Estimation Techniques in Agile Software Development

By

Sandeep RC

M.Sc, Østfold University College, 2020

Thesis Submitted in Particular Fulfillment of the Requirement for the
Degree of Master in Applied Computer Science

Supervised by: Professor Dr. Ricardo Colomo-Palacios

Co-supervised by: Associate Professor Dr. Mary Sánchez-Gordón

June 2020

Degree of Master in Applied Computer Science

Table of Contents

List of Tables

List of Figures

ABBREVIATIONS AND TERMS

AM	Agile Manifesto
ASD	Agile Software Development
ASDM	Agile Software Development Model
ASE	Agile Software Estimation
COCOMO	Constructive Cost Model
DSDM	Dynamic systems development method
FDD	Feature-driven development
GSD	Global Software Development
HELENA	Hybrid dEveLopmENt Approaches
IDE	Ideal days estimation
LSD	Lean Software Development
ML	Machine Learning
NN	Neural Networks
OORI	Over Optimism Related Issue
PP	Planning Poker
PMRI	Project Management Related Issue
RRI	Requirement Related Issue
SD	Software Development
SLR	Systematic Literature Review
SS	Story Size
TRI	Team Related Issue

TDD	Test Driven Development
TEG	Team Estimation Game
WM	Waterfall Model
XP	Extreme Programming

Abstract

Software development is one of the most important factors to govern in a world in which software is increasing in significance. However, and despite its importance, software development is still a challenge due to the need to meet the market demands in a faster way. One of the approaches to tackle the problem is Agile methods, however, they also pose several challenges in which inaccuracy is one of them.

In this scenario, this master thesis is devoted to investigate agile effort estimation techniques in practice including their benefits and challenges. To do so, an online survey was designed based on a literature review, then a mixed qualitative and quantitative approach was used to analyze the results.

From the online survey a total of 62 responses were collected, among them, 53 were valid responses to the research question, since they came from practitioners that are involved in the software estimation process. After the data collection process was completed data filtration was done, followed by, a descriptive statistical analysis, and an inferential approach.

It is expected that this study will provide insights into the agile estimation techniques, and, in addition, improve the understanding of #NoEstimates and #NoProject movement.

The result shows that not only mixed software development approaches are used but also different effort estimation techniques, and those techniques mainly used the Fibonacci series as a measurement unit. Regarding the perceived benefits, all the respondents agree with the listed category. The most important one is to Drive the team to complete the project successfully. Similarly, the major perceived reason for the inaccuracy is one of the requirements related issues called - Complexity and Uncertainty followed by Missing and changing requirements. However, further research should be conducted to gain a broader and more comprehensive understanding of this area.

Keywords: Agile Software Development (ASD), Estimation, #Noestimates, #Noproject, Software Development

1. Introduction

With the new industrial revolution, the software industry has been introducing new methodologies and different approaches to software development over the last half-century. As software has become a crucial part of society, it is creating an impact on business and everyday life. For instance, personal and work life, business and economic, civil and industrial politics, education, and entertainment are few occupied and directed areas by software applications (Fuggetta & Di Nitto, 2014). Thus, the software industry is playing a significant role in fulfilling the increasing demand and extensive use of software (S. Hastie & Wojewoda, 2017;Stankovic, Nikolic, Djordjevic, & Cao, 2013). In spite of that fact, software projects still fail to get success associated with cost, quality, time, and are unable to get the expected returns from their investment (Kulathunga & Ratiyala, 2018). Therefore, software development has grown as a risky activity that needs careful examination, understanding, support, and improvement (Casado-Lumbreras et al., 2009; Fuggetta & Di Nitto, 2014).

In the ’90s, the software industry had an excessive due to "over planning, inadequate conversation, and all-at-once delivery" plus, ill-famed projects for their missed deadline, over budget, incomplete deliverable, and dissatisfied customers (Cooke, 2012). Different methods and policies have emerged to overcome these issues, but the best-known approach is based on the Agile Manifesto (Cooke, 2012). According to (Alliance, 2016) ”*Agile software development is an umbrella term for a set of frameworks and practices based on the values and principles expressed in the Manifesto for Agile Software Development and the 12 Principles behind it*.” Basically, it emphasizes the value of customer, iterative, incremental delivery process, powerful collaboration, small combined teams, self-organized, as well as minor and ongoing progress (Abrahamsson et al., 2017). These days, the flexible nature of agile has become an integral part of different types of organizations.

The survey carried out by ISTQB, (2016) revealed that 69.6% of organizations use agile methodologies (Scrum, Kanban, Extreme Programming) during software development. The results were based on 3200 respondents from 89 countries. In 2017, the VersionOne survey revealed that 94% of the respondents practice agile for software development (*VersionOne*, 2017).

In 2018, the ISTQB survey (ISTQB, 2018) revealed that 79% of the organization uses agile methodologies. This time, 2000 respondents from 92 countries were surveyed. On the other hand, a large-scale international survey was carried out by the HELENA initiative in 2018, to investigate the use of mixed software development approaches. In this case, 75 researchers were involved in a 2-year endeavor from 55 countries. The respondents reported that 76.8% of them use hybrid (mixed) development methods (Kuhrmann et al., 2018). Thus, it shows that the adoption of agile methodologies in the world is growing and, it is promising that it will reduce software development issues such as low productivity, team motivation, excessive cost, and schedule delays (Inayat et al., 2015).

Among the most popular agile methodologies used for software development are Scrum, Extreme Programming, Feature-driven development, Test-driven developments, and Lean software development (Dora & Dubey, 2015). In this context, estimation helps to ensure the success of a software project, as it provides the information required to develop a software development plan, to fulfill the specification and meets the commitments (Ceschi et al., 2005). For instance, a review of surveys of software effort estimation conducted by McKinsey and Company in collaboration with the University of Oxford evaluated 5400 IT projects in 2012. Such a survey showed that 45% of large IT projects exceed the estimated budget and 7% took extra time than planned, whereas 56% had a lesser value than predicted while delivering (Bloch et al., 2012). In 2013, another study (Flyvbjerg & Budzier, 2013) revealed, that one in six of the projects had a cost overrun of 200%, on average, and a schedule overrun of nearly 70%. Those findings are based on 1,471 software projects. The same year, according to (Eberendu et al., 2018), one study on Innotas carried out by Planview shows that 50% of the software projects were unable to deliver what was required within the previous 12 months. Besides this, in 2015, the study conducted by Hastie & Wojewoda, (2017) revealed a higher failure rate, 94% of larger projects, 91% of medium projects, and more than 50% of the projects overrun cost and time. From these studies (Bloch et al., 2012; Flyvbjerg & Budzier, 2013; Dora & Dubey, 2015; Eberendu et al., 2018), it can be concluded that time and cost determine the overall performance of the software project and its success-failure rate.

Estimation is an essential part of software development, however, no specific approach is available to estimate software that can provide the highest degree of accuracy (Rashmi Popli & Chauhan, 2013). According to Beck et al., (2001), the major principle of agile software development method

is "welcoming the changing requirements" but, a requirement change can cause an estimation problem in any software development. Likewise, Molokken & Jorgensen, (2003) identified that most of the project overrun (60-80%) so that they face a significant challenge in terms of effort or schedule. In fact, (Popli & Chauhan, 2014) identified that the cause of inaccurate estimates in agile software development (ASD) are: 1) software development process has various unstable factors (software requirement, distributed teams, and other hardware) associated together can affect the estimation 2) Software development environment is evolving and changing so it can cause a problem in estimation 3) Plus, unavailability of tools for measuring the complexity of software systems. More recently, Anooja & Rajawat, (2018) have identified four reasons why estimation does not work 1) Limited information of the problem 2) Communication gap among client and development team 3) Pressure of the deadline to team 4) Flexibility among team member.

In this context, the value of estimation techniques has been highly criticized. Since 2012, a #NoEstimate movement has been growing and is revolving around twitter feed and among various blog posts. The main principle of the movement is that estimates do not directly add value to the software process, so practitioners should find ways to reduce the estimation process or even stop it if possible. It was promoted by developers, including Woody Zuill and Neil Killick, who have raised the questions of efficiency (*What Is #noestimates*, 2017). Thus, #NoEstimates camp began, to fill up the gap estimate has, primarily, estimation took extra time to discuss the task that will generate no business value, secondly, unclear project requirement that needs to be estimated, finally, the frequent requirement change, adds extra pressure to the development team to complete the committed task (Duarte, 2015).

Although #NoEstimates seems to be trending, it has some limitations as well. For instance, #NoEstimates requires an extra budget to forecast the data. Thus, when stories of the project are broken down into smaller and manageable chunks to predict their delivery the project stories need calculated data. In line with this, the #NoProjects (Leybourn & Hastie, 2018) movements also focus on the delivery value so there is some similarity with #NoEstimates (Duarte, 2015).

1.1. Objective

The objective of the study is to identify the benefits and challenges of effort estimation techniques in the agile practice and the impact of #NoEstimates and #NoProjects movements. To do so, this paper conducts a:

- Literature review of the prior works related to effort estimation in agile.
- Mixed quantitative and qualitative research to identify the benefits and challenges of effort estimation techniques and the impact of #NoEstimates and #NoProjects movements
- Data will be collected using a survey questionnaire.

1.2. Structure of the thesis

The thesis is organized into five different sections. First chapter presents the introduction of the topic, problem statement, and objectives of the study. The literature review chapter presents the background of the software development model followed by, estimation scale, and agile estimation techniques, including #NoEstimates and #NoProjects movements. Finally, related works are presented. Third chapter provides an overview of the research methodology used in this study. While the fourth chapter presents the results of the survey. Fifth, contains the discussion of the results and finally, the conclusion is included in the sixth chapter.

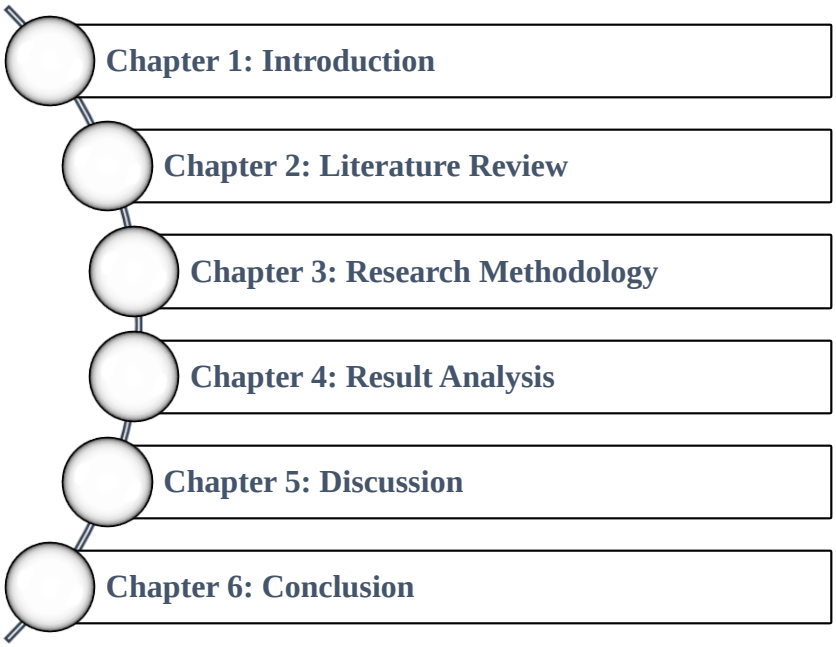


Figure 1: Structure of the thesis

2. Literature Review

2.1. Software Development Models

According to the HELENA study (Kuhrmann et al., 2018), not only agile methodologies (Beck et al., 2001) but also the waterfall model (Royce, 1987) are important software development approaches. In fact, mixed software development approaches are a reality as they are widely used. Therefore, although the focus of this study is an agile estimation, ASD and waterfall are explained briefly below.

2.1.1. Waterfall Development Model

Waterfall is a well-known software development model in which, the development process is carried out in a sequential pattern (Royce, 1987). In this model, once the requirements are collected from the client they are analyzed and frozen. Each phase should be completed in a specific time constraint after that, the team should start developing the next phase. However, in practice, the development overlaps, and different phases feed each other (Sommerville, 2007).

According to (Royce, 1987), after the testing phase the requirement might need to be changed for fixing the problem that is existing in the system, because of this, the development process starts all over again. It means that there might be a 100% schedule or cost overrun in the project.

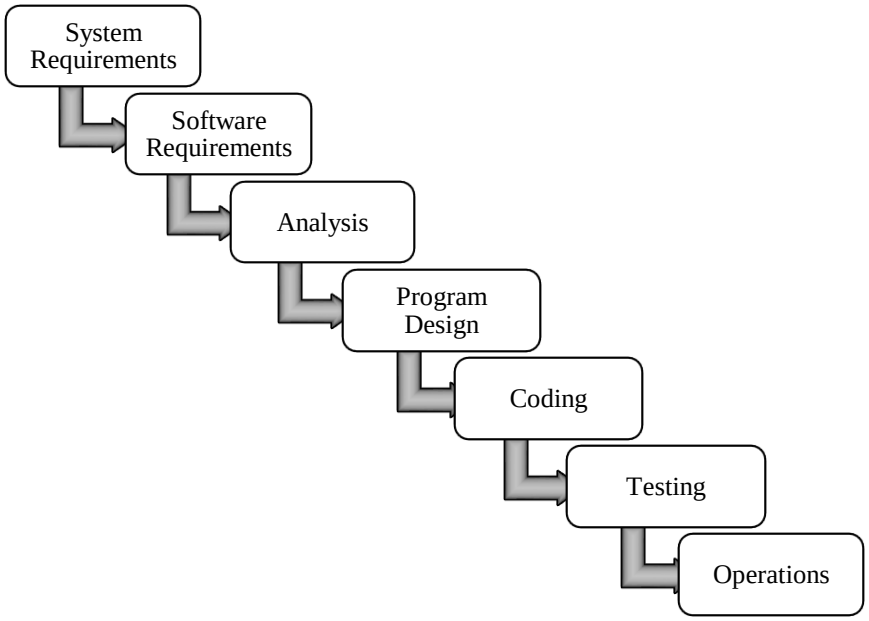


Figure 2: The Waterfall model (Royce, 1987)

Estimation in traditional software development is done by the managers. They first calculate the capacity of the entire team and then determine the time required to complete the specific tasks, based on that they assign the work to the team members (Ziauddin & Zia, 2012).

2.1.2. Agile Software Development Model

During 2001 the Agile Alliance was formed, and the Agile Manifesto was published. In this way, agility was formally introduced in the software development panorama (Sommerville, 2007). The four important value of Agile Manifesto are (Beck et al., 2001),

"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 software development is an iterative development model that is evolutionary and incremental. It is also identified as a "modern" software development model since it substitutes popular traditional approaches such as the waterfall model (Larman & Basili, 2003). As agile is an incremental development process, it has small iterations and after each iteration, feedback from the customer is taken and served as the input for next iteration (Vyas et al., 2018).

According to (Javdani Gandomani et al., 2015) popular approaches of agile are Scrum, Extreme Programming (XP), Crystal methods, Feature-driven development (FDD), Dynamic systems development method (DSDM), Test Driven Development (TDD), and lean software development. Although, they pose different practices and activities according to the goals, generally, they have similar values. Some approaches focus on project management (Scrum, DSDM) while others focus on software development (XP, Crystal).

Every agile approach uses estimation during the software development process at some stage. However, estimation is particularly done by the group rather than a single individual. Besides, it is done by the experts who are going to work on the software project. But in the case of agile, it is unknown in advance who is going to work in a particular task (Cohn, 2005).

In traditional software development, estimation is done by a manager. Whereas, in agile software development, estimating team member's capacity has a different approach. Firstly, the work is assigned to the entire team rather than an individual. Theoretically, the emphasis is given on group effort. Secondly, the work will not measure in terms of time, since this would weaken the self-organizing principle to the accomplishment of the methodology. Thirdly, in the case of scrum team members, themselves use effort and degree of difficulty to determine their work instead of manager determining it as happen in traditional methods. More importantly, the team shares their opinion of the scale that is being used, so that everyone in the team is comfortable with the scale's values (Ziauddin & Zia, 2012).

2.2. Estimation Scale

To measure the story points some sort of scales are used, these scales are relative in nature and the values can be numbers or any objects that are easy to compare (Ziauddin & Zia, 2012). However, there are two types of estimation scales: Relative estimation and Absolute estimation.

Relative estimation is the estimation process in which the related items are estimated based on 3 parameters of a user story - COMPLEXITY, UNCERTAINTY, and EFFORT.

According to the (Alliance, 2016), "*Relative estimation is one of the several distinct flavors of estimation used in Agile teams, and consists of estimating tasks or user stories, not separately and in absolute units of time, but by comparison or by grouping items of equivalent difficulty*".

For instance, if someone says, "I think this feature (B) is twice as complex as this other feature (A)". But that someone has not mentioned about the specific time required to complete the feature (A). So, if feature A took 3 weeks then it is obvious that feature B will take 6 weeks.

Some popular relative estimation scales are:

Fibonacci sequence: Fibonacci series is a sequence of numbers where each number is the sum of the previous two numbers. The numbers are 1, 2, 3, 5, 8, 13, 21, etc (Raslan et al., 2015).

T-shirt sizes: Story points are estimated in the form of size if the task is small then S, for heavy task XL, and so on (Anooja & Rajawat, 2018). The sizes are (XS, S, M, L, XL, XXL, XXXL).

Absolute estimate refers to the estimation method in which user stories are measured based on the time like, estimation based on ideal hours (Alliance, 2016).

Idea day: the estimation scale used to estimate user stories based on the ideal time. It includes only the workable time.

2.3. Agile Software Estimation Approaches

This study is focused on identifying the effort estimation techniques used including the benefits and challenges of estimation in ASD. Therefore, the following section provides an overview of some agile estimation approaches.

2.3.1. Agile Estimation

McConnel in his book" Software Estimation" define estimation as (McConnell, 2006);

"*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*"

Software estimation is an integral part of the software development process, it helps to set up the budget of any software projects, while it, fulfill the demand of the software projects, and monitors project progress (Chemuturi, 2009). Estimation particularly takes place during the initial stage of planning and serves in allocating resources that are needed, set the deadline of the project, and validate the delivery of the commitment (Chemuturi, 2009). In other words, the estimation process starts from the very beginning of the software development process, and lots of decisions are made throughout the development process which might lead to uncertainty (Dagnino, 2013). Also, unclear requirement form the client and continuing transformation of the project leads to inaccuracy in estimation (Dagnino, 2013).

Estimation in traditional software development approaches is based on initial development plans (Cao, 2008). However, estimation in agile development is based on the release plan and an iteration plan. A release plan is a list of the features that will be developed for the next release, here, developers select the highest priority features based on their estimates for the release (Cao, 2008).

As estimation is an integral part of software development, Cohn, (2005) has mentioned that even though it is important, an accurate estimate is always a big challenge. Thus, the inaccuracy arises due to uncertainty estimation (Dagnino, 2013).

In particular, McConnell, (2006), remarks that uncertainty arises due to improper decision making. Improper decisions are made due to unstable requirements and continuous refinement of features. So, to reduce the uncertainty in software projects proper decisions need to be made.

Figure 3 depicts the cone of uncertainty (McConnell, 2006). It shows the various levels of uncertainty during the development phases and how estimation becomes more accurate at each phase. The estimates made at the early stage of the software development life cycle have higher chances of error occurrence (McConnell, 2006). Thus, the estimation made at the initial concept phase has the inaccuracy of 4x divided by 0.25x or a total of 16x, this inaccuracy arises because of uncertainty in software development itself. This is the reason why reducing the uncertainty in the estimates will automatically reduce the uncertainty in software projects (McConnell, 2006).

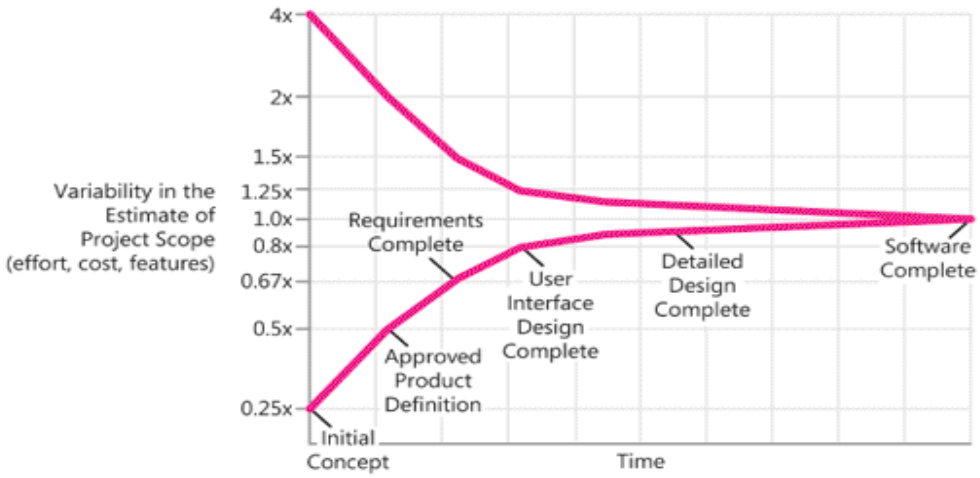


Figure 3: The cone of uncertainty based on project Milestone (McConnell, 2006)

2.3.2. Estimation Approaches

Estimation focuses on three aspects namely effort, schedule, and cost (Dagnino, 2013). This thesis is focused on effort estimation.

Effort estimation is a prediction done to find out the total time required for completing the project (Popli & Chauhan, 2014). Moreover, effort estimation is one of the essential factors of the software development process as it drives the team to complete the project successfully (Schweighofer et al., 2016).

An accurate estimate is made in the presence of historical data, the absence will lead to inaccuracy (R. Popli & Chauhan, 2014). Besides, these data provide the estimation outline associated with

cost and the number of resources required to complete the project. Alongside this, it achieves the functional and non-functional requirements of the project and fulfills the customer's demand by delivering the project successfully (Trendowicz & Jeffery, 2014).

Furthermore, Trendowicz & Jeffery summarized several reasons for practicing effort estimation. Firstly, it manages and reduces the risks that might arise in a project, secondly, evaluate the system's progress, thirdly, discover the general plans and efficiency measurement, fourthly, helps in identifying the resources and project scope, and finally, manages the product changes by reducing the cost (Trendowicz & Jeffery, 2014).

Similarly, a systematic literature review carried out by Britto, Usman, & Mendes, (2014) in the context of global software development and ASD identified the effort estimation techniques reported in that context (see *Table 1*). Moreover, this study identified that the most popular estimation method was expert judgment, use case points (UCP), planning poker, and Delphi.

Table 1: Popular effort estimation method (Britto et al., 2014)

#	**Estimation method**
1	Case-based reasoning
2	Planning poker
3	Function point count
4	Use case point count
5	Use case point test effort estimation model
6	Expert judgment
7	Delphi
8	No estimation approach

Later, a survey study conducted by (Usman, Mendes, & Börstler, 2015) collected data from 16 different countries and 60 agile practitioners that were involved in the state of practice of effort estimation in ASD. The findings revealed that planning poker (63%) was the most used effort estimation technique followed by, analogy (47%) and expert judgment (38%). *Table 2* presents the popular estimation methods. Besides this, it was found that story points were the most frequently (62%) reported a size metric.

Table 2: Popular estimation method (Usman, Mendes, & Börstler, 2015)

#	Estimation method
1	Planning Poker
2	Analogy
3	Expert judgment

Usman, Börstler, & Petersen, (2017) carried out an effort estimation taxonomy in which they compared two previous studies: a systematic literature review (Britto et al., 2014) that identified the state of art on effort estimation in ASD, and a survey that was a follow-up study conducted by (Usman, Mendes, & Börstler, 2015) Table 3 shows, the result of such a comparison.

Table 3: Estimation techniques used in ASD (Usman, Börstler, & Petersen, 2017)

SLR (Systematic Literature Review) (Britto et al., 2014)	**Survey** (Usman, Mendes, & Börstler, 2015)
Expert judgment	Planning poker
Planning Poker	Estimation by analogy
Use Case Point Method	Expert judgment
Use Case Point modification	Use Case Point method
Regression	Delphi
AI	Other
Other	COCOMO

In what follows, the most reported estimation techniques in the literature, however Machine Learning, and Neural Network are not mentioned in the above studies, even if so, they are presented and explained briefly as these are also the part of estimation techniques.

- **Expert Estimation Method**

Expert judgment, according to (Jørgensen, 2004; Jorgensen & Shepperd, 2007), is the effort estimation technique based on the final judgment of the expert. Here, experts are those individuals who have the competency of estimating any software development effort, i.e. people who have experience in this field or software developers. In this technique, experts are asked about their opinion about how long the project will take to complete or how big is it. So, the expert will estimate the project based on their guts and intuitions. In agile projects, this approach is less useful

because estimates are allocated to user stories or any other user considered functionality (Cohn, 2005).

- **Planning poker**

Planning poker is a popular card game introduced by (Grenning, 2002) and it is the popular estimation techniques mostly used by the scrum team for estimating the effort. More specifically, it is the combination of expert opinion, analogy, and disaggregation into an enjoyable estimation approach.

Particularly, it helps to estimate the user stories of the project, and the development teams are responsible for implementing the specific story. During estimation, a deck of cards is given to the estimator. With those cards, all the team members including the product owner start discussions about the product backlog requirement to reach an agreement for estimates (Ram et al., 2017).

The steps in planning poker are:

- The requirement of one user story is explained by the product owner.
- After that, if any team member is confused about the requirements then (s)he will ask the product owner.
- When everyone in the team understands the user story then they will individually estimate the required effort of story point (story point is used to calculate the size of a user story or feature) by writing on the card or choosing the existing card.
- Then everyone flips their card to reveal their estimate.
- If the team found much difference in the estimation, then they will discuss it and the low and high estimator should share their reason; the scrum team continues the process until they reach an agreement before estimating.

- **Algorithmic Model**

The algorithmic model is also known as the Parametric model, it is based on the mathematical equations and is constructed from collected data or the theory. The basic way to calculate the effort estimation by using algorithmic model is: (Shepperd et al., 1996)

$effort = \alpha \times size^{\beta}$

Where α=algorithmic models apply a productivity coefficient (α)

β=economies (β) on the estimated size of the task, which can be the line of code or the function point.

- **COCOMO**

COCOMO stands for the Constructive Cost Model, it was created by Boehm, (1981) and published in 1981. This model is used for estimating effort, cost, and schedule for software projects. During estimation, it uses a regression formula in which parameters are taken from the historical data and the future project.

The systematic literature review carried out by Britto, Usman, & Mendes, (2014) to report the state of the art of the effort estimation in ASD. It revealed that traditional algorithmic models like COCOMO were not popular in this context. Similarly, in 2015, the survey carried out by Britto et al.,(2015) about effort estimation in agile global software development found that only 5.26% of the respondents used COCOMO along with other methods.

- **Machine Learning**

Machine Learning model is the first effort estimation model, that was initially purposed to improve the estimation accuracy since the 1990s. But it is gaining popularity recently (Wen et al., 2012). The systematic literature review conducted by (Wen et al., 2012) found eight machine learning techniques that are used for estimating effort. They are case-based reasoning, artificial neural networks, decision trees, Bayesian networks, support vector regression, genetic algorithms, genetic programming, and association rules. Among these techniques, case-based reasoning, artificial neural networks, and decision trees are the three most frequently used technique (Wen et al., 2012).

- **Neural Network**

The element processing the nets that are used to learn the mapping between input and output data is known as artificial neural networks. They change their structure during the learning period based on provided sample data because of an adaptive nature. Usually, it is used to solve the input-output data relationships of the complex problem. Various studies (Finnie et al., 1997; Gray & MacDonell, 1997; Park & Baek, 2008; Shepperd & Kadoda, 2001) have examined the capability

of neural networks for estimating the effort of the software development and have proved their usefulness.

- **Estimation by story points**

According to (Osman & Musa, 2016) story point is a number that is used to measure the size of a user story. A user story is a number used to calculate the effort, complexity, and doubt of the story. Generally, an effort is the combination of actual development time and review time required to evaluate the user story whereas, complexity is measured based on the difficulty of the story which includes the dependencies of the task. Finally, doubt means if everyone in the team understands the task clearly. Normally, estimation of a story is done by the development team, while estimating, they use their previous experience or with the help of historical data (Coelho & Basu, 2012). As mentioned by (Osman & Musa, 2016) there are two ways to define the story point, firstly, the development team selects the smallest user story, and they assign 1 to the selected story point based on their opinion. Similarly, another way is by choosing medium size user stories and assign medium points. Finally, the medium size is decided by the development team based on the middle range of their story points.

User Story

A description of the functionality useful for both user or purchaser of a system or software is called a user story it comprises of three aspects: (Cohn, 2005)

A written summary of the story or the function used during planning as a remainder.

- The conversation of the story that serves to flesh out the details of the story
- Text that conveys and document details, which can be used to determine when a story is complete.

Story Point

"*Story points are a unit of measure for expressing the overall size of a user story, feature, or other pieces of work*" (Cohn, 2005). In other words, story points are the relative values used for measuring the effort required for the story that is being developed. For instance, a user story that has 8 points is twice big or complex as a user story that has 4 story points.

Furthermore, if the story points are not linked with time, it is possible to estimate duration using velocity.

Velocity

Velocity measure is used to calculate the productivity of the team, or the ability of the team to complete the number of stories within an iteration (Cohn, 2005).

For instance, if the team completes 3 stories each having 5 story points each than their velocity would be fifteen, and 2 stories estimated with 5 story point then the velocity would be 10.

The formula for calculating average velocity is:

$$MV\ n\ = \frac{1}{n}\sum_{i=1}^{n} \text{Vi}$$

where:

- n = current iteration
- Vi = Velocity of iteration i (number of story points achieved in the iteration)
- MV n = Average velocity for the n first iterations

- **Ideal days estimation**

Ideal time is one of the metrics in ASD related to the amount of time required to complete the task (Cohn, 2005). Another metric is elapsed time that is the combination of the ideal time and unproductive time (Cohn, 2005). Using the ideal day estimation technique, the development team members estimate any piece of work or user story based on their experience, i.e. they include only the workable time. It is worth noting that, although it is easier and accurate to estimate in ideal time than elapsed time, usually it creates confusion among elapsed time and the ideal time in the organization (Cohn, 2005). Ideal days estimations are easier and simple to understand, so that, estimation based on user story is usually faster (Osman & Musa, 2016).

- **T-Shirt Sizes**

T-shirt size estimation is a simpler and more informal technique compared to the story point estimation. It is also useful for large backlogs. The two basic steps used for t-shirt size estimation are (Anooja & Rajawat, 2018):

1. Before estimating all team members jointly and openly discussed the size.

2. Based on the user stories participants give sizes XS, S, M, L, XL, XXL. For instance, if the task is small then S, for heavy task XL and so on.

- **Dot Voting**

Dot Voting is an agile estimation technique which allows the agile team to immediately select or prioritize items with input from teams (Dalton, 2019). It is a simple group activity:

1. The teams are given an equal number of dot stickers.
2. They place dot stickers next to the options presented that they like.
3. The options with most dot win.

It is used during sprint retrospective to prioritize changes.

- **The Bucket System**

The bucket system is an estimation technique used for a large number of backlog items that starts from a small to medium-sized with a large group of participants (Anooja & Rajawat, 2018).

The main steps are:

1. Create the backlog buckets based on the Fibonacci series.
2. According to the complexity of backlog items, select a very low number to highest. For instances, Low=1, Medium=2, High=3
3. Now add all the backlog items and check which bucket is near to that backlog value then decide story point.

- **Large/Uncertain/Small**

This technique is the simplification of the bucket system. There are 3 sizes available: Large, Small, and Uncertain. (Anooja & Rajawat, 2018). For this technique, estimators are asked to set the items

in one category. First, simple user stories are selected, and they are placed in large and small categories. Similarly, complex items are selected and classified. It is beneficial if there are comparable items in the product backlog.

- **Team Estimation Game**

The Team Estimation Game is known as the best technique used for estimating the larger user stories (Anooja & Rajawat, 2018).

The main steps are:

1. The story cards in a pile are placed in the table.
2. The estimator picks the top card from the pile and places it in the playing surface wherever it is easier for them.
3. They now continue the same steps for all the cards and no player wishes to move a card.

After completing the final round team assign numerical estimations to groups of cards. To it, they may use estimations of previously completed tasks.

- **Analogy**

It is a substitute for an expert opinion that comes in the form of estimating by analogy. Saying, "This story is a little bigger than that story." means the estimator compares the present story with the previous one, and if found the story being estimated larger than the one that has already been estimated then, it is given an estimate twice as large (Cohn, 2005).

- **Use Case Point**

Use-Case Points (UCP) was purposed by Jacobson, (1993). It is a software estimation technique used to measure the software size with use cases. When the size of the software is known, the software development effort can be estimated. Use cases usually depend on four factors, which are: i) the number and the complexity of the use cases, ii) the complexity of the actors, iii) some non-functional requirements such as usability and portability, and iv) some environmental factors where the software will be developed (Nassif, Capretz, et al., 2016).

- **Swimlane Sizing**

Swimlane Sizing is one of the popular backlog estimation techniques. This technique works best for those who do not have an idea of story points beforehand (Captain, 2011).

Initially

The deck of incomplete user stories on readable cards or stickies should be collected. After that, a large flat clear surface should be used to and make 7 lines using string or sticky tape mark out 8 “swim-lanes” on the surface.

Steps

The deck of stories should be divided among the team. Ask the team to spend a maximum of 5 minutes and then place the stories in the swim lanes, and the stories should be placed in ascending order. For instance, the smallest stories in left-most lane. Once the stories are placed there might be “clumping” of stories. So, give team 5-10 minutes to move their stories to different lanes to make their story size relative to others and the stories should be in a single lane. Now, ask the team member to rate their satisfaction/confidence regarding relative sizing and placement of the cards (use a scale of 1-5). Once the relative size is finalized by the team with a moderate level there appear the question "For all the stories in the left-most column, do you think you’ll be working on anything smaller?". Now, yes, represent 3 or 5; no represent 1 or 2 (a single number from these options – use your judgment and instinct). Now the remaining column is numbered using Fibonacci sequence applied for estimating story point (0, 0.5, 1, 2, 3, 5, 8, 13, 20, 40, 100, !/?).

As it is expected that every estimation approach has its strengths and weaknesses associated with their abilities. Table 4 shows the comparison results of various (Muketha, 2016; Captain, 2011; Nassif, Azzeh, Capretz, & Ho, 2016; Anooja & Rajawat, 2018).

Table 4: Comparison of software effort estimation approaches

Approaches	**Strength**	**Weakness**
Expert Estimation Method	• Fewer data required • Adopt to special projects	• Its success depends on the expert
Planning Poker	• Adopt to changes in requirements • Reduced bias by involving a team of experts	• Its success depends on a team of experts • Estimation is relative to a team.
COCOMO	• Transparent technique • It provides clear results	• Much data required • The requirements must be clear.

	• Independent on programming language	• Not adopted to changes in requirements. • Require historical data which are not available sometimes
Machine Learning (ML)	• The good quality of the data helps in constructing the machine learning-based effort estimation model used for validating the model's accuracy.	• The ML model needs existing historical data.
Neural Network	• Change the structure during the learning process because of its adaptive nature.	• The main limitation of implementing this model is that there is no consensus on which model is the best.
Story Point	• Easy to compare the sizes of the features/ functionality to each other to determine the relative size. • Require less time.	• With the pressure of the performance improvement, teams may increase story points, e.g. something that was 3 story points, now becomes 5 story points.
T-Shirt Sizes	• It is better to have something that implies a range rather than an absolute number	• They are not additive; it is difficult to tell a boss you will be done in 3 mediums.
Analogy	• Based on similar project experience • More accurate	• Information about past projects is required • Historical data may not be accurate
Use Case Point (UCP)	• It is based on use cases and can be measured very early in the project life cycle. • Easy to use and does not call for additional analysis.	• It can be used only when requirements are written in the form of use cases. • Technical and environmental factors have a high impact on UCP.
Swimlane Sizing	• Work best with teams at any experience also work with the team having less or no experience with story points already. • Work for the team that had never used planning poker before.	• If the estimating team is big, it may not fit in the board.
Dot Voting	• It helps to prioritize the improvements during the sprint retrospective.	• It sometimes gives confusing or false results. • Participants can cheat by adding extra dots, peeling off dots or moving dots
The Bucket System	• It can also be used with larger groups.	• Estimate roughly. • Relative results.

	• Very large numbers of items can be estimated.	
Large/Uncertain/Small	• It is beneficial if there are comparable items in the product backlog.	• Difficult to estimate in the absence of comparable items.
Team Estimation Game	• Faster than planning poker. • Easier to play. • Simple setup • Team members having no experience of the method can be directly involved in estimation rounds.	• Once introduced, the method can be difficult to replace. • From a group size of more than nine people, a single round takes several minutes.

2.3.3. #NoEstimates

Estimation is the main concern for the software development industry, in particular, an accurate estimate always has become a matter of concern (R. Popli & Chauhan, 2014). Due to this, #NoEstimates has been creating a buzz in the software industry and it is an emerging movement within the agile community.

So far, only one book "NoEstimates: How To Measure Project Progress Without Estimating" by Vasco Duarte (Duarte, 2015) and study (Hannay et al., 2018) is available. According to Duarte (2015), estimation is a great area of concern but it is hard to calculate the accurate estimates of the unknown parts. In this situation, many teams are unable to deliver the committed work, because their estimates are often way off. They face project overruns and this restricts the companies from achieving their goal due to delayed delivery.

The #NoEstimate movement was first started in 2012 by Woody Zuill as a Twitter trend. However, in 2011, it was first publicly talked in Europe, Berlin on #NoEstimates at Agile Lean. After that, Woody Zuill and Neil Killick have escorted different seminars in various countries (Cork, 2015).

However, they lead two different groups based on their views. The first group (Woody Zuill) argues on breaking the larger tasks into smaller tasks. Once the tasks are broken, the team should immediately start working on those tasks so that, it will give feedback which helps in delivering the project. The second group (Neil Killick) was similar to the first one, except that, they argue on grouping the stories according to their priorities. The highest value or higher priorities (maybe risk) should be grouped, so that, the delivery helps on generating the feedback and throughput.

Throughput is the process of measuring the task that has been completed and delivered in a specific period. Apart from this, it has become the measuring tool to understand when to start working on a specific task for delivery. Following on, the average throughput can be used by many organizations to forecast the project delivery ("So What Exactly Is This #NoEstimates Movement?," 2015).

#NoEstimates is a hashtag for the topic of exploring alternatives to estimates for making decisions in software development (Duarte, 2015) i.e, ways to make decisions with #NoEstimates. Therefore, #NoEstimates in software development does not mean to eliminate estimation, but exploring the distinct methods of solving the problems without asking the question "How long will it take?" (Heusser, 2013). It could be expected that estimates work perfectly by providing an accurate estimation of an agile software project. However, Duarte, (2015) has presented three clear reasons why they do not work: Hofstadter's Law, Parkinson's Law, and Accidental Complication.

According to "**Hofstadter's Law**: It always takes longer than you expect, even when you take into account Hofstadter's Law". It is one of the popular time management techniques within project planning. Even if so, it always takes more time compared to the expected time to complete the specific task, especially, it applies to software development. Similarly, **Parkinson's Law** points out that "Work expands to fill the time available for its completion". This law explains that the work keeps on expanding even if the specific time allocated to the project. For instance, if you are stuck on some specific task that you are working, the following tasks will automatically get delayed or at least they will end after the scheduled time. Therefore, it automatically affects estimation. Finally, **Accidental Complication** could be another reason why estimating is hard. In the case of newly added tasks or features, there are two explanations. Firstly, **Essential complication or g(e)** means the problem becoming hard within itself. For instance, the problem is hard, so the system is complicated. Secondly, **Accidental complication or h(a)** means not being good at our jobs or "*we stuck at our jobs*" due to various factors supposedly, the pressure from the organizational structure (till when will we be able to get approval for a new test environment?) and from how programs are written (development keeps on evolving so, things keeps on changing and functionality will be evolving). Therefore, the cost of the feature can be calculated as: Cost of feature = f **g(e)**, **h(a)**

#NoEstimates usually divide the larger projects stories into smaller user stories where the requirement for estimation is decreased and the focus will be in measuring the development progress and predicting the future of the project (*#NoEstimates - Alternative to Estimate-Driven Software Development*, 2015). According to Boiten (2017), some of the notable features of #NoEstimates are:

1. The team can concentrate from day one to create the value for the project.

2. Relying on the project improvement good decisions can be made.

3. Frequently reviewing the project development and predicting a project will help to launch the project without any difficulty or trouble.

2.3.4. #NoProject

Based on the (Leybourn & Hastie, 2018) book the concept of the project in software development is inherited from the engineering field, as they have a strong background in project management. However, due to agile and lean approaches, there is an ongoing discussion about whether to follow project management or not.

According to (Leybourn & Hastie, 2018) in software development, the project model is a “temporary endeavor undertaken to create a unique product, service, or result” into an organization. The purpose of using a project is to predict and control budgets and financial spend. In fact, the software project model seems a bad fit for software development organizations because of its emphasis on allocated meeting time and cost criteria destroys the values of the project. But without the project model, it is difficult to guide, control, and governed the work (Leybourn & Hastie, 2018).

The #NoProject concept started in 2005 and it is gaining popularity in recent years. According to (Leybourn & Hastie, 2018), "*#NoProject is a movement and a philosophy that represents a set of principles, practices, and ideas that any organization can apply*". It can be seen as a modern agile approach that directs companies on continuous and market-validated value delivery. There are no hard and fast rules for the organization that adopts #NoProject, but if they want then they can evaluate it, experiment it, and then adopt #NoProject rules. (Leybourn & Hastie, 2018)

Some of the common activities required in all #NoProject approaches are listed below (Leybourn & Hastie, 2018) :

- The expected outcomes should be defined in terms of metrics that truly provide values, instead of easy-to-measure vanity metrics.
- Recognizing the first small step or experiment to validate the assumptions that are being made for obtaining the outcomes.
- Execute that step.
- Measure the results.
- Examine the method and adapt to the reality of your learning.
- Finally, continue the above steps, pivot, or stop if its already done enough (reached maximum value, or learned enough).

#NoEstimates and #NoProject are related to each other. Both of them focus on delivery value. Indeed, to plan a project estimation is required, and estimation is the part of the planning of software projects. In other words, #NoEstimate removes the justification of estimates and helps the organization focus on value delivery first (Duarte, 2015). And, #NoProjects is the modern agile approach that directs companies on continuous and market-validated value delivery (Leybourn & Hastie, 2018).

However, it is worth noting that the book #NoEstimate by Duarte, (2015) has not mentioned anything about #NoProject. Whereas, the book #NoProject written by Leybourn & Hastie, (2018) shows that both #NoEstimate and #NoProject have the same motivations even if both are independent. Leybourn & Hastie, (2018) have mentioned that #NoEstimates talks about breaking the task into smaller pieces, prioritizing them, start working on them and predicting their completion based on the actual delivery rate rather than guessing.

2.4. Related Work

In 2003, Molokken & Jorgensen, (2003) analyzed the most appropriate software effort estimation surveys. The preliminary findings show that expert judgment-based estimation to be the most used one. Similarly, it was claimed to be the most popular, possibly, because of the absence of evidence related to the formal estimation models that direct to better estimates.

During 2004, a survey on software estimation was carried out in the Norwegian industry by (Moløkken-Østvold et al., 2004). The findings revealed that expert judgment-based estimations were the most popular among practitioners. Furthermore, the authors found that formal estimation methods did not demonstrate any improvement in the accuracy of the estimation covering expert estimations. Indeed, managers believe to have better accuracy than it already is.

Additionally, Haugen (2006) conducted an empirical study that analyzed 101 user stories using planning poker from four different projects. The result revealed that the user stories estimated by groups are superior to individuals. However, group estimation was reduced by dominant personalities and anchoring effects (a cognitive bias in which people make a decision based on the first piece of information available). Similarly, the findings show that the estimation done by the Agile development team had better performance than estimating the unstructured group estimation process. The study concluded that the estimation done by using planning poker gives a more realistic release plan.

Likewise, the survey performed by Trendowicz et al., (2008), to analyze the current industrial practice of effort estimation in software development found that various expert-based techniques were accepted by the software industry. Later, in 2011, another survey conducted by Mansor et al. (2011) to understand the most commonly used effort estimation method showed that the most reported method was expert judgment followed by, price to win and algorithmic models. In this survey, 13 responses were collected from 30 software companies.

Furthermore, Usman et al. (2014) conducted a systematic literature review in 2014 on effort estimation in ASD. The authors selected 25 primary studies that summarize the state of the art in this field. The four main findings are: 1) subjective estimation methods like expert judgment, planning poker, use case points estimation method are often used for agile estimation; 2) Use Case Points (UCP) and Story Points (SP) are the most often used size matrices; 3) Mean Magnitude of Relative Error (MMRE) and Magnitude of Relative Error (MRE) is the frequently used metrics in ASD; and 4) Team skills, prior experience, and task size are included as the 3 fundamental -cost drivers in ASD.

More recently, Lenarduzzi, Lunesu, Matta, & Taibi, (2015) evaluated functional size measures and effort estimation based on the small agile projects. The result showed that the effort estimated by software developers were more realistic than the estimates achieved by functional size measure.

Tanveer, Guzmán, & Engel, (2016) carried out a study to examine and understand the estimation process related to the accuracy of ASD. In this study, three agile teams that worked on different web applications were involved. The result showed that developers' knowledge, experience, and complexity affect the accuracy of the estimation.

A comparative analysis study was carried out by (Usman & Britto, 2016). In this study, they compared the co-located and globally distributed teams to identify the similarities and differences of effort estimation practice in ASD. The result shows that planning poker is the most reported effort estimation technique for both teams. Moreover, story points are the frequently used size metrics in both teams.

A recent case study conducted by Pozenel & Hovelja, (2019) estimated the time required and accuracy of user stories. Here, authors compare planning poker and team estimation games (a relatively new estimation technique) for ASD, which has not gained attention from researchers despite its growing popularity among practitioners. This study concludes that the Team Estimation Game produces more accurate story estimates than Planning Poker.

To sum up, most of the previous studies are focused on outlining the various estimation techniques available. The majority of studies also identified expert judgment to be the most popular estimation technique. However, to the best of our knowledge, there are no published studies on the benefits and challenges of effort estimation techniques available for ASD whereas, there are few studies related to #NoEstimates and #NoProjects movements. Therefore, this study is focused on identifying the benefits and challenges (inaccuracy) of estimation techniques including the #NoEstimates and #NoProjects movements by collecting data through a survey.

3. Research Methodology

This chapter presents the research questions and the theory behind the research methods used in this thesis.

3.1. Research Question

Research Questions (RQs) of this study are formulated based on the research objectives:

RQ1: What are the effort estimation techniques used in ASD?

RQ2: What are the benefits of estimation techniques in ASD?

RQ3: What are the reasons for inaccurate estimates in ASD?

Table 5: Research questions and motivation

Research Question	Motivation
RQ1: What are the effort estimation techniques used in ASD?	This question is prepared to collect the software practitioners' opinions related to the estimation technique software company use.
RQ2: What are the benefits of estimation techniques in ASD?	This question is used to collect the software practitioners' opinions related to the benefits of estimation techniques used in ASD.
RQ3: What are the reasons for inaccurate estimates in ASD?	This question is used to collect the software practitioners' opinions about the factors for inaccurate estimates in ASD.

3.2. Research Method

In the first place, the research design must be defined. Research design means the inquiry among qualitative, quantitative, and mixed methods approach which leads the researcher in a specific direction during the research process it is also the strategies of inquiry (Lincoln et al., 2011). However, the researcher can choose their method based on various factors like research questions being studied, the method that has been used in previous research, and the researcher's philosophical learnings (Creswell & Miller, 2000).

The detailed discussion of the qualitative, quantitative, and mixed-method will be done in the following section. Usually, the quantitative analysis provides the results in numeric and statistic

format. Qualitative analysis in the form of words, opinions, and expert thoughts. Finally, a mixed methodology is a combination of qualitative and quantitative studies. Commonly, these methods are individually used to collect the information needed to solve the research question, yet, it is possible to combine both the methods.

Table 6: Research Design (Creswell & Miller, 2000)

Quantitative	Qualitative	Mixed Methods
• Experimental Design • Non-experimental Designs, Surveys	• Narrative Research • Phenomenology • Grounded theory • Ethnographies • Case Study	• Convergent • Explanatory • Exploratory

3.2.1. Quantitative Methods

The quantitative method of data collection is done by using prearranged instruments in which data are examined statistically (Creswell, 1994). These data are based on evidence or the number which are broken down in into quantitative information by using various design pattern graphs, tables, or charts. This method facilitates the researcher to see the appropriate design of response and helps in concluding that design (Oates, 2005). Apart from this, quantitative research has closed-ended questions like "how much?" and "how often?" (Coleman & O'Connor, 2007).

Some of the popular Quantitative strategies of inquiry are Experiential Research and Survey Research which are explained in detail:

- **Experiential Research:** The experimental method is a classical method adopted by the scientific method in the quantitative analysis, here, the research helps to decide if any input has an impact on its output (Coleman & O'Connor, 2007). Usually, the experiment takes place with the involvement of two groups, one group receives special treatment, whereas, no specific treatment for another group, afterward, the result groups are determined.
- **Survey Research:** Creswell & Miller, (2000) have defined surveys as '*quantitative or numeric descriptions of trends, attitudes, or opinions of a population*'. Typically, researchers use a survey method for collecting data and, they are in the form of questionnaires or structured interviews (Fowler Jr, 2013). Additionally, two main goals of

a survey methodology are reducing the errors during data collection and measuring the errors as they are undeniably the part of surveys (Fowler Jr, 2013). However, a survey is fulfilled if the researcher can collect the required result (Babbie, 1989).

3.2.2. Qualitative Methods

The qualitative research method is non-numeric which consists of words, images, sounds, video, and field notes. Basically, this method helps in gathering in-depth information from the expert in the related field. Typically, the purpose of qualitative research is to collect and interpret non-numeric data and it has open-ended questions like “why?” and “how?”(Coleman & O’Connor, 2007). At this point, small groups are chosen purposefully to collect relevant data like thoughts, opinions, experiences, and feelings. The data are obtained in the form of interviews and surveys. (Oates, 2005). The various important strategies of inquiries are presented below:

- **Narrative research:** It is the research phenomenon where the researcher tries to find the life experience of the individuals based on the stories told by them. Here, the information narrative should be presented by the researcher in chronological order (Czarniawska, 2004). Specifically, former researchers should understand the stories of an individual for collecting "data; their stories" and should describe their experience in sequential order (Ivankova et al., 2006).
- **Phenomenology:** A qualitative research method used for describing a certain phenomenon, experienced by a human being refers to Phenomenology (Creswell & Garrett, 2008). This methodology helps in understanding people's lived experience (lived spaced, lived body, lived time, and lived human relations) in specific situations. Not only, phenomenological research focuses on in-depth conversations and interviews for data collection but also, from diaries, drawings, or observation. The result will outline people's experiences (Giorgi, 2012).
- **Grounded theory:** Creswell, (1994) has defined Grounded theory (GT) as “*a strategy of inquiry where the researcher derives a general, abstract theory of a process, action or interaction grounded in the views of the participants”*. Necessarily, it is a data collection process that has multiple stages, and it also helps in generating the theory from those data. Ultimately, this methodology focuses on generating the theory from multiple comparisons of the data, concept, and categories (Birks & Mills, 2015).

- **Ethnographies:** It is the process of studying the behavior of groups in a natural setting for a longer period. During this period, the researcher collects the data by examining the work of participants' behavior, beliefs, and language (Murchison, 2010). Similarly, this study evaluates the actions and behavior of people, therefore, the researcher is fully involved in the everyday life of participants to gain an in-depth understanding. Likewise, observation and interviews can also be used for data collection. (Creswell, 1994)
- **Case study:** A qualitative research method in which the researcher acquires an in-depth summary of the study under evaluation during case studies, which can be done through program, event, activity, and process. Additionally, cases are bounded by time and activity, therefore, they must collect the required data in a specific period. Similarly, case studies design should take into account when: to answer "how" and "why" questions (Creswell, 1994; Yin, 2009).

3.2.3. Mixed-Method

A mixed-method is the combination of both qualitative and quantitative research methodologies in which the strength of both qualitative and quantitative methods is used to solve the complex problem (Creswell & Miller, 2000). These two elements qualitative and quantitative are linked together at some point during the research to solve the research question that is deeper and needs an integrated response (Glogowska, 2015; Zhang & Creswell, 2013). Mostly, integration can be done at any stage during the research process. Moreover, it helps in generating a better understanding of the phenomenon (Glogowska, 2015)

Johnson, Onwuegbuzie, & Turner, (2007) has defined:

"*Mixed methods research is the type of research in which a researcher or team of researchers combines elements of qualitative and quantitative research approaches (e. g., use of qualitative and quantitative viewpoints, data collection, analysis, inference techniques) for the broad purposes of breadth and depth of understanding and corroboration*."

Not only, qualitative and quantitative method, but also, mixed research methods are growing independently and increasingly, and becoming the third methodological movement (Denscombe, 2008). In fact, R. B. Johnson & Onwuegbuzie (2004) believed that this model could help in bridging the gap between quantitative and qualitative research. Initially, this method was championed by researchers who have already published their mixed method books in different

disciplines like education, sociology, and the health sciences (Čížek, 2009; Mertens, 2014; Niglas, 2007; Teddlie & Tashakkori, 2009).

Quite often mixed-method and multi-method are considered the same. Essentially, mixed methods research solves the single research question of the study by using both qualitative and quantitative research methods, whereas multi-method research uses two different methods qualitative and quantitative to solve the two different research questions of the same study (Babbie, 1989).

The three basic mixed designs are exploratory sequential, explanatory sequential, and convergent. These methods are based on qualitative and quantitative data for the research questions, but the only difference is the order in which data are collected.

- **Convergent Parallel Mixed Methods Design:** For the convergent mixed method both the quantitative and qualitative data are collected at about the same time and are then used together to triangulate the findings and answer the research questions. According to Creswell & Miller, (2000) "*Triangulation is the combination methodologies in the study of the same phenomenon*".

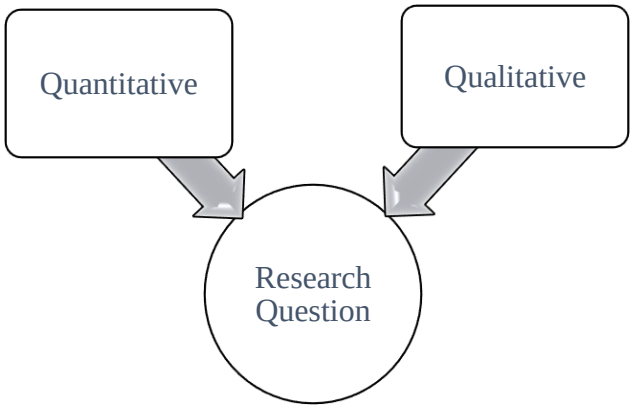


Figure 4: Convergent Mixed Method (Creswell & Miller, 2000)

- **Explanatory Sequential Mixed Methods Design:** In the case of the explanatory mixed-method the quantitative data are collected first, afterward qualitative data are collected and meant to serve the purpose of explaining the result of the data. Although data are collected at different times, still they work together to answer the same research questions (Creswell & Miller, 2000).

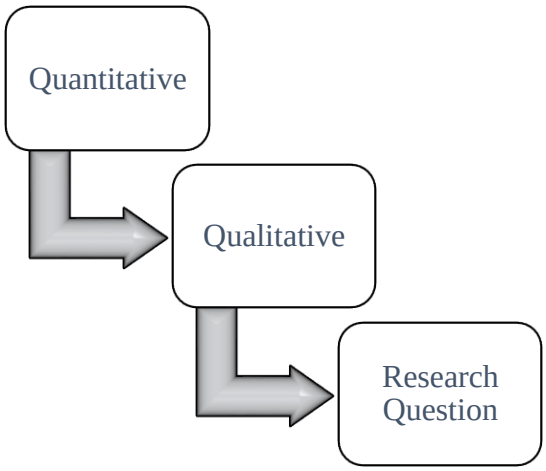


Figure 5: Explanatory Mixed Method (Creswell & Miller, 2000)

- **Exploratory Sequential Mixed Methods Design:** In the case of the exploratory mixed-method the qualitative data are first collected and then analyzed. Once the analysis is done the result is used to notify the collection of the quantitative data. Although the data are collected at different phases, still they work together to answer the same research questions (Creswell & Miller, 2000).

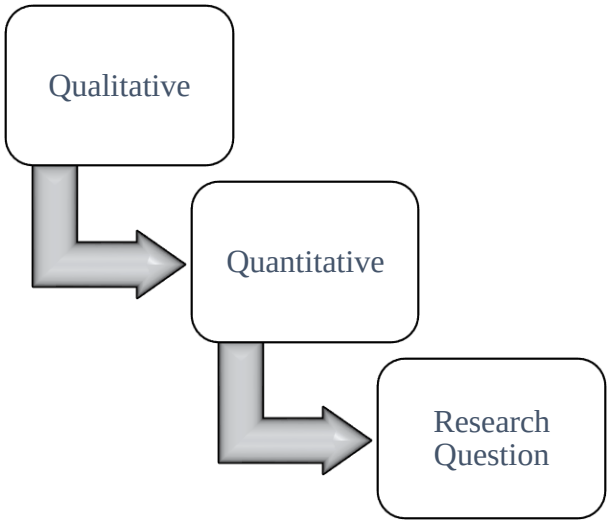


Figure 6: Exploratory Mixed Method (Creswell & Miller, 2000)

For this study, a mixed methodology will be used for information collection. The data collection technique used for this study a survey.

3.3. Data Collection

In this section, the data collection process is briefly described. Given that it will collect the software practitioners' options to identify the benefits and limitations of the effort estimation techniques used, a survey is carried out.

A survey is the systematic form of information collection with the purpose of forming a quantitative description of the attributes of the specific population that reflects the behavior of individuals like attitude, behavior, opinion, and belief.

A survey as data collection uses different techniques like personal interviews, telephone interviews, direct observation, or self-administered questionnaires (Scheaffer et al., 1990). In this study, the data collection technique used to survey is a questionnaire. Here, an online web-based questionnaire tool (google forms) will be used for data collection because of its practical mechanism for data collection from respondents. It also allows us to collect data from a wider and globally distributed population.

- **Survey design**

Figure 7 shows the plan of survey design. First, survey questions are design based on the research questions. This is then followed by pilot testing, participant selection, and survey. Once the data collection is completed the analysis will begin.

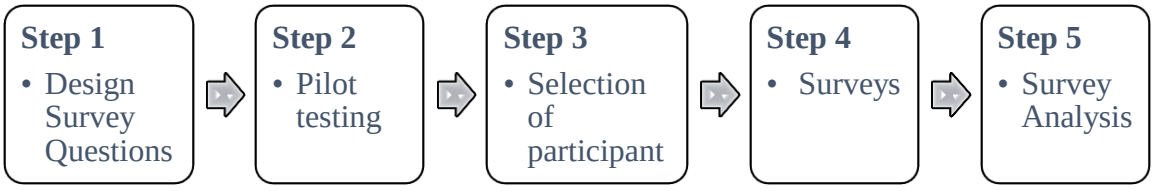


Figure 7: Survey analysis design

a. Design of survey question

Initially, a survey questionnaire was prepared based on the research questions, to gather the information from the software practitioners having experience in agile development. The questionnaire is based on the literature review and prepared with multiple discussions with supervisors. The questionnaire is divided into 7 sections: 1. Demographics, 2. Software

Development Project, 3. Effort Estimation Technique, 4. Estimation Benefits, 5. Inaccurate Estimates 5. Movements and, 6. Closing.

The first section contains Demographic information and it has questions related to the work location, genre, and agile experience. Second, the software development project contains the questions related to role, team size, project length, project domain, development approaches, and the perceived importance of estimation. Third, the Effort Estimation Technique contains the questions related to estimation techniques and measurement units. This helps us to get the answer to our first research question. Fourth, estimation benefits/advantages, it helps us to get the answer to the second research question. Fifth, Inaccurate estimates, this section contains the survey questions to get answers related to the factors for inaccurate estimates. Sixth contains the questions related to #NoEstimates and #NoProject. Finally, the closing includes the participants' comments, or issues not addressed in the questionnaire and the potential interest in the future interview phase. The designed survey questions are presented in APPENDIX A. *Figure 8* shows the mapping between survey questions and research questions.

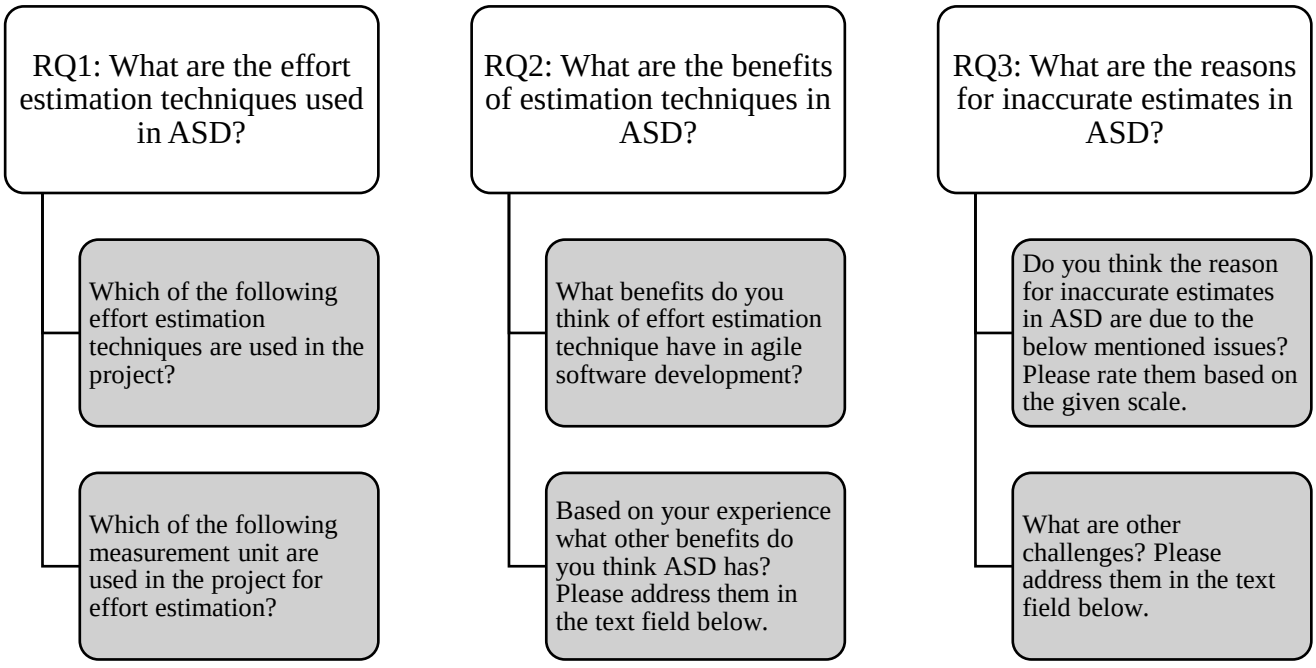


Figure 8: Mapping of research questions and survey questions.

Survey Question(s) for RQ1: (1) Which of the following effort estimation techniques are used in the project? Respondents could select multiple items from a list of possible techniques and can provide responses based on the Likert scale (0- Do not know, 1- Never Use, 2- Rarely, 3- Sometimes, 4- Very Often, 5- Always). Besides, a text field was provided for further answers. (2) Which of the following measurement unit are used in the project for effort estimation? Likert scale with multiple answer options was provided.

Survey Question(s) for RQ2: (1) What benefits do you think of effort estimation techniques have in agile software development? A list of benefits was added with the Likert scale (0- Do not know, 1- Strongly Disagree, 2- Disagree, 3- Neutral, 4- Agree, 5- Strongly Agree). (2) Based on your experience what other benefits do you think ASD has? Please address them in the text field below. Only a free text field was provided for answering.

Survey Questions(s) for RQ3: (1) Do you think the reason for inaccurate estimates in ASD are due to the below-mentioned issues? Please rate them based on the given scale (0- Do not know, 1- Strongly Disagree, 2- Disagree, 3- Neutral, 4- Agree, 5- Strongly Agree). Respondents could select multiple items from a list of possible techniques and can provide responses based on the Likert scale. (2) According to your experience, if there are other challenges (related or not related to inaccurate estimates)? Please address them in the text field below. Only a free text field was provided for answering.

b. Pilot testing the questionnaire

After the questionnaire was designed, a pilot test was done to make sure that the questions work as expected, both context-wise and technically as the survey was web-based. According to (Kasunic, 2005), a pilot test is done to simulate the survey implementation with a few members from the target group, to detect the problems in the questions, as well as, layout, process.

To collect the feedback on the questionnaire, an evaluation form with a few questions was used, as proposed by (Kasunic, 2005). The following questions were asked

- Are there any unclear questions or answer options?
- Is something relevant missing from the questions?
- Are there any unclear terms used?
- Was the ordering of the questions logical?

- How long did it take to answer the questionnaire?
- Were there any problems with the web survey system?

For the pilot test, 4 software practitioners having experience in ASD and knowledge of the estimation process were involved. Participants from 2 countries were asked to do pilot testing, Norway, and Nepal. We followed two rounds of pilot testing. In round 1, one participant was selected to get insight into the survey. So, the feedback was received specifically from 3 areas: 1. The total time survey took 2. Options on the Likert scale, 3. Questions related to #NoEstimates and #NoProject.

The time expected was around 10-15 minutes, but the participant told that it took approximately 25 minutes. Similarly, for the Likert scale, the participant told that there were too many options to consider which took more time to complete the survey. Finally, questions related to #NoEstimates and #NoProject were confusing. Therefore, some modification was made, and others were removed.

After collecting the feedback, some changes were made for the second round. For round 2, three participants were selected. This time there was no problem with the questionnaire and the time taken for the survey was 10 minutes approximately.

c. Target Population

The targeted population of this study is those who have experience in ASD. According to the aim and objectives of this study, software practitioners having experience from Agile software development can provide valuable and reliable results.

d. Carrying out a survey

After conducting the pilot study and fixing the discovered issue, the survey was made open for answering. Before sending the questionnaire link to my current colleagues, I asked for permission from my project manager. Once the approval was received the link to the questionnaire was emailed to personal contacts as well as, potential participants. Apart from this, participants were asked to circulate the link to their colleagues and others who were working in ASD. The period to answer the questionnaire was about two weeks starting from the 24th of June 2019 to the 8th of July 2019.

After the week of sending the questionnaire, a reminder to the survey participation was sent to increase the response rate (Dillman, 2011; Kasunic, 2005), the remainder were sent by email.

e. Survey Analysis

Data were analyzed using a statistical analysis tool pack in Excel, SPSS, and SQL (Standard Query Language. Initially, survey data were collected, followed by, filtration of valid responses, data transposed into the numeric format, frequency, and percentage of the data were calculated, descriptive statistical analysis, finally, a hypothesis test was done.

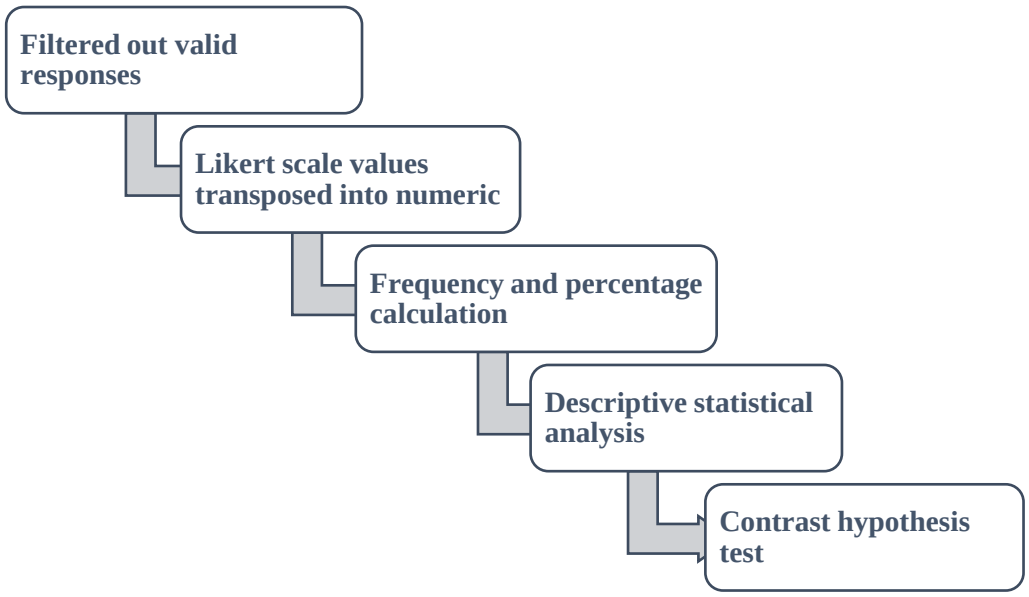


Figure 9: Data analysis process

4. Result Analysis

This chapter contains the analysis of the survey results that were collected. The chapter is structured based on the design of the questionnaire. The first sub-section contains the demographic information, then the software development project information, third Effort Estimation Techniques; fourth Estimation Benefits; fifth Inaccurate Estimates; sixth Movements and, finally Closing.

Initially, 62 responses were gathered from the survey. However, it was identified that only 53 respondents have experience in effort estimation and ASD. To do so, at the end of the software development project section, one question "*Do you participate in effort estimation in your team?*" was asked to find out their experience and participation estimation. Here, if the respondents answered yes, they were redirected to the question related to the main section. Else, they were redirected to the closing section.

Therefore, the demographics section and software development project section contains the result of the analysis of all 62 responses. Whereas, the other sections contain the analysis result of 53 responses. Demographics section contains various questions like working place, current role, experience. Whereas, Software Development Project contains the questions related to the job role, size of the team, project size, business domain, software development approach, how important effort estimate was, effort estimation experience, and finally, participation in estimation was asked. The following section contains the first research question related to effort estimation technique and measurement unit, followed by estimation benefits and other benefits, finally, the reason for inaccurate estimates.

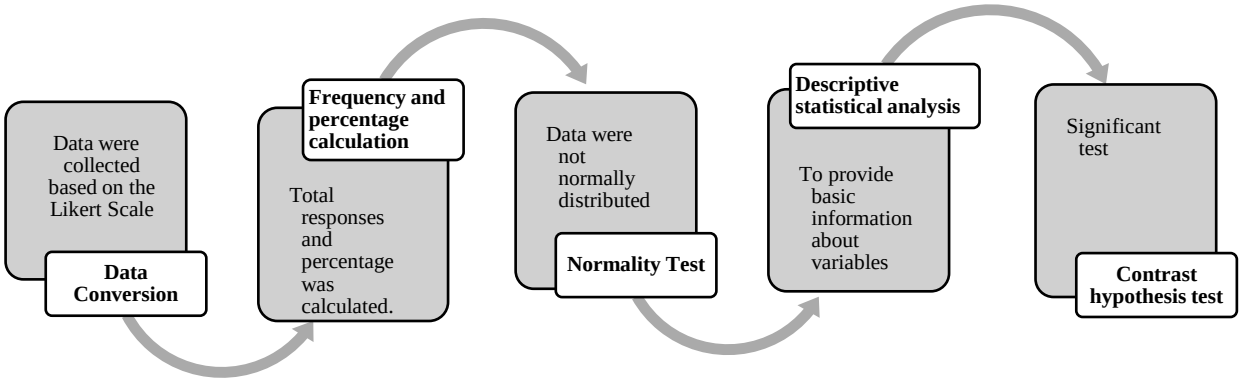


Figure 10: Data Analysis process

4.1. Demographics

To understand the respondent's background, the first part of the questionnaire included questions related to the work location, genre, and experience in ASD. For each question, each participant was able to select the option (answer) that best describes them.

4.1.1. Country/Location

A total of 62 respondents from 4 continents and 7 countries participated in the survey (see *Table 7*). Most of the respondents were from Nepal followed by Norway. Therefore, the number of respondents per continent, the majority came from Asia (54.8%) followed by Europe (32.2%), North America (11.3%), and South Africa (1.6%).

Table 7: Respondents by continents and countries

Continent	Countries	Frequency (%)
Africa	South Africa (1)	1 (1.6%)
North America	USA (7)	7 (11.3%)
Asia	Nepal (26), India (7), Korea (1)	34 (54.8%)
Europe	Norway (18), Germany (2)	20 (32.2%)

4.1.2. Genre

Figure 11 depicts the information about the respondents' genre along with the frequencies and percentages. A total of 80.6% were male who takes part in this survey. Remaining 17.7% are female, and only one participant (1.6%) does not reveal their gender.

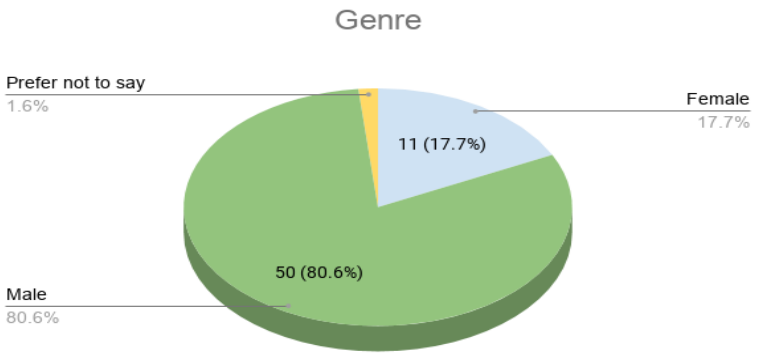


Figure 11: Gender of the respondents

4.1.3. Agile Experience

Table 8 shows the answers about the respondent's experience in ASD along with the corresponding frequencies and percentages. A total of 56.4% of the respondents have more than three years of agile experience, while only 9.7% have less than a year of experience. Moreover, 33.9% of respondents have more than one year and less than 3 years of experience.

Table 8: Experience Table

Experience	**Frequency**	**Percentage (%)**
Less than 1 year	6	9.7%
1-3 years	21	33.9%
3-5 years	23	37.1%
5-10 years	9	14.5%
11-20 years	3	4.8%
Total	**62**	**100.00%**

4.2. Software Development Project

This section contains information related to a software project. Therefore, this section aims at collecting the information related to a job role, team size, project length, business domain, development approach, and finally, the importance of estimation.

4.2.1. Job Role

Table 9 shows the job role of the participants. In this survey, the highest number of respondents was software developers, a total of 59.7%. This is followed by software tester (19.4%), scrum master (8.1%), and product owner (6.5%). Finally, software architects and program Manager have the same number of responses (3.2%).

Table 9: Role by frequency and percentage

Role	Frequency	Percentage
Software Developer	37	59.7%
Software Tester	12	19.4%
Scrum Master	5	8.1%
Product Owner	4	6.5%
Software Architect	2	3.2%
Program Manager	2	3.2%

4.2.2. Team Size

Figure 12 shows that most of the respondents (50%, 31) work in teams of 6 to 10 team members; 38.7% of the respondent reported a team size of 1-5, 6.5% a team size greater than 20. Only some respondents reported a team size of 11-20 (4.8%, 3) people.

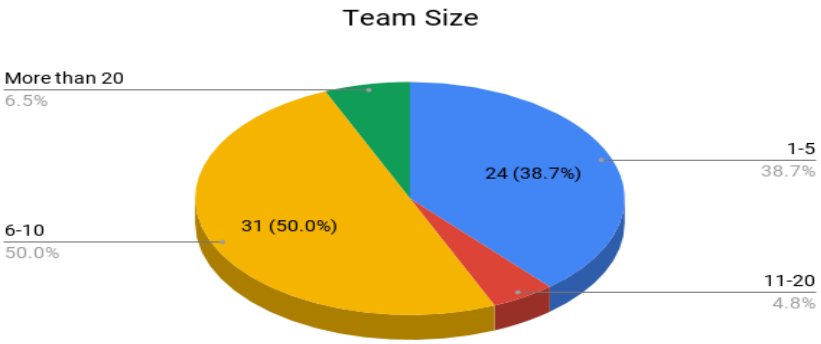


Figure 12: Team size

4.2.3. Project Size

From 62 respondents, around one-third of them reported that the project was more than 1 year (74.2%, 46). While 11.3% responded that the project was 2-4 weeks, followed by project size of 1-6 months (9.7%, 6). Only 4.8% (3) of respondents reported that the project size was 7-12 months. Figure 13 depicts the project size reported by the participants.

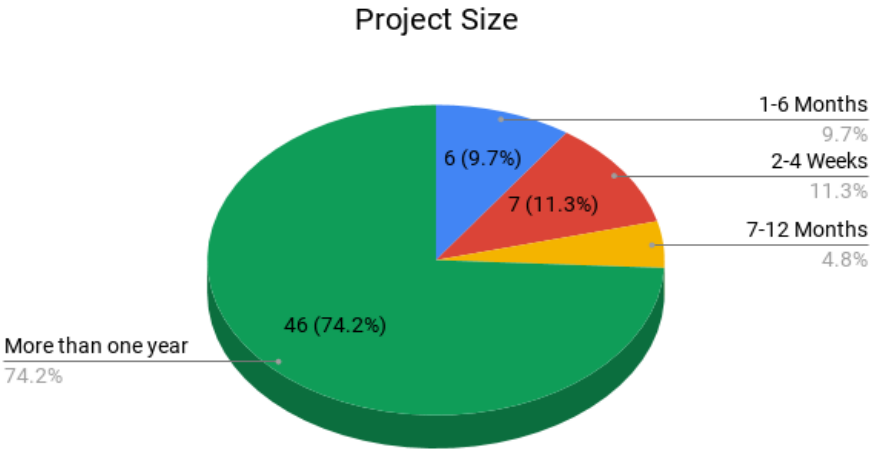


Figure 13: Project Size

4.2.4. Business Domain

Figure 14 depicts that most of the respondents reported that they developed E-commerce applications (48.4%, 30). Some of them also worked on energy technology (8.1%, 8), health insurance (3.2%, 2), data processing (3.2%), banking (3.2%) and few develops others (33.9%), like medical (1.6%, 1), telecom (1.6%), auto finance (1.6%). It is worth noting that, given that the question was open, the answers were categorized. For instance, some respondents entered text like *ecommerce, Ecommerce, E-comerce E-commerce, and FontExplorer X, as well as E-commerce and Fonts technology* which were classified as E-commerce. Similarly, *energy app, energy* were categorized into Energy Technology.

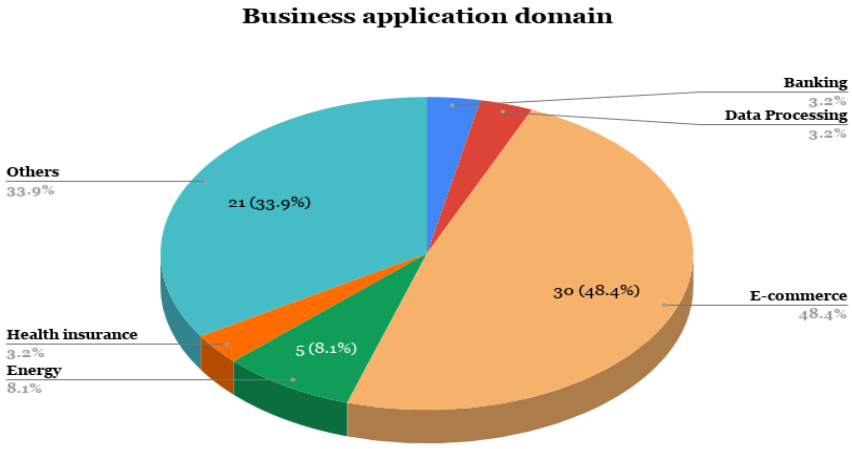


Figure 14: Business Application Domain

Furthermore, one question about the importance of effort estimation was asked: "*How important do you think software development effort estimation is?*". In this case, a Likert scale was used (from 1 not important to 5 very important). Figure 15 shows that 75.8% (47) of the respondents perceive that it is *very important*, followed by 19.4% (12) *important*, 4.8% (3) *50-50*.

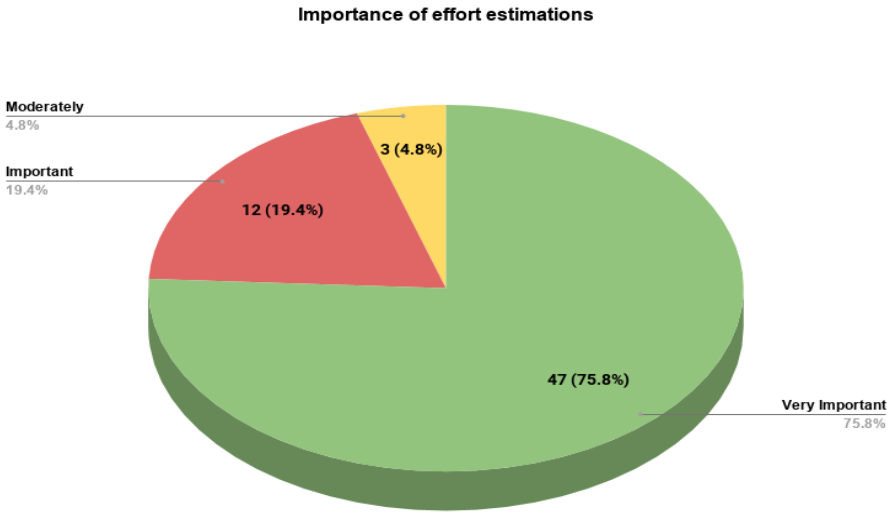


Figure 15: Importance of effort estimation

Given that work experience on the topic gives relevant and reliable answers to this study, the following question (*Do you participate in effort estimation in your team*?) was asked to know the participation of respondents in effort estimation. Figure 16 shows that the majority of respondents took part in the estimation. Based on how they answer that question, participants were involved in the next part of this study, in which the research questions are related to estimation.

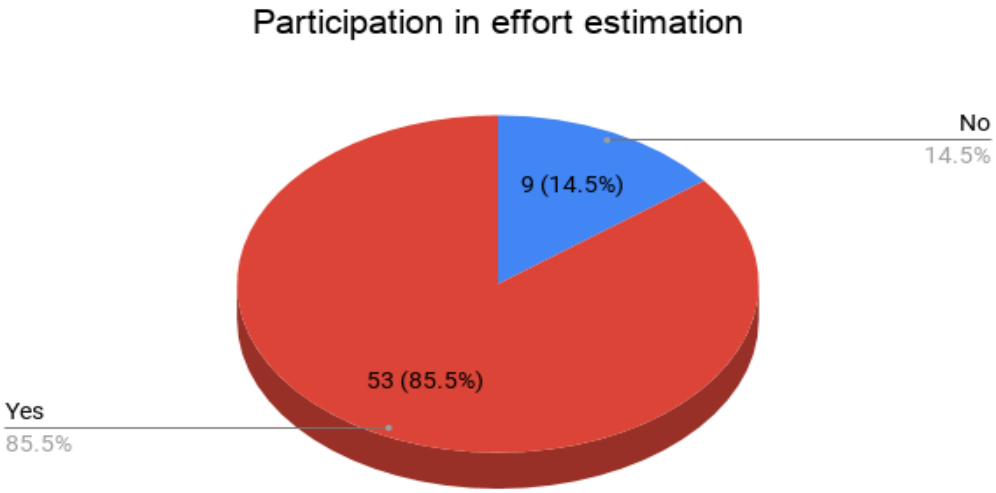


Figure 16: Respondent participation in effort estimation

Hereafter, 85.5% (53) of the participants in this study is the sample to answer the research questions. Therefore, 53 responses were considered valid for our research questions.

Although 9 (14.5%, 9) respondents state that they do not participate in effort estimation techniques, their responses were further analyzed. From this analysis, two responses draw attention to the role and experience of the participants. One of them was given by a Program/Product Manager (PM) and the other was given by a developer. They reported 11-20 and 5-10 years of the agile experience but they claim that they did not participate in Agile estimation. Moreover, the PM reported that "always" uses DevOps and "Often" used Scrum whereas, the developer reported that "always" uses Extreme Programming as a software development approach. Even more, despite that, they were not involved in the effort estimation both of them reported that estimation is "important" and "very important" respectively. The remaining 7 responses were not so curious as most of the participants have 1-3 years of experience and did not use the Agile software development approach.

4.3. Normality Test

Two approaches to test the normality of the data are statistical tests and visual inspection (Mazlan & NUR, 2012). For the statistical test, well-known normality tests are Kolmogorov-Smirnov and Shapiro-Wilk. Shapiro-Wilk is normally used if the sample size is less than in 2000, which is suitable for this study as the sample size is 53. Whereas, for visual inspection frequency distribution, skewness, kurtosis, and Normal Q-Q plots can be used.

In this research, the Shapiro-Wilk test is used. **Normality, a** test that obtained **a** significance value greater than 0.05 then the data are normally distributed. However, the significance value of the data in this study is less than 0.05, therefore the data are not normally distributed. Test results are presented in the tables in section 4.4 as **statistics** and **sig**.

In what follows, a descriptive statistical analysis and normality test results of estimation techniques are presented, followed by the same analysis of the measurement units, estimation benefits, and inaccurate estimates.

4.4. Descriptive Statistical Analysis

As the survey questions for this research were designed using a Likert scale. Those responses based on Likert scale values were converted in numeric form as presented in Table 10 shown. After converting the data into a numeric format, the data were filtered out because the answer "*Do not*

know it" is excluded from the analysis. In other words, even if 53 responses were considered valid the count might not be the same after filtration. However, the diverging stacked bar chart contains all 53 responses.

Table 10:Value used for Likert scale

0	Do not know	Do not know it
1	Strongly Disagree	Never
2	Disagree	Rarely
3	Neutral	Sometimes
4	Agree	Often
5	Strongly Agree	Always

The descriptive statistical analysis was done using excel and SPSS. From this analysis Count, Sum, Mean, Standard Deviation (SD), Variance, Range, and finally, Statistics and Sig. (normality test results) values are presented in tables.

The column, "sum" is the weighted value. Table 11 shows an example of the calculation based on the Effort Estimation Technique called Dot Voting. The frequency distribution tables for Techniques, Benefits, and Inaccuracy are presented in APPENDIX B.

Table 11: Sum Calculation

Weight (W)	**Value (V)**	**Total W.V**
0	16	0
1	23	23
2	6	12
3	5	15
4	1	4
5	2	10
ΣW = 15	ΣV= 53	Σ = 64

4.4.1. Development Approach

Table 12 shows the result of the descriptive statistics analysis of the data. The first column presents the list of software development approaches, followed by Mean, Standard Deviation (SD), Range, and sum. The column sum is weighted based on the scale see an example of calculation in Table 11). The descriptive statistical analysis was done to find out the most used software development approach. Based on the highest mean values and the frequency of use, the most commonly reported

software development approaches are Scrum and DevOps. Whereas, the least reported development approach is *Kanban* (2.708) and *Waterfall* (1.714). Moreover, just one participant reported that s/he used the RAD (Rapid application development) approach as well. The frequency distribution table is added in the Appendix (see Appendix B1).

Table 12: Software Development Approach

Software development Approach	Mean	SD	Range	Sum	Count
DevOps (D)	3.600	1.262	1 - 5	180	50
Extreme Programming (XP)	2.791	1.390	1 - 5	120	43
Kanban (K)	2.708	1.237	1 - 5	130	48
Scrum (S)	4.481	0.896	1 - 5	233	52
Waterfall (W)	1.714	0.913	1 - 4	84	49

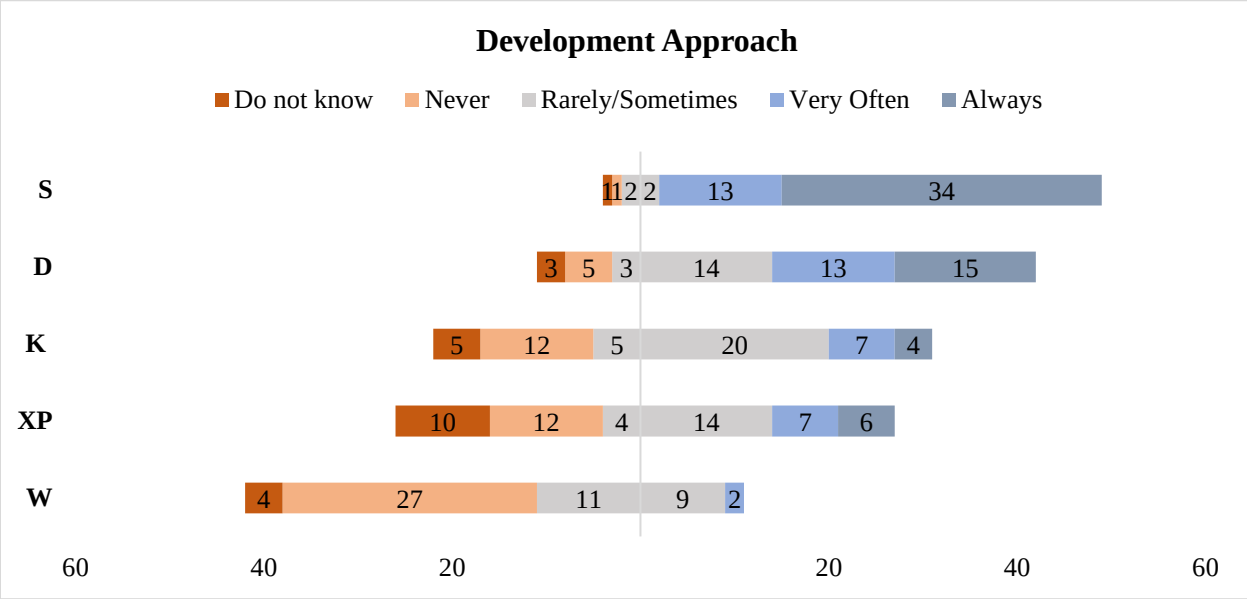


Figure 17: Frequency of Development Approaches

4.4.2. Effort estimation techniques and measurement unit

This part of the questionnaire was prepared to identify the effort estimation techniques that participants are using. Specifically, two survey questions were formulated for the first research questions which were: effort estimation techniques, and measurement units used for estimation. Here, the estimation techniques identified in the literature review (Chapter 2) were added along with a Likert scale. The descriptive statistical analysis of responses after removing "*Do not know it*" is presented in the tables below. Followed by diverging stacked bar chart for easier visibility

and interpretation of the results. For this "Rarely Use" and "Sometimes" are added up together and created a single group.

a. Effort estimation techniques

First glance, results in Table 13 shows that the most frequently used estimation technique is *story point* as it has the highest mean value (4.520), followed by, *Planning Poker* (3.000), and *Expert Estimation Method* (2.733). Whereas, the least used techniques are *Swimlane Sizing* (1.400), *The bucket System* (1.474), and others. Moreover, one participant stated that "*the organization uses COCOMO for estimation*". Figure 18 shows the same results.

Table 13: Descriptive statistical results of effort estimation techniques.

Groups	Count	Sum	Mean	SD	Variance	Range	Statistic	Sig.
Dot Voting (DV)	37	64	1.730	1.146	1.314	1 - 5	0.806	0
Expert Estimation (XPE)	45	123	2.733	1.321	1.745	1 - 5	0.914	0.001
Planning Poker (PP)	50	150	3.000	1.355	1.837	1 - 5	0.904	0
Story Point (SP)	50	226	4.520	0.814	0.663	1 - 5	0.604	0
Swimlane Sizing (SS)	35	49	1.400	1.006	1.012	1 - 5	0.681	0
Team Estimation Game (TEG)	39	79	2.026	1.386	1.920	1 - 5	0.819	0
The bucket System (BS)	38	56	1.474	0.922	0.851	1 - 5	0.775	0
Use Case Point (UCP)	41	83	2.024	1.275	1.624	1 - 5	0.854	0

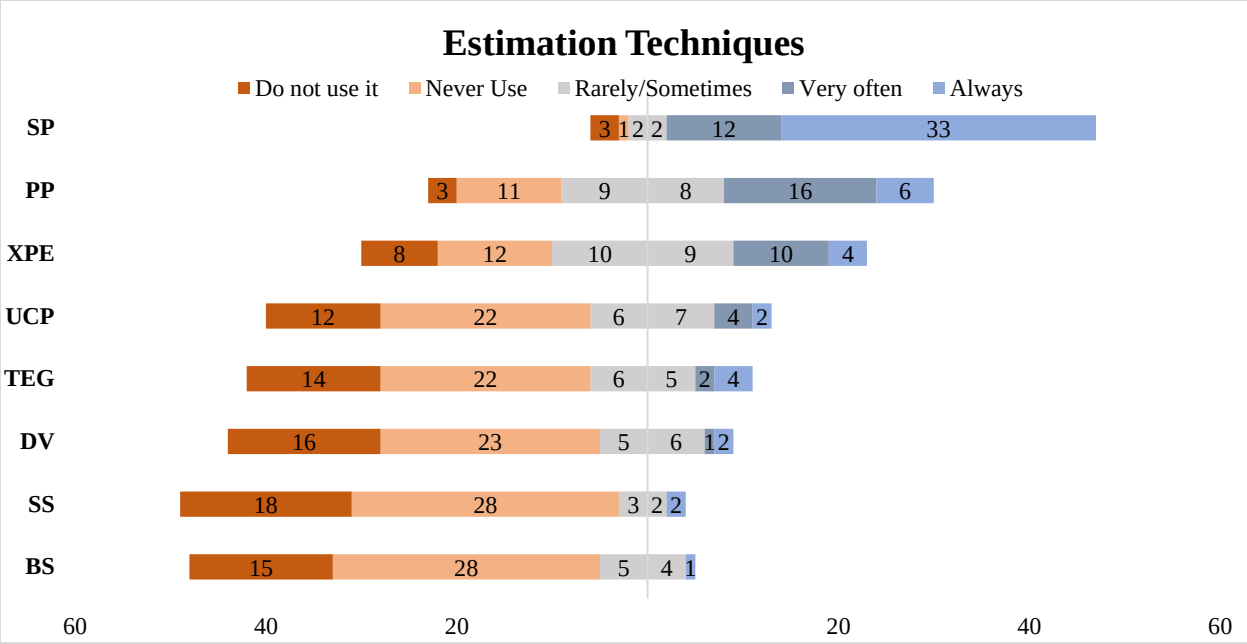


Figure 18: Frequency of estimation techniques

b. Measurement unit

Table 14 and Figure 19 shows the most used measurement unit results. As *Fibonacci Sequence* (4.420) has the highest mean value it is the major measurement unit, followed by *ideal day* (2.524), and *T-shirt size, and Dog size* (1.513).

Table 14: Descriptive statistical result of the measurement unit

Groups	Count	Sum	Mean	SD	Variance	Range	Statistic	Sig.
Fibonacci Sequence (FS)	50	221	4.420	1.247	1.555	1 - 5	0.573	.000
Ideal days (ID)	42	106	2.524	1.383	1.914	1 - 5	0.899	.000
T-shirt size, Dog size (TS)	39	59	1.513	1.097	1.204	1 - 5	0.719	.000

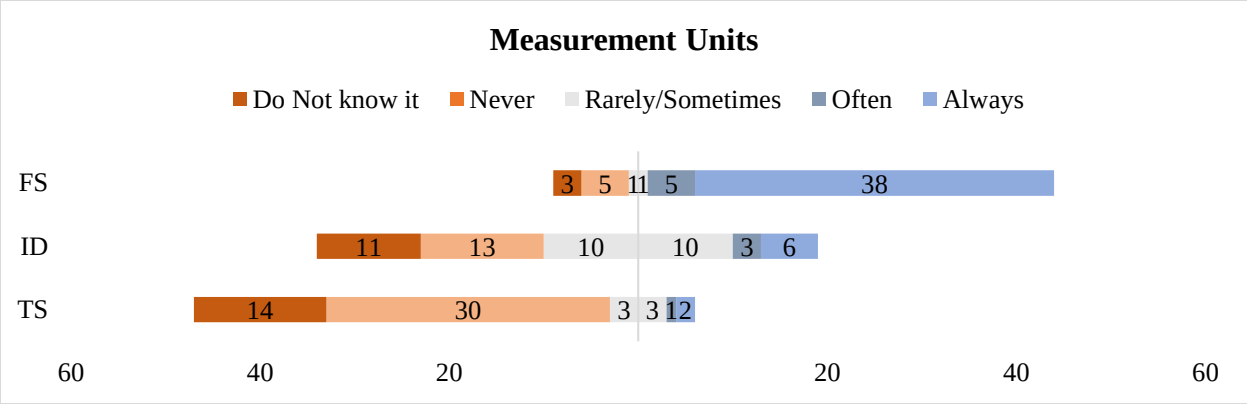


Figure 19: Frequency of Measurement units

4.4.3. Benefits

To understand the benefits of effort estimation techniques in agile software development two questions were prepared. First multiple-choice questions with six categories followed by, an open question. In what follows are descriptive statistical analysis and diverging bar chart. As there were 0 responses for "I do not know", therefore it is not included in the diverging bar chart.

a. Benefits

Table 15 shows the result of estimation benefits. The overall result shows that all groups are equally important whereas, the most important according to the highest mean values are *Drive the team to complete the project successfully* (4.38) and *Identify the resources and project scope* (4.36). Figure 20 shows the same result as a descriptive analysis result. Only one respondent disagreed with the statement "*Helps to identify important issues earlier*"

Table 15: Descriptive statistical result of estimation benefits

Groups	Count	Sum	Mean	SD	Variance	Range	Statistic	Sig.
Drive the team to complete the project successfully	53	232	4.38	0.69	0.47	2 - 5	0.749	.000
Identify the resources and project scope	53	231	4.36	0.62	0.39	2 - 5	0.711	.000
Helps to identify important issues earlier	53	221	4.17	0.83	0.68	1 - 5	0.77	.000
Monitors project progress	53	225	4.25	0.65	0.42	2 - 5	0.752	.000
To create transparency	53	223	4.21	0.66	0.44	3 - 5	0.786	.000
To gain accuracy	53	216	4.08	0.78	0.61	2 - 5	0.834	.000

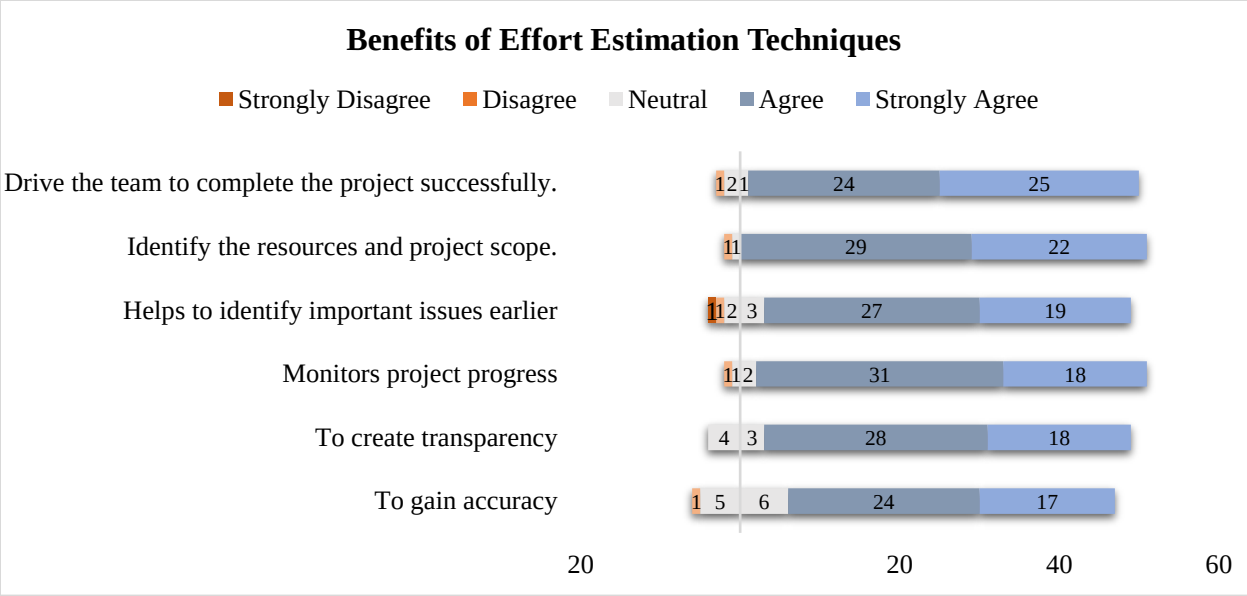


Figure 20: Frequency of estimation benefits

b. Other benefits

Regarding the other benefits, an open question was formulated in which a total of 11 responses were received out of 53 respondents. The responses were grouped based on the resemblance to find the frequency of the result. Table 16 shows the result of other benefits received.

Table 16: Other benefits according to respondents

#	Benefits	Frequency
1.	Quick and timely delivery	4
2.	Provide a sense of teamwork	3
3.	Increase adaptability of the team for accepting new feature	1
4.	Remove conflict among development and management teams as they agree on specific estimation.	1
4.	Easier forecast	1
5.	Cross-Functional team	1

4.4.4. Inaccuracy

To get the understanding of inaccuracy in estimation 5 categories were prepared: *Requirement Related Issues*, *Project Management Related Issues*, *Team Related Issues*, *Over Optimism*, and *Others*. Additionally, one open-ended question was added at the end to collect other kinds of answers. In what follows, is analysis result and diverging bar chart for the easier representation of

the responses. As mentioned before, "I do not know" responses are discarded however, it is included in the Frequency and percentage of results presented in Appendix B.

a. Requirement Related Issues

The category representing requirements-related issues contains 4 statements. Table 17 shows that most of the respondents agree with the statements that were presented. Whereas, the most reported issue is *Complexity and Uncertainty* as it has the highest mean (4.250) value. Which is then followed by, *Missing and changing requirements* (4.058), *Overlooking non-functional requirements* (4.00), and *Poor user stories* (3.981).

Table 17: Descriptive analysis of requirement related issue

Requirement Related Issue	Count	Sum	Mean	SD	Variance	Range	Statistic	Sig.
Complexity and Uncertainty	52	221	4.250	0.711	0.505	2- 5	0.688	.000
Missing and changing requirements	52	211	4.058	0.916	0.840	1- 5	0.737	.000
Overlooking non-functional requirements	50	200	4.000	0.833	0.694	2- 5	0.749	.000
Poor user stories	52	207	3.981	1.000	1.000	1- 5	0.802	.000

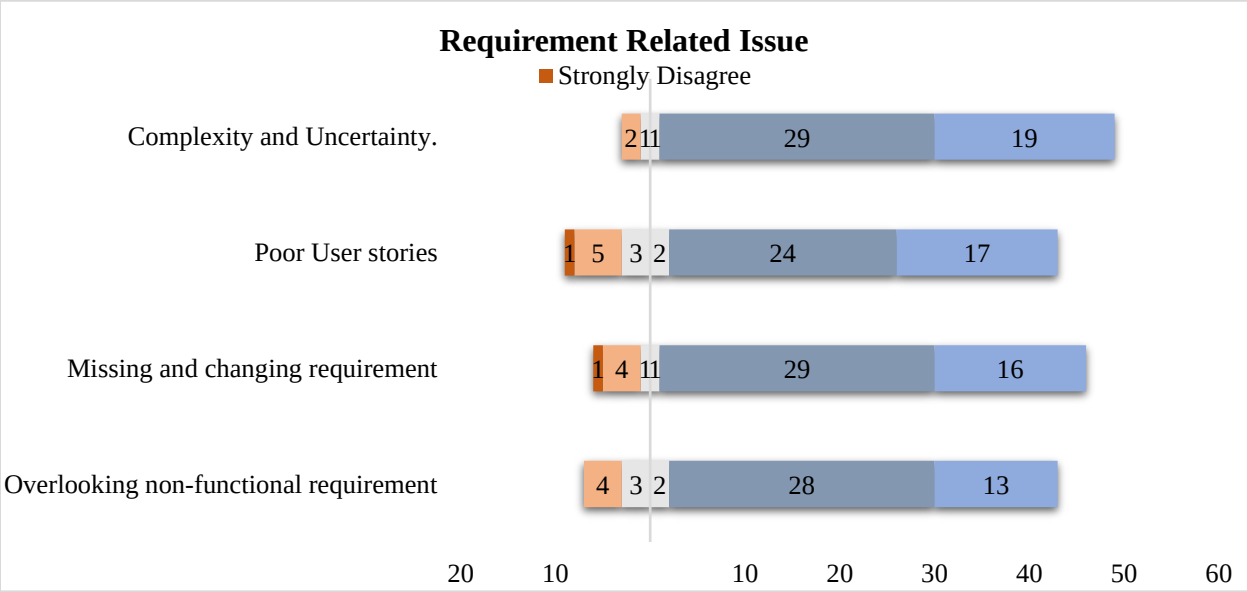


Figure 21: Frequency of requirement related issues

b. Project Management Related Issues

The category representing project management related issues contains 4 statements Table 18 shows that most of the respondents agree with the statements that were presented. The most reported issues based on the highest mean values are the *Poor change control* (3.860), *Scope creeps* (3.796), *Unstructured group estimation process* (3.750) and, *Scrum Master not guiding the team* (3.547). Therefore, the respondents' opinion reveals that not managing such kinds of issues could lead to inaccurate estimates.

Table 18: Descriptive analysis of project management related issue

Project Management Related Issue	Count	Sum	Mean	SD	Variance	Range	Statistic	Sig.
Poor change control	50	193	3.860	0.783	0.613	1 – 5	0.732	.000
Scope creep	49	186	3.796	0.935	0.874	1 – 5	0.817	.000
Scrum Master not guiding the team	53	188	3.547	1.102	1.214	1 – 5	0.892	.000
Unstructured group estimation process	52	195	3.750	1.046	1.093	1 – 5	0.848	.000

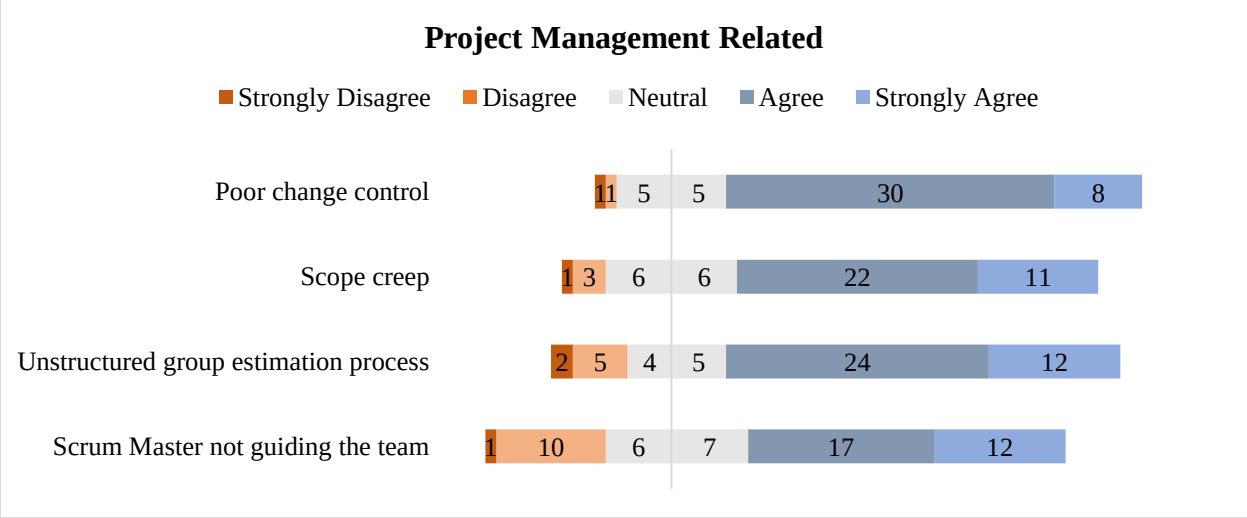


Figure 22: Frequency of project management related issues

c. Team Related Issues

The category of team-related issues contains 6 statements. Table 19 shows the result of responses. The most reported issues based on the highest mean value are the *knowledge-sharing problem in the team* (3.962), followed by, *Unskilled team members* (3.943), *Pressure of timeline*, and

Inexperience (3.698). Whereas, issues having the least mean values are *distributed teams* (3.077) and *Dominant Personalities* (3.434).

Table 19: Descriptive analysis of the team-related issue

Team Related Issue	Count	Sum	Mean	SD	Variance	Range	Statistic	Sig.
Distributed teams	52	160	3.077	1.135	1.288	1 – 5	0.91	0.001
Dominant Personalities	53	182	3.434	0.930	0.866	1 – 5	0.88	.000
Inexperience	53	196	3.698	1.085	1.176	1 – 5	0.839	.000
Knowledge sharing problem in team	53	210	3.962	0.980	0.960	1 – 5	0.786	.000
Pressure of timeline	53	196	3.698	0.972	0.946	1 – 5	0.772	.000
Unskilled team members	53	209	3.943	0.886	0.785	1 – 5	0.812	.000

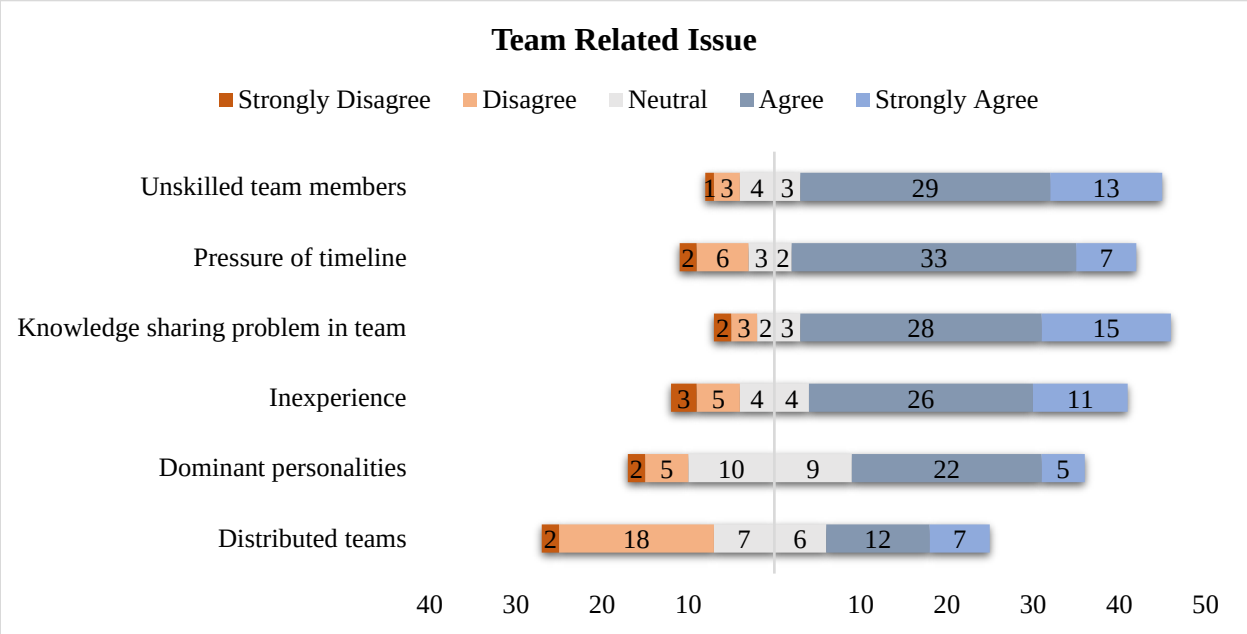


Figure 23: Frequency of team related issues

d. Over-Optimism

Table 20 presents the statistical result of over-optimism. Here, the top issue based on the highest mean value is *Considering the best-case scenario* (3.96) followed by *Purposely underestimating to obtain work* (3.538).

Table 20: Descriptive analysis of Over-Optimism

Over Optimism	Count	Sum	Mean	SD	Variance	Range	Statistic	Sig.
Considering best case scenario	53	210	3.96	0.759	0.575	2 – 5	0.798	.000
Purposely underestimating to obtain work	52	184	3.538	1.056	1.116	1 – 5	0.82	.000

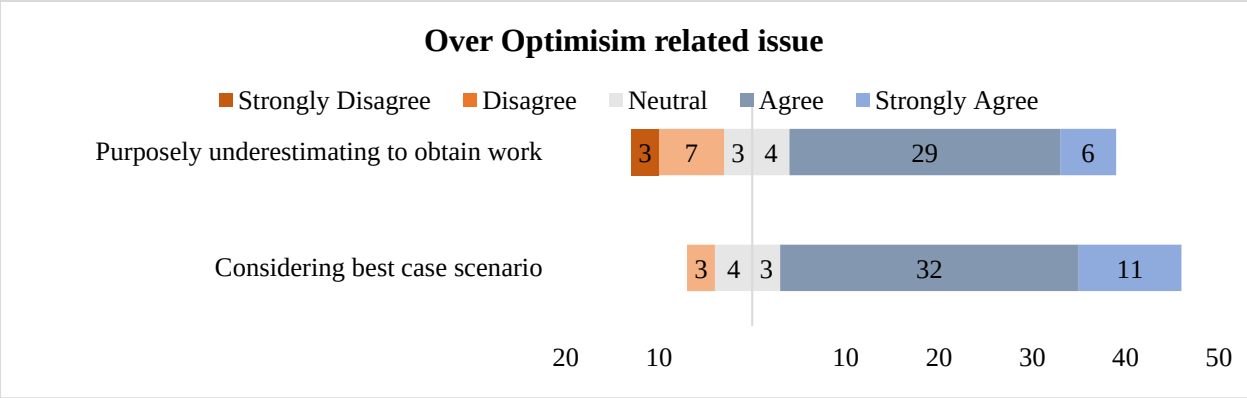


Figure 24: Frequency of overoptimism related issues

e. Others

The category about other issues contains 4 statements. Table 21 shows the statistical analysis result of other types of reasons related to inaccurate estimates. The most-reported statement based on the highest mean value is *Ignoring testing effort* (3.94) followed by, *Lack of formal estimation process* (3.68), *Insufficient customer involvement during the estimation process* (3.47), and *Hardware* (3.32).

Table 21: Descriptive analysis of others

Others	Count	Sum	Mean	SD	Variance	Range	Statistic	Sig.
Hardware	53	176	3.32	0.98	0.95	1 - 5	0.902	.000
Ignoring testing effort	53	209	3.94	0.91	0.82	1 - 5	0.753	.000
Insufficient customer involvement during estimation process	53	184	3.47	1.03	1.06	1 - 5	0.871	.000
Lack of formal estimation process	53	195	3.68	1.01	1.03	1 - 5	0.836	.000

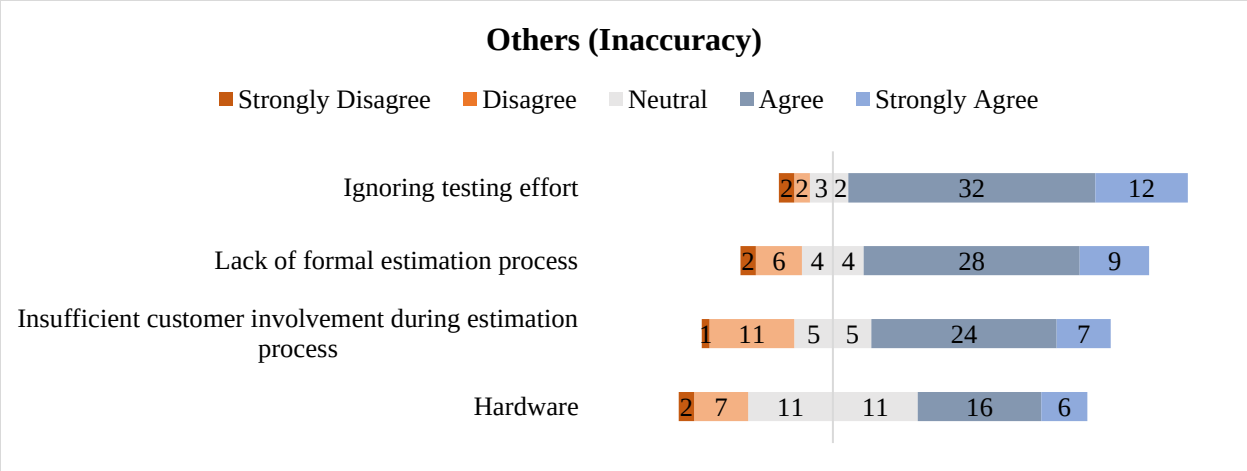


Figure 25: Frequency of other related issues

f. Other challenges apart from the above mention

Apart from the above-mentioned inaccurate estimate issue, an open question was included, in order to understand if respondents have anything else to say about other challenges. In this case, a total of 3 responses were received from 53 participants. Table 22 presents the results of other challenges.

Table 22: Other challenges according to respondents

#	Challenges
1.	Estimates are often used by sales/project managers in a way that a customer gets high expectations. An estimate of 1 month may mean it takes 3 months to deliver because of parallelism etc. If the customer only sees 1 month, they get upset if it takes 3 in practice.
2.	Not understanding the impact areas of changes being done on user stories can lead to inaccurate estimates. Not understanding the impact of third-party integration or third-party dependencies also causes the issue.
3.	Accurate demand for a specific project in a given timeline.

Once the descriptive analysis of the data was completed, a comparative analysis of estimation techniques based on the development approach was done. Then, another comparative analysis of benefits and inaccuracy based on the experience was done. In what follows, the result of descriptive statistical analysis is presented.

4.4.5. Effort estimation techniques by development approach

Apart from the previous analysis, this section contains further analysis 1st research questions based on the development approaches. This analysis is done to identify the relation between the software

development approach and the use of different estimation techniques. For this analysis SQL (Standard Query Language was used).

Steps followed for this analysis

Step1: Initially, all the possible combinations of 5 development approaches were created. As a result, 31 groups were created discarding the repeated combination.

Table 23: Combination of the development approach

Combination of Development Approaches				
D	D X	D X K	D X K S	D X K S W
X	D K	D X S	D X K W	
K	D S	D X W	D X S W	
S	D W	D K S	D K S W	
W	X K	D K W	X K S W	
	X S	D S W		
	X W	X K S		
	K S	X K W		
	K W	X S W		
	S W	K S W		

Step 2: Once it was done, valid responses of development approaches were taken into consideration by filtering out 0 and 1 responses (i.e. 0- I do not know it and 1- Never Used it), and the following result was received.

Table 24: Combination and frequency of development approaches

Combination and frequency of development approaches							
Approach	**Count**	**Approach**	**Count**	**Approach**	**Count**	**Approach**	**Count**
D	1	D X	0	D X K	0	D X K S	14
X	0	D K	0	D X S	1	D X K W	0
K	1	D S	7	D X W	0	D X S W	4
S	3	D W	0	D K S	3	D K S W	4
W	0	X K	0	D K W	0	X K S W	1
			0	D S W	0	D X K S W	11
			0	X K S	0		
			1	X K W	0		
			0	X S W	0		
			1	K S W	1		

Step 3: Now, the combination having 0 counts in step 2 were discarded and a total of 14 combinations were presented based on the applicable count.

Table 25: Frequency of development approaches

Approach	Count
D	1
K	1
S	3
D S	7
K S	1
S W	1
D X S	1
D K S	3
K S W	1
D X K S	14
D X S W	4
D K S W	4
X K S W	1
D X K S W	11
Total	53

Step 4: After that, one can compare the used combination of development approaches with estimation techniques. For instance, take into consideration the following combination (D X K S W) and filter out 0 and 1 responses (0- I do not know and 1- Never Use) responses of swimline sizing, as a result, there are 3 responses.

Table 26: Example of development approaches and Swimlane sizing

DevOps	Extreme Programming	Kanban	Scrum	Waterfall	Swimlane Sizing
3	5	5	5	3	5
5	3	3	4	2	3
5	3	3	4	2	2

Table 27 shows the result of all 14 combinations of development approaches and estimation techniques. The first column contains the frequency (F) followed by, 14 combination of development approaches, Estimation Techniques [*Swimlane sizing (SS), The bucket system (BS),*

Dot voting (DV), Team estimation game (TEG), Use case point (UCP), Expert Estimation (EE), Planning Poker (PP), and Story point (SP)], finally, Total and Percentage (%).

The first review of the result in Table 27 shows that the *Story point* (26.1%) is the most used estimation technique as it has the highest percentage, followed by, *Planning Poker* (20.7%), *Expert Estimation* (17.6%), and *Use Case Point* (10.1%). Whereas the least used estimation techniques are *Team Estimation Game* (9.0%) followed by, *Dot Voting* (7.4%), *The bucket System* (5.3%), and *Swimlane Sizing* (3.7%).

Table 27: Overview of estimation techniques and software development approaches

		Estimation Techniques									
F	**Approach**	**SS**	**BS**	**DV**	**TEG**	**UCP**	**EE**	**PP**	**SP**	**Total**	**%**
1	K								**1**	1	0.5%
1	KSW					1				1	0.5%
1	D							**1**		1	0.5%
1	KS				1				**1**	2	1.1%
1	DXS					1			**1**	2	1.1%
1	SW						**1**	**1**	**1**	3	1.6%
1	XKSW				1	1	**1**		**1**	4	2.1%
3	S			1	1		**2**	**2**	**3**	9	4.8%
3	DKS	1	1		1	1		**2**	**3**	9	4.8%
4	DKSW		1	2	1	2	**4**	**2**	**2**	14	7.4%
7	DS				1	1	**3**	**6**	**7**	18	9.6%
4	**DXSW**		**2**	**2**	**3**	**4**	**3**	**3**	**4**	**21**	11.2%
11	**DXKSW**	**3**	**2**	**5**	**4**	**4**	**7**	**9**	**11**	**45**	23.9%
14	**DXKS**	**3**	**4**	**4**	**4**	**4**	**12**	**13**	**14**	**58**	30.9%
53	**Total**	7	10	14	17	19	33	39	49	188	100.0%
	%	3.7%	5.3%	7.4%	9.0%	10.1%	**17.6%**	**20.7%**	**26.1%**	100%	

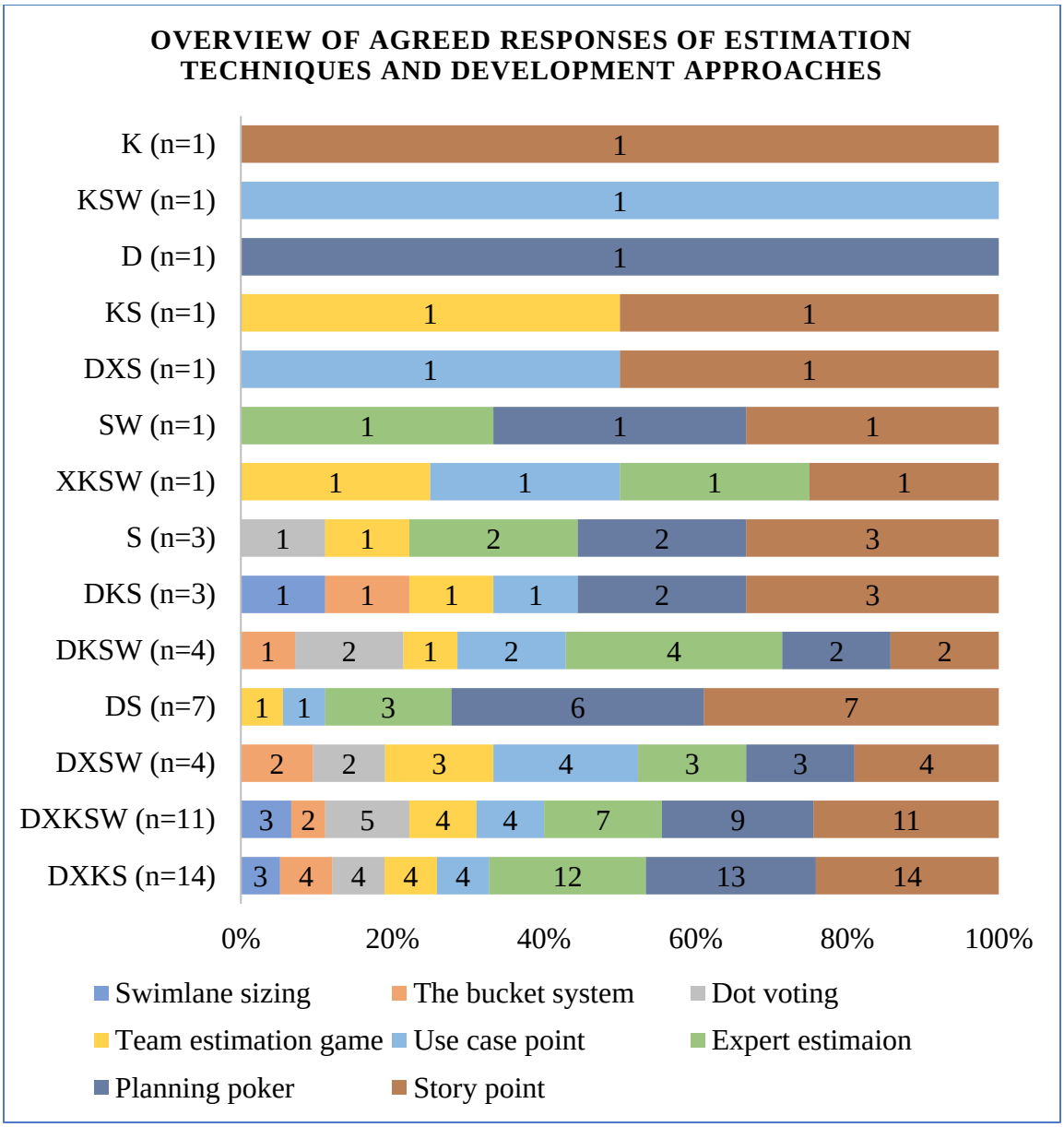


Figure 26: Overview of effort estimation techniques and development approaches

4.4.6. Benefits and inaccuracy by experience

Apart from the previous analysis, experience analysis was done to identify how practitioners having different years of experience perceive the importance of benefits and accuracy in the estimation processes. First, descriptive analysis was done followed by, Inferential Statistical Analysis.

As experience has 5 groups in total (see Table 8), some groups have fewer responses, therefore, the experience was grouped into two categories: less than 3 years and more than 3 years of

experience. Moreover, the frequency may vary because "I do not know" responses are removed for the analysis. However, it is worth noting that there were no "I do not know" responses for *Benefits* and Inaccuracy-*Others*, therefore, there are altogether 53 responses for these categories.

a. Estimation benefits

First column presents the benefits followed by, mean and Standard Deviation (std. dev) of less than 3 years and more than 3 years.

Initial review of the result in Table 28 shows that *Identify the resources and project scope* (4.50) have the highest mean value, followed by, *Drive the team to complete the project successfully* (4.35) and, *Helps to identify important issues earlier* (4.30) for less than 3 years. However, the highest mean value for more than 3 years is, *Drive the team to complete the project successfully* (4.39) followed by, *Identify the resources and project scope* and *Monitor project progress* (4.27).

Whereas, the lowest mean value for less than 3 years are for *Monitors project progress* and *To create transparency* (4.20) followed by, *To gain accuracy* (3.90). However, the lowest for more than 3 years was *To create transparency* (4.21), followed by, *To gain accuracy* (4.18) and *Helps to identify important issues earlier* (4.09).

Table 28: Statistical analysis of benefits based on experience.

Estimations benefits	**Less than 3 years (n=20)**		**More than 3 years (n=33)**	
	Mean	**Std. dev**	**Mean**	**Std. dev**
Drive the team to complete the project successfully	4.35	0.671	4.39	0.704
Identify the resources and project scope	4.50	0.607	4.27	0.626
Helps to identify important issues earlier	4.30	0.733	4.09	0.879
Monitors project progress	4.20	0.616	4.27	0.674
To create transparency	4.20	0.696	4.21	0.650
To gain accuracy	3.90	0.852	4.18	0.727

b. Inaccuracy in estimation

First column in Table 29 presents the inaccuracy list followed by, number (n) mean and Standard Deviation (std. dev) of less than 3 years and more than 3 years.

The first review of the result for *requirement related issues* shows that all the issues for more than 3 years have mean values greater than 4. Whereas, it is less than four for less than 3 years of experience. However, the most reported issue by less than 3 years of experience is *Complexity and Uncertainty* (4.16), followed by, *Overlooking non-functional requirements* (3.88) and *Missing and changing requirements* (3.84). Whereas, for more than 3 years are *Complexity and Uncertainty* (4.30) followed by, *Poor user stories* (4.24) and *Missing and changing requirements* (4.18).

Regarding the less reported issue for less than 3 years is for *Poor user stories* (3.53), however, for more than 3 years is *Overlooking non-functional requirements* (4.06). Moreover, the most reported issue by both groups is *Complexity and Uncertainty* (Less=4.16 and More=4.30).

For *Project management-related issues*, the most reported issue for less than 3 years is the *Unstructured group estimation process* (3.85), followed by, *Poor change control* (3.58) and *Scope creep* (3.58). Whereas, for more than 3 years are *Poor change control* (4.03) and *Scope Creep*(3.93). However, the least reported issue for less than 3 years is *Scrum master not guiding the team* (3.20) and for more than 3 years is an *Unstructured group estimation process* (3.69).

Similarly, the most reported *team-related issues* for less than 3 years are the *Knowledge sharing problem in the team* (4.00) followed by, *Unskilled team members* (3.90), *Pressure of timeline* (3.55), and *Inexperience* (3.50). Whereas, for more than 3 years are *Unskilled team members* (3.97), followed by, *Knowledge sharing problem in the team* (3.94), *Inexperience* (3.82), and *Pressure of timeline* (3.79). However, the least reported by both groups are *Dominant Personalities* (Less= 3.35, More=3.48) and *Distributed teams* (Less=3.42 and More=2.88).

Additionally, the most reported *over-optimism related issue* (OORI) by groups is *Considering the best case scenario* (Less=3.90 and More=4.00). However, the least reported for both is *Purposely underestimating to obtain work* (Less=3.50 and More=3.56).

Finally, the most reported *other related issues* by less than 3 years of experience are *Ignoring testing effort* (3.65), followed by, *Lack of formal estimation process* (3.35) and *Hardware* (3.30). Whereas, for more than 3 years of experience are *Ignoring testing effort* (4.12) followed by, *Lack of formal estimation process* (3.88) and *Insufficient customer involvement during estimation process* (3.67). However, the least reported issue for less than 3 years is *Insufficient customer involvement during the estimation process* (3.15) and for more than 3 years is *Hardware* (3.33).

Table 29: Statistical analysis of inaccuracy based on experience

Inaccuracy in estimation		Less than 3 years			More than 3 years		
		N	Mean	Std. dev	N	Mean	Std. dev
Requirement	Complexity and Uncertainty	19	4.16	0.765	33	4.30	0.684
	Missing and changing requirements	19	3.84	1.119	33	4.18	0.769
	Overlooking non-functional requirements	17	3.88	0.857	33	4.06	0.827
	Poor user stories	19	3.53	1.172	33	4.24	0.792
Project	Poor change control	19	3.58	0.902	31	4.03	0.657
	Scope creep	19	3.58	1.071	30	3.93	0.828
	Scrum master not guiding the team	20	3.20	1.056	33	3.76	1.091
	Unstructured group estimation process	20	3.85	0.988	32	3.69	1.091
Team	Distributed teams	19	3.42	0.902	33	2.88	1.219
	Dominant Personalities	20	3.35	0.933	33	3.48	0.939
	Inexperience	20	3.50	1.100	33	3.82	1.074
	Knowledge-sharing problem in team	20	4.00	1.026	33	3.94	0.966
	Pressure of timeline	20	3.55	1.050	33	3.79	0.927
	Unskilled team members	20	3.90	0.718	33	3.97	0.984
OOR I	Considering best case scenario	20	3.90	0.718	33	4.00	0.803
	Purposely underestimating to obtain work	20	3.50	1.147	32	3.56	1.014
Others	Hardware	20	3.30	1.031	33	3.33	0.957
	Ignoring testing effort	20	3.65	1.089	33	4.12	0.740
	Insufficient customer involvement during estimation process	20	3.15	1.089	33	3.67	0.957
	Lack of formal estimation process	20	3.35	1.089	33	3.88	0.927

As there are some differences among the perception of the practitioners regarding the benefits and inaccuracies, it could be interesting to know if those differences are significant based on the experience. In what follows the hypothesis test of benefits and inaccuracy is tested.

4.5. Inferential Statistical Analysis

Inferential statistics are chosen for this study to draw the conclusion from the random sample and make inferences about the population in general (Bettany-Saltikov & Whittaker, 2014). As

presented in section 4.3, it is known that data are not normally distributed. In case of the not normal distribution, one should use the Mann-Whitney test also known as the Wilcoxon rank-sum test (Dexter, 2013). However, given that the sample size is smaller (> 30 or 40), for this study, both parametric (Independent samples t-test) and non-parametric (Mann-Whitney U test) tests are used to find out the significance of the differences.

For this analysis, the following hypothesis was created for both benefits and inaccuracy.

H0: There is no significant difference for estimation benefits and inaccuracy reported by both groups (Less than 3 and more than 3 years of experience)

H1: There is a significant difference for estimation benefits reported by both groups (Less than 3 and more than 3 years of experience)

4.5.1. Parametric (Independent samples t-test)

The alpha value for this test was selected to be $p \leq 0.05$, the first column contains the Estimation (Benefits and Inaccuracy), followed by, Levene's t-test result (Equal variances assumed "=" and Equal variances not assumed "≠" along with F and Sig.), and t-test for equality of Means (t (t Stat), df (degree of freedom), sig. (2 tailed) (sig.2t), Mean Difference(μ.Diff), and Standard Deviation Error Difference (SD. ED)). At This Point, the statically significant result is distinguished by "*" in the table. And Levene's p-value less than alpha value by "**".

Moreover, significant is evaluated based on Levene's t-test, if Levene's test p-value (sig.) is less than or equal to .05 the button row also known as *equal variances not assumed* is selected for further analysis. Whereas, if greater than 0.05 the top row *equal variances assumed* is selected for result interpretation. Based on this result the conclusion is made for either to accept or reject the null hypothesis *on t-test for equality of Means.*

a. Benefits of estimation

The contrast hypothesis test results in Table 30 shows that there is no significant difference based on Levene's test p-values as all the p-values are greater than 0.05. Therefore top row *equal variances assumed* are considered for analysis. As a result, there is no significant difference in the result.

Table 30: Contrast hypothesis of estimation benefits (Independent Sample t-test)

Estimations benefits	Levene's Test for Equality of Variances			t-test for Equality of Means				
		F	Sig.	t	df	Sig. 2t	μ.Diff	SD. ED
Drive the team to complete the project successfully	=	0.001	0.978	-0.224	51	0.824	-0.044	0.196
	≠			-0.227	41.797	0.822	-0.044	0.194
Identify the resources and project scope	=	0.470	0.496	1.296	51	0.201	0.227	0.175
	≠			1.306	41.228	0.199	0.227	0.174
Helps to identify important issues earlier	=	0.234	0.631	0.892	51	0.377	0.209	0.235
	≠			0.933	45.879	0.356	0.209	0.224
Monitors project progress	=	0.192	0.663	-0.393	51	0.696	-0.073	0.185
	≠			-0.402	43.132	0.690	-0.073	0.181
To create transparency	=	0.104	0.748	-0.064	51	0.949	-0.012	0.189
	≠			-0.063	38.079	0.950	-0.012	0.192
To gain accuracy	=	1.839	0.181	-1.282	51	0.206	-0.282	0.220
	≠			-1.232	35.366	0.226	-0.282	0.229

b. Inaccuracy in estimation

The result in Table 31 shows that based on the Levene's test, three factors have a p-value less than the alpha value ($p \leq 0.05$): Requirement related issue [*Poor user stories*], Project management related issue [*Poor change control*] and, others [*Ignoring testing effort*]. Therefore, the second row (equal variances not assumed) of the result is considered, whereas, rest are evaluated based on the first row (equal variances assumed).

As a result, there is only one significant difference in the inaccuracy reported by both groups which is: Requirement related issue- *Poor user stories* (t (27.625)= -2.370, p=0.025)

Table 31: Contrast hypothesis of inaccuracy in estimation (Independent Sample t-test)

Inaccuracy		Levene's Test for Equality of Variances			t-test for Equality of Means				
			F	Sig.	t	df	Sig. 2t	*μ.*Diff	SD. ED
Requirement	Complexity and Uncertainty	=	0.018	0.895	-0.706	50	0.484	-0.145	0.206
		≠			-0.685	34.306	0.498	-0.145	0.212
	Missing and changing requirements	=	1.582	0.214	-1.296	50	0.201	-0.340	0.262
		≠			-1.174	27.955	0.250	-0.340	0.289

	Overlooking non-functional requirements	=	0.460	0.501	-0.713	48	0.479	-0.178	0.250
		≠			-0.705	31.398	0.486	-0.178	0.253
	Poor user stories	=	5.515	0.023**	-2.627	50	0.011	-0.716	0.273
		≠			-2.370	27.625	0.025*	-0.716	0.302
PM-related issues	Poor change control	=	4.228	0.045**	-2.052	48	0.046	-0.453	0.221
		≠			-1.903	29.749	0.067	-0.453	0.238
	Scope creep	=	1.772	0.190	-1.302	47	0.199	-0.354	0.272
		≠			-1.229	31.412	0.228	-0.354	0.288
	Scrum master not guiding the team	=	0.201	0.655	-1.825	51	0.074	-0.558	0.305
		≠			-1.840	41.256	0.073	-0.558	0.303
	Unstructured group estimation process	=	1.769	0.190	0.541	50	0.591	0.163	0.300
		≠			0.554	43.495	0.582	0.163	0.293
Team related Issues	Distributed teams	=	2.630	0.111	1.689	50	0.097	0.542	0.321
		≠			1.830	46.710	0.074	0.542	0.296
	Dominant Personalities	=	0.025	0.876	-0.508	51	0.614	-0.135	0.266
		≠			-0.509	40.444	0.614	-0.135	0.265
	Inexperience	=	0.043	0.837	-1.036	51	0.305	-0.318	0.307
		≠			-1.030	39.466	0.309	-0.318	0.309
	Knowledge-sharing problem in team	=	0.037	0.849	0.216	51	0.830	0.061	0.280
		≠			0.213	38.340	0.832	0.061	0.284
	Pressure of timeline	=	1.108	0.297	-0.861	51	0.393	-0.238	0.276
		≠			-0.835	36.379	0.409	-0.238	0.285
	Unskilled team members	=	1.064	0.307	-0.275	51	0.784	-0.070	0.253
		≠			-0.297	49.090	0.768	-0.070	0.235
OORI	Considering the best-case scenario	=	0.022	0.883	-0.462	51	0.646	-0.100	0.217
		≠			-0.473	43.291	0.639	-0.100	0.211

	Purposely underestimating to obtain work	=	0.187	0.668	-0.206	50	0.838	-0.063	0.304
		≠			-0.200	36.723	0.843	-0.063	0.313
Others	Hardware	=	0.151	0.699	-0.119	51	0.905	-0.033	0.279
		≠			-0.117	37.9	0.907	-0.033	0.284
	Ignoring testing effort	=	6.006	**0.018	-1.876	51	0.066	-0.471	0.251
		≠			-1.71	29.727	0.098	-0.471	0.276
	Insufficient customer involvement during estimation process	=	2.028	0.161	-1.808	51	0.077	-0.517	0.286
		≠			-1.75	36.236	0.089	-0.517	0.295
	Lack of formal estimation process	=	2.74	0.104	-1.883	51	0.065	-0.529	0.281
		≠			-1.809	35.307	0.079	-0.529	0.292

4.5.2. Non- parametric (Mann-Whitney U test)

Mann-Whitney U test (Freund & Wilson, 1993) is the non-parametric equivalent of the student t-test which is used to compare the two independently sampled distributions (Freund & Wilson, 1993).

Before analyzing the data the following assumptions were checked:

1. Assumption #1: The dependent variable is ordinal (i.e. Likert scale value).
2. Assumption #2: The independent variable should consist of two categories (Less and More than 3 years of experience).
3. Assumption #3: Independent of observations (no relation between two groups).
4. Assumption #4: Distribution of score for “Less than 3” and “More than 3” should have the same shape. If so, we need to compare the median else compare mean.

However, in this study, the data has different shapes therefore mean rank is considered for analysis and result interpretation. Figure 27 shows an example of the assumption #4 test.

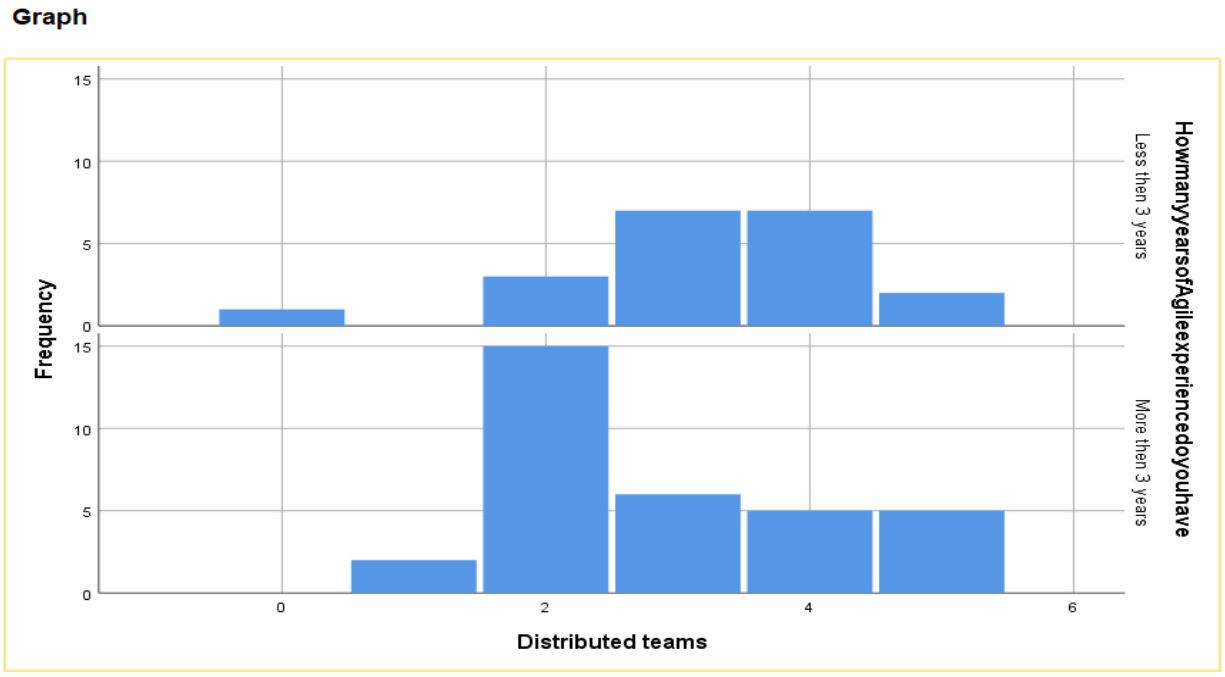


Figure 27: Example of assumption #4 for Mann- Whitney U test

In what follows, the analysis result of estimation benefits and inaccuracy based on experience is presented. Benefits and inaccuracy are dependent variables whereas, the experience is an independent variable. The following hypothesis was created:

> **H0**: There is no significant difference for estimation benefits and inaccuracy reported by both groups (Less than 3 and more than 3 years of experience)
>
> **H1:** There is a significant difference for estimation benefits reported by both groups (Less than 3 and more than 3 years of experience)

The alpha value for this test was selected to be $p \leq 0.05$, the first column contains the benefits, followed by Mann- Whitney test (N, Mean Ranks and Sum of Ranks) and Test Statistics (Mann-Whitney U, Wilcoxon W, Z, and Asymp. Sig. (2-tailed)). Statistically, significant results are marked by "*" sign.

a. Benefits of estimation

The review of the result in Table 32 shows that there is not much difference between the mean ranks of both groups (less than and more than 3 years of experience). The Mann-Whitney test

statistics (p-value) also show no significant differences among the benefits reported by both groups as the p-value is greater than alpha value.

Table 32: Contrast hypothesis of estimation benefits (Mann-Whitney Test)

Estimations benefits	**Mann-Whitney Test**			**Test Statistics**			
	N	**Mean Ranks**	**Sum of Ranks**	**Mann-Whitney U**	**Wilcoxon W**	**Z**	**Asymp. Sig. (2-tailed)**
Drive the team to complete the project successfully	20	26.18	523.5	313.5	523.5	-0.338	0.735
	33	27.5	907.5				
Identify the resources and project scope	20	30.28	605.5	264.5	825.5	-1.374	0.169
	33	25.02	825.5				
Helps to identify important issues earlier	20	28.95	579	291	852	-0.79	0.43
	33	25.82	852				
Monitors project progress	20	25.65	513	303	513	-0.568	0.57
	33	27.82	918				
To create transparency	20	26.93	538.5	328.5	538.5	-0.031	0.976
	33	27.05	892.5				
To gain accuracy	20	23.65	473	263	473	-1.321	0.186
	33	29.03	958				

b. Inaccuracy in estimation

The review of the result in Table 33 shows that there are some differences between the mean ranks of both groups (less than and more than 3 years of experience) among the inaccuracy, i.e. **Requirement related issue**- *Poor use story* and **Project Management Related Issue**- *Poor Control chang* and others. However, there are two significant results in particular, as the Mann-Whitney test statistics (p-value) is less than the alpha value for

Requirement related issue- *Poor use story* (U= 199.5, p=0.02)

Project Management Related Issue- *Poor Control chang* (U=204, p=0.04)

Table 33: Contrast hypothesis of inaccuracy in estimation (Mann-Whitney Test)

Inaccuracy	**Mann-Whitney Test**			**Test Statistics**			
	N	**Mean Ranks**	**Sum of Ranks**	**Mann-Whitney U**	**Wilcoxon W**	**Z**	**Asymp. Sig. (2-tailed)**

Requirement Related	Complexity and Uncertainty	19	24.84	472.00	282	472	-0.679	0.497
		33	27.45	906.00				
	Missing and changing requirements	19	24.05	457.00	267	457	-0.99	0.322
		33	27.91	921.00				
	Overlooking non-functional requirements	17	23.21	394.50	241.5	394.5	-0.89	0.374
		33	26.68	880.50				
	Poor user stories	19	20.50	389.50	199.5	389.5	-2.329	0.02*
		33	29.95	988.50				
Project M related	Poor change control	19	20.74	394.00	204	394	-2.058	0.04*
		31	28.42	881.00				
	Scope creep	19	22.55	428.50	238.5	428.5	-1.015	0.31
		30	26.55	796.50				
	Scrum master not guiding the team	20	22.30	446.00	236	446	-1.785	0.074
		33	29.85	985.00				
	Unstructured group estimation process	20	27.93	558.50	291.5	819.5	-0.57	0.568
		32	25.61	819.50				
Team Related Issues	Distributed teams	19	31.47	598.00	219	780	-1.864	0.062
		33	23.64	780.00				
	Dominant Personalities	20	25.95	519.00	309	519	-0.411	0.681
		33	27.64	912.00				
	Inexperience	20	24.03	480.50	270.5	480.5	-1.171	0.241
		33	28.80	950.50				
	Knowledge sharing problem in team	20	27.95	559.00	311	872	-0.383	0.702
		33	26.42	872.00				
	Pressure of timeline	20	25.00	500.00	290	500	-0.845	0.398
		33	28.21	931.00				
	Unskilled team members	20	25.30	506.00	296	506	-0.689	0.491
		33	28.03	925.00				
OORI	Considering best case scenario	20	25.10	502.00	292.0	502.0	-0.598	0.550
		32	27.38	876.00				
	Purposely underestimating to obtain work	20	26.45	529.00	319	529	-0.021	0.983
		32	26.53	849.00				
Others Related Issues	Hardware	20	27.03	540.5	329.5	890.5	-0.01	0.992
		33	26.98	890.5				
	Ignoring testing effort	20	22.8	456	246	456	-1.759	0.079
		33	29.55	975				
	Insufficient customer involvement during estimation process	20	22.5	450	240	450	-1.751	0.08
		33	29.73	981				
	Lack of formal estimation process	20	22.3	446	236	446	-1.879	0.06
		33	29.85	985				

4.6. Conclusion

Once the data collection process was completed, then data filtration was done and 53 out of 62 responses were considered as valid responses. Secondly, as the responses were collected by using a Likert scale conversion of the scales to numbers was done. Thirdly, frequency and percentage were calculated followed by descriptive statistical analysis. Finally, a contrast hypothesis test was done to find out the significance of the data. For this analysis, two different tests were used independent sample t-test and Mann-Whitney Test. As a result, no statically significant result was found on the benefits of estimation. However, regarding independent sample t-test, there was one significant result for **Requirement related issue**- *Poor user stories* (t (27.625)= -2.370, p=0.025). While, for Mann-Whitney Test **Requirement related issue**- *Poor use story* (U= 199.5, p=0.02) and **Project Management Related Issue**- *Poor Control chang* (U=204, p=0.04) respectively.

Even though, Mann - Whitney Test result was considered best for this study based on the sample size <30, an independent sample t-test was carried out to find out the differences in the result and it is found that there is some difference for one inaccuracy (Project management related issue).

4.7. Movements

This section is prepared to get insights into the impact caused by #NoEstimates and #NoProject movements. To do so, each question allows choosing one of the 5 options formulated (*1. I've never heard of it; 2. I've HEARD of it and Not interested; 3. I've HEARD of it and WOULD like to learn it; 4. I've USED it before, and would NOT use it again; 5. I've USED it before, and WOULD use it again*). Also, an open question was added to the questionnaire to get an understanding of the potential benefits.

The result presented in Figure 28 leads to conclude, that 86.6% of respondents have *never heard of #NoEstimates* and 5.8% *HEARD of it and Not interested.* However, 9.6% of the participants have *heard of it and they would like to know more about it*.

Regarding the #NoProject result, 88.5% *have never heard on #NoProject* and 7.7% *have heard of it and are not interested in it*. However, only 3.8% *would like to learn more about #NoProject*.

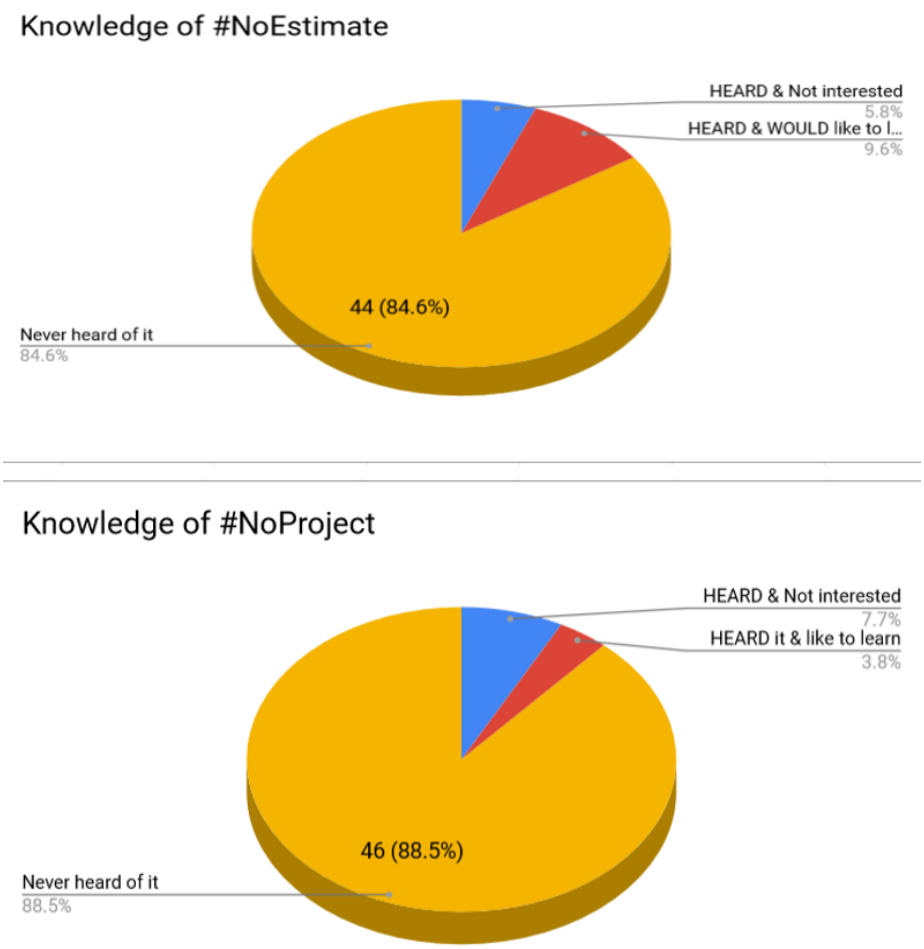


Figure 28: Knowing about #NoEstimates and #NoProject

For the open question about the *benefits or challenges related to effort estimation from #NoEstimate and #NoProject*, a total of 8 responses were received for #NoEstimate and 3 answers for #NoProject.

Among the 8 responses received for #NoEstimate, 5 of them were not relevant, e.g. “do not know”, “not used”, and “not suitable for their project” whereas only three responses presented below were suitable.

1. *“#NoEstimates is faster since it needs very less time for estimating time”*
2. *“Sometimes overshadowed the project scope”*

3. *"Proper estimations will give you the clear timelines and will provide the best results in delivery"*

Regarding the 3 responses received for #NoProject, all the responses were not relevant as participants reported that *"[#NoProject] is not suitable for the project", "never tried in the real project"* and "*NA*".

5. Discussion

This chapter presents the main findings of this study and discusses them along with previous literature. In what follows, each of the research questions will be answered and discussed sequentially.

5.1. Demographics

The demographics section was prepared to understand the background of the respondents. In particular, the demographic section contains 3 questions: Country, Gender, and Agile Experience.

A total of 62 responses were collected from the survey research, among the respondent, more than 50% were from Asia (34, 54.8%) follow by, Europe (20, 32.2%), North America (7, 11.3%) and South Africa (1, 1.6%). Moreover, more than 80.6% were male participants (50) while, females made up 17.7% (11), and one participant prefer not to say (1.6%). From the analysis results of the participant experience, it is found that more than 50% of them have more than 3 years of agile experience (35, 56.4%), whereas, 33.9% have 1-3 years of experience and 9.7% have less than a year of experience.

5.2. Software Development Project

This section was prepared to get an understanding of the background of the software project and it can provide insights about the estimation. To do so, the questionnaire included job role, team size, project length, business domain, development approach, and finally, the importance of estimation.

Among the 62 responses collected, the majority were software developers (37, 59.7%), and the majority (50%, 31) work in the team size of 6-10 people. Regarding the business domain, most of the respondents (30, 48.4%) reported that they work in e-commerce, and the project length was longer than 1 year (74.2%, 46). Among the software development approaches, Scrum and Kanban are the most frequently practice approaches in ASD. This result is aligned with the previous studies conducted by ISTQB, (2016) and the HELENA initiative (Kuhrmann et al., 2018). These studies also reported that agile development approaches are widely used but the last study highlights that mixed approaches are commonly used which is in line with the results in this thesis.

Regarding the perceived importance of estimation, the majority of respondents reported that estimation is very important in software development (47, 75.8%). And, regarding participation,

more than 85% reported that they participate in the estimation process (53, 85.5%). Therefore, these responses were considered valid responses, and further questions were asked to the participants about estimation.

5.3. Effort Estimation Techniques and Measurement unit

Questions were prepared based on a Likert scale and a list of different estimation techniques and measurement units that were identified in the literature review.

From the responses received, among the 8 estimation techniques presented, it is found that more than 60% of the respondents "Always use" *Story points* as their estimation techniques (see Figure 18) followed by, *Planning Poker* and *Expert Estimation Method*. The descriptive statistical analysis also shows the same result, based on the highest mean values, i.e. *Story Point* (4.520), *Planning Poker* (3.00), and *Expert Estimation Method* (2.733) are the most frequently used estimation techniques in ASD.

Besides this, an analysis of the estimation techniques and the development approaches (see Figure 26) show that *Story Point* (26.1%) is the most used estimation technique followed by, *Planning Poker* (20.7%), *Expert Estimation* (17.6%), *Use Case Point* (10.1%). These findings are in line with the previous studies done by (Pozenel & Hovelja, 2019; Usman et al., 2015; Usman & Britto, 2016) that have mentioned story point as the most used estimation technique. Moreover, the most used estimation measurement unit is the *Fibonacci series* as (38, 71.1%) *always use it*, followed by, *Often* 5, 9.4%, and *Sometimes* 2, 3.8%.

5.4. Estimation Benefits

Among the 6 categories of perceived benefits (see Figure 20), more than 45 respondents agree (Agree and Strongly Agree) with almost all categories except for "To gain accuracy" as only 41% respondents agree and, there were 0 responses for the "I do not Know-0" scale. However, it is worth noting that "*To gain accuracy*" is the only category having more than 20% of neutral responses, followed by, "*To create transparency*" (13.2%) and "*Helps to identify important issues earlier*" (9.4%).

Besides this, based on the highest mean value most reported estimation benefits is *Drive the team to complete the project successfully* (4.38), followed by, *Identify the resources and project scope* (4.36), and *Monitors project progress*(4.25).

These findings are in line with the previous studies done by (Rashmi Popli & Chauhan, 2013; Trendowicz & Jeffery, 2014), these studies stated that estimation helps to gain accuracy and also monitors the project progress. Moreover, the contrast hypothesis test result presented in Table 30 and Table 32 shows no significant difference in the benefits presented in this study.

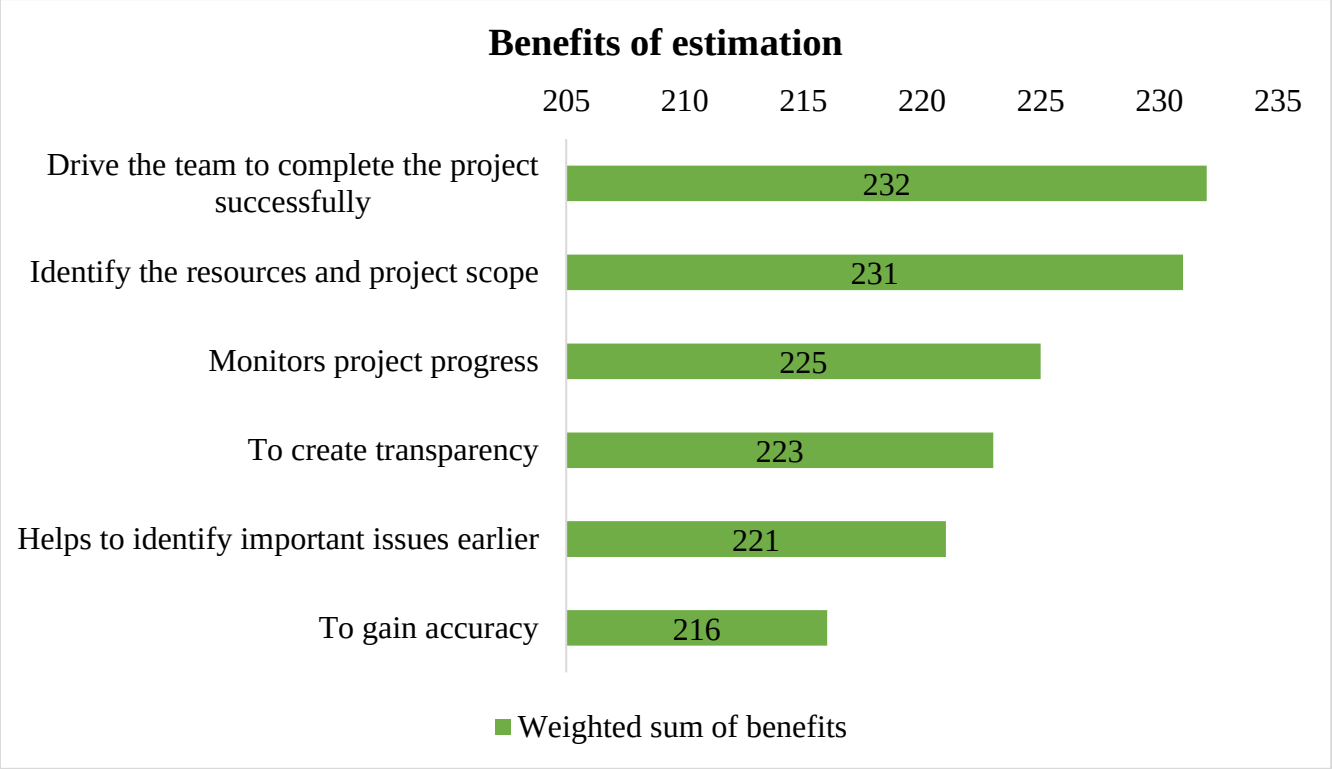


Figure 29: Benefits presented based on the weighted sum

5.5. Inaccuracy

To get the insights about the inaccuracy in the estimation, 20 factors were analyzed, essentially, these factors were divided into 5 major categories (*Requirement Related Issue*, *Project Management Related Issue*, *Team Related Issuequestions*, *Over-Optimism*, and *Others*).

Based on the descriptive statistical analysis result, most reported inaccurate estimates based on the mean values are: Requirement Related Issue-*Complexity and Uncertainty* (4.250), *Missing and changing requirements* (4.058) followed by, Team Related Issue- *Knowledge sharing problem in the team* (3.96), *Unskilled Team Member* (3.943), Over Optimism- *Considering best case scenario* (3.962), Others- *Ignoring Testing Effort* (3.94) and Project Management Related Issue- *Poor change control* (3.860).

These findings are in line with the previous studies done by (Anooja & Rajawat, 2018; Dagnino, 2013; R. Popli & Chauhan, 2014) as these studies have stated that various unstable factors of inaccuracy are a requirement, distributed teams issue, deadline pressure, flexibility among team members, hardware and others.

5.5.1. Requirement related issue

The result presented in Table 17 shows that *complexity and uncertainty* (4.250), as well as *missing and changing requirements* (4.058) are the top factors for the requirement related issues. Besides this, frequency and percentage results presented in APPENDIX B5 show that 48 participants out of 53 Agree (Agree and Strongly Agree) that *Complexity and uncertainty* and 45 Agree with *Missing and changing requirements* are the reason for inaccuracy in estimation. This result supports previous evidence where both *missing and changing requirements*, as well as *overlooking the non-functional requirements* were found to be the top reasons for inaccuracy in effort estimation (Leinonen, 2016).

Besides this, more than 3 years of experience analysis result shows that almost all factors have a mean value greater than 4, this means that respondents having more experience believe that these factors are the topmost reason for the inaccuracy in estimation. Similarly, both respondents considered that *Complexity and Uncertainty* (Less=4.16 and More=4.30) as the topmost reason for the inaccuracy in the estimation, however, *Poor user stories* (3.53) as the least reported factor for less than 3 years and *Overlooking non-functional* requirements (4.06) as the least reported factor for more than 3 years based on the least mean value among the presented factors.

Moreover, the contrast hypothesis test result based on experience shows only one significant difference in *Poor user stories* for both independent sample t-test (t (27.625)= -2.370, p=0.025) and Mann-Whitney Test (U= 199.5, p=0.02).

5.5.2. Project Management Related Issue

Based on the result in Table 18 the most reported Project management related issues are the *Poor change control* (3.860) and *Scope creep* (3.796) *Unstructured group estimation process* (3.750). Besides this, frequency and percentage result in APPENDIX B5 shows that 38 participants out of 53 Agree (Agree and Strongly Agree) that *Poor change control* and 36 agree that the *Unstructured group estimation process* as the top 2 reasons for project management related issue. Therefore,

managing these issues properly could have a positive impact on the development time and project cost, and therefore decrease the risk for estimation errors (Usman et al., 2015).

Besides this, the experience analysis result shows that 3 out of 4 inaccuracies presented in Table 29 are in favor of more than three years of experience. Therefore, the only issue that is in favor of less than 3 years of experience is the *Unstructured group estimation process*. Furthermore, the independent sample t-test result shows no significant differences among the inaccuracy reported by respondents having less and more than 3 years of experience. However, the Mann-Whitney test shows one significant difference for *Poor change control* (U=204, p=0.04).

5.5.3. Team Related Issue

Table 19 shows that the *Knowledge sharing problem in a team* (3.962), as well as, *Unskilled team members* (3.943), are the top factors for the inaccuracy in team-related issues. Besides this, 43 out of 53 respondents agree (Agree and Strongly Agree) with the statement *Knowledge sharing problem in the team,* followed by, *Pressure of timeline* (40), and *Inexperience* (36). This result is in line with the previous studies done by (Keaveney & Conboy, 2006; R. Popli & Chauhan, 2014). These studies have mentioned that lack of team experience is one of the reasons for inaccuracy in estimation. Therefore, these factors should be considered during estimation to avoid inaccuracy.

Besides this, experience analysis shows that 4 out of six factors included are in favor of more than 3 years of experience. However, only inaccuracies that are in favor of less experience are the *Knowledge sharing problem in the team* (4.00) and *Distributed team* (3.42). Moreover, the hypothesis test result shows no significant difference between the reported inaccuracy.

5.5.4. Over Optimism

Among the two reasons for over-optimism included in Table 20 shows that the topmost reason for *Considering the best case scenario* (3.96). Besides this, 43 out of 53 respondents agree (Agree and Strongly Agree) with the statement *Considering the best-case scenario*, followed by 35 on *Purposely underestimating to obtain work*. This result is in line with the previous studies by (Liskin et al., 2014; Mahnič & Hovelja, 2012) these studies presented that only considering the best-case scenario by developers can create inaccuracy in estimation.

Besides this, experience analysis shows that among the 2 overoptimism presented one in favor of less experience other for more experience. Moreover, the hypothesis test result shows no significant difference between the reported inaccuracy.

5.5.5. Others

Table 21 shows that the topmost *other related issues* for inaccuracy in estimation are *Ignoring testing effort* (3.94) and *Lack of formal estimation process* (3.68). Besides this, 44 out of 53 respondents agree (Agree and Strongly agrees) with the statement *Ignoring testing effort* and 37 *Lack of formal estimation process*. This study is in line with the previous studies done by (Keaveney & Conboy, 2006; Molokken-Ostvold & Furulund, 2007), these studies have included that testing effort and customer involvement are important for the estimation process. Doing so will reduce the inaccuracy to some extend in the estimation process.

Besides this, experience analysis shows that among the 4 factors included, all are in favor of more than 3 years of experience. However, the most reported by both groups is the *Ignoring testing effort* (Less=3.65, More=4.12). Besides, the hypothesis test result shows no significant difference between the reported inaccuracy.

5.6. Impact of #NoEstimate and #NoProject

To get the understanding of the impact caused by #NoEstimate and #NoProject and their potential benefits two types of questions were formulated: one closed-ended question (5 options added) and open-ended question. The result shows that more than 84% *have never heard of #NoEstimate* (84.6%) and #NoProject (88.5%). Only less than 10% *heard of it* and *wanted to know about #NoEstimate* (9.6%) and *#NoProject* (3.8%).

Despite that fact, 3 participants provided valid answers related to the benefits of #NoEstimate, [*1. Faster, 2. Overshadow Project Scope 3. Provide a clear timeline for delivary*]. Moreover, no valid responses were received for #NoProject.

The previous work done by (Hannay et al., 2018), have mentioned that even though the #NoEstimates movement attracts agile practitioners, it does not have much benefit-over-cost optimization that is more importantly needed in agile. However, only a few practitioners involved in this study know about the #Noestimated, even more, some of them are not interested in knowing it.

5.7. Threats to validity

The results of this study may have been impacted by the exposure of the study search, inaccuracy in study information extraction as well as the sample size of the study. In what follows, the main threats of validity as the category provided by (Wohlin et al., 2012): Construct Validity, Internal Validity, Conclusion Validity, and External Validity.

Construct validity is related to the issue caused by poor data extraction and the recording process, including the correctness and the clarification of the concept of the related studies. For this study, google scholar and the college library were used for the literature review. Some papers not accessible through the college library have been missed during the search process. To minimize this most of the time asked the supervisor for the papers. For the survey, some respondents might have misinterpreted the questionnaire, or they were confused. To ensure the correct understanding of the questionnaire, 2 rounds of pilot testing were done.

Internal validity is related to examining the selected data. Google scholar and college library were used for a literature review. However, some relevant papers could be missing. For the survey, Likert scale questions were formulated and they were validated by two supervisors. Although multiple options were added to the questionnaire, respondents might not get the answer as they want. To reduce this threat, "Other" was included at the end of all the questions.

Conclusion validity is to make sure that sensible conclusions are outlined based on data collected, sometimes the result may vary and it may lead to incorrect conclusions. Therefore, to minimize this issue survey questionnaires were revised by two supervisors, followed by, 2 rounds of pilot studies, and finally one of the supervisors validates the results and she found some differences. Hence, it was reviewed and resolved. However, further research is a need.

External validity is concerned with the generalization of the selected studies as regards the overall goal of the study. The results of this exploratory study are related to effort estimation in Agile software development. However, the sample is small, which limits the generalization of the results. Therefore, the sample size should be expanded to a larger group in order to increase the generalizability of the results. In fact, given the small sample size, the statistically significant is threaten. Besides, the analysis has some limitations due to the sample, too. In this case, collected data were analyzed using Excel and SPSS, and supervisors also reviewed the results.

6. Conclusion

This chapter contains the findings of this exploratory study that aims to identify the effort estimation techniques in practice including their benefits and challenges, particularly inaccuracy. To do so, a literature review was carried out which provided us some insights regarding the topic. Based on that, a survey was carried out among software practitioners to get the answer to the research questions. Most of the survey questions were formulated using a Likert scale, however, the questions were divided into both open and closed-ended.

For data analysis initially, frequency and percentage were calculated besides, normality test, statistical analysis, and contrast hypothesis test were done.

6.1. Main Findings

The result shows that not only mixed software development approaches are used but also different effort estimation techniques. The most commonly used are *Story Point*, *Planning Poker*, *Expert Estimation*, and *Use Case Point*. However, it is also observed that the most frequently used measurement units during estimation are the *Fibonacci series* and the *ideal day.*

Similarly, regarding the benefits of estimation, it is observed that most of the respondents agree that *Drive the team to complete the project successfully* was one of the topmost benefits, in addition, other benefits were also reported similarly.

The results presented here are categorized based on the highest mean value, the factors having the highest mean value are reported as the major issue of inaccuracy. Among the 5 major inaccuracy, Requirement Related Issue-*Complexity and Uncertainty* (4.250), *Missing and changing requirements* (4.058) followed by, Team Related Issue- *Knowledge sharing problem in the team* (3.96), *Unskilled Team Member* (3.943), Over Optimism- *Considering best case scenario* (3.962), Others- *Ignoring Testing Effort* (3.94) and finally, Project Management Related Issue- *Poor change control* (3.860) is the least reported reason for the inaccuracy in estimation. More importantly, the inferential statistical analysis compares opinions of the respondents based on their experience, here, the experience was divided into two groups (less than 3 years and more than 3 years of experience). As a result, there were two statistically significant results as presented in Table 34.

Even though, Mann - Whitney Test result was considered best for this study based on the sample size <30, an independent sample t-test was also used to find out the differences in the result and it is found that there is only one difference for the inaccuracy (Project management related issue).

Table 34: Inferential statistical analysis result

Inaccuracy	**Independent sample t-test**	**Mann-Whitney Test**
Requirement related issue	*Poor user stories* (t (27.625)= -2.370, p=0.025)	*Poor user stories*= (U= 199.5, p=0.02)
Project management related issue		*Poor change control* (U= 191.00, p=0.045).

Finally, more than 85% of the respondents *did not know* about either #NoEstimate or #NoProject. However, only 9.6% wanted to know more about #NoEstimate and 3.8% for #NoProject respectively. Therefore, the impact created by these movements among the respondents in this study is less than 10%.

6.2. Future Work

The most obvious opportunity for further research in the context of this study is to collect more responses to increasing the generalizability of the results. Although #NoEstimate and #NoProject are promoted as practitioners' movement, more than 84% of the respondents *did not know about it*, so further research is needed to better understand the principles behind those movements and the impact in practice.

Finally, among the 9 participants who did not participate in effort estimation, two of them (Program/Product Manager (PM) [11-20] and developer[5-10]) have more than 5 years of agile experience and they reported the use of DevOps, Scrum, and Extreme Programming. Therefore, it would be interesting to get insights about the no participation in the effort estimation of these types of profiles.

7. References

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). Agile Software Development Methods: Review and Analysis. *ArXiv:1709.08439 [Cs]*. http://arxiv.org/abs/1709.08439

Alliance, A. (2016). What is Agile. Agile Alliance. Https://Www. Agilealliance. Org/Agile101/What-Is-Agile.

Anooja, A., & Rajawat, S. (2018). Analyzing Agile Estimation Techniques and Software Development.

Babbie, E. R. (1989). *The practice of social research*. Wadsworth Publishing Company.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., & Jeffries, R. (2001). *Manifesto for agile software development*.

Bettany-Saltikov, J., & Whittaker, V. J. (2014). Selecting the most appropriate inferential statistical test for your quantitative research study. *Journal of Clinical Nursing*, *23*(11–12), 1520–1531.

Birks, M., & Mills, J. (2015). *Grounded theory: A practical guide*. Sage.

Bloch, M., Blumberg, S., & Laartz, J. (2012). Delivering large-scale IT projects on time, on budget, and on value. *Harvard Business Review*.

Boehm, B. W. (1981). *Software engineering economics* (Vol. 197). Prentice-hall Englewood Cliffs (NJ).

Boiten, R. (2017). *The Necessity of Estimation in Software Development Projects*. https://brage.bibsys.no/xmlui/handle/11250/2456085

Britto, R., Mendes, E., & Börstler, J. (2015). An empirical investigation on effort estimation in agile global software development. *Global Software Engineering (ICGSE), 2015 IEEE 10th International Conference On*, 38–45.

Britto, R., Usman, M., & Mendes, E. (2014). Effort estimation in agile global software development context. *International Conference on Agile Software Development*, 182–192.

Cao, L. (2008). Estimating agile software project effort: An empirical study. *AMCIS 2008 Proceedings*, 401.

Captain, T. (2011, July 2). Swimlane Sizing – Complete & Fast Backlog Estimation. *The Agile Pirate*. http://theagilepirate.net/archives/109

Casado-Lumbreras, C., Colomo-Palacios, R., Gomez-Berbis, J. M., & Garcia-Crespo, A. (2009). Mentoring programmes: A study of the Spanish software industry. *International Journal of Learning and Intellectual Capital*, *6*(3), 293–302.

Ceschi, M., Sillitti, A., Succi, G., & Panfilis, S. D. (2005). Project management in plan-based and agile companies. *IEEE Software*, *22*(3), 21–27. https://doi.org/10.1109/MS.2005.75

Chemuturi, M. (2009). Software estimation best practices, tools & techniques: A complete guide for software project estimators. J. Ross Publishing.

Čížek, T. (2009). Designing and conducting mixed methods research. *Central European Journal of Public Policy*, *3*(1), 92–94.

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.

Coleman, G., & O'Connor, R. (2007). Using grounded theory to understand software process improvement: A study of Irish software product companies. *Information and Software Technology*, *49*(6), 654–667.

Cooke, J. L. (2012). Everything you want to know about Agile: How to get Agile results in a less-than-agile organization. IT Governance Ltd.

Cork, P. (2015). Empirical study of project management practices.

Creswell, J. W. (1994). Research design: Qualitative, quantitative, and mixed methods approaches.

Creswell, J. W., & Garrett, A. L. (2008). The "movement" of mixed methods research and the role of educators. *South African Journal of Education*, *28*(3), 321–333.

Creswell, J. W., & Miller, D. L. (2000). Determining validity in qualitative inquiry. *Theory into Practice*, *39*(3), 124–130.

Czarniawska, B. (2004). Narratives in social science research. Sage.

Dagnino, A. (2013). Estimating software-intensive projects in the absence of historical data. *2013 35th International Conference on Software Engineering (ICSE)*, 941–950. https://doi.org/10.1109/ICSE.2013.6606643

Dalton, J. (2019). Dot Voting. In *Great Big Agile* (pp. 165–166). Springer.

Denscombe, M. (2008). Communities of practice: A research paradigm for the mixed methods approach. *Journal of Mixed Methods Research*, *2*(3), 270–283.

Dexter, F. (2013). Wilcoxon-Mann-Whitney test used for data that are not normally distributed. LWW.

Dillman, D. A. (2011). Mail and Internet surveys: The tailored design method—2007 Update with new Internet, visual, and mixed-mode guide. John Wiley & Sons.

Dora, S. K., & Dubey, P. (2015). Software Development Life Cycle (SDLC) Analytical Comparison and Survey on Traditional and Agile Methodology. *National Monthly Referred Journal of Research Science and Technology*, *2*(8).

Duarte, V. (2015). *NoEstimates: How To Measure Project Progress Without Estimating*. https://www.amazon.com/NoEstimates-Measure-Project-Progress-Estimating-ebook/dp/B01FWMSBBK

Eberendu, A. C., Akpan, E. O. P., Ubani, E. C., & Ahaiwe, J. (2018). *A Methodology for the Categorisation of Software Projects in Nigeria Based on Performance*. https://doi.org/DOI: 10.9734/AJRCOS/2018/43819

Finnie, G. R., Wittig, G. E., & Desharnais, J.-M. (1997). A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models. *Journal of Systems and Software*, *39*(3), 281–289.

Flyvbjerg, B., & Budzier, A. (2013). Why your IT project might be riskier than you think. *ArXiv Preprint ArXiv:1304.0265*.

Fowler Jr, F. J. (2013). *Survey research methods*. Sage publications.

Freund, R. J., & Wilson, W. J. (1993). Inferences for two or more means. *Statistical Methods. Academic Press, Inc., San Diego, CA*, 203–260.

Fuggetta, A., & Di Nitto, E. (2014). Software process. *Proceedings of the on Future of Software Engineering*, 1–12.

Giorgi, A. (2012). The descriptive phenomenological psychological method. *Journal of Phenomenological Psychology*, *43*(1), 3–12.

Glogowska, M. (2015). Paradigms, pragmatism and possibilities: Mixed-methods research in speech and language therapy. *International Journal of Language & Communication Disorders*, 1–10.

Gray, A. R., & MacDonell, S. G. (1997). A comparison of techniques for developing predictive models of software metrics. *Information and Software Technology*, *39*(6), 425–437.

Grenning, J. (2002). Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting*, *3*, 22–23.

Hannay, J. E., Benestad, H. C., & Strand, K. (2018). Agile Uncertainty Assessment for Benefit Points and Story Points. *IEEE Software*, *36*(4), 50–62.

Hastie, S., & Wojewoda, S. (2017). Standish Group 2015 Chaos Report-Q&A with Jennifer Lynch (2015).

Haugen, N. C. (2006). An empirical study of using planning poker for user story estimation. *Agile Conference, 2006*, 9 pp. – 34.

Heusser, M. (2013, November 5). *"No Estimates" in Action: 5 Ways to Rethink Software Projects*. CIO. https://www.cio.com/article/2381167/agile-development/-no-estimates-in-action-5-ways-to-rethink-software-projects.html

Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, *51*, 915–929.

ISTQB®. (2016). *ISTQB® Worldwide Software Testing Practices Report 2015-2016*. https://www.istqb.org/references/surveys/istqb-worldwide-software-testing-practices-report-2015-2016.html

ISTQB®, I. (2018). *ISTQB® International Software Testing Qualifications Board*. ISTQB® International Software Testing Qualifications Board. https://www.istqb.org/references/surveys/istqb%C2%AE-worldwide-software-testing-practices-survey-2017-18.html

Ivankova, N. V., Creswell, J. W., & Stick, S. L. (2006). Using mixed-methods sequential explanatory design: From theory to practice. *Field Methods*, *18*(1), 3–20.

Jacobson, I. (1993). Object-oriented software engineering: A use case driven approach. Pearson Education India.

Javdani Gandomani, T., Zulzalil, H., Abdul Ghani, A. A., Md. Sultan, A. B., & Meimandi Parizi, R. (2015). The impact of inadequate and dysfunctional training on Agile transformation process: A Grounded Theory study. *Information and Software Technology*, *57*, 295–309. https://doi.org/10.1016/j.infsof.2014.05.011

Johnson, R. B., & Onwuegbuzie, A. J. (2004). Mixed methods research: A research paradigm whose time has come. *Educational Researcher*, *33*(7), 14–26.

Johnson, R. B., Onwuegbuzie, A. J., & Turner, L. A. (2007). Toward a definition of mixed methods research. *Journal of Mixed Methods Research*, *1*(2), 112–133.

Jørgensen, M. (2004). A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, *70*(1–2), 37–60.

Jorgensen, M., & Shepperd, M. (2007). A systematic review of software development cost estimation studies. *IEEE Transactions on Software Engineering*, *33*(1).

Kasunic, M. (2005). *Designing an effective survey*. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

Keaveney, S., & Conboy, K. (2006). Cost estimation in agile development projects. *ECIS*, 183–197.

Kuhrmann, M., Tell, P., Klünder, J., Hebig, R., Licorish, S., & MacDonell, S. (2018). *HELENA Stage 2 Results*. https://doi.org/10.13140/RG.2.2.14807.52649

Kulathunga, D., & Ratiyala, S. D. (2018). Key Success Factors of Scrum Software Development Methodology in Sri Lanka. *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)*, *45*(1), 234–252.

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. A brief history. *Computer*, *36*(6), 47–56.

Leinonen, J. (2016). Evaluating Software Development Effort Estimation Process in Agile Software Development Context. *University of Oulu*.

Lenarduzzi, V., Lunesu, I., Matta, M., & Taibi, D. (2015). Functional size measures and effort estimation in agile development: A replicated study. *International Conference on Agile Software Development*, 105–116.

Leybourn, E., & Hastie, S. (2018). *# noprojects: A Culture of Continuous Value*. Lulu. com.

Lincoln, Y. S., Lynham, S. A., & Guba, E. G. (2011). Paradigmatic controversies, contradictions, and emerging confluences, revisited. *The Sage Handbook of Qualitative Research*, *4*, 97–128.

Liskin, O., Pham, R., Kiesling, S., & Schneider, K. (2014). Why we need a granularity concept for user stories. *International Conference on Agile Software Development*, 110–125.

Mahnič, V., & Hovelja, T. (2012). On using planning poker for estimating user stories. *Journal of Systems and Software*, *85*(9), 2086–2095.

Mansor, Z., Kasirun, Z. M., Yahya, S., & Arshad, N. H. H. (2011). Current practices of software cost estimation technique in Malaysia context. *International Conference on Informatics Engineering and Information Science*, 566–574.

Mazlan, M., & NUR, A. (2012). Students' Perception of Motivation to Learn: Does an Avatar Motivate? Durham University.

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

Mertens, D. M. (2014). Research and evaluation in education and psychology: Integrating diversity with quantitative, qualitative, and mixed methods. Sage publications.

Molokken, K., & Jorgensen, M. (2003). A review of software surveys on software effort estimation. *Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium On*, 223–230.

Molokken-Ostvold, K., & Furulund, K. M. (2007). The relationship between customer collaboration and software project overruns. *Agile 2007 (AGILE 2007)*, 72–83.

Moløkken-Østvold, K., Jørgensen, M., Tanilkan, S. S., Gallis, H., Lien, A. C., & Hove, S. W. (2004). A survey on software estimation in the Norwegian industry. *10th International Symposium on Software Metrics, 2004. Proceedings.*, 208–219.

Muketha, G. (2016). A Review of Agile Software Effort Estimation Methods.

Murchison, J. (2010). Ethnography essentials: Designing, conducting, and presenting your research (Vol. 25). John Wiley & Sons.

Nassif, A. B., Azzeh, M., Capretz, L. F., & Ho, D. (2016). Neural network models for software development effort estimation: A comparative study. *Neural Computing and Applications*, *27*(8), 2369–2381.

Nassif, A. B., Capretz, L. F., & Ho, D. (2016). Enhancing use case points estimation method using soft computing techniques. *ArXiv Preprint ArXiv:1612.01078*.

Niglas, K. (2007). Introducing the quantitative-qualitative continuum: An alternative view on teaching research methods courses. *Learning and Teaching of Research Methods at University*, 185–203.

#NoEstimates—Alternative to Estimate-Driven Software Development. (2015). http://www.methodsandtools.com/archive/noestimates.php

Oates, B. J. (2005). Researching information systems and computing. Sage.

Osman, H. H., & Musa, M. E. (2016). A Survey of Agile Software Estimation Methods. *International Journal of Computer Science and Telecommunications*, *7*(3).

Park, H., & Baek, S. (2008). An empirical validation of a neural network model for software effort estimation. *Expert Systems with Applications*, *35*(3), 929–937.

Popli, R., & Chauhan, N. (2014). Agile estimation using people and project related factors. *2014 International Conference on Computing for Sustainable Global Development (INDIACom)*, 564–569. https://doi.org/10.1109/IndiaCom.2014.6828023

Popli, Rashmi, & Chauhan, D. N. (2013). *Research Challenges of Agile Estimation*. *7*(1), 4.

Pozenel, M., & Hovelja, T. (2019). A comparison of the planning poker and team estimation game: A case study in software development capstoneproject course. *The International Journal of Engineering Education*, *35*(1), 195–208.

Ram, A., Juarez-Ram, R., Licea, G., & Mart, Y. (2017). Analysis of Planning Poker Factors between University and Enterprise. *Software Engineering Research and Innovation (CONISOFT), 2017 5th International Conference In*, 54–60.

Raslan, A. T., Darwish, N. R., & Hefny, H. A. (2015). Towards a fuzzy based framework for effort estimation in agile software development. *International Journal of Computer Science and Information Security*, *13*(1), 37.

Royce, W. W. (1987). Managing the Development of Large Software Systems: Concepts and Techniques. *Proceedings of the 9th International Conference on Software Engineering*, 328–338. http://dl.acm.org/citation.cfm?id=41765.41801

Scheaffer, R. L., Mendenhall, W., & Ott, L. (1990). *Elementary survey sampling, 4th edn. PWS*. KENT Publishing Company, Boston, Massachusetts, USA.

Schweighofer, T., Kline, A., Pavlic, L., & Hericko, M. (2016). How is Effort Estimated in Agile Software Development Projects? *SQAMIA*, 73–80.

Shepperd, M., & Kadoda, G. (2001). Comparing software prediction techniques using simulation. *IEEE Transactions on Software Engineering*, *27*(11), 1014–1022.

Shepperd, M., Schofield, C., & Kitchenham, B. (1996). Effort estimation using analogy. *Proceedings of the 18th International Conference on Software Engineering*, 170–178.

So what exactly is this #NoEstimates movement? (2015, April 4). *Premios*. https://premiosgroup.com/exactly-noestimates-movement/

Sommerville, I. (2007). *Software engineering*. Addison-wesley.

Stankovic, D., Nikolic, V., Djordjevic, M., & Cao, D.-B. (2013). A survey study of critical success factors in agile software projects in former Yugoslavia IT companies. *Journal of Systems and Software*, *86*(6), 1663–1678. https://doi.org/10.1016/j.jss.2013.02.027

Tanveer, B., Guzmán, L., & Engel, U. M. (2016). Understanding and improving effort estimation in agile software development: An industrial case study. *Proceedings of the International Conference on Software and Systems Process*, 41–50.

Teddlie, C., & Tashakkori, A. (2009). Foundations of mixed methods research: Integrating quantitative and qualitative approaches in the social and behavioral sciences. Sage.

Trendowicz, A., & Jeffery, R. (2014). Software project effort estimation. Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO Pags, 277–293.

Trendowicz, A., Münch, J., & Jeffery, R. (2008). State of the practice in software effort estimation: A survey and literature review. *IFIP Central and East European Conference on Software Engineering Techniques*, 232–245.

Usman, M., Börstler, J., & Petersen, K. (2017). An effort estimation taxonomy for agile software development. *International Journal of Software Engineering and Knowledge Engineering*, *27*(04), 641–674.

Usman, M., & Britto, R. (2016). Effort estimation in co-located and globally distributed agile software development: A comparative study. *2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, 219–224.

Usman, M., Mendes, E., & Börstler, J. (2015). Effort estimation in agile software development: A survey on the state of the practice. *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, 12.

Usman, M., Mendes, E., Weidt, F., & Britto, R. (2014). Effort estimation in agile software development: A systematic literature review. *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, 82–91.

VersionOne. (2017). VersionOne. https://www.versionone.com/about/press-releases/versionone-releases-11th-annual-state-of-agile-report/

Vyas, M., Bohra, A., Lamba, C. S., & Vyas, A. (2018). A Review on Software Cost and Effort Estimation Techniques for Agile Development Process.

Wen, J., Li, S., Lin, Z., Hu, Y., & Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, *54*(1), 41–59.

What is #noestimates. (2017). http://www.agileguru.xyz/agile-blog/article/what-is-noestimates/

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Yin, R. K. (2009). Case study research: Design and methods (applied social research methods). *London and Singapore: Sage*.

Zhang, W., & Creswell, J. (2013). The use of "mixing" procedure of mixed methods in health services research. *Medical Care*, *51*(8), e51–e57.

Ziauddin, S. K. T., & Zia, S. (2012). An effort estimation model for agile software development. *Advances in Computer Science and Its Applications (ACSA)*, *2*(1), 314–324.

Appendix A: Survey Design

Estimation Techniques in Agile Software Development

Dear Participant,

I am Sandeep RC, a student of Master of Applied Computer Science at the Østfold University College. This survey is a part of my master's thesis related to "Estimation Techniques in Agile Software Development". The aim of this survey is to identify the benefits and challenges of estimation techniques that agile teams face in practice.

The survey takes around 5-10 minutes to complete and it is open until 2019/07/08.

Please remember that the survey is voluntary, and your identity remains anonymous.

If you have any questions or concerns, please do not hesitate to contact us.

Student: Sandeep RC

Supervisor: Dr. Ricardo Colomo-Palacios and Dr. Mary Sánchez-Gordón

College: Østfold University College- Høgskolen i Østfold

Email: sandeepr@hiof.no

Section 1: Demographics

First, we need to capture your current working context. Therefore, the following questions aim at collecting some basic demographic information about country, genre, and experience.

1. In which country are you located?

If you are working in different countries, please select the country in which you spent most of your time.

- O Norway
- O USA

- O Nepal
- O India
- O Others

2. What is your genre?

Please select from the following options:

- O Male
- O Female
- O Others
- O I prefer not to answer

3. How many years of Agile Experience do you have?
    - O Less than one year
    - O 1-3 years
    - O 4-5 years
    - O 6-10 years
    - O 11-20 years
    - O More than 20 years

Section 2: Software Development Project

This section contains the information related to software project. Therefore, the following section aims at collecting the information related to role followed by team size, project length, business domain, development approach and finally, importance of estimation.

Please consider your current or last project to answer the following questions

1. What is your major role in the project?

Please choose from the following

- O Designer
- O Developer
- O Program Manager

- O Product owner
- O Scrum Master
- O Software Architect
- O Tester
- O Other (Please Specify) [ ]

2. How big is the team of the project?
   - O 1-5
   - O 6-10
   - O 11-20
   - O More than 20
3. How long is the project?
   - O 2-4 weeks
   - O 1-6 months
   - O 7-12 months
   - O More than one year
4. Please specify the business application domain that you are working on in the textbox below. (E.g. E-commerce, data processing)

[ ]

5. Which are the software development approaches used in the project?

	Do not know it	Never Use	Rarely Use	Sometimes	Often	Always
DevOps	0	1	2	3	4	5
Extreme Programming	0	1	2	3	4	5
Kanban	0	1	2	3	4	5
Scrum	0	1	2	3	4	5

Waterfall	0	1	2	3	4	5
Others						

6. How important do you think software development effort estimation is?
    - O Not Important
    - O Less Important
    - O 50-50
    - O Important
    - O Very Important
7. Do you participate in effort estimation in your team?
    - O Yes
    - O No

If the participant chooses "YES", the following will appear otherwise redirect to the end screen.

Section 3: Effort Estimation Techniques

The following questions aim to collect the information on the current estimation process its advantages and the reason for inaccurate estimates.

RQ1: Estimation Techniques

a. Which of the following effort estimation techniques are used in the project? Please choose all that apply:

	Do not know it	Never Use	Rarely Use	Sometimes	Often	Always
Dot Voting	0	1	2	3	4	5
Expert Estimation Method	0	1	2	3	4	5
Planning Poker	0	1	2	3	4	5
Story Point	0	1	2	3	4	5

Swimlane Sizing	0	1	2	3	4	5
Team Estimation Game	0	1	2	3	4	5
The bucket System	0	1	2	3	4	5
Use Case Point	0	1	2	3	4	5

b. According to your experience please address other techniques that are not covered above. Please address them in the text field below.

Measurement Unit

Which of the following measurement units are used in the project for effort estimation?

	Do Not know it	Never	Rarely	Sometimes	Often	Always
Fibonacci Sequence	0	1	2	3	4	5
Ideal days	0	1	2	3	4	5
Line of code	0	1	2	3	4	5
T-shirt size, Dog size	0	1	2	3	4	5

Others

Section 4: Estimation Benefits

These questions are prepared to understand the benefits of software effort estimation in ASD.

RQ2: Benefits of estimation techniques in ASD?

a. What benefits do you think effort estimation techniques have in agile software development? Please choose all that apply

	Do not know	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Drive the team to complete the project successfully.	0	1	2	3	4	5
Identify the resources and project scope.	0	1	2	3	4	5
Helps to identify important issues earlier.	0	1	2	3	4	5
Monitors project progress.	0	1	2	3	4	5
To create transparency	0	1	2	3	4	5
To gain accuracy	0	1	2	3	4	5

b. Based on your experience what other benefits do you think agile software development has? Please address them in the text field below.

Section 5: Inaccurate Estimates

RQ2 Reasons for inaccurate estimates in agile software development

This section specifically focuses on identifying the reason of inaccurate estimates in agile software development.

Do you think the reason for inaccurate estimates in ASD are due to the below mentioned issues? Please rate them based on the given scale.

a. Requirements' related issues

	Do not know it	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Complexity and Uncertainty	0	1	2	3	4	5
Missing and changing requirements	0	1	2	3	4	5
Overlooking non-functional requirements	0	1	2	3	4	5
Poor user stories	0	1	2	3	4	5

b. Project management issues

	Do not know	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Poor change control	0	1	2	3	4	5
Scrum Master not guiding the team	0	1	2	3	4	5
Unstructured group estimation process	0	1	2	3	4	5

c. Team related issues

	Do not know	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Distributed teams	0	1	2	3	4	5
Dominant Personalities	0	1	2	3	4	5
Inexperience in agile	0	1	2	3	4	5
Knowledge sharing problem in team	0	1	2	3	4	5
Pressure of timeline	0	1	2	3	4	5
Technically unskilled team members	0	1	2	3	4	5

d. Over optimism

	Do not know	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Considering only the best-case scenario	0	1	2	3	4	5
Purposely underestimating project to obtain work/contract	0	1	2	3	4	5

e. Others

	Do not know	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Hardware	0	1	2	3	4	5
Ignoring testing effort	0	1	2	3	4	5
Insufficient customer involvement during estimation process	0	1	2	3	4	5
Lack of formal estimation process	0	1	2	3	4	5

According to your experience, if there are (any) other challenges (related or not related with inaccurate estimates)? Please address them in the text field below.

Section 6: Movements

This section is prepared to get the insights of the movements. To understand the impact caused by #NoEstimates and the benefits of #NoProject.

#NoEstimates

These questions are prepared to identify the impact caused by no estimate.

1. Have you heard about #NoEstimates movement?

- O I've never heard of it
- O I've HEARD of it and am Not interested
- O I've HEARD of it and WOULD like to learn it
- O I've USED it before, and would NOT use it again
- O I've USED it before, and WOULD use it again

2. If you see some benefits or challenges related to effort estimation from #NoEstimate, please address them in the text field below.

#NoProject

These questions are prepared to identify the impact caused by no estimates.

1. Have you heard of #NoProject?

- O I've never heard of it
- O I've HEARD of it and am Not interested
- O I've HEARD of it and WOULD like to learn it
- O I've USED it before, and would NOT use it again
- O I've USED it before, and WOULD use it again

2. If you see some benefits or challenges related to effort estimation from #NoProject, please address them in the text field below.

Section 7: Closing

The final part of the survey in which further comments or unaddressed issue, and their interest in interview are asked.

Do you have further comments or issues not addressed so far? Please write your answer in the free text available below:

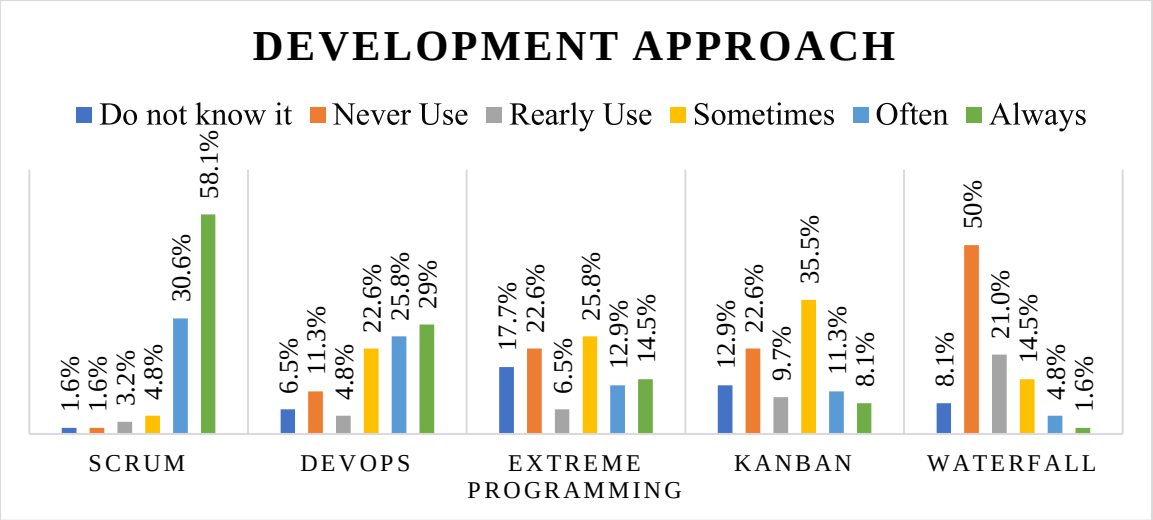
Would you like to collaborate in the interview? Please drop your email in the free text.

Thank you for completing this questionnaire!

I would like to thank you very much for helping me.

Appendix B: Survey Result

B1: Development Approach

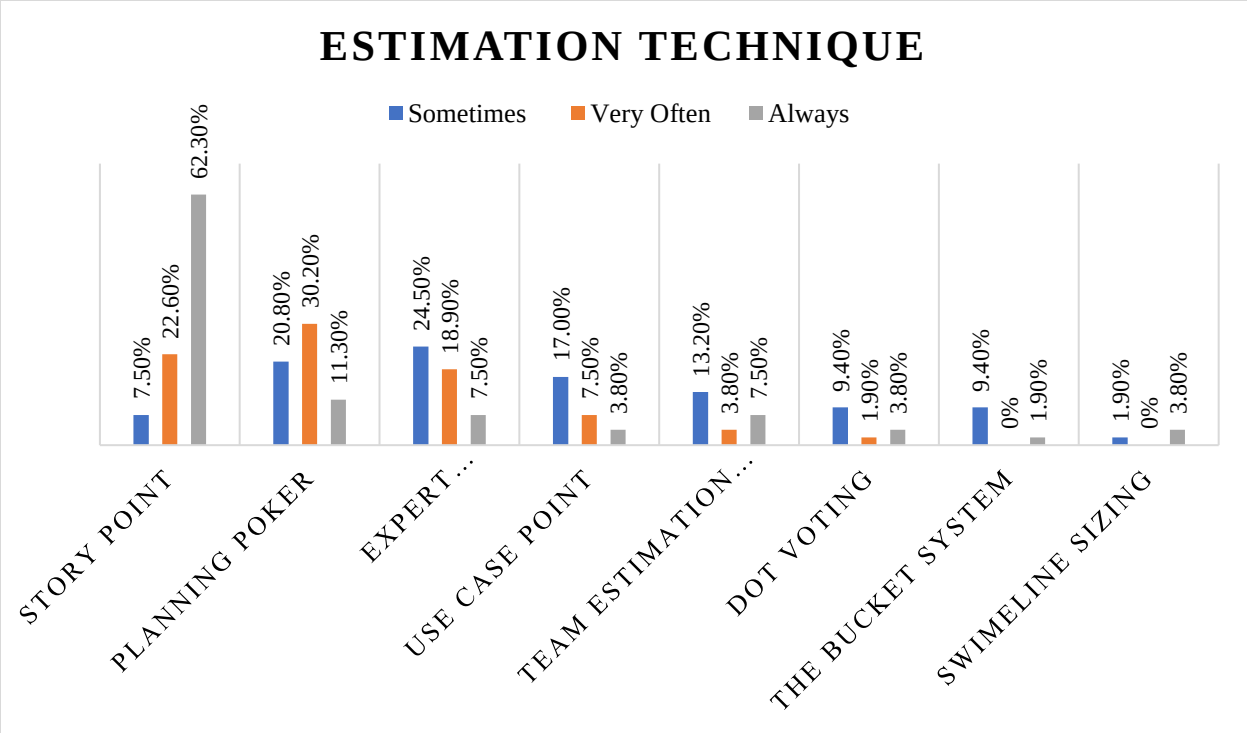


Scale	0	1	2	3	4	5		
Method	Do not know	Never Use	Rarely Use	Sometimes	Very Often	Always	Total	Average
DevOps	6.5% (4)	11.3% (7)	4.8% (3)	22.6% (14)	25.8% (16)	29.0% (18)	62	3.4
Extreme Programming	17.7% (11)	22.6% (14)	6.5% (4)	25.8% (16)	12.9% (8)	14.5% (9)	62	2.4
Kanban	12.9% (8)	22.6% (14)	9.7% (6)	35.5% (22)	11.3% (7)	8.1% (5)	62	2.3
Scrum	1.6% (1)	1.6% (1)	3.2% (2)	4.8% (3)	30.6% (19)	58.1% (36)	62	4.4
Waterfall	8.1% (5)	50% (31)	21.0% (13)	14.5% (9)	4.8% (3)	1.6% (1)	62	1.6

B2: Effort Estimation Technique

Scale / Level	0 Do not know it	1 Never use	2 Rarely	3 Sometimes	4 Very often	5 Always	Total	Average
Planning Poker	3 (5.7%)	11 (20.8%)	6 (11.3%)	11 (20.8%)	16 (30.2%)	6 (11.3%)	53	2.8
Story Point	3 (5.7%)	1(1.9%)	0	4 (7.5%)	12 (22.6%)	33 (62.3%)	53	4.3
Dot Voting	16 (30.2%)	23 (43.4%)	6 (11.3%)	5 (9.3%)	1(1.9%)	2 (3.8%)	53	1.2
Expert Estimation	8 (15.1%)	12 (22.6%)	6 (11.3%)	13 (24.5%)	10 (18.9%)	4 (7.5%)	53	2.3
Swim line	18 (34%)	28 (52.8%)	4 (7.5%)	1 (1.9%)	0	2 (3.8%)	53	0.9
Team Estimation Game	14 (26.6%)	22 (41.5%)	4 (7.5%)	7 (13.2%)	2 (3.8%)	4 (7.5%)	53	1.5
The Bucket System	15 (28.3%)	28 (52.8%)	4 (7.5%)	5 (9.4%)	0	1 (1.9%)	53	1.1
Use Case Point	12 (22.6%)	22 (41.5%)	4 (7.5%)	9 (17%)	4 (7.5%)	2 (3.8%)	53	1.6

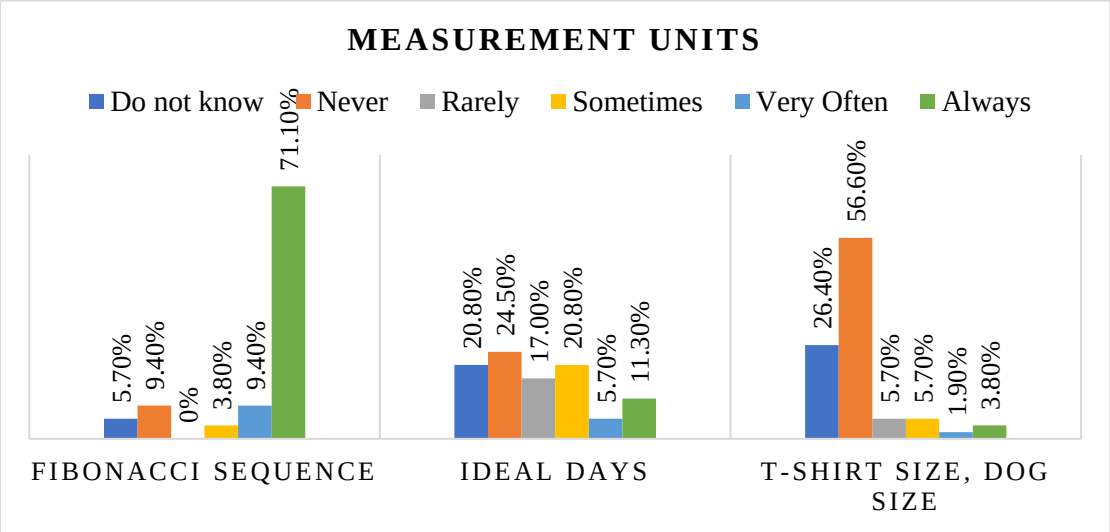
a. Estimation technique percentage



B3: Measurement Unit

Scale	0	1	2	3	4	5	
Units	Do Not know it	Never	Rarely	Sometimes	Often	Always	Total
Fibonacci Sequence	5.7% (3)	9.4% (5)	0	3.8% (2)	9.4% (5)	71.1% (38)	53
Ideal days	20.8% (11)	24.5% (13)	17% (9)	20.8% (11)	5.7% (3)	11.3% (6)	53
T-shirt size, Dog size	26.4% (14)	56.6% (30)	5.7% (3)	5.7% (3)	1.9% (1)	3.8% (2)	53

a. Measurement unit percentage



B4: Benefits of Estimation

Scale / Categories	0 Do not know	1 Strongly Disagree	2 Disagree	3 Neutral	4 Agree	5 Strongly Agree	Total
Drive the team to complete the project successfully.	0	0	1.9% (1)	5.7% (3)	45.3% (24)	47.2% (25)	53
Identify the resources and project scope.	0	0	1.9% (1)	1.9% (1)	54.7% (29)	41.5% (22)	53
Helps to identify important issues earlier	0	1.9% (1)	1.9% (1)	9.4% (5)	50.9% (27)	35.8% (19)	53
Monitors project progress	0	0	1.9% (1)	5.7% (3)	58.5% (31)	34% (18)	53
To create transparency	0	0	0	13.2% (7)	52.8% (28)	34.0% (18)	53
To gain accuracy	0	0	1.9% (1)	20.8% (11)	45.3% (24)	32.1% (17)	53

B5: Reason for inaccurate estimate

B5A: Requirement related issue

Scale	0	1	2	3	4	5	
Category	Do not know	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Total
Complexity and Uncertainty.	1.9% (1)	0	3.8% (2)	3.8% (2)	54.7% (29)	35.8% (19)	53
Missing and changing requirement	1.9% (1)	1.9% (1)	7.5% (4)	3.8% (2)	54.7% (29)	30.2% (16)	53
Overlooking non-functional requirement	5.7% (3)	0	7.5% (4)	9.4% (5)	52.8% (28)	24.5% (13)	53
Poor User stories	1.9% (1)	1.9% (1)	9.4% (5)	9.4% (5)	45.3% (24)	32.1% (17)	53

B5B: Project Management related issue

Scale	0	1	2	3	4	5	
Category	Do not know	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Total
Poor change control	5.7% (3)	1.9% (1)	1.9% (1)	18.9% (10)	56.6% (30)	15.1% (8)	53
Scope creep	7.5% (4)	1.9% (1)	5.7% (3)	22.6% (12)	41.5% (22)	20.8% (11)	53
Scrum Master not guiding the team	0	1.9% (1)	18.9% (10)	24.5% (13)	32.1% (17)	22.6% (12)	53
Unstructured group estimation process	1.9% (1)	3.8% (2)	9.4% (5)	16.9% (9)	45.3% (24)	22.6% (12)	53

B5C: Team Related Issue

Scale	0	1	2	3	4	5

Method	Do not know	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Total
Distributed teams	1.9% (1)	3.8% (2)	34.0% (18)	24.5% (13)	22.6% (12)	13.2% (7)	53
Dominant personalities	0	3.8% (2)	9.4% (5)	35.9% (19)	41.5% (22)	9.4% (5)	53
Inexperience	0	5.7% (3)	9.4% (5)	15.1% (8)	49.1% (26)	20.8% (11)	53
Knowledge sharing problem in team	0	3.8% (2)	5.7% (3)	9.4% (5)	52.8% (28)	28.3% (15)	53
Pressure of timeline	0	3.8% (2)	11.3% (6)	9.4% (5)	62.3% (33)	13.2% (7)	53
Unskilled team members	0	1.9% (1)	5.7% (3)	13.2% (7)	54.7% (29)	24.5% (13)	53

B5D: Over-optimism

Scale	0	1	2	3	4	5	
Method	Do not know	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Total
Considering best case scenario	0	0	5.7% (3)	13.2% (7)	60.4% (32)	20.8% (11)	53
Purposely underestimating to obtain work	1.9% (1)	5.7% (3)	13.2% (7)	13.2% (7)	54.7% (29)	11.3% (6)	53

B5E: Others (Paired sample t-test result)

Scale	0	1	2	3	4	5	
Method	Do not know	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Total
Hardware	0	3.8% (2)	13.2% (7)	41.5% (22)	30.2% (16)	11.3% (6)	53
Ignoring testing effort	0	3.8% (2)	3.8% (2)	9.5% (5)	60.4% (32)	22.6% (12)	53
Insufficient customer involvement	0	1.9% (1)	20.8% (11)	18.9% (10)	45.3% (24)	13.2% (7)	53

during estimation process							
Lack of formal estimation process	0	3.8% (2)	11.3% (6)	15.1% (8)	52.8% (28)	17.0% (9)	53