University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Industrial Economy / Project Management	Spring semester, 2017 Open access
Writer: Rolf Andreas Boiten	.... (Writer's signature)
Faculty supervisor: Sigbjørn Tveterås External supervisor(s): Sigbjørn Tveterås	
Thesis title: The Necessity of Estimation in Software Development Projects.	
Credits (ECTS): 30	
Key words: Software development, Programming, Estimation, Agile Project Management, Scrum, #NoEstimates, UX Design	Pages: **43** + enclosure: **3** Stavanger, **13.07/2017** Date/year

Front page for master thesis
Faculty of Science and
Technology
Decision made by the Dean October 30th 2009

1 Abstract

Today most Software Development Companies use Agile Project Management to manage their project, here the decisions are made as the project progresses to keep the agility in the project. This has been a successful transition with a lot of benefits, but many agree that estimation remains a great area of concern. The most important reasons for this are the unknown parts of the project that make it hard to calculate accurate estimates. Many teams are unable to deliver what they have committed to, because their estimates are often way off. This results in large project overruns, which prevent the organizations from reaching the market early enough and can result in bad market performance.

In this thesis, the necessity of estimation in Software Development Projects is discussed. Interviews with experienced project managers where held to find out how theory coincides with the Software Development Industry, Also, some of the main goals of these interviews were to find out more about how estimates in Software Development Projects are created within the industry today, and how necessary it is to have accurate estimates or estimates at all, for project managers.

This thesis shows that there are a lot of benefits and often necessary to have good estimates. The thesis also shows that it is important to have early specification of parts of the project, where the decisions about the software should be made as early as possible to create the most accurate estimates. But only the parts of the project which are soon going into development must be heavily specified, otherwise agility will be lost. For less specified parts of the project rough estimates should be enough to make the right decisions.

I Accreditation

First and foremost, I would like to thank my faculty supervisor Sigbjørn Tveterås, for taking the role as my supervisor even without having any special knowledge about Software Development. He helped me to change the thesis when I came to him for guidance on what to do, when I could not gather enough information and data to complete my previous topic. He has also helped me structure the thesis, answered my questions and given me information to prevent me from falling into one of the many pitfalls when writing a master’s thesis.

II Motivation

I chose this thesis because I have a degree as a Computer Engineer, and I wanted to write a thesis combining my background from computer engineering with Project Management. Through my experience as a Software Developer, the difficulty of making accurate estimates has fascinated me for a long time. I therefore chose this thesis in order to find out more about why accurate estimates are needed and how accurate they need to be.

III Figure list

IV Table of content

1 Introduction

1.1 Estimation in Software Development Projects

Today, most Software Development Organizations use an agile process for Software Development where decisions are made as the project progresses in order to keep the agility in the project. These organizations claim that they have had large benefits from transitioning to Agile Software Development, but many of them still agree that estimation remains a great area of concern due to the unknown part of the project which makes it hard to calculate accurate estimates. Many teams are unable to deliver what they have committed to, and their estimates are often way off. These inaccurate estimation results in large project overruns, which prevents the organizations from reaching the market early enough and can result in bad market performance (Lakkaraju & Goswami, 2016). Some even goes as far as saying that most of the time spent making estimates is just a waste of time, because the accuracy is so bad and often a lot of time is used making them (Holub, 2015).

To challenge this, the thesis is going to set light on “The Necessity of Estimation in Software Development Projects”. By finding out:

1. What estimates are?
2. Why estimates are needed?
3. How estimates should be used?
4. How estimates should be made?
5. Who should make the estimates?
6. How accurate the estimates should be?

1.2 Limitations

This thesis is mostly going to focus on Agile Projects within Software Estimation, even though some methods discussed in this thesis can possibly make the projects less agile. Much of the theory discussed can also be used for managing and estimating within less agile projects managed by a more traditional like Project Management Structure, where projects have a target or fixed price and more strict requirement documents. Scrum will be used and referenced to as the main framework for Project Management within Software Development in this thesis, because Scrum is agile and appears to be the Project Management Structure or Framework mostly used in Software Development Projects.

1.3 Methodology

For gathering information about estimation theory within Software Development, this thesis uses Steve McConnell’s book “Software Estimation: Demystifying the Black Art”. McConnell is one of the most well known people when it comes to estimation in Software Development, and he is referenced in almost all lectures and presentations about Software Estimation. This thesis uses presentations and lectures given by Vasco Duarte and Allen Holub about #NoEstimates to establish how this new methodology works and its usefulness in replacing the traditional estimates. To gather information about terms used in this thesis and to establish a notion of how Project Management is done within Software Development. Scrum Guides written by The International Scrum Institute and Ken Schwaber & Jeff Sutherland are used as well as various articles.

To find out how theory coincides with the Software Development Industry, interviews with experienced project managers where held. Also, some of the main goals of these interviews were to find out more about how estimates in Software Development Projects are created within the industry today, and how necessary it is to have accurate estimates (or estimates at all) for project managers.

2 Theory

2.1 Estimates, Targets, Commitments and Planning

An estimate is a term that is often confused or misused especially when it comes to Software Development. Estimates done by a software developer are often considered to be sufficiently accurate to be used as actual commitments and a plan to meet a target (McConnell, 2006). A target is a statement of a wanted business objective, for example “the first release needs to be finished before summer”. A good reason for businesses to establish targets for projects is to define what the desirable business objectives are and getting as much value as possible from the project. Even if these targets can be classified as desirable or even mandatory, it does not necessarily make them achievable. A commitment is a promise to deliver exact functionality at a specific level of quality by a certain date. Commitments are mostly based on estimates, but this does not mean that the commitment must be the same as the estimate (McConnell, 2006). An estimate should not be a promise nor binding. A reason for this is that if people know the estimates are going to be used as a promise and can be used against them, this can affect the estimate they make (Beckerleg, 2014).

Estimates should be unbiased and the goal of the estimate should be accuracy and not to seek a particular result. The goal of planning is on the other hand to seek a particular result, and the plans are deliberately created to achieve specific outcomes (McConnell, 2006). Plans are based and created from the knowledge gained by the estimates, and some of the risk in the project can be assessed by comparing the plans with the estimates. Though even if the planned target is the same as the unbiased estimate there is still a level of risk in the project. In Steve McConnel own words “A good estimate is an estimate that provides a clear enough view of the project reality to allow the project leadership to make good decisions about how to control the project to hit its targets” (McConnell, 2006). This statement means that according to McConnel estimates do not need to be highly accurate, only accurate enough for the project managers to use them for making good decisions to control the project and its future.

2.2 Benefits of Accurate Estimates

2.2.1 Desirable Project Attributes

Software Development Projects try to achieve many goals for a project, but the most common goals they strive for are (McConnell, 2006):

- Shortest possible schedule for getting the desired functionality with the best quality level.
- Lowest possible cost to deliver desired functionality within the desired time.
- Most functionality and highest possible feature richness for the time and money available.

These and other goals are prioritized very differently from person to person. Agile Development focuses on flexibility, repeatability, robustness, sustainability, and visibility (Cockburn, 2002). Executives tend to focus on predictability (McConnell, 2006). Following an Agile Project Management style is therefore difficult if one of the goals is to please the executives also. To accomplish this, accurate estimates are necessary.

2.2.2 Benefits of Accurate Estimates

Having accurate estimates creates many benefits when it comes to Software Development Projects. Some of these are:

- ***Better budgeting***. Accurate estimates lead to accurate budgets and forecasting costs are important (McConnell, 2006).
- ***Higher quality***. Schedule-stress-related quality problems are avoided. When schedule pressure is extreme, there are about four times as many defects reported in the release software (Jones, 1994).
- ***Early risk information.*** A very common mistake in software development is to fail to correctly interpret the meaning of an initial mismatch between project goals and project estimates (McConnell, 2006). Detecting a mismatch early gives more options that will be available for corrective action.
- ***Improved status visibility.*** A good way to track progress is to compare planned progress with actual progress. A project with a large mismatch between planned an actual progress will stop using the actual plan and stop comparing progress (McConnell, 2006).
- ***Better coordination with nonsoftware functions***. Without a reliable software schedule, related functions like sale or marketing campaigns can start to slip which can cause the entire project schedule to slip (McConnell, 2006).
- ***Increased credibility for the Development Team.*** A project team that holds its ground and insists on an accurate estimate will improve its credibility within its organization (McConnell, 2006).

Unfortunately, accurate estimates are often very hard and sometimes almost impossible to make.

Because of this, various estimation errors ranging from small to business bankrupting errors, still affect the Software Development Industry.

2.3 Estimation Errors

Estimation errors come in many different sizes. These errors can affect the project in countless ways and there are endless possible sources that can create them. Therefore, removing all of them would be close to impossible and the effort should be put into preventing estimation errors with serious consequences. There are four general sources for estimation errors (McConnell, 2006):

- Inaccurate information about the project being estimated.
- Inaccurate information about the capabilities of the organization that will perform the project.
- Too much chaos in the project to perform accurate estimations.
- Inaccuracies created by the estimation process itself.

In Software Development, wanted and needed functionality and design of the product gradually change during the project. In the beginning, there is a general product concept (the vision of the software), this concept is refined based on the product and product goals. The goal can be to deliver a specific amount of functionality or to deliver as much functionality as possible within a fixed budget (McConnell, 2006). In every Software Development Project, thousands of decisions must be made on how to meet requirements and achieve specific functionality in the software. These decisions will affect the time needed to implement the specific functionality and influence the rest of the project. Therefore, as more decisions are made the uncertainty of estimation will be reduced. The Cone of Uncertainty in Figure 2.1 shows how estimates becomes more accurate as a project progresses.

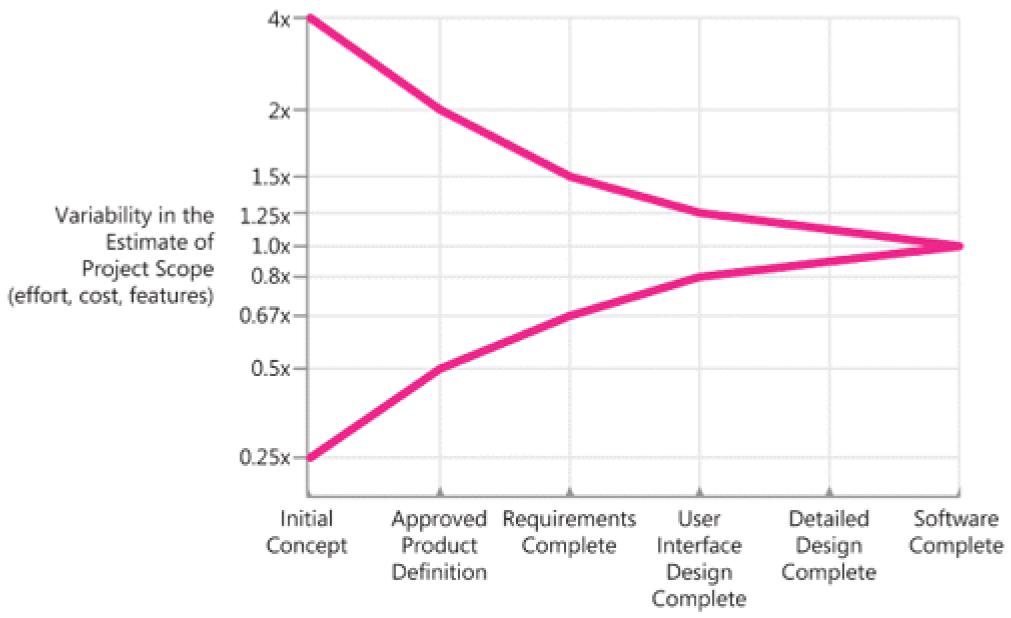


Figure 2.1: The Cone of Uncertainty *(McConnell, 2006). This graph shows the best possible accuracy on estimates, with the two lines showing the smallest high and low points for the variance related to the actual estimate, at different milestones in the project*

The graph shown in figure 2.1 shows that estimates created early in the project are subject to a high degree of error. In the initial concept phase the total range of the inaccuracy factor for the estimate is 4x divided by 0.25x, or a total of 16x. The reason the estimate contains uncertainty is that the Software Development Project itself contains uncertainty. Because of this, the best way to reduce uncertainty in estimates is to reduce uncertainty in the project itself. This can be done by making decisions (McConnell, 2006). At first glance on the Cone of Uncertainty it would be easy to presume that it takes a long time before having good accuracy on estimates, and that estimates are not very good before the project is nearly done. The reason for this is because of the milestones in the figure that are equally spaced, and the assumption that the horizontal axis shows calendar time. The figure shows that as early as when the user interface design is completed, the total range is 1.25x divided by 0.8x, or 1.56x. This creates a much more acceptable accuracy on the estimates. It is important to consider that the Cone of Uncertainty represents the best-case accuracy that it is possible to have in software estimates. It is therefore possible to do much worse, but not possible to make more accurate estimates at given points, only to be more lucky (McConnell, 2006).

2.4 Scrum

2.4.1 Definition of Scrum

Scrum is an Agile Project Management framework, mostly used within Software Development. Scrum uses an iterative, incremental approach to optimize predictability and control risk (Schwaber & Sutherland, 2016). Today Scrum is widely used, and most companies use a management framework resembling or based on Scrum for their Software Development Projects. Scrum can be used internal and external for all kinds of Software Development Projects from parts of bigger systems to complete software (International Scrum Institute, 2016). Figure 2.2 shows the objects and events that is part of traditional Scrum and their relations within the Scrum Framework. The Sprint is not shown in this figure, but its actual location would be behind the Sprint Backlog and the Scrum Team, since the Scrum Team is responsible for doing the Sprint and they use the Sprint Backlog to keep track of the Sprint. Objects and events from the Scrum Framework that have an impact on the estimates will be explained.

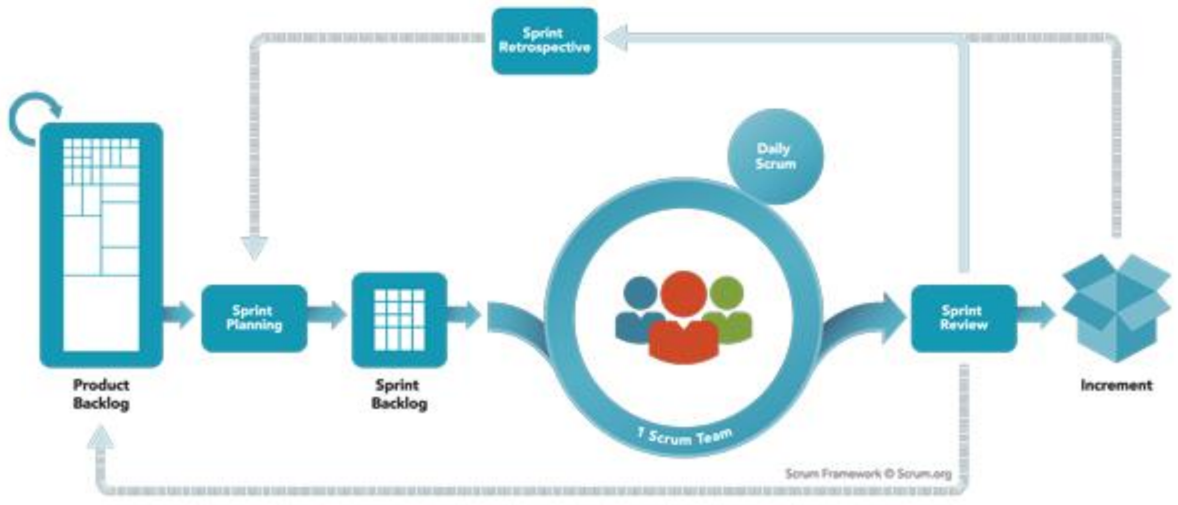


***Figure 2.2: Scrum Framework**. The flow and relations within the Scrum Framework (scrum.org, 2017)*

In the Scrum Framework, there are three main pillars that need to be followed in all parts (Schwaber & Sutherland, 2016):

- ***Transparency***. Important aspects of the process and the project must be visible to those who are responsible for the outcome of the project. All observers need to share a common understanding of what is being seen.

- ***Inspection.*** The progress towards the Sprint Goal and Scrum Items must be frequently inspected by Scrum users to detect undesirable variances. Still the inspection should not get in the way of the development process.
- ***Adaptation.*** The process or the material being processed must be adjusted when one or more aspects of a process deviate outside acceptable limits. Otherwise the resulting product will be unacceptable.

2.4.2 The Scrum Team

The Scrum Team consists of the Scrum Master, the Product Owner, and the Development Team. Scrum Teams are cross functional and can accomplish the work without depending on others that are not a part of the team. The Product Owner role is to maximize the actual business value of the product and the work done by the Development Team. The Product Owner is responsible for maintaining and managing the Product Backlog so that this is transparent, clear to all, informative and updated. The Development Team is s self-organizing unit consisting of more than two but less than nine people. This team is responsible for delivering a potentially releasable product at the end of each Sprint. The Scrum Master is a servant-leader role for the Scrum Team, and makes sure that the team understands the Scrum Theory, practice and rules, and that these are followed through the project. The Scrum Master helps the Product Owner to get an understanding of how to maintain and manage the Product Backlog and the Development Team with self-organizing, removing impediments from progress and creating a high value product. The Scrum Master also facilitates Scrum Events and overall causes changes that increase the productivity of the Scrum Team (Schwaber & Sutherland, 2016).

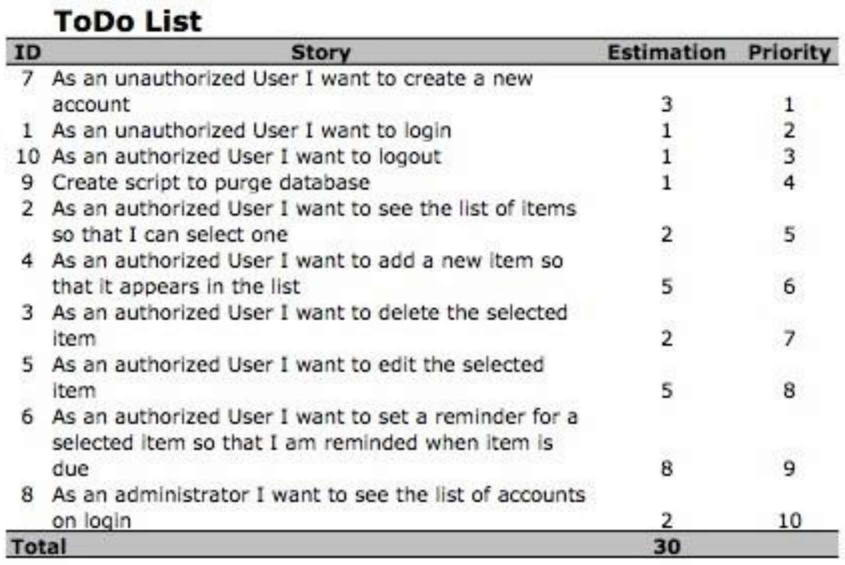
2.5 Events

2.5.1 Sprint

The Sprint is the backbone of Scrum, which represents a limited period of one month or less. During the Sprint, a usable and potential releasable product increment is created. Therefore, each Sprint may be considered as a project with a short horizon. During the Sprint, no changes that could endanger the Sprint Goal are made. Before the Sprint begins, a Sprint Planning Meeting is held where a plan is created by the collaborative work of the Scrum Team. The goal of the Sprint Planning Meeting is to find out what can be delivered in the next Sprint and how this can be achieved. The Product Owner discusses the objective that the Sprint should achieve and the wanted Product Backlog Items to be done. This and the Development Teams projected capacity is used to forecast the Product Backlog Items the Development Team will deliver during the next Sprint, and the Scrum Team crafts a Sprint Goal. The Sprint Goal is an objective that will be met within the Sprint, by implementing the Product Backlog Items selected for the Sprint. These selected Product Backlog Items plus the plan for delivering them is called the Sprint Backlog. The Sprint Goal also works as a guidance to the Development Team to clarify why they are building this increment (Schwaber & Sutherland, 2016).

2.5.2 Product Backlog

The Product Backlog is basically a list of all things that need to be done within the project. The items in the Product Backlog can be technical or user-centric e.g. in the form of User Stories, and replaces the traditional requirements specification artifacts. Figure 2.3 shows a typical Product Backlog with ten User Stories, where each User Story is briefly explained, estimated and given a priority. As most Product Backlogs, the User Stories in Figure 2.3 are sorted after priority, with the highest prioritized User Stories at the top.



ToDo List

ID	Story	Estimation	Priority
7	As an unauthorized User I want to create a new account	3	1
1	As an unauthorized User I want to login	1	2
10	As an authorized User I want to logout	1	3
9	Create script to purge database	1	4
2	As an authorized User I want to see the list of items so that I can select one	2	5
4	As an authorized User I want to add a new item so that it appears in the list	5	6
3	As an authorized User I want to delete the selected item	2	7
5	As an authorized User I want to edit the selected item	5	8
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9
8	As an administrator I want to see the list of accounts on login	2	10
Total		30	

Figure 2.3: The Scrum Product Backlog *(Scrum Institute, 2017). A typical Product Backlog, where the User Stories are briefly explained, estimated and given a priority*

The Product Owner uses the Product Backlog during the Sprint Planning Meeting to describe the upcoming entries to the team. Entries in the Product Backlog should always add create value for the customer, where they are listed and arranged according to prioritization. The level of detail depends on the position of the item within the Product Backlog, and all items regardless of position are estimated. There should not be any no action-items or low-level tasks in the Product Backlog. The Product Backlog is a living document, so new items are discovered, described and added to the list during the project, while existing items are changed or removed as appropriate (International Scrum Institute, 2016). During the Sprint Planning Meeting items from the Product Backlog are used to create a Sprint Backlog, containing all items that could be fully implemented within the next Sprint. To further maintain the Product Backlog a Backlog Grooming sessions can be held.

2.5.3 Backlog Grooming

During Backlog Grooming, the Product Owner and some or all members of the team review the Backlog Items. They ensure that the Backlog contains the appropriate items, that they are prioritized, estimated to an appropriate degree and that the items on the top of the Backlog is ready for delivery (Agile Alliance, 2017).

2.6 Dividing the Project

2.6.1 Project Scope

The Project Scope defines the software being built and what it should deliver, this is often shown in a specification document of some kind. In Agile Projects the Scope is mostly only defined by high level requirements in the form of User Stories, scheduled in the release plan. Detailed requirements are also necessary but mostly created when needed (Thomas, 2008). Often, customers try to add more functionality or they changes their minds about what they want. In this case the team may be asked to do something outside the initial agreement, often without increasing budget or time frame. When this happens, it is called Scope Creeps (Yatzeck, 2012). In Agile Projects, it is not considered Scope Creeps if the team has not started working on the specification and detailing on that part of the project. Because being able to change parts of the project while the project progresses is what makes it an Agile Project.

2.6.2 Epics, Stories and Tasks

In Agile Projects, it is common to have different terms to classify the things that need to be done, related to how much they are specified within the projects. The terms used varies from organization to organization. In this thesis the terms Epics, Stories and Tasks will be used. These terms appear to be the most used terms, though Scrum Guides mentions small and large Stories (International Scrum Institute, 2016; Schwaber & Sutherland, 2016). The relationship between the terms are shown in Figure 2.4.

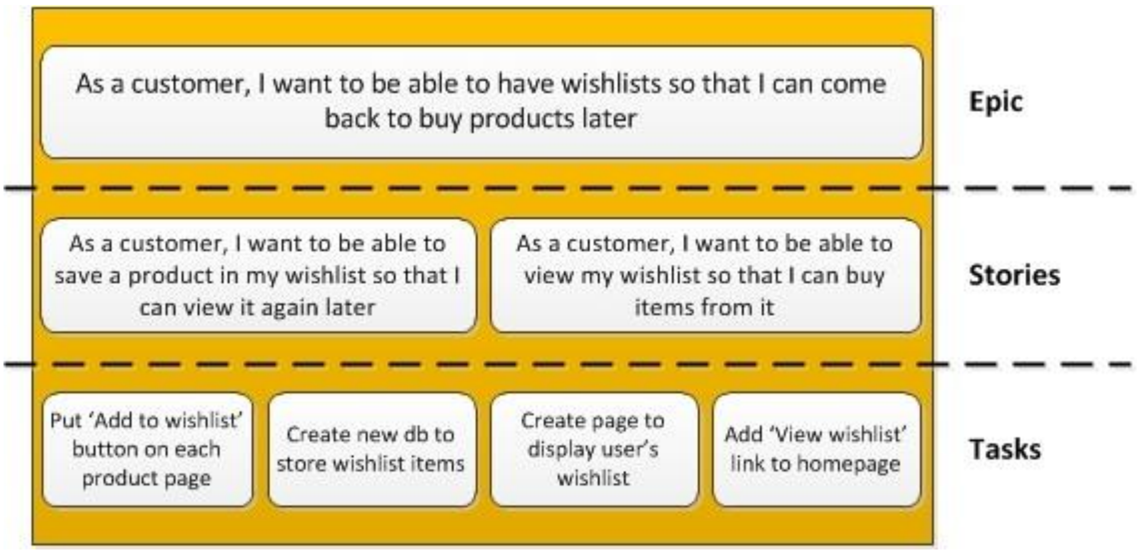

As a customer, I want to be able to have wishlists so that I can come back to buy products later
Epic
As a customer, I want to be able to save a product in my wishlist so that I can view it again later
As a customer, I want to be able to view my wishlist so that I can buy items from it
Stories
Put 'Add to wishlist' button on each product page
Create new db to store wishlist items
Create page to display user's wishlist
Add 'View wishlist' link to homepage
Tasks

Figure 2.4: Epics, Story, Task (Lowe, 2014). *Relationship between the terms for classification of items used in this thesis*

2.6.2.1 Epics

Epics are large bodies of work and can be broken into several smaller stories that can span across multiple Sprints. In some cases, they can even span over more than one project. Epics often change in Scope during the project as a natural aspect of Agile Development (Radigan, 2017).

2.6.2.2 User Stories or Stories

User Stories also known as Stories, are Software System Requirements that are expressed in a few short sentences, preferably using only non-technical language (Radigan, 2017). The User Stories focuses on exactly what the user needs/wants without going into more specific details on how to achieve it. The notion of User Stories was created to address the issue that estimation culture is focusing on time and not functionality of a program. Stories are now a central part of putting software together, but Stories are not functional requirements and are mostly hard to estimate (Holub, 2015). A typical template for a User Story is (International Scrum Institute, 2016):

As a [user], I [need/want] [action/functionality] so that [goal/achievement].

2.6.2.3 Tasks

Tasks are the elements of User Stories, but they are not used by all teams (Lowe, 2014). The term task is often mixed with the terms acceptance criteria and conditions, and they are hard to separate. The SMART acronym is often followed when creating tasks. What actually SMART stands for is a hot topic, but often the letters represent: **S**pecific, **M**easurable, **A**chievable, **R**elevant, **T**ime-boxed (Lowe, 2014).

2.7 Estimation Methods

All items within the Product Backlog must be estimated according to Scrum Theory. The Product Owner needs to know how difficult the work will be, although it is not recommended that the Product Owner attends when the estimations are done (International Scrum Institute, 2016). In traditional Scrum, it is not common to use terms of time, but more abstract metrics like t-shirt sizes (XS, S, M, L XL) or Story Points.

2.7.1 Story Points

Story Points represents the actual effort needed to develop a Story. There are several factors that needs to be considered when creating an estimate (Chon, 2016):

- The amount of work to do
- The complexity of the work
- Any risk or uncertainty in doing the work

If Story Points are used for the estimation, everyone who is doing the estimations needs to agree on the same metrics, a common scale or sequence to use and roughly how big a Story Point is (International Scrum Institute, 2016).

2.7.2 Fibonacci Format

When creating estimates, the Fibonacci sequence (1,2,3,5,8,13,21) or Fibonacci-like sequences (1,3,8,12,20) is often used as values instead of just using all possible numbers (International Scrum Institute, 2016). This abstraction tends to work better than more exact estimates. The reason that the sequences works better is debatable, but it is probably a combination the fact that it is easier to use a high number for an estimate and that most estimates will be rounded up to nearest number in the sequence. Using sequences instead of a more exact estimate can also be a faster way of estimating.

2.7.3 Planning Poker

Planning Poker is an agile estimating technique that has become very popular when following the Scrum Framework, even though the basics of this technique has been used for a long time (Hartman, 2009). Poker Planning is very simple and accurate enough to use for Agile Project Planning, where the only thing needed is a deck of numbered cards or notes per participant. The game plays as followed (International Scrum Institute, 2016):

1. Product Owner presents the Story to be estimated, and the Scrum Team asks questions to understand the Story if it is unclear.
2. Each member of the Scrum Team chooses their own card from their deck, representing their estimate.
3. When everyone has chosen their cards, all cards are revealed at once.
4. Those with high or low estimates can explain their estimate.
5. The estimation starts again, and repeats itself until a consent is found.
6. This is done for all Stories.

2.7.4 Other Estimation Techniques

There are a lot of other possible estimation techniques. McConnel suggests in his book that when something is being counted, the countable items must be (McConnell, 2006):

- Highly correlated to the size of the software.
- Available for counting early in the project.
- Where you can count to at least 20.
- Understand what you are counting.
- Request minimal effort to count.

Another much used technique is Three-Point Estimation, where three values are produced for every estimated item based on experience or best-guess (Maksimovic, 2012):

- The best-case estimate.
- The most likely estimate.
- The worst-case estimate.

How the actual result estimate is calculated from the Three-Point Estimation can vary from project to project.

Expert judgment by one person with or without using statistical analysis or algorithms to compare current project with previous projects are also often used as an estimation method (Jørgensen, 2005). This method ranges all the way from the most basic one where one person estimates by just writing down an estimate with his gut feeling to estimates done by a group of people that afterwards gets compared by an advanced computer program using a large database of other projects before a new estimate with including variance or standard deviation is calculated.

2.8 #NoEstimates

2.8.1 The Concept

The idea behind #NoEstimates is to implement a method for helping manage a project and collaborating with stakeholders in a transparent and productive way. The team will remain focused doing only valuable work which is the work that matters the most. This goes well with the Agile principle that states, “Individuals and interactions over processes and tools” (Lakkaraju & Goswami, 2016). #NoEstimates focuses on dividing projects into small User Stories to reduce the need for estimation and starting to measure progress and foresee the future of the project. Some of the most important features with #NoEstimates are (Lakkaraju & Goswami, 2016):

- The team is allowed to focus on creating value from day one.
- Good decisions can be made based on project progress.
- Monitoring actual progress and predicting a release date can be done without much effort.

2.8.2 #NoEstimates Product Backlog

When Product Backlog Items are being estimated, the top 10 Stories are probably not going to change, but the next 30 are going to change based on information gained from the first 10 Stories. After this, most Stories are just wild fantasy and could be thrown away. Estimating the first 10 Stories is mostly waste, because these Stories are under production or being produced so soon that an estimate will not have much value. For the next 30 Stories estimates are mostly invaluable, because they are going to be based on incorrect assumptions about how the software works and what it is going to accomplish (Holub, 2015).

2.8.3 Replacing Estimations with Projections

When running a business, it is necessary to have some sort of projections, because time matters and there is often a need to have some control over or projection for the future. #NoEstimates solves this by counting User Stories instead of estimating them. A Cumulative Flow Diagram can be used to measure progress, which is the actual rate the team is completing Stories. The diagram can also be used to show when the project is most likely going to be finished. Figure 2.5 is a Product Backlog Cumulative Flow Diagram, and shows the amount of User Stories in the Product Backlog at different dates. In the Figure, the upper line shows the predicted increase in total User Stories over time and the diagonal line shows the predicted number of User Stories that are finished. The point where the two lines crosses is the predicted date the project is finished.

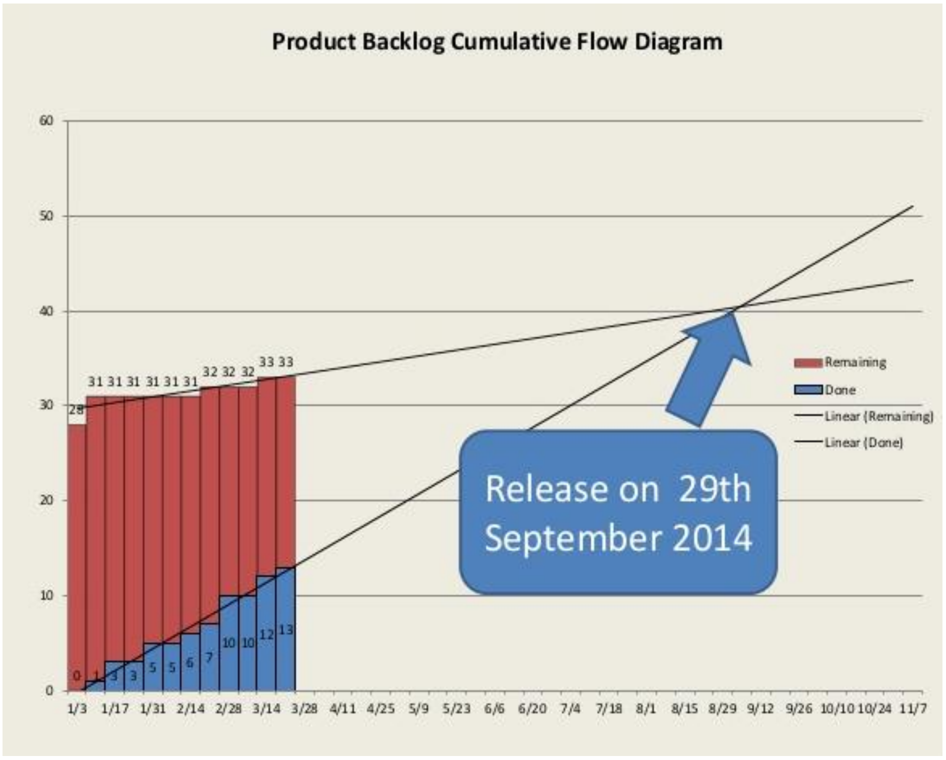


Figure 2.5: Product Backlog Cumulative Flow Diagram *(Duarte, 2014). The y-axis shows number of User Stories in the backlog and the x-axis shows time. The red part of the bars shows remaining User Stories and the blue part shows the finished ones. The upper line shows the predicted increase in total User Stories over time and the diagonal line shows predicted number of User Stories that are finished. The point where the two lines crosses is the predicted date the project is finished*

2.8.4 The Software Development Process

Duarte, one of the most well-known spokes persons for #NoEstimates, recommended to follow these simple steps during the project:

1. Select the most important piece of work you need to do.
2. Break that work down into risk-neutral chunks of work. Risk neutral means that if the delivery fails, this will not lead to bankruptcies nor have large negative impact on the project.

3. Develop each piece of work
4. Iterate and refactor

If the system is stable it is possible to accurately predict how many Stories the team will deliver over a low period. For a system to be considered being stable it must follow two rules:

1. The velocity cannot be outside limits three times in a row.
2. There cannot be five or more points trending up or down.

If the system is not stable one simply has to wait to find out what the new throughput will be (Duarte, 2014).

2.9 Specification

2.9.1 General specification

The specification process of a Software Project, is a process to find out and describe the requirement of the software the customer wants in a gradually greater level of detail. The specification should be unambiguous and accurate so that the requirements only has one interpretation (Thakur, 1999). Often in projects with a Traditional Project Management structure it is common that the customer has created a highly detailed requirements document that contains the basic specifications for the whole project. Though in Agile Project Management these documents are not detailed and sometimes not created at all. Specification can be done part by part in a project or the whole project at once. In Agile Projects, it is important to keep the agility, and therefore only specify in great detail the parts of the software or items that are soon going into development. Much of the specification proses can be done by a UX designer, who will collaborate with the customer to design the software.

2.9.2 User Experience Designer

User Experience Designers mostly known as UX Designers in a Software Development Project has the responsibility of making sure the product feels good to use. They ensure that the software's logic flows from one step to the next (Ming, 2014). UX Designers makes sure that the software is useful, easy to use, intuitive and delightful to interact with. Their main goal is to enhance the experience the user have while using the software, and make sure they find value in the product (Lanoue, 2015).

The UX Designers in Software Development Projects often interact with the customers and users to specify the Software. They establish the customer’s need, helps them make decisions on what the User Interface should look like and which functions they need I the software. Mostly based interaction and experience gained from the potential users of the software.

One other benefit from UX Design is that it prevents time being used on coding before most decisions are made. It also makes estimating and the software development easier for the Development Team since they know more specific what the customer wants and how it is supposed to look like.

3 Results

3.1 Setup of the Results

One of the main goals of this thesis was to find out more about how estimates in Software Development Projects are created in companies today, and how necessary it is to have accurate estimates (or estimates at all) for project managers. To solve this, four well experienced project managers from two different companies were selected for interviews. All project managers had background as software developers and experience ranging from 18 to 35 years within Software Development.

All interviews were anonymous and held at the project managers own company locations. The project managers where informed about #NoEstimates and the results presented by Vasco Duarte (Duarte, 2014). The project managers were also encouraged to answer all questions according to their opinion and not how they thought they were supposed to make estimates or how it was done in their organization. The full interview guide is added as Appendix 1 where more information about the questions and the interview process can be found.

The results from each question are calculated as percentage. The answers for each question are created by the project managers, and the value given to each answer is calculated by:

- How the answer is prioritized by the person interviewed if he had more than one answer.
- How many of the people that were interviewed gave the same answer.

An example of how the calculation is done is shown in Appendix 2.

3.2 Results from Interviews

In this section, the results of each question in the interviews is shown and commented.

Figure 3.1 shows the answers for the question: " In your opinion, how accurate should estimates be before a project proposal?" Answers from this question shows that most of the project managers agreed that only rough estimates are needed before a project proposal is given. Even though one project manager wanted every single item in the project broken down to tasks and estimated before the proposal was made. Where the specification part would be done using UX Design, and the estimation part afterwards by the Development Team. The project manager had firsthand experience with projects where things were done like this, where the pre-project would be paid by the customer.

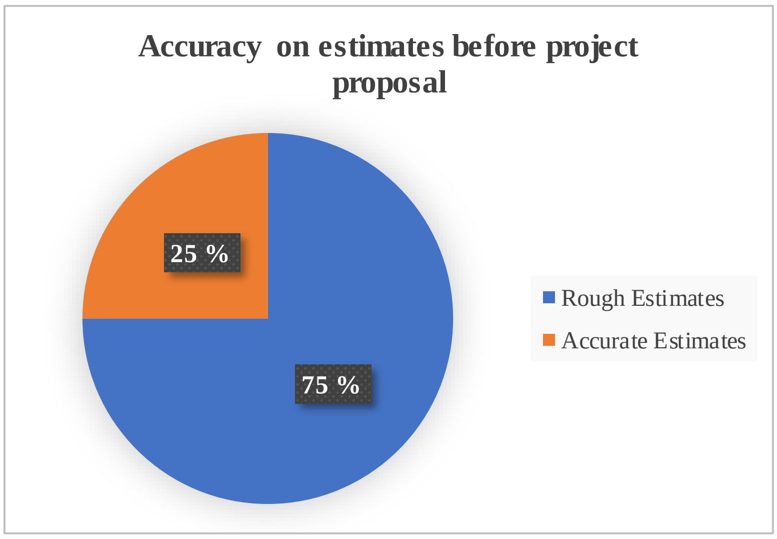


Figure 3.1: Accuracy on estimates before project proposal. *Distribution of the answers rough estimates and accurate estimates*

Figure 3.2 shows the answers to the question: " In your opinion, how accurate should estimates be before Epics are divided into smaller Stories?" The answers from this question show that most of the project managers agreed that only rough estimates are needed before Epics are divided into smaller Stories. Most of the Epics already had an estimate from the pre-project proposal so no further estimation was needed for those items at this point.

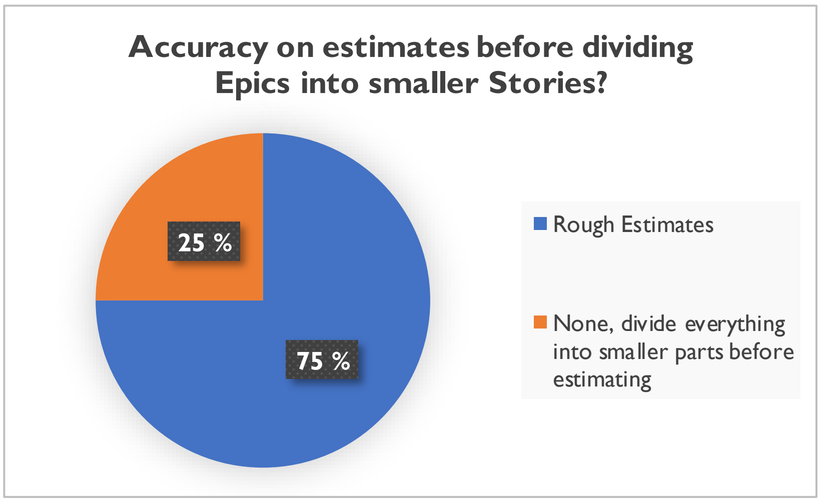


Figure 3.2: Accuracy on estimates before dividing Epics into smaller Stories. *Distribution of the answers rough estimates and none, divide everything into smaller part before estimating*

Figure 3.3 shows the answers to the question: " In your opinion, which method should be used for estimation of Epics?" The answers from this question show that the project managers have different preferences when it comes to which estimation method is the best, but Three-Point Estimation appears the one mostly used if more accurate estimates where needed. The project managers who answered Three-point estimation also said that the estimation was done by a group, and that they also used forth number for complexity in addition to most likely, max and min values.

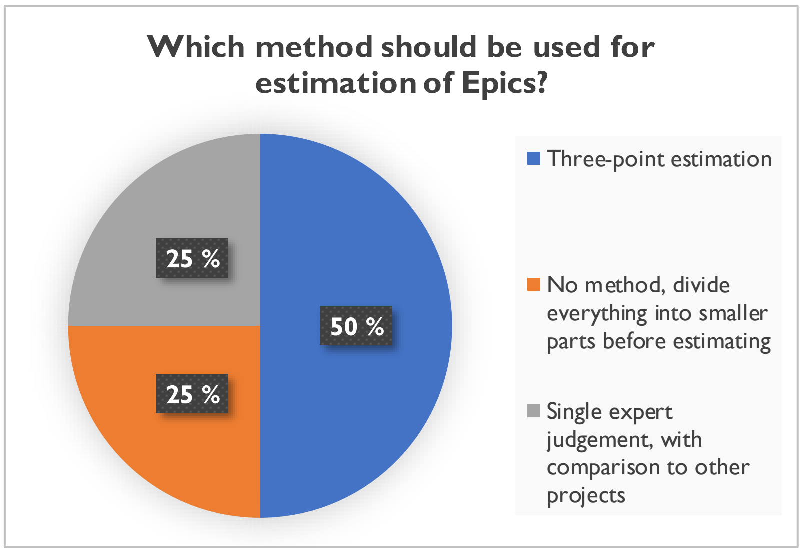


Figure 3.3: Methods used for estimation of Epics. *Distribution of the answers Three-Point Estimation, No method, divide everything into smaller parts before estimating and single experts judgement, with comparison to other projects*

Figure 4.4 shows the answers to the question: "In your opinion, should Stories be estimated before divided into Tasks?" The answers from this question are divided, but everyone agreed that it should at least be a rough estimation. All the project managers also added that Stories always should be divided into Tasks if possible.

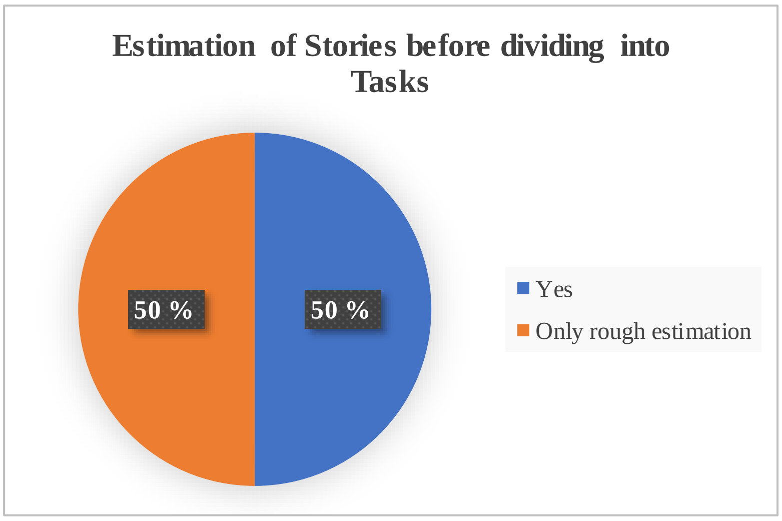


Figure 3.4: Estimation of Stories before dividing into Tasks. *Distribution of the answers yes and only rough estimation*

Figure 3.5 shows the answers to the question: " In your opinion, which method should be used for estimation of Stories?" The answers from this question show that every project manager had their preferred method if they were to estimate User Stories. Though Three-Point Estimation and Poker Planning appeared to be the methods they mostly used if an accurate estimate was needed.

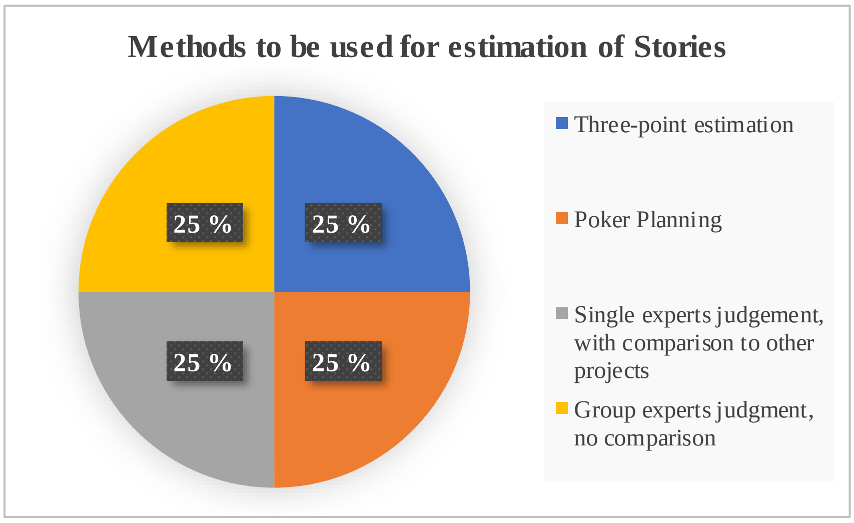


Figure 3.5: Which method should be used for estimation of Stories. *Distribution of the answers Three-Point Estimation, Poker Planning, Single expert judgement, with comparison to other projects and group experts judgement, no comparison*

Figure 3.6 shows the answers to question: " In your opinion, what are the greatest sources of error when estimating Stories?" The answers from this question show that most project managers regard the lack of understanding of the Task/Stories and the needed technology to accomplish the tasks to be one of the greatest sources of error. All the sources listed by the project managers except one appear to be related to the complexity of the Stories.

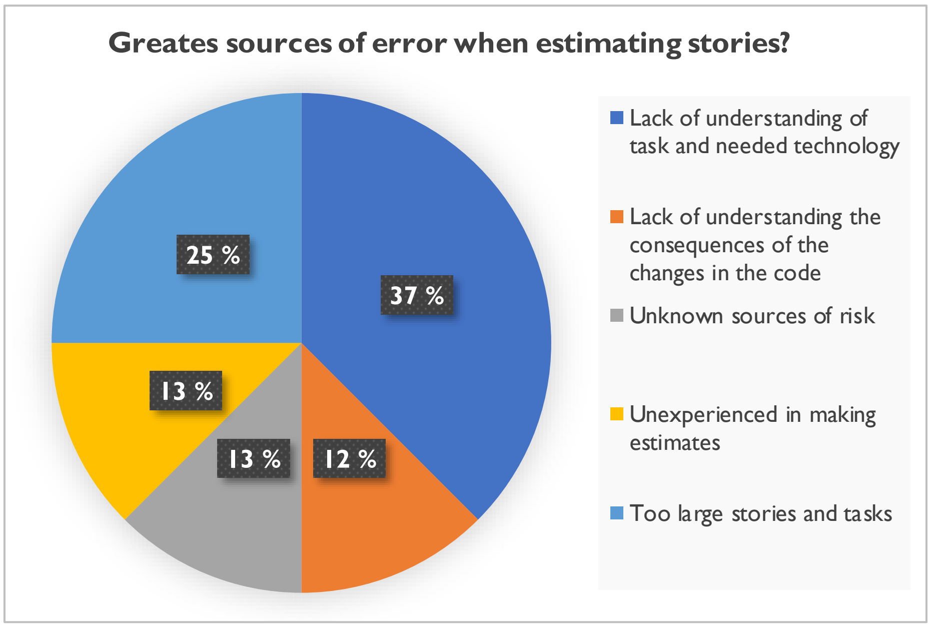


Figure 3.6: Greatest source of error when estimation Stories. *Distribution of the answers lack of understanding of task and needed technology, lack of understanding the consequences of the changes in the code, unknown sources of risk, unexperienced in making estimates and too large Stories and Tasks*

Figure 3.7 shows the answers to the question: " In your opinion, how accurate should Backlog Items be estimated?" The answers from this question show that most of the project managers agreed that Backlog Items only should be roughly estimated before going into the Sprint. If the items should be more accurately estimated during Sprint Planning is not agreed upon, but most of them answered that they should.

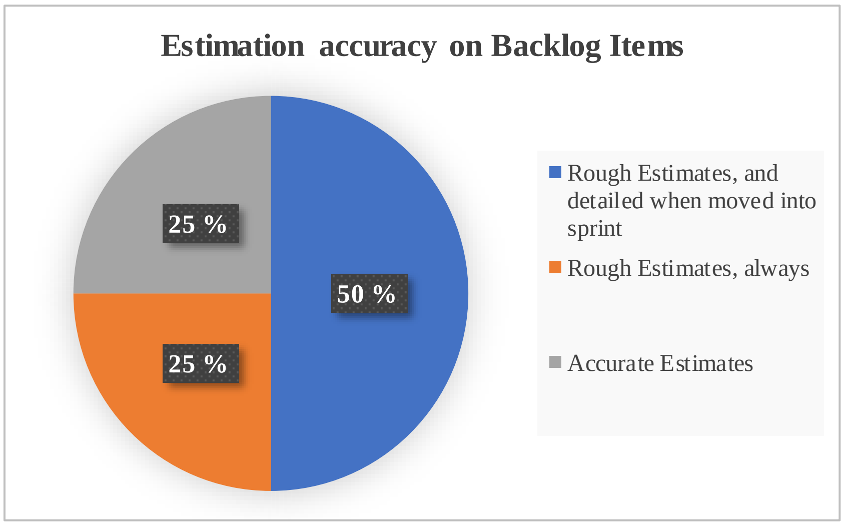


Figure 3.7: Estimation accuracy on Backlog Items. *Distribution of the answers rough estimates, and detailed when moved into Sprint, rough estimates, always and accurate estimates*

Figure 3.8 shows the answers to the question: " In your opinion, should items be re-estimated if new information appears?" The answers from this question show that most of the project managers agreed that to have better control of the project items should be re-estimated if new information appears. The information though must have a significant impact on the items, if they should be re-estimated, or even have a very large impact according to one of the project managers.

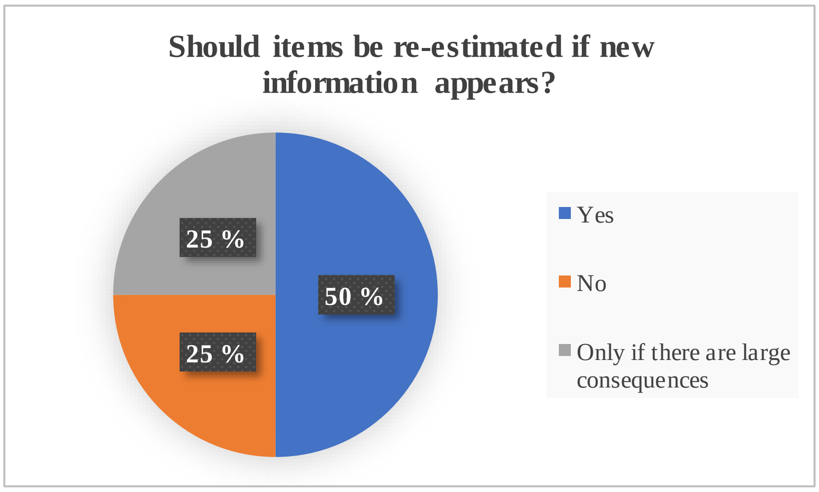


Figure 3.8: Should items be re-estimated if new information appears. *Distribution of the answers yes, no and only if there are large consequences*

Figure 3.9 shows the answers to the question: " In your opinion, what type of project requires the largest amount of time spent on estimation?" The answers to this question show that most of the managers agreed that projects with the risk of losing money for developer company are the projects that require the most time to be spent on making estimates. One project manager also added that more time should be spent on estimates on small projects because they often get underrated because of their size. The relationship with the customer is also important. If there is not much trust between development company and the customer, this forces more time to be spent on estimates, according to one of the project managers.

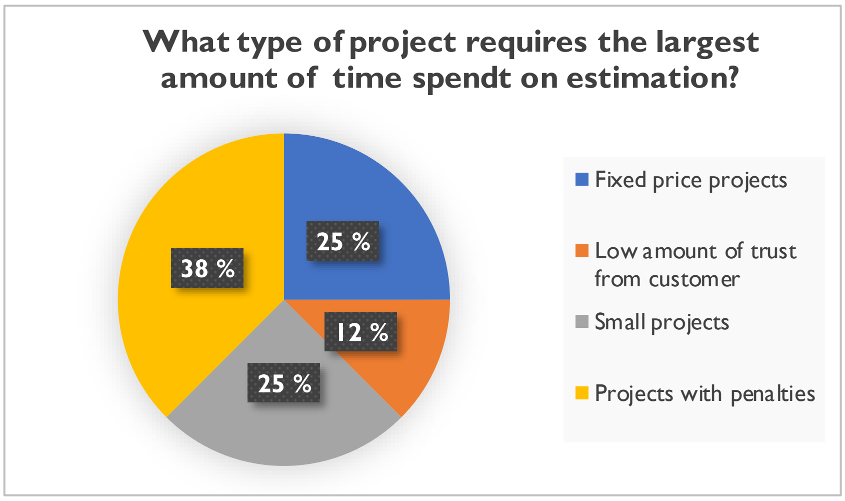


Figure 3.9: Type of project that requires the largest amount of time spent on estimation.
Distribution of the answers fixed price projects, low amount of trust from customer, small projects and projects with penalties

Figure 3.10 shows the answers to the question: " How accurate has the project estimates been in projects you have worked on?" The answers to this question show that most of the project the interviewed project manager has been a part of has been finished on time or on overtime. Even though a few of the projects has been finished earlier than estimated, most of the project managers answered that projects tend to use all time available and rather be finished on time.

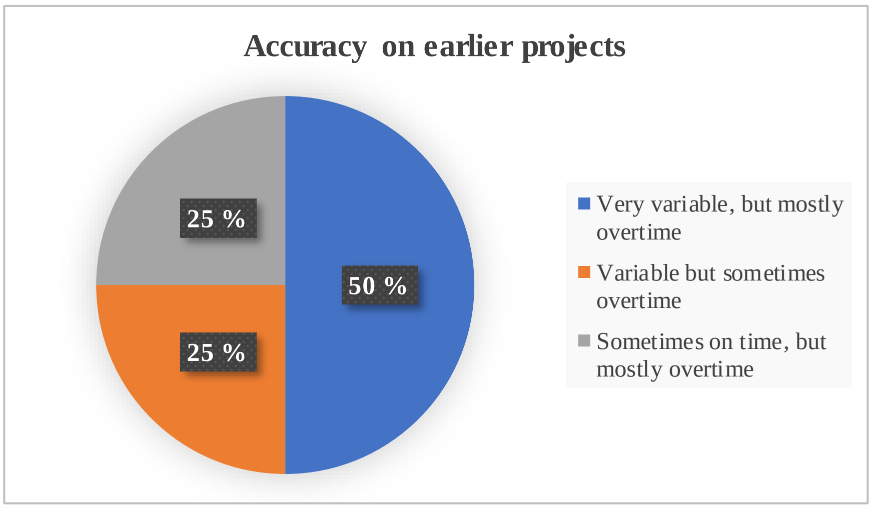


Figure 3.10: Accuracy on earlier project. *Distribution of the answers very variable, but mostly overtime, variable but sometimes overtime and sometimes on time, but mostly overtime*

Figure 3.11 shows the answers to the question: " What have the consequences been on projects that were not completed on delivery time?" The answers to this question show that most of the project managers agreed that overtime or increased time frame is the most common consequence on a project that did not complete on delivery time or appeared to take longer time than budgeted. Whether the customer had to pay more or the developing company earned less or even lost money, happened just as frequent and are very dependent on the contract and the project. Often the consequence are a combination of both customer paying more and the developing companies earning less. All project managers agreed that reduction of Scope is not a good solution since it only has a minor impact, because removing major parts of the Scope is not an option in most projects.

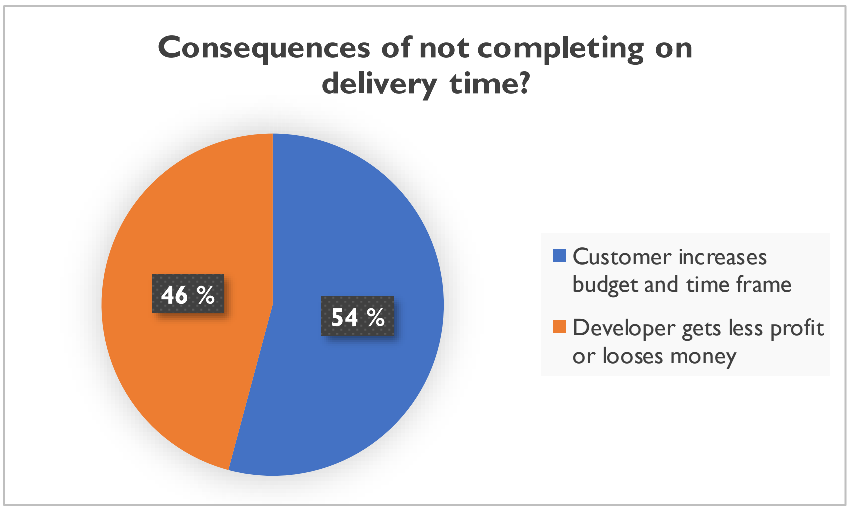


Figure 3.11: Consequences of not completing on delivery time. *Distribution of the answers customer increases budget and time frame and developer gets less profit or looses money*

Figure 3.12 shows the answers to the question: " Do you think the methods explained in #NoEstimates would work?" The answers to this question shows that most of the project managers were positive to #NoEstimates methods by projecting instead of estimating every object. One project manager answered that they had already tried similar methods with success on one project. Another project manager reported that they mostly did not estimate at all on a current project they were working on. Still, all of them agreed that this method will have some limitations and problems.

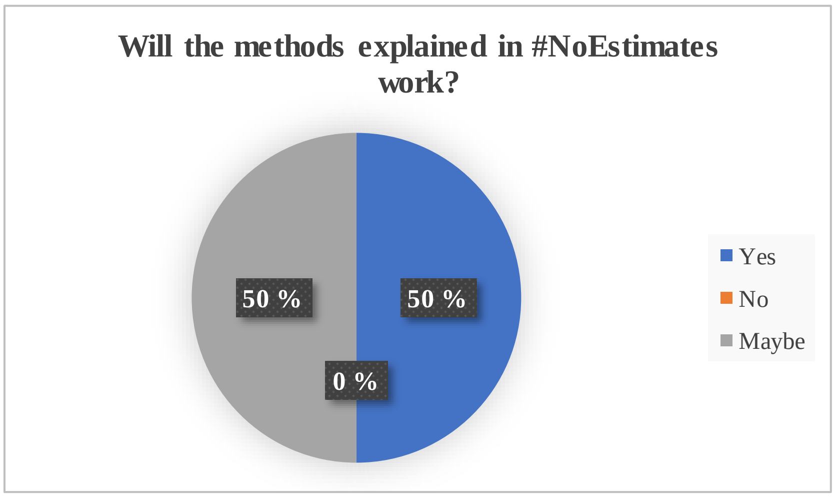


Figure 3.12: Will the methods explained in #NoEstimates work. *Distribution of the answers yes, no and maybe*

Figure 3.13 shows the answers to the question: " What are the biggest problems with #NoEstimates?" The answers to this question show that the project managers believed that the biggest problems with #NoEstimates would be sizing the Stories and changing the customers and organizations view on the estimation.

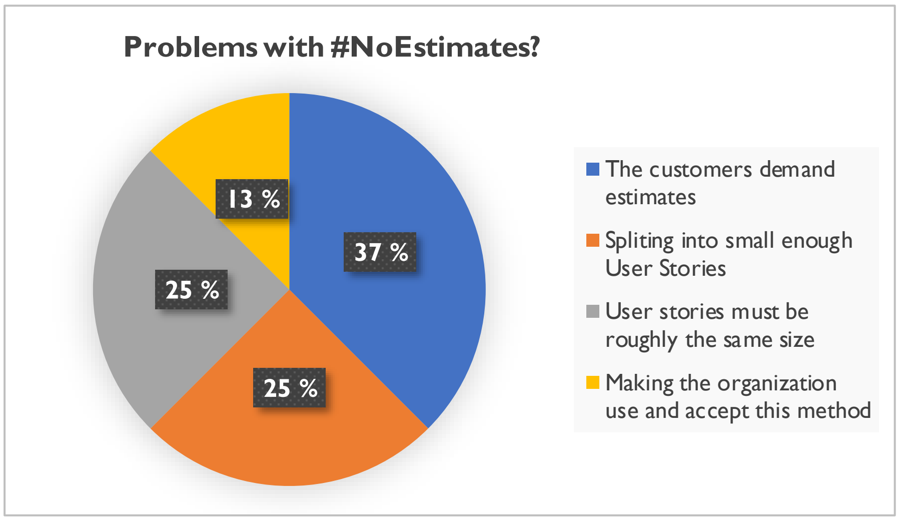


Figure 3.13: Problems with #NoEstimates. *Distribution of the answers the customers demand estimates, splitting into small enough User Stories, User Stories must be roughly the same size and making the organization use and accept this method*

Figure 3.14 shows the answers to the question: " In your opinion, what is the most important reason to make estimates?" The answers to this question show that the project managers agreed that budgeting and controlling projects are the main reasons for making estimates. Followed by planning resources and establishing risk in the project.

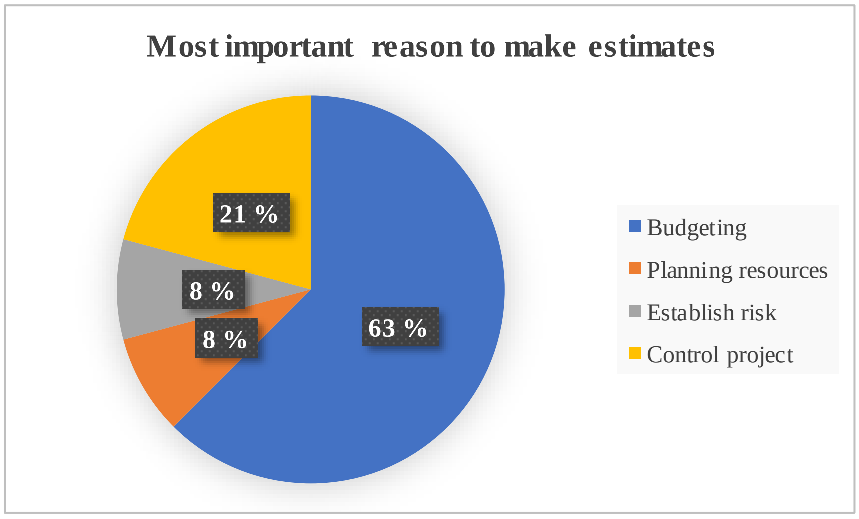


Figure 3.14 Most important reason to make estimates. *Distribution of the answers budgeting, planning resources, establish risk and control project*

Figure 3.15 shows the answers to the question: " In your opinion, what are the greatest sources of error when estimating project cost and delivery time?" The answers to this question show that the project managers disagreed on what the greatest sources for error are when estimating project cost and delivery time. Though, the most common answers were Scope Changes and lack of understanding of complexity and needed technology. Unmanaged or unforeseen risk which was the third most common answer and can also partly be regarded as a consequence of lack of understanding of needed technology and complexity.

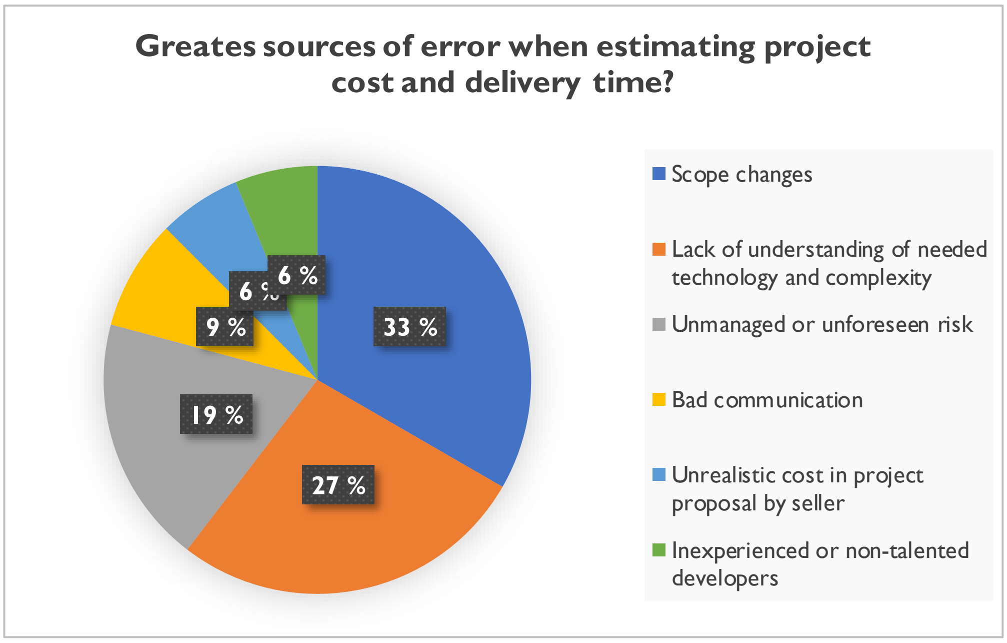


Figure 3.15 Greatest sources of error when estimating project cost and delivery time. *Distribution of the answers scope changes, lack of understanding of needed technology and complexity, unmanaged or unforeseen risk, bad communication, unrealistic cost in project proposal by seller and inexperienced or non-talented developers*

3.3 Other Findings from the Interviews

Besides answering all the questions in the interview there were some important additional information gained during the interviews, that also will be used in the discussion.

- All project managers said that they were using hours or days as metrics for estimates. Some of them had tried using Story Points earlier, but they did not like the concept of estimating by using something abstract and then having to re-calculate to hours if the customer asked when something would be finished.
- One of the companies used UX design for the specification process to make a common understanding between the development company and the customer of the project and every User Story in it.

4 Discussion

Estimation is a great concern when it comes to managing Software Development Projects. Because customers most often want a certainty about what they will get and when they will get it. This can be very hard to deliver to the customer on an accurate level when the projects are agile and change all the time. There is a limit on how accurate estimates can be at different points in the project's life cycle, and how large the uncertainties in the estimates are. This is the case especially in the beginning of a project and can be hard to explain to the customer (McConnell, 2006). Unfortunately, this forces the Development Team to make early estimates without enough knowledge about the project. These estimates will be used to set targets and make commitments for the project and will often be way off. This can in turn lead to a lot of stress for the developers, project overruns and possibly bad market performance or project failure (Lakkaraju & Goswami, 2016).

Accurate estimates in Software Development Projects have a lot of benefits. Among them are: better budgeting, higher quality, early risk information, improved status visibility, better coordination with nonsoftware functions and increased credibility for the Development Team (McConnell, 2006). The project managers that were interviewed regarded budgeting, controlling project, planning resources and establishing risks as the most important reasons for making estimates. With budgeting and controlling the project as the most important ones. These reasons given by the project managers are almost the same as the benefits listed except the increased credibility to the Development Team and higher quality. This shows that many of the benefits are regarded as important, where most of them are also to some degree necessary for the projects when it comes to the successfulness of the project.

How important the benefits are and how much they increase based on time spent on estimating is related to how much the accuracy increases with the time spent on estimation. The increase in benefits must be weighed against the cost and the time lost on doing the actual estimates. It is also important that no time is spent on estimation that does not increase the benefits noticeably. Because any activity that does not directly contribute to putting something of value into the customer's hands is a waste (Holub, 2015).

Some sort of knowledge about the time frame of a project is undoubtedly necessary for a project with a customer or if the software should be used by someone else than the developer. Therefore, some estimation or projection as it is called in the #NoEstimates theory is necessary, but how accurate should they be? McConnell regards a good estimate to be an estimate that provides clear enough view of the project to allow the project leadership to make good decisions about how to control the project to hit its target (McConnell, 2006). The usage of the words “clear enough view” is very different from person to person, but in this context, it does not sound like the estimates need to be very accurate. Only to the point that the estimates makes it possible the project leadership to use them for making good decisions.

It is well known that making good estimates is easier when more about the project and each item is known. Therefore it is very important to make as much decisions about the software as possible to narrow down the uncertainty of the estimates (McConnell, 2006). This is also backed up by the Cone of Uncertainty that shows how the variance in the estimates are reducing as the projects moves towards having the user interface design or the detail design completed. The project managers interviewed also answered that non, or rough estimates were the only estimates that seemed necessary before the items in the Backlog have been specified on a Task level.

A good way of helping the customer to make decisions early is by using UX Design on the start of the project or on a pre-project. This will move the project past the “user interface design completed” milestone and according to the Cone of Uncertainty the best-case accuracy of estimates will then be reduced to [-20%, +25%] of the estimate. Using UX Design for the whole software in the entire project in a pre-project or before the Developer Team starts coding, would arguably remove much of the agility during the project. This could be solved by splitting the project into smaller parts and then designing part by part of the project as the project progresses. The Development Team can estimate and start developing a part of the project when that part is designed. The UX Designer should not design further ahead in time than necessary for the Development Team to have enough to do. This will keep most of the agility in the project. If someone demands estimates for a part of the projector or the entire project, before the needed UX Design is done they should be informed that these estimates are very uncertain.

Instead of using the word estimating, #NoEstimates uses the word projecting. In this method, the number of User Stories are used for projecting when it is likely for parts of the project to be finished. This is done by comparing the amount of User Stories in the project with the rate the Development Team are finishing the User Stories (Duarte, 2014). The biggest difference from a regular estimation process is that in the #NoEstimates method, no estimates are made before the developers starts coding. Therefore, no more time than it takes to count User Stories and keeping score of how many User Stories that have been finished during the Sprints are spent on estimation. Still, the project must be divided down and specified until there is nothing bigger than User Stories in the Backlog for the method to work and project a time frame. Duarte even claims that this works better than actual estimates and has reasonable data from a lot of projects to back him up on that claim (Duarte, 2014).

Though, the #NoEstimates method has some challenges. The project managers interviewed answered that one of the biggest problems of the #NoEstimates methods would be changing the view of the customers and the organization to not always have and use estimates. The other issues the project managers interviewed highlighted with the #NoEstimates method were sizing the User Stories properly into small Stories or to almost equally sized Stories. This shows that the project managers interviewed did not believe that Vasco Duarte #NoEstimates method of estimation works as well as he claims it does.

Duarte highlights that this method will work regardless of the size of the User Stories. One of the project managers interviewed said that "this might work but it sounds a bit extreme". He might have a fair point, why not make very rough estimates instead? This would not take more time than one person reading the actual Stories. The main reason to use the #NoEstimates method would be if Duarte is correct, and that this method is more accurate than other estimates and works for all User Stories almost regardless of size. Another reason would be that there is no actual estimate to hand over to the customer before the first work period/Sprint is finished. In this case, no one would be held accountable for creating a bad estimate before the Development Team has had the time to finish some of the User Stories and created projections from that.

It is very important that estimates are used for what they really are. They are estimates with degree of uncertainty related to a lot of known and unknown variables. Estimates should never be used as promises, they are not commitments and should not be used as targets either. If commitments are needed or targets must be set, they should be based on the estimates because that is better than basing them on nothing. Still the large amount of uncertainty in the estimates must be accounted for if commitments should be made or targets set.

According to some of the project managers that were interviewed this is often done by multiplying with factors like 2 or pi or other numbers, though this is mostly used within projects that have a fixed price or target price. This is a reasonable way of addressing the risk in the projects because the developing company does not want to lose money and take all the risk themselves. Especially since studies shows that developers also tend to make optimistic estimates (Genuchten, 1991; Jørgensen & Grimstad, 2005). This can create some problems though. If the developers know that no matter how they estimate the estimates will be multiplied with the same factor, the developers might start making lower estimates. Therefore, this is a way of reducing risk for the development company but will probably not increase the accuracy of the estimates.

There is also a well-known problem with estimating called Parkinson’s Law which states: "that work expands to fill the time available for its completion". The project managers also confirmed that projects mostly use all time available. Due to of Parkinson’s Law, making higher estimates will not necessarily increase the revenue for a development company. Still over estimating is much better than losing money. If possible, estimates should therefore be overestimated to reduce risk, when commitments are made or targets are set based on the estimates (McConnell, 2006).

All the project managers said that they preferred using hours or days as metrics for estimates, instead of Story Points. They had tried using Story Points earlier, but they did not like the concept of estimating by using something abstract, and then having to re-calculate it to hours to hours if the customer asked when something could be finished. The abstraction of estimating by using Story Points, has most advantages for the Development Team that creates the estimates. Because they can blame the system and show that these estimates really are not accurate.

When the projects use a traditional project manager instead of just using the Scrum Master he will be the middleman and in charge of projecting what can be done in the future. The project manager will have to find out how long time it will take to finish a certain amount of Story Points. For this to work the speed of the Development Team or the throughput of the team must be calculated or already known which is not always the case. It can also be hard for many customers to relate to Story Points, and possibly make them feel like they do not have full control of the project. Still the project managers said that they preferred estimating using the Fibonacci-scale or similar scales. This is probably a combination of it being easier to use a high number for an estimate and that most estimates will be rounded up to nearest number in the sequence.

The project managers answered very differently when they were asked about which methods they preferred to use when estimating at different stages of the project. Though, Three-Point Estimation appeared to be the one most of them would use if more accurate measurements were needed. It is important to notice that these answers were given by most of them under the assumption that they needed to select a known method. All except one, answered that only a rough estimate done by one person would be good enough until that part was divided in smallest items if more accurate estimates were not required.

There are a lot of great sources of error within Software Development. As mentioned earlier the greatest of them are the uncertainty in the project itself and all the possible scope changes. The project managers also regarded lack of understanding complexity and needed technology, unmanaged or unforeseen risks, bad communication, unrealistic cost in project proposal by seller and inexperienced or non-talented developers as great sources of risk. Almost all of these sources of error can be reduced if a detailed and thorough specification process is done. Since Agile Projects are supposed to change over time, there are always going to be some errors in the estimates. Especially estimates for entire projects or large parts of the project. This will be explained more directly in the conclusion.

5 Conclusion

It is necessary to have estimations within Software Development Projects, because the customer needs to have some knowledge about the time frame and how much that will be finished by then. The customer should be informed about the uncertainty of the estimates, and that the uncertainty will be reduced when more decisions are made.

There are a lot of benefits by having accurate estimates and often it is also necessary to have them. Still, it is important that only the needed amount of time is spent on making the estimates. The estimates just need to be good enough. Specifying the project by splitting up Stories and making decisions about the software is the best way of narrowing down the uncertainty of the estimates, as well as making it possible to create more accurate estimates. Therefore, only rough estimates should be made before the project is more specified and divided up. The specification part should be done by a UX Designer who will collaborate with the customer to make software that enhances the experience the user has while using the software, and make sure the users find value in the product.

It is important that only the parts of the project that is soon going into development are heavily specified. Otherwise agility will be lost in the process. If time should be used to divide the User Stories into Tasks before they are estimated will depend on the customers demand for accuracy and total control over the project. #NoEstimates will probably work if it is implemented in an organization, but the project should be specified into small User Stories for the projection to really work. For less specified parts of the project rough estimates should be enough to make the right decisions. The people working on the project should also make the estimates since they can make the most accurate ones.

6 References

Agile Alliance. (2017). "Backlog Grooming" Agile Alliance, Retrieved June 20, 2017 from https://www.agilealliance.org/glossary/backlog-grooming

Beckerleg, G. (2014). "#NoEstimates - Stop lying to yourself and your customers and stop estimating" Gerard Beckerleg, Retrieved May 12, 2017 from https://www.slideshare.net/gerardbeckerleg/noestimates-stop-lying-to-yourself-and-your-customers-and-stop-estimating

Chon, M. (2016). "What are story points" Mountain Goat Software, Retrieved May 22, 2017 from https://www.mountaingoatsoftware.com/blog/what-are-story-points

Cockburn, A. (2002). *Agile software development*: Addison-Wesley Longman Publishing Co., Inc.

Duarte, V. (2014). "No estimates - a controversial way to improve estimation with result-handouts" Vasco Duarte, Retrieved May 5, 2017 from https://www.slideshare.net/duartevasco/no-estimates-a-controversial-way-to-improve-estimation-with-resultshandouts

Genuchten, M. v. (1991). Why is Software Late? An Empirical Study of Reasons for Delay in Software Development. *IEEE Trans. Softw. Eng., 17*(6). doi:10.1109/32.87283

Hartman, B. (2009). "Introduction to Planning Poker" Agile Zone, Retrieved May 20, 2017 from https://dzone.com/articles/introduction-planning-poker

Holub, A. (2015). "#NoEstimates" Allen Holub, Retrieved May 12, 2017 from https://holub.com/slides/NoEstimates.key.pdf

International Scrum Institute. (2016). *Scrum Revealed* Vol. 1.

Jones, C. (1994). *Assessment and control of software risks*: Yourdon Press.

Jørgensen, M. (2005). Practical guidelines for expert-judgment-based software effort estimation. *IEEE Software, 22*(3). doi:10.1109/MS.2005.73

Jørgensen, M., & Grimstad, S. (2005). Over-Optimism in Software Development Projects: "The Winner's Curse". *IEEE Computer Society.*

Lakkaraju, R., & Goswami, S. (2016). "#NoEstimates, An analytical perspective" Scrum Alliance, Retrieved July 2, 2017 from https://www.scrumalliance.org/community/articles/2016/june/noestimates

Lanoue, S. (2015). "What is UX Design? 15 User Experience Experts Weigh In" User Testing, Retrieved June 28, 2017 from https://www.usertesting.com/blog/2015/09/16/what-is-ux-design-15-user-experience-experts-weigh-in/

Lowe, D. (2014). "Theme, Epic, Story, Task" Scrum & Kanban, Retrieved May 20, 2017 from http://scrumandkanban.co.uk/theme-epic-story-task/

Maksimovic, Z. (2012). "Easy task estimation with Three-point estimation" Retrieved June 12, 2017 from http://www.agile-code.com/blog/easy-task-estimation-with-three-point-estimation-technique/

McConnell, S. (2006). *Software Estimation: Demystifying the Black Art*: Microsoft Press.

Ming, L. M. (2014). "UI UX who does what? A designers guide to the tech industry" CO.DESIGN, Retrieved June 28, 2017 from https://www.fastcodesign.com/3032719/ui-ux-who-does-what-a-designers-guide-to-the-tech-industry

Radigan, D. (2017). "Epics, stories, versions, and sprints" Atlassian, Retrieved May 20, 2017 from https://www.atlassian.com/agile/delivery-vehicles

Schwaber, K., & Sutherland, J. (2016). *The Definitive Guide to Scrum: The Rules of the game*

Scrum Institute. (2017). "The Scrum Product Backlog" scrum-institute.org, Retrieved May 20, 2017 from http://www.scrum-institute.org/The_Scrum_Product_Backlog.php

scrum.org. (2017). "What is Scrum" Scrum.org, Retrieved April 12, 2017 from https://www.scrum.org/resources/what-is-scrum

Thakur, D. (1999). "What is Software Requirements Specification? Explain Structure and Characteristics of SRS." Computer Notes, Retrieved June 28, 2017 from http://ecomputernotes.com/software-engineering/softwarerequirementsspecification

Thomas, S. (2008). "Agile Project Scope" It's a delivery Thing, Retrieved May 22, 2017 from http://itsadeliverything.com/agile-project-scope

Yatzeck, E. (2012). "How to Contol Scope Creep in Agile" Thought Works, Retrieved June 29, 2017 from https://www.thoughtworks.com/insights/blog/how-control-scope-creep-agile

7 Appendices

Appendix 1

Interview Guide

The main goal of the interviews is to find out how accurate estimation the project managers regards as necessary at the different stages of a project, as well as which methods they recommend using at the different stages of the project. The other goals of the interviews are to gain information about:

1. Why they consider it important to make estimates.
2. What they regard as the greatest sources of error.
3. How accurate their estimates normally are and what the consequences of bad estimates have been in their projects.
4. If they believe it would be possible to change to the #NoEstimates method, and what they assume to be the biggest problems with #NoEstimates.

All interviews will be anonymous and held at the project managers own company locations. The project managers will be informed about #NoEstimates and the results presented by Vasco Duarte (Duarte, 2014). The project managers will be encouraged to answer all questions according to their opinion and not how the think they are supposed make estimates or how it is done in their organization.

The interview questions:

1. What is your name?
2. How many years have you worked within Software Development?
3. What is your background from before you became a project manager?
4. In your opinion, how accurate should estimates be before a project proposal?
5. In your opinion, how accurate should estimates be before Epics are divided into smaller Stories?
6. In your opinion, which method should be used for estimation of Epics?
7. In your opinion, should Stories be estimated before divided into Tasks?

8. In your opinion, which method should be used for estimation of Stories?
9. In your opinion, what are the greatest sources of error when estimating Stories?
10. In your opinion, how accurate should Backlog Items be estimated?
11. In your opinion, should objects be re-estimated if new information appears?
12. In your opinion, what type of project requires the largest amount of time spent on estimation?
13. How accurate have the project estimates been in projects you have worked on?
14. What have the consequences been on projects that were not completed on delivery time?
15. Do you think the methods explained in #NoEstimates would work?
16. What are the biggest problems with #NoEstimates?
17. In your opinion, what is the most important reason to make estimates?
18. In your opinion, what are the greatest sources of error when estimating project cost and delivery time?

Appendix 2

The Calculation Method of Interview Results

The results from each question are calculated as percentage. The answers for each question are created by the project managers, and the value given to each answer are calculated by:

- How the answer is prioritized by the person interviewed if he has more than one answer.
- How many of the interviewed people did give that answer.

For example:

In your opinion, what are the greatest sources of error when estimating project cost and delivery time?

Person A answers: Scope Changes and Bad Communication

Person B answers: Lack of understanding of needed technology and complexity, Unmanaged or unforeseen risk, Inexperienced or non-talented developers and Unrealistic cost in project proposal by seller.

Person C answers: Scope changes and a small part might be Bad Communication

These answers will result in the following calculations:

	A	A %	B	B %	C	C %	Total:	Percentage:
Scope changes	1	0,5		0	4	0,8	1,3	43,33 %
Lack of understandment		0	1	0,3		0	0,25	8,33 %
Unmanaged or unforeseen risk		0	1	0,3		0	0,25	8,33 %
Bad communication	1	0,5		0	1	0,2	0,7	23,33 %
Un realistic cost in project proposal		0	1	0,3		0	0,25	8,33 %
Inexperienced or non-talented developers		0	1	0,3		0	0,25	8,33 %
Sum	2	1	4	1	5	1	3	1