Effort Estimation in Agile Software Development: A Exploratory Study of Practitioners' Perspective

Sandeep RC [1], Mary Sánchez-Gordón [*1] [0000-0002-5102-1122], Ricardo Colomo-Palacios [1] [0000-0002-1555-9726] and Monica Kristiansen[1]

[1] Østfold University College, Halden, Norway
{sandeep.rc, mary.sanchez-gordon,ricardo.colomo-palacios, monica.kristiansen}@hiof.no

Abstract. Software is increasingly important for our society. However, software industry presents flaws to meet market demands in a faster and reliable way. Agile methods are a way to tackle this problem. However, this approach also poses several challenges, including effort estimation as one of them. In this scenario, #NoEstimates and #NoProject movements emerged as another way to solve estimation issues. In this new scenario, this study aims to provide further empirical evidence on agile effort estimation techniques in practice. To do so, an online survey was designed based on a literature review. Researchers gathered 53 valid questionnaires from agile practitioners. Result shows the importance of hybrid software development approaches and mixed effort estimation techniques. However, it is important to note that *Story Points* and *Fibonacci series* are often used as well. Moreover, the most perceived benefit of estimation in agile contexts is to *drive the team to complete the project successfully*. C*omplexity and uncertainty* are perceived as key factors in estimation accuracy. Finally, further research should be conducted to gain a better understanding of #NoEstimates and #NoProject movements.

Keywords: Effort Estimation, Agile Software Development, Distributed software development

1 Introduction

Software industry is playing a significant role in fulfilling the increasing demand and extensive use of software in our society [1]. Despite that, software projects are challenged in aspects like cost, quality, time, or expected returns on investment [2]. In this scenario, software development needs careful examination, understanding, support, and improvement [3].

Estimation in software projects contains the assessment of the effort, size, staffing, schedule (time), and cost involved in creating a unit of the software product [4]. Estimation is one of the main concerns for the software development industry [5], playing an important role in software development [6] supporting key software process decisions, such as feasibility analysis, resource allocation, risk mitigation, and project planning [7]. However, there is even difficulty in assessing the accuracy of different approaches to effort estimation [8]. Although estimation accuracy directly influences the utility of the estimation results, different software management decisions may require different degrees of accuracy [7]. While, according to [9], effort estimation is not critical for constructing a project's scheduling and planning, it is important to facilitate understanding. Also, software practitioners require effective effort estimation models to facilitate project planning [10] as well as to create baseline budgets and schedules [11].

In particular, effort estimation in agile software development (ASD) is challenging as the requirements are constantly evolving and they are developed as the project progresses [12]. As a consequence, effort estimations in such environments need to be progressively adjusted for every sprint [4] to ensure delivery in required times [12].

Although different estimation techniques exist in ASD and it was reported as an active research area, accuracy was also reported as a clear gap in this field [13].

In 2014, Usman et al. [13] conducted a systematic literature review on effort estimation in ASD. As a result, 25 primary studies were identified. The four main findings are: i) subjective estimation methods like expert judgment, planning poker, use case points estimation method are often used for agile estimation; ii) Use Case Points (UCP) and Story Points (SP) are the most often used size matrices; iii) Mean Magnitude of Relative Error (MMRE) and Magnitude of Relative Error (MRE) is the frequently used metrics in ASD, and iv) Team skills, prior experience, and task size are included as the 3 fundamental cost drivers in ASD.

In 2015, a survey on the state of the practice [14] collected data from 16 different countries and 60 agile practitioners that were involved in the effort estimation. The findings revealed that planning poker (63%) was the most used effort estimation technique followed by, analogy (47%) and expert judgment (38%). In 2016, Tanveer et al. [15] carried out a study to understand the accuracy of the estimation process by examining three agile teams that worked on different web applications. The authors conclude that developers' knowledge, experience, and complexity affect it. The same year, a comparative analysis study on effort estimation practice in ASD was carried out by Usman and Britto [16]. In this study, they compared two co-located and globally distributed teams to identify the similarities and differences of effort estimation practice. The result shows that planning poker and story points are the most reported effort estimation technique and size metric for both teams. More recently, Fernández-Diego et al. [10] updated previous works from Usman et al. [13]. In the last years, several intelligent approaches based have also impacted the ASD effort estimation e.g. [17–20].

According to Duarte [21], in software projects, it is hard to estimate unknown parts. As a response to this challenge, #NoEstimates and #NoProjects movements have emerged. #NoEstimates started back in 2012 promoted by Woody Zuill as a Twitter trend. In this movement, it is claimed to stop estimating backlog because accurate estimation is not possible, and estimations put useless pressure on teams. As a result of this, estimation is seen as waste. #NoProjects concept started back in 2005 [22] and stands for modern management methods to offer proven techniques and tools that go beyond "meeting requirements". These approaches are normally based on continuous value. Both movements are related, #NoEstimate removes the justification of estimation and helps the organization focus on value delivery first [21] whereas #NoProjects is an agile approach towards continuous and market-validated value delivery [22].

In this scenario, despite those previous studies provide valuable insights into effort estimation in ASD, to the best of our knowledge, there is little empirical evidence about the benefits and inaccuracies of effort estimation techniques in actual practice and the impact of #NoEstimates and #NoProjects movements in the arena. Therefore, this study aims to provide further empirical evidence on agile effort estimation techniques in practice.

The remaining of the paper is structured as follows. Section 2 presents the research method adopted describing formulated research questions. The results of this study are presented in Section 3. Section 4 contains the limitations of the study. Finally, some conclusions and future work are given in Section 5.

2 Research Method

2.1 Research Questions

The main objective of this study is to better understand the state of the practice on effort estimation in ASD including benefits and challenges. Based on that, four research questions were formulated:

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 estimations in ASD?
RQ4: What is the repercussion of #NoEstimates and #NoProjects in ASD?

2.2 Survey design

Survey research is one possible design choice for quantitative research. Survey research produces quantitative data about trends, attitudes, or opinions among the population under study [23]. There are different approaches for data collection in a survey, such as personal interviews, telephone interviews, direct observation, or self-administered questionnaires [24]. In the study presented in this paper, a draft questionnaire was designed considering the guidelines for software engineering proposed by Molléri et al. [25]. The two first authors developed an initial version in the English language that is informed by previous literature, e.g. [5, 12, 13, 26, 27]. Then, the other authors reviewed it for validity checking. To obtain as many responses as possible, and to not distract participants unnecessarily, it was decided to keep the number of questions to a minimum.

In this study, an online web-based questionnaire tool (Google forms) was used for data collection. The questionnaire contained sections on software development projects as well as demographic information. Questions were presented to subjects with multiple response answers. The frequency of use software development approaches (see Fig. 1) and estimation techniques (see Fig. 2) was reported by using a five-point scale *Never Use* (1); *Rarely Use* (2); *Sometimes* (3); *Often* (4); and *Always* (5). In addition, the option *I do not know* (0) was included.

The questionnaire also asks subjects for a set of perceived benefits (see Table 2) and 20 reasons for the inaccuracy (see Table 3). These reasons/factors were grouped into 5 major categories: *Requirement Related Issue (RrI)*, *Project Management Related Issue (PMrI)*, *Team Related Issue (TrI)*, *Over-Optimism (Oo)*, and *Others*. The respondents' agreement regarding benefits and inaccuracies was reported using a five-point scale with the following values: *Strongly Disagree* (1); *Disagree* (2); *Neutral* (3); *Agree* (4); and *Strongly Agree* (5). Additionally, the option *I do not know* (0) was included. Moreover, an additional open question encouraging participants to voice other options was included in each category. Authors also provided a text box at the end to gather any further comments or suggestions from participants.

To get an understanding of the impact caused by #NoEstimates and #NoProjects and their potential benefits and challenges, two types of questions were formulated: one closed-ended question (5-point scale) and one open-ended question. The 5-point scale was: *I've never heard of it* (0); *I've HEARD of it and Not interested* (1); *I've HEARD of it and WOULD like to learn it* (2); *I've USED it before, and would NOT use it again* (3); *I've USED it before, and would use it again* (4).

Survey Execution and Sampling Strategy. Participants were identified among the networks of researchers (Convenience Sampling). An e-mail was sent out to contacts detailing the purpose of the study and inviting software practitioners to participate. Authors underlined that the questionnaire was anonymous. The period to answer the questionnaire was about two weeks starting from the 24th of June 2019 to the 8th of July 2019 and one email reminder invitation was sent out after one week of the survey being open. In consequence, recruiting participants was based on availability −a convenience sampling. Despite the drawbacks and bias in such a sample, it does not mean that is inappropriate. Indeed, such a sampling method is reported as the dominant survey approach in software engineering [28, 29].

Data Analysis and Synthesis Approach. As mentioned before, the survey primarily contained questions with predefined lists from which participants could choose a value (e.g., job role, gender, and ASD approach), or code simple data such as integers or strings (e.g., country and years of experience). After reviewing the raw data, 53 out of 62 questionnaires were considered valid. These respondents provided relevant and reliable answers since they were involved in the effort estimation process. In this phase, anonymous IDs were assigned to the respondents and their data records, i.e., we used the "Pi" format [P1 to P62].

To investigate benefits and accuracy challenges in ASD estimation as well as statistical differences between years of experience, the data was analyzed using statistical tests chosen based on certain pre-conditions. In addition, *"I do not Know-0"* answers are excluded from the analysis. For all reported statistical tests, we used a significance level of 0.05. Before the actual test, we tested each skill for normality with the Shapiro-Wilk test. The results of the tests of normality indicated that our sample was not normally distributed. Therefore, we used a non-parametric statistical test called Wilcoxon signed-rank test. Results were analyzed using SPSS.

3 Results

In this section, first, an overview of the study population is presented then the results of the survey answer the three research questions.

Study population. A total of 53 valid responses were collected from seven countries, however, almost 70% of them come from Nepal (22, 41.5%) and Norway (15, 28.3%). Most of the responses were male participants (83%) while females made up 15% (8), and one participant preferred not to say (2%). Regarding years of experience, most of the respondents have more than 3 years of agile experience (33, 62.3%), whereas 32.1% (17) have 1-3 years and 5.7% (3) have less than a year. Most of the respondents also were software developers (31, 58.5%). 50.9% (27) of respondents work in a team size of 6-10 people while 41.5% (22) are in teams of 1-5 people. The project length was reported longer than 1 year by most of the respondents (77.4%, 41). The business domain most reported was e-commerce (56.6%, 30).

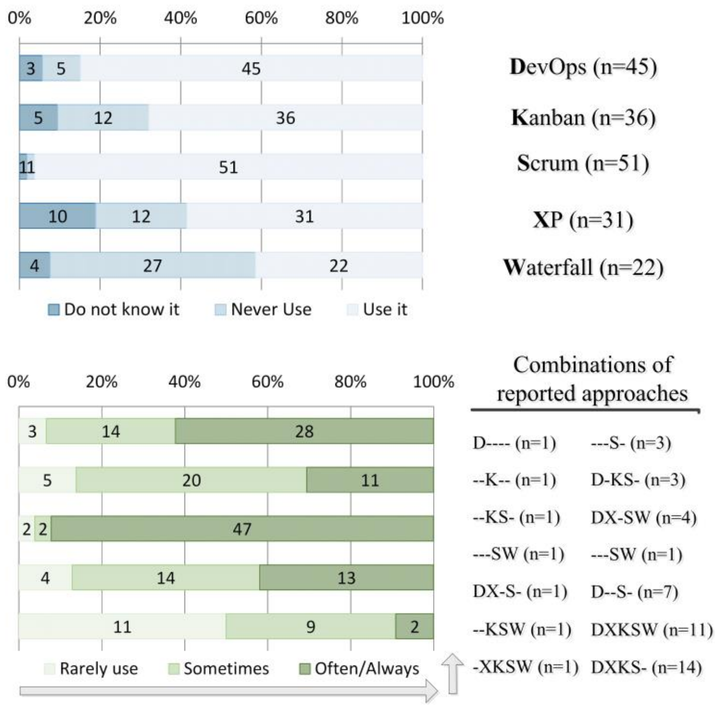
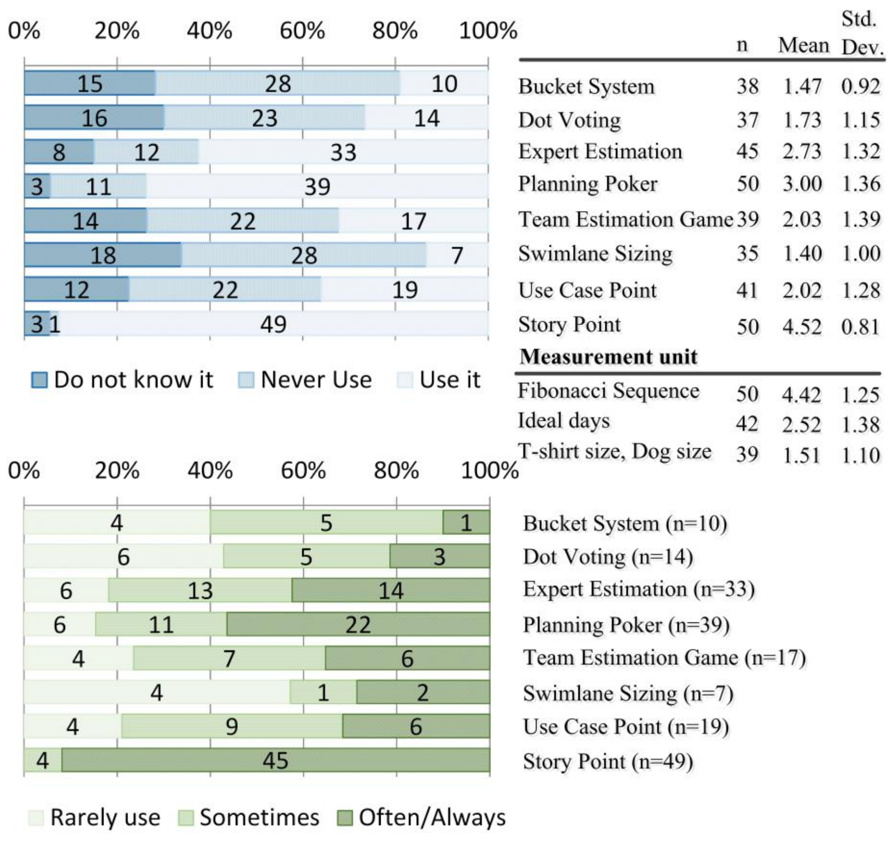


Fig. 1. Frequency and combinations of software development approaches

Fig. 1 shows the frequency and combinations of software development approaches. Although **Scrum** and Kanban are the most frequently practiced approaches in ASD, it is worth noting that many combinations of them are reported, e.g., ***D**evOps and **S**crum* (DS, 7) or ***D**evOps, **X**P, **K**anban, and **S**crum* (***DXKS***, 14) or all of them along with ***W**aterfall* (***DXKS***, 11). This result is aligned with a large previous survey, namely HELENA [30] in which mixed approaches were reported as commonly used.

Finally, the perceived importance of the estimation process by most respondents (75.8%, 43) was that *estimation is very important in ASD* whereas 7 perceived it as important and only 3 were neutral. Moreover, subjects who reported that they were not involved in estimation processes, perceived it as very important (4) and important (5).

RQ1: What are the effort estimation techniques used in ASD?
The descriptive analysis of seven effort estimation techniques reported in this study is shown in Fig. 2.



	n	Mean	Std. Dev.
Bucket System	38	1.47	0.92
Dot Voting	37	1.73	1.15
Expert Estimation	45	2.73	1.32
Planning Poker	50	3.00	1.36
Team Estimation Game	39	2.03	1.39
Swimlane Sizing	35	1.40	1.00
Use Case Point	41	2.02	1.28
Story Point	50	4.52	0.81
Measurement unit			
Fibonacci Sequence	50	4.42	1.25
Ideal days	42	2.52	1.38
T-shirt size, Dog size	39	1.51	1.10

Fig. 2. Descriptive statistical results of the effort estimation techniques

In this study, we included *Bucket system*, *Dot Voting*, *Expert estimation*, *Planning Poker*, *Team estimation game*, *Swimlane sizing*, *Use case point*. According to [31], *Planning Poker* is an estimation technique similar to the *Team estimation game* so we considered each technique separately. Moreover, *Story points* were included in the group of effort estimation techniques, although, *Story points* are a unit of measurement used to represent an estimate of the entire effort necessary to completely perform a piece of software work. It was decided because *Story points* are usually expressed either in numbers that follow the Fibonacci series, t-shirt sizes, or even dog sizes that were included as measurement units.

As we expected, more than 90% of the respondents reported that "Often/Always use" *Story points* while *Planning Poker* and *Expert Estimation Method* were the most common estimation techniques. *Story point* has the highest mean value (4.52) followed by *Planning Poker* (3.00) and *Expert Estimation Method* (2.73). However, it is worth noting that one respondent stated —using the open question— that "*the organization uses COCOMO for estimation*".

Based on the mean values of the measurement units, the *Fibonacci Sequence* was preferred (4.42) followed by *Ideal days* (2.52) and *T-shirt size/Dog size* (1.51).

Estimation Techniques

N	**A S D**	Swimlane Sizing	**Bucket** System	**Dot** Voting	**Team Esti-mation Game**	**Use Case** Point	Expert Estimation	**Planning Poker**	**Story Point**	**Frequency**	**%**
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	**Frequency**	7	10	14	17	19	33	39	49	188	100%
	%	3.7%	5.3%	7.4%	9.0%	10.1%	17.6%	20.7%	26.1%	100%	

Table 1. Overview of estimation techniques by software development approaches

Table 1 shows the frequency of the use of estimation techniques by software development approaches. The first column contains the frequency (#) followed by the (14) combinations of software development approaches (see Table 1), estimation techniques —*Swimlane sizing (**SS**), **B**ucket **S**ystem (**BS**), Dot voting (**DV**), Team estimation game (**TEG**), Use case point (**UCP**), Expert Estimation (**EE**), Planning Poker (**PP**)*, and *Story point (**SP**)*—, and finally Total and Percentage (%).

As it was expected, *Story point* (26.1%) is the most used estimation technique as it has the highest percentage of usage followed by *Planning Poker* (20.7%), *Expert Estimation* (17.6%), and *Use Case Point* (10.1%). These findings are in line with the previous studies [14, 16, 32] that mentioned *Story point* as the most used estimation technique. On the other hand, the large survey —1319 full responses— carried out by VersionOne [33] reveals that 61% of respondents chose Planning poker/team estimation as agile techniques that their companies use. In addition, the findings also reveal that not only hybrid software development approaches are used but also mixed effort estimation techniques.

RQ$_2$: What are the benefits of estimation techniques in ASD?

The six categories of perceived benefits are shown in Table 2. More than 75% (41) respondents "agree/strongly agree" with them. Moreover, "*To gain accuracy*" is the only benefit in which 20% are neutral responses followed by "*To create transparency*" (13.2%) and "*Helps to identify important issues earlier*" (9.4%).

		Less than 3 years			**More than 3 years**			**Total**		
	Benefits	**n**	**Mean**	**Std. Dev.**	**n**	**Mean**	**Std. Dev.**	**n**	**Mea n**	**Std. Dev.**
1	Drive the team to complete the project successfully	20	4.35	0.67	33	**4.39**	0.70	53	**4.38**	0.69
2	Identify the resources and project scope*	20	**4.50**	0.61	33	4.27	0.63	53	4.36	0.62
3	Helps to identify important issues earlier*	20	4.30	0.73	33	4.09	0.88	53	4.17	0.83
4	Monitors project progress	20	4.20	0.62	33	4.27	0.67	53	4.25	0.65
5	To create transparency	20	4.20	0.70	33	4.21	0.65	53	4.21	0.66
6	To gain accuracy	20	3.90	0.85	33	4.18	0.73	53	4.08	0.78

Table 2. Descriptive statistical results of the estimation benefits

Based on the highest mean value, the most perceived benefit is to *Drive the team to complete the project successfully* (4.38) followed by *identifying the resources and project scope* (4.36) and *Monitors project progress* (4.25). Thus, effort estimation is one of the essential factors of the software development process since it drives the team to complete the project successfully [34]. For two benefits —*Identify the resources and project scope* and *Help to identify important issues earlier*, less experienced respondents reached a higher agreement than more experienced ones (marked as * in Table 1, higher mean values are bolded).

On the other hand, it would be interesting to explore if there is a significant difference between the answers based on the experience of the respondents. To do so, the respondents were grouped into two categories "less than 3 years of experience" (n=20) and "more than 3 years of experience" (n=33). We tested the null hypotheses H_0: μ_{Bx}(<3 years) = μ_{Bx}(3 years+) using Wilcoxon signed-rank test. We used that non-parametric statistical test method because it does not require the data sets to follow a normal distribution. The results show that there is no significant difference in the respondents' perceived value of the benefits based on their experience. Although the practitioners in this study rated benefits in a similarly positive way, it is worth noting that 13% of respondents from the 2019 VersionOne survey [33] pointed out that estimation accuracy is one measure of success.

RQ$_3$: What are the reasons for inaccurate estimations in ASDSD?

To get insights about the inaccuracy in the estimation, 20 potential factors/reasons were analyzed. These factors were grouped into 5 major categories: *Requirement Related Issue (RrI)*, *Project Management Related Issue (PMrI)*, *Team Related Issue (TrI)*, *Over-Optimism (Oo)*, and *Others*. Table 3 shows the lists of the descriptive statistical results of each factor.

Table 3 also shows that less experienced respondents reached a higher agreement than more experienced ones for 3 out of 20 factors —*Unstructured group estimation process*, *Distributed team*, and *Knowledge sharing problem in team* (marked as *). The descriptive statistical analysis result shows that most reported inaccurate estimates based on the mean values are two *Complexity and Uncertainty* (4.25) and *Missing and changing requirements* (4.06). Both are in category RrI. The higher mean values in the other categories are *Knowledge sharing problem in the team* TrI (3.96), *Considering best case scenario* OO (3.96), *Ignoring Testing Effort* Others (3.94), and *Poor change control* PMrI (3.86).

On the other hand, one can see differences in the hindering factors influencing accuracy based on the experience of the respondents. In consequence, we tested the null hypotheses H_0: μ_{Ax}(<3 years) = μ_{Ax}(3 years+) using a Wilcoxon signed-rank test. The results show that there are two significant differences:

- *Poor user stories* (U= 199.5, p=0.02)
- *Poor change control* (U= 191.00, p=0.045)

		Less than 3 years			More than 3 years			Total		
Inaccurate		**n**	**Mean**	**Std. Dev.**	**n**	**Mean**	**Std. Dev.**	**n**	**Mean**	**Std. Dev.**
RrI	Complexity and Uncertainty	19	**4.16**	0.76	33	**4.30**	0.68	52	**4.25**	.71
	Missing and changing requirements	19	3.84	1.12	33	4.18	0.77	52	4.06	.92
	Overlooking non-functional requirements	17	3.88	0.86	33	4.06	0.83	50	4.00	.83
	Poor user stories	19	3.53	1.17	33	4.24	0.79	52	3.98	1.00
PMrI	*Poor change control*	19	3.58	0.90	31	**4.03**	0.66	50	**3.86**	0.78
	Scope creep	19	3.58	1.07	30	3.93	0.83	49	3.80	0.93
	Scrum Master not guiding the team	20	3.20	1.06	33	3.76	1.09	53	3.55	1.10
	Unstructured group estimation process*	20	**3.85**	0.99	32	3.69	1.09	52	3.75	1.05
TrI	Distributed teams*	19	3.42	0.90	33	2.88	1.22	52	3.08	1.13
	Dominant Personalities	20	3.35	0.93	33	3.48	0.94	53	3.43	0.93
	Inexperience	20	3.50	1.10	33	3.82	1.07	53	3.70	1.08
	Knowledge sharing problem in team*	20	**4.00**	1.03	33	3.94	0.97	53	**3.96**	0.98
	Pressure of timeline	20	3.55	1.05	33	3.79	0.93	53	3.70	0.97
	Unskilled team members	20	3.90	0.72	33	**3.97**	0.98	53	3.94	0.89
OO	Considering best case scenario	20	**3.90**	0.72	32	**4.00**	0.80	53	**3.96**	0.76
	Purposely underestimating to obtain work	20	3.50	1.15	32	3.56	1.01	52	3.54	1.06
Others	Hardware	20	3.30	1.03	33	3.33	0.96	53	3.32	0.98
	Ignoring testing effort	20	**3.65**	1.09	33	**4.12**	0.74	53	**3.94**	0.91
	Insufficient customer involvement during estimation process	20	3.15	1.09	33	3.67	0.96	53	3.47	1.03
	Lack of formal estimation process	20	3.35	1.09	33	3.88	0.93	53	3.68	1.01

Table 3. Descriptive statistical results of the inaccurate estimates

RQ4: What is the repercussion of #NoEstimates and #NoProjects in ASD?

The result shows that around 85% of the respondents (45 and 47) *have never heard of* #NoEstimate and #NoProject. Moreover, three respondents claim that they "*have heard of it and are not interested"* in both movements. While less than 10% *heard of it* and *wanted to know about* #NoEstimate (5) and only one of them about #NoProject. Despite that fact, 3 participants provided valid answers related to the benefits of #NoEstimate —*1. Faster, 2. Overshadow Project Scope, and 3. Provide a clear timeline for delivery*—. However, no valid responses were received for #NoProject.

The aforementioned reveals the scarce impact of these movements on the effort estimations among the respondents in this study. Although, a previous study [35] about "agile uncertainty assessment for benefit points and story points" highlights that "*the #NoEstimates movement is gaining attention of agile practitioners"*, our findings rather point out little attention. The authors also mention that it could not offer enough benefit-over-cost optimization in the context of large agile projects however our findings neither support nor deny such a claim.

4 Limitations

In this study, authors followed the survey guidelines for software engineering proposed by Molléri et al. [25]. However, this study still has some limitations:

The researchers' bias is always a threat. To reduce that bias, the survey questionnaire was iteratively designed and updated by the authors based on the results of the literature review, and its completeness and readability were validated by one senior researcher. However, further research should make clear that story points are a unit of measurement and include man-hours as measurement units, as well. In this sense, it is worth noting that *Fibonacci numbers* are just numbers so that they can refer to *ideal days* or *man-hours*.

Irrelevant respondents could introduce a systematic error or bias in the study results. To reduce that threat some steps were taken. Firstly, respondents were assured of their anonymity to avoid evaluation apprehension. Secondly, it was explicitly stated in the survey introduction that only practitioners with experience in ASD should participate.

Additionally, respondents were asked about their experience in ASD and effort estimation to ensure that all respondents were agile practitioners and active participants in the effort estimation process. Although 62 agile practitioners were involved in this study, 53 of them reported work experience on effort estimation. Therefore, only 53 were valid answers that could provide relevant and reliable insights on this area. Thirdly, some respondents might have misinterpreted the questionnaire, or they could be confused. To ensure the correct understanding of the questionnaire, 2 rounds of pilot testing were done. Moreover, although multiple options were added to the questionnaire, respondents might not get the answer they want. To reduce this threat, "Other" option was included at the end of all the questions.

The sample is small, which limits the generalization of the results, as well as the important part of the sample coming from Nepal, meaning that it is not representing a generic population. Although we believe that such a sample is quite heterogeneous in terms of experience, job role, and country, the sample size should be expanded to a larger group to increase the generalizability of the results. The statistical significance is threatened by the small sample size. Finally, it is worth noting that the *"I do not Know-0"* answers are excluded from the analysis.

5 Conclusion and Future Work

This paper presents the findings of our exploratory study that aims to identify agile effort estimation techniques in practice including their benefits and challenges related to inaccuracy. To identify the effort estimation techniques a previous literature review was carried out. Based on those results, a questionnaire was designed to get the answers to our research questions. Most of the questions were formulated using a six-point scale however the questions were divided into both open and closed-ended. It means that our survey was intentionally designed to explore effort estimation in agile contexts. Therefore, a subjective evaluation made by the respondents based on a pre-defined list of options and agile artifacts such as user stories were considered.

After inviting agile practitioners, 62 answers were collected but only 53 were valid since those practitioners were involved in the effort estimation process. The most used effort estimation technique based on the higher value mean is *Planning Poker* (3.00) along with *Story Point* (4.52). In this context, the most frequently used measurement unit also is the *Fibonacci series* (4.42). In addition, most of the respondents agree that *Drive the team to complete the project successfully* (4.38) was the top perceived benefit.

Regarding the reasons for inaccuracy, 20 factors were grouped into five categories. By each category, the factors most agreed were *Complexity and Uncertainty "*RrI" (4.25), *Knowledge sharing problem in the team* "TrI" (3.96), *Considering best case scenario* "OO" (3.96), *Ignoring Testing Effort* "Others" (3.94), and *Poor change control* "PMrI" (3.86). The respondents were also grouped into two categories "less than 3 years of experience" (n=20) and "more than 3 years of experience" (n=33) to identify if there are significant differences.

A richer investigation of agile artifacts to estimate effort accurately be conducted. The most obvious opportunity for further research in the context of this study is to collect more responses. Moreover, although #NoEstimate and #NoProject are promoted as practitioners' movements, more than 84% of the respondents *did not know about it*, so further research is also needed to better understand the principles behind those movements and their impact in practice.

References

1. 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 86:1663–1678. https://doi.org/10.1016/j.jss.2013.02.027

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

3. Fuggetta A, Di Nitto E (2014) Software process. In: Proceedings of the on Future of Software Engineering. ACM, pp 1–12

4. Jorgensen M, Shepperd M (2007) A Systematic Review of Software Development Cost Estimation Studies. IEEE Transactions on Software Engineering 33:33–53. https://doi.org/10.1109/TSE.2007.256943

5. Popli R, Chauhan N (2014) Agile estimation using people and project related factors. In: 2014 International Conference on Computing for Sustainable Global Development (INDIACom). pp 564–569

6. Usman M, Mendes E, Weidt F, Britto R (2014) Effort estimation in agile software development: a systematic literature review. In: Proceedings of the 10th International Conference on Predictive Models in Software Engineering. ACM, Turin, Italy, pp 82–91

7. Qi K, Boehm BW (2019) Process-Driven Incremental Effort Estimation. In: 2019 IEEE/ACM International Conference on Software and System Processes (ICSSP). pp 165–174

8. Sommerville I (2010) Software Engineering, 9th ed. Addison-Wesley

9. Altaleb A, Altherwi M, Gravell A (2020) A Pair Estimation Technique of Effort Estimation in Mobile App Development for Agile Process: Case Study. In: Proceedings of the 2020 The 3rd International Conference on Information Science and System. Association for Computing Machinery, New York, NY, USA, pp 29–37

10. Fernández-Diego M, Méndez ER, González-Ladrón-De-Guevara F, et al (2020) An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review. IEEE Access 8:166768–166800. https://doi.org/10.1109/ACCESS.2020.3021664

11. Rosa W, Clark BK, Madachy R, Boehm B (2021) Empirical Effort and Schedule Estimation Models for Agile Processes in the US DoD. IEEE Transactions on Software Engineering 1–1. https://doi.org/10.1109/TSE.2021.3080666

12. Tanveer B, Guzmán L, Engel UM (2017) Effort estimation in agile software development: Case study and improvement framework. Journal of Software: Evolution and Process 29:e1862. https://doi.org/10.1002/smr.1862

13. Usman M, Mendes E, Weidt F, Britto R (2014) Effort Estimation in Agile Software Development: A Systematic Literature Review. In: Proceedings of the 10th International Conference on Predictive Models in Software Engineering. ACM, New York, NY, USA, pp 82–91

14. Usman M, Mendes E, Börstler J (2015) Effort estimation in agile software development: a survey on the state of the practice. In: Proceedings of the 19th international conference on Evaluation and Assessment in Software Engineering. ACM, p 12

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

16. Usman M, Britto R (2016) Effort estimation in co-located and globally distributed agile software development: A comparative study. In: 2016 joint conference of the international workshop on software measurement and the international conference on software process and product measurement (IWSM-MENSURA). IEEE, pp 219–224

17. Arora M, Sharma A, Katoch S, et al (2021) A State of the Art Regressor Model's comparison for Effort Estimation of Agile software. In: 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM). pp 211–215

18. Sinha RR, Gora RK (2021) Software Effort Estimation Using Machine Learning Techniques. In: Goar V, Kuri M, Kumar R, Senjyu T (eds) Advances in Information Communication Technology and Computing. Springer, Singapore, pp 65–79

19. Weflen E, MacKenzie CA, Rivero IV (2022) An influence diagram approach to automating lead time estimation in Agile Kanban project management. Expert Systems with Applications 187:115866. https://doi.org/10.1016/j.eswa.2021.115866

20. Ramessur MA, Nagowah SD (2021) A predictive model to estimate effort in a sprint using machine learning techniques. Int j inf tecnol 13:1101–1110. https://doi.org/10.1007/s41870-021-00669-z

21. Duarte V (2015) NoEstimates: How To Measure Project Progress Without Estimating. https://www.amazon.com/NoEstimates-Measure-Project-Progress-Estimating-ebook/dp/B01FWMSBBK. Accessed 25 Feb 2019

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

23. Creswell JW (2009) Research design: qualitative, quantitative, and mixed methods approaches, 3rd ed. Sage Publications, Thousand Oaks, Calif

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

25. Molléri JS, Petersen K, Mendes E (2016) Survey Guidelines in Software Engineering: An Annotated Review. In: Proceedings of the 10th ESEM '16. ACM, New York, NY, p 58:1-58:6

26. Usman M, Börstler J, Petersen K (2017) An Effort Estimation Taxonomy for Agile Software Development. International Journal of Software Engineering and Knowledge Engineering 27:641–674. https://doi.org/10.1142/S0218194017500243

27. Usman M, Börstler J, Petersen K (2017) An Effort Estimation Taxonomy for Agile Software Development. Int J Soft Eng Knowl Eng 27:641–674. https://doi.org/10.1142/S0218194017500243

28. Sánchez-Gordón M, O'Connor RV (2015) Understanding the gap between software process practices and actual practice in very small companies. Software Quality Journal. https://doi.org/10.1007/s11219-015-9282-6

29. Sjoeberg DIK, Hannay JE, Hansen O, et al (2005) A survey of controlled experiments in software engineering. IEEE Transactions on Software Engineering 31:733–753. https://doi.org/10.1109/TSE.2005.97

30. Kuhrmann M, Tell P, Klünder J, et al (2018) HELENA Stage 2 Results

31. Dalton J (2019) Team Estimation Game. In: Dalton J (ed) Great Big Agile: An OS for Agile Leaders. Apress, Berkeley, CA, pp 255–257

32. 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 35:195–208

33. VersionOne (2019) 13th Annual State of Agile Report. https://explore.versionone.com/state-of-agile/13th-annual-state-of-agile-report

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

35. Hannay JE, Benestad HC, Strand K (2018) Agile Uncertainty Assessment for Benefit Points and Story Points. IEEE Software 36:50–62