. |
| *Find PM_Priority | Informs if "PM_Priority" tag is found in the combined tags. |
| *Cycle time divided by story points | Numeric value of Cycle time in days divided by the amount of story points. |

*The names and descriptions of the data columns from Features table.*

| Column name | Description of the content |
|---|---|
| Iteration End Date | The end date of the sprint in which the feature data was recorded in the form of a timestamp. |
| Iteration Start Date | The start date of the sprint in which the feature data was recorded in the form of a timestamp. |
| Iteration Path | The "Path" of the sprint in the system. It has the name of the project and the name of the sprint separated with a "/" -character. Usually the sprint name has the current year and the week number it was started. |

| | |
|---|---|
| Release | In which release the feature is planned on releasing. |
| Parent Work Item Id | The Work Item Id of the work item which is one step higher in the hierarchy if they are linked. |
| *First story started date | The date when the first user story under this feature has been changed to active state in the form of a timestamp. |
| Work Item Id | Unique id for the work item. |
| State | The state of the work item. Includes New, In development, Released, Removed, Concepting, Closed. |
| *Total Story Points | The total amount of story points under the feature. Numeric value. |
| *Completed Story Points | The sum of completed story points under the feature. Numeric value. |
| *Story Points left | The sum of story points of the not completed user stories under the feature. Numeric value. |
| *Story Points Completed/Total | A text value informing how many story points have been completed out of the total. |
| Tags | Optional tags, which can be added to any work item. |
| Title | Title/Name of the work item. |
| *Estimated time to complete an old feature | Estimated time to complete a feature. This value is used in evaluating the accuracy of the estimates. |
| *Estimated days left | Estimated days left to complete the remaining user stories. |
| *Latest story completed date | The latest date when a user story under this feature has been completed. |
| *Teams in feature | Combined teams from the user stories under the feature. |

| | |
|---|---|
| *Feature done percent | The completed story points divided by the total story points of a feature. |
| Team | Team which is marked responsible for the feature. |
| *Avg developers in feature per sprint | The average amount of developers developing the feature per sprint. |

*The names and descriptions of the data columns from Story points table.*

| Column name | Description of the content |
|---|---|
| Story Points | The distinct values of story points amount in numeric format. |
| *Story Points text | The distinct values of story points amount converted to text format. |
| *Median Cycle Time | The median cycle time of user stories of the same story points amount. |
| *PM_Priority Median Cycle Time | The median cycle time of user stories of the same story points amount and with the tag PM_Priority. Also, the most significant outliers are filtered away. |

*The names and descriptions of the data columns from Story history -table, which were used.*

| Column name | Description of the content |
|---|---|
| Iteration Path | The "Path" of the sprint in the system. It has the name of the project and the name of the sprint separated with a "/" -character. Usually the sprint name has the current year and the week number it was started. |
| Assigned To | The name of the person assigned to complete the user story. |
| Parent Work Item Id | The Work Item Id of the work item which is higher in the hierarchy if they are linked. |

| | |
|---|---|
| Iteration Start Date | The start date of the sprint in which the feature data was recorded in the form of a timestamp. |
| Work Item Id | Unique id for the work item. |
| Story Points | The effort estimation for the user story in the form of story points. Numeric value. |
| Title | Title/Name of the work item. |

# APPENDIX E: TABLES FROM THE 4.4.3 DEVELOPMENT CHAPTER

*The names and explanation of the data columns, which were made by the author.*

| Column name | Explanation |
|---|---|
| Find PM_Priority | PM_Priority was the program manager's tag to mark the most important features and stories, which could be used for the artifact. This column informed if the item had the tag. |
| Cycle time divided by story points | This value was calculated for the comparing of the different sized stories and measuring the linearity of the story points. |
| Story Points text | Story points' values were needed in text format to divide the stories to these story point size groups with the same size. With numeric values the calculations happened differently. |
| Median Cycle Time | The median cycle time of the group which would be specified by the story point text field. For example, the median cycle time of all stories with the size estimate 2 story points. Calculated in days. |
| Estimated days left | Value calculated by summing up the median cycle days of all the uncompleted user stories. |
| Teams in feature | Several teams can develop user stories for a feature, but only one can be assigned to it. This column gathers the teams from the user stories under the feature. |
| Avg developers in feature per sprint | This is needed for calculating the estimates in V-method, it is relevant information when forming an estimate. |

| | |
|---|---|
| PM_Priority Median Cycle Time | The Median Cycle Time of the important features. Information to compare to the other user stories. |
| Estimated time to complete an old feature | Value calculated by summing up the median cycle days of all the user stories in a feature and dividing it with the average developers in feature per sprint. |

*The names of measures and their descriptions.*

| Measure name | Description |
|---|---|
| Actual days to finish a feature | The amount of days between first story started and last story completed. |
| Amount of developers with active user stories | The amount of developers who currently have active user stories related to a specific feature. |
| Completed % at date | A completed percentage of a feature which changes dynamically depending on the date chosen. |
| Completed points at date | The amount of completed story points in a feature which changes dynamically depending on the date chosen. |
| Count of assigned to | Counts different names who have been developing a specific feature. |
| Difference: actual and AVG | Calculates the difference between actual days to finish a feature and the estimated time to complete an old feature using AVG method. |
| Difference: actual and FSS | Calculates the difference between actual days to finish a feature and the estimated time to complete an old feature using FSS method. |
| Difference in % (AVG) | The percentage of measured Difference: actual and AVG. |

| | |
|---|---|
| Estimated complete date | The date calculated by adding the estimated days left divided by the active developers to Estimation start date. |
| Estimation start date | Estimation start date was decided to be the start date of the latest active sprint. |
| Feature specific velocity: Points per sprint | The amount of story points completed divided by the number of sprints done. |
| First story completed date | The earliest completion date of a user story in a feature. |
| First user story started | The earliest activated date of a user story in a feature. |
| FSS dynamic estimate | An FSS estimate of days to complete a feature which changes dynamically depending on the date chosen. |
| Last user story completed | The latest user story completed date in a feature. |
| MAE | Mean average error of which formula has been presented earlier in this research. |
| Points per day per developer | The completed story points divided by the amount of days the story has been active divided by the average amount of developers per sprint. |
| Velocity at date | The amount of story points completed per sprint related to a specific feature, which changes dynamically depending on the date chosen. |

# APPENDIX F: THE FINAL DASHBOARD