Pertanika J. Sci. & Technol. 29 (2): 837 - 861 (2021)

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Review Article

A Review Article on Software Effort Estimation in Agile Methodology

Pantjawati Sudarmaningtyas[1,2] and Rozlina Mohamed[1]*

[1]Faculty of Computing, Universiti Malaysia Pahang, 26300 UMP, Gambang, Kuantan, Pahang, Malaysia
[2]Department of Information System, Universitas Dinamika, 60298 Surabaya, Jawa Timur, Indonesia

ABSTRACT

Currently, Agile software development method has been commonly used in software development projects, and the success rate is higher than waterfall projects. The effort estimation in Agile is still a challenge because most existing means are developed based on the conventional method. Therefore, this study aimed to ascertain the software effort estimation method that is applied in Agile, the implementation approach, and the attributes that affect effort estimation. The results showed the top three estimation that is applied in Agile, are machine learning (37%), Expert Judgement (26%), and Algorithmic (21%). The implementation of all machine learning methods used a hybrid approach, which is a combination of machine learning and expert judgement, or a mix of two or more machine learning. Meanwhile, the implementation of effort estimation through a hybrid approach was only used in 47% of relevant articles. In addition, effort estimation in Agile involved twenty-four attributes, where Complexity, Experience, Size, and Time are the most commonly used and implemented.

Keywords: Agile, effort estimation attributes, expert judgement, hybrid approach, software effort estimation

ARTICLE INFO

Article history:
Received: 18 July 2020
Accepted: 20 October 2020
Published: 30 April 2021

DOI: https://doi.org/10.47836/pjst.29.2.08

E-mail addresses:
pantja@dinamika.ac.id (Pantjawati Sudarmaningtyas)
rozlina@ump.edu.my (Rozlina Mohamed)
*Corresponding author

INTRODUCTION

Software development projects widely use Agile software development method, especially Scrum methodology with iteration planning techniques. This is in accordance with the survey which showed that 94% of

ISSN: 0128-7680
e-ISSN: 2231-8526

respondents employ Agile, and 60% have more than three years of experience. Scrum is the most common methodology used by respondents' organizations (58%), while the top Agile techniques are iteration planning (90%) (VersionOne.com, 2017). Besides, statistics showed that the success rate of Agile is two times more likely to succeed, and one-third less likely to fail than waterfall projects (Mersino, 2018).

Based on a survey, it was found that 45% of IT projects transcend budget because it is not established according to the factual requirement (Bloch et al., 2012). The estimation of cost in software development is essential to avoid excessive costs. In general, the costs are based on effort estimation (Bloch et al., 2012). Therefore, effort estimation is a crucial part of the software development project.

The objective of this estimation process is to provide an approximation of the resources needed to complete a project, in order to deliver outputs in the form of products or services in accordance with the specified characteristics of functional and non-functional (Institute, 2017). Estimates are usually internally generated and periodically conducted. Meanwhile, early effort estimation, schedule, and cost are a repetitive work to be compromised and reviewed between stakeholders to reach an agreement regarding the requirement of resources and time to complete the projects (Bourque & Fairley, 2014).

The precision measurement of a single effort estimate is not straightforward; therefore, the value is better communicated in the interval. Although there is no connection between the implied effort intervals and confidence level, but there is estimator confidence of a possibility that the actual efforts will be within range (Jørgensen, 2016).

The estimate can be interpreted from the requirement of resources, such as people and tools (Bourque & Fairley, 2014). This effort is a composite of person and time, which indicates the number of thoroughly productive working hours necessary to get a work done. Also, the units of effort are typically stated in person-hours, person-days, person-months, and person-years (Trendowicz & Jeffery, 2014).

Agile substantially is quite different from the waterfall method in the manageability of software development. The collaborative and cooperative approach between all stakeholders is the main characteristic of the agile software development method that perform by involving users actively, empower the team to make decisions, capture requirements in lightweight and visual, focus on frequent delivery of products through developing small, incremental release, and iterate (Project-Management.com, 2019).

Following the characteristics of agile that focus on delivery products, effort estimation uses to determine the number of iterations to predict the date delivery project. In contrast, the other software development methodology uses effort estimation as a basis for calculating the cost. Thereby effort estimation remains the main challenge in agile software development projects because there were not yet commonly accepted standardized effort estimation techniques for agile software development. Therefore, this study aimed to review the methods and approaches of software effort estimation, in order to ascertain the most appropriate method for Agile development.

This study was conducted to provide a basis for further development of effort estimation method to be used in Agile software development. Towards achieving the target as mentioned above, this paper was organized into five parts. Beginning with an introduction, followed by overview of effort estimation techniques, materials and methods, results, and discussion, as well as the conclusion.

OVERVIEW OF SOFTWARE EFFORT ESTIMATION METHOD

Currently, many software effort estimation methods are implemented in projects such as Constructive Cost Model (COCOMO), Function point Analysis, Source Line of Code (SLOC), SEER-SEM (Software Evaluation and Estimation of Resources-Software Estimation Model). Furthermore, they are implemented in Linear, Multiplicative, and Putnam Models, Brake Down Estimation, Artificial Neural Network, as well as Fuzzy. The existing methods are shown in Figure 1, and three researchers have classified them according to their perspective.

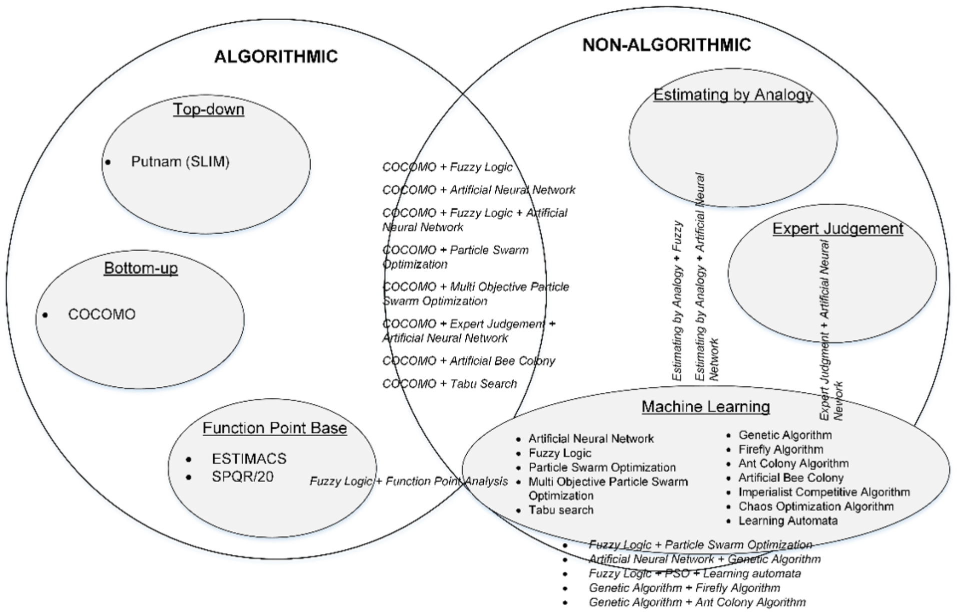
ALGORITHMIC
NON-ALGORITHMIC
Top-down
Putnam (SLIM)
Bottom-up
COCOMO
Function Point Base
ESTIMACS
SPQR/20
Fuzzy Logic + Function Point Analysis
COCOMO + Fuzzy Logic
COCOMO + Artificial Neural Network
COCOMO + Fuzzy Logic + Artificial Neural Network
COCOMO + Particle Swarm Optimization
COCOMO + Multi Objective Particle Swarm Optimization
COCOMO + Expert Judgement + Artificial Neural Network
COCOMO + Artificial Bee Colony
COCOMO + Tabu Search
Estimating by Analogy
Expert Judgement
Estimating by Analogy + Fuzzy
Estimating by Analogy + Artificial Neural Network
Expert Judgment + Artificial Neural Nevork
Machine Learning
Artificial Neural Network
Fuzzy Logic
Particle Swarm Optimization
Multi Objective Particle Swarm Optimization
Tabu search
Genetic Algorithm
Firefly Algorithm
Ant Colony Algorithm
Artificial Bee Colony
Imperialist Competitive Algorithm
Chaos Optimization Algorithm
Learning Automata
Fuzzy Logic + Particle Swarm Optimization
Artificial Neural Network + Genetic Algorithm
Fuzzy Logic + PSO + Learning automata
Genetic Algorithm + Firefly Algorithm
Genetic Algorithm + Ant Colony Algorithm

Figure 1. Existing Software Effort Estimation Method

The first researcher (Boehm, 1984) classified software estimation methods from the perspective of the technique used. Based on this, the classification was divided into seven categories, namely Algorithmic models, Expert Judgment, Analogy, Parkinson, Price-to-Win, Top-Down, and Bottom-Up.

The second researcher (Srivastava & Wadhwa, 2013) classified software estimation methods based on an algorithm. From that perspective, four categories of software effort estimation were obtained, which are Algorithmic, Non-Algorithmic, Parametric, and Machine learning. Some popular methods in Algorithmic models include Function point Analysis, Source Line of Code (SLOC), SEER-SEM (Software Evaluation and Estimation of Resources-Software Estimation Model), Linear Model, Multiplicative Model, Putnam Model, and Constructive Cost Model (COCOMO). Furthermore, Analogy and Expert Judgment are part of Non-algorithmic models, while Brake Down Estimation is one of the methods in Parametric. In addition, the estimation methods that include Machine learning models are Artificial Neural Network and Fuzzy.

The third researcher classified software estimation methods by the data input type and the principle of estimation that was employed (Trendowicz & Jeffery, 2014). Based on this, the methods were classified into three categories, namely Data-driven, Expert-based, and Hybrid. The Data-driven has two groups, which are Proprietary and Non-proprietary. Furthermore, the Non-proprietary is divided into three classes, namely Model-based, Memory-based, and Composite. The Model-based consists of Parametric, Non-parametric, and Semi-parametric.

Table 1 shows a summary of existing methods classification based on the above explanation.

Table 1
Existing Effort Estimation Methods Classification

Classification	(Boehm, 1984)	(Srivastava & Wadhwa, 2013)	(Trendowicz & Jeffery, 2014)
Algorithmic models	√	√	
Expert Judgment/Expert base	√		√
Analogy	√		
Parkinson	√		
Price-to-Win	√		
Top-Down	√		
Bottom-Up	√		
Non-algorithmic		√	
Parametric		√	√
Non-parametric			√
Semi-parametric			√
Machine Learning		√	
Data-driven			√
Hybrid			√

Based on the previous research, this study classified software effort estimation methods based on three aspects, namely (1) estimation principle that is employed (2) estimation strategy that is applied, and (3) data that are required. In general, each of these aspects are divided into two parts. The estimation principle issue consists of algorithmic and non-algorithmic. The facet of the strategy is divided into two types, namely Top-Down and Bottom-Up. While the data aspect indicates that some methods have a high dependency on historical data, but others do not. Figure 2 shows the classification that was used in this study and existing software effort estimation methods.

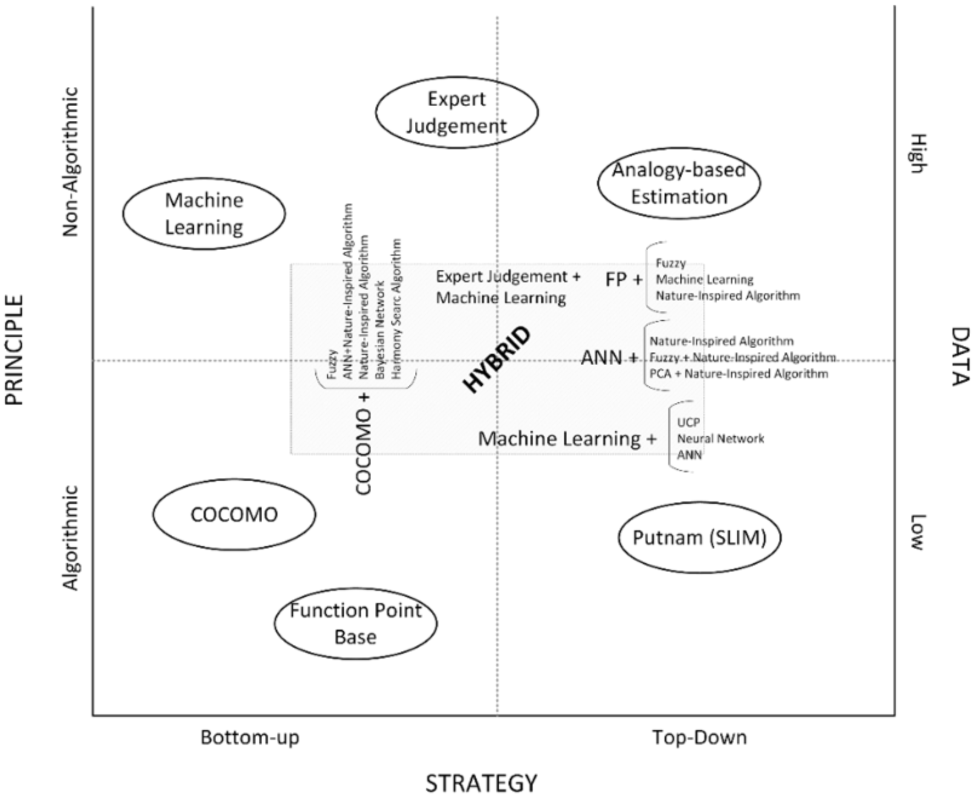

Figure 2. Software Effort Estimation Classification

In the development, the problem in estimation is better resolved when using a combination of several elements, this model is called hybrid. A hybrid model can combine components in a different aspect, and also combine attributes in the aspect of itself. There have been no perfect single estimation methods, therefore it is more suggested to use multiple methods. The confluence amongst the estimates generated by distinct methods shows that the estimation is most likely accurate. In fact, the discrepancy between the estimates indicate that the possibility has been the neglect of certain factors. Therefore, it is essential to determine the factors that cause the difference, and then reexamine to converge result to preferable estimates (Bourque & Fairley, 2014).

MATERIALS AND METHODS

The method of conducting the literature review was based on Kitchenham & Charters (2007) that comprised three main phases, which include planning, conducting, and reporting. Therefore, the development of the systematic literature review protocol in this study consisted of planning and conducting, as shown in Figure 3.

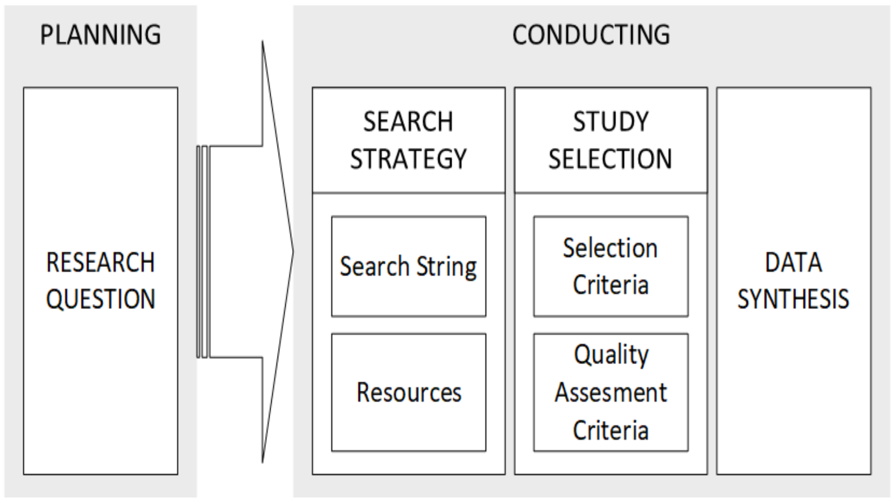

Figure 3. Systematic Literature Review Protocol

Planning

To achieve this research aim, the software effort estimation that is used in Agile project development needs to be known in advance. Next up is knowing how the estimation is implemented and what variables are involved.

The planning stage is the formulation of research questions related to Agile effort estimation as follows:
RQ#1: What kind of method is used for effort estimation in Agile?
RQ#2: How it works to implements estimation effort in Agile?
RQ#3: What are the attributes involved in the estimation of effort in Agile?

Conducting

The conducting phase comprises of three activities, namely search strategy, study selection, and data synthesis. In contrast to data synthesis that is independent, the search strategy and study selection are each formed by two different activities.

On search strategy, a term that is used is known as search strings. This phase also determines resources, where the search will be executed. This process can be conducted after search strings and resources are identified. The strings used in the process is "Agile" ("effort" OR "cost") and "Estimation" ("technique" OR "model").

The process was conducted by embedding the search strings on seven resources as shown in Table 2.

Table 2
List of Resources

No.	Source Name	URL
1	IEEExplore	www.ieeexplore
2	ACM Digital library	https://dl.acm.org/
3	Google Scholar	scholar.google.com
4	Inspec	https://digital-library.theiet.org/
5	ScienceDirect	www.sciencedirect.com
6	SpringerLink	www.springerlink
7	World Scientific	https://www.worldscientific.com

The study selection involved two activities. The first activity was filtering the result of the search process based on inclusion and exclusion criteria. The second activity was to implement quality assessment criteria on the first result.

The inclusion criteria consist of four aspects, which are publication year, the language used, the context of research assessed from title and abstract, and valid DOI. Meanwhile, the exclusion criteria is determined by two aspects, namely the type of work and kind of research. In this case, type of work may be in the form of a journal, conference proceeding, book, chapter, thesis, dissertation, or course material, while the kind of research consists of the study, literature review, comparative study, or survey. The inclusion and exclusion criteria are explained in Table 3.

After the first activity was performed, the next step was selection using quality assessment criteria. As seen in Table 4, the assessment criteria were derived from the research question. The relevant studies fulfilled the quality assessment criteria with an accumulated score value greater than 1.

Data synthesis is an activity to resume the selected studies evidence, in order to synchronize it with the research question. The primary purpose of this activity is to provide clear answers to the research questions.

Table 3
Inclusion and Exclusion Criteria

Inclusion Criteria	Exclusion Criteria
Publication Year: 2009 – 2019	Type of work: Book, Book Chapter, Thesis, Dissertation, course material
Language: English	Kind of Research: Literature Review, Comparative Study, Survey, Report
Title or Abstract: Contains search string	
DOI: valid DOI	

Table 4
Quality Assessment Criteria

ID	Question (Q)	Answer/ Score Points
RQ#1	Does the study discuss the kind of technique that used to estimate effort in Agile?	Yes | Partially | No / 1 | 0.5 | 0
RQ#2	Does the study explain the approach to implements the estimating effort methods in Agile?	Yes | Partially | No / 1 | 0.5 | 0
RQ#3	Does the study declare attributes affecting the estimating of effort in Agile?	Yes | Partially | No / 1 | 0.5 | 0

RESULTS AND DISCUSSION

Conducting Phase Result

The result of the conducting phase from seven resources obtained 171 articles. The number of articles for each resource is shown in Table 5. Furthermore, the top three articles are from Google Scholar (30.99%), SpringerLink (28.07%), and IEEExplore (22.22%).

The inclusion criteria were implemented to obtain 118 articles and exclusion criteria to obtain 100. Table 6 shows the results of both.

Table 7 shows the number of articles of each resource generated from the quality assessment criteria.

Table 5
Result Conducting Phase

No.	Source Name	URL	Number of Articles
1	IEEExplore	www.ieeexplore	38
2	ACM Digital library	https://dl.acm.org/	15
3	Google Scholar	scholar.google.com	53
4	Inspec	https://digital-library.theiet.org/	1
5	ScienceDirect	www.sciencedirect.com	14
6	SpringerLink	www.springerlink	48
7	World Scientific	https://www.worldscientific.com	2
TOTAL			171

Table 6
Result of Inclusion and Exclusion Criteria

No.	Source Name	URL	Number of Articles	
			Inclusion Criteria	Exclusion Criteria
1	IEEExplore	www.ieeexplore	37	33
2	ACM Digital library	https://dl.acm.org/	15	15
3	Google Scholar	scholar.google.com	11	5
4	Inspec	https://digital-library.theiet.org/	1	0
5	ScienceDirect	www.sciencedirect.com	13	12
6	SpringerLink	www.springerlink	40	34
7	World Scientific	https://www.worldscientific.com	1	1
TOTAL			118	110

Table 7
Result of Quality Assessment Criteria

No.	Source Name	URL	Number of Articles
1	IEEExplore	www.ieeexplore	16
2	ACM Digital library	https://dl.acm.org/	6
3	Google Scholar	scholar.google.com	4
4	Inspec	https://digital-library.theiet.org/	0
5	ScienceDirect	www.sciencedirect.com	5
6	SpringerLink	www.springerlink	7
7	World Scientific	https://www.worldscientific.com	0
TOTAL			38

Data Synthesis Phase Results

Effort Estimation Method that Implemented in Agile. According to the relevant articles, effort estimation method that is implemented in Agile can be classified into four, which are Expert Judgement (EJ), Algorithmic (A), Machine Learning (ML), and Statistic (St).

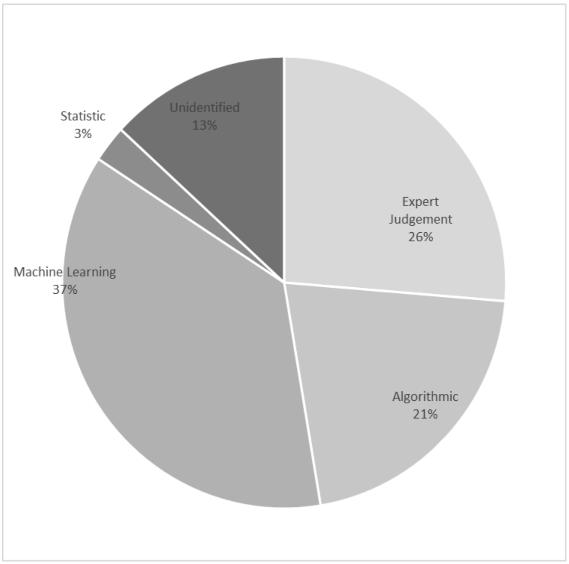

Figure 4. Distribution of Software Effort Estimation Method in Agile

Figure 4 shows the distribution of effort estimation method applied in Agile. From the relevant articles, most of them (87%) clearly stated the method that was used, and the rest (13%) did not describe the method. The top three estimation methods were Machine Learning (37%), Expert Judgement (26%), and Algorithmic (21%).

Expert Judgement Method. All the Expert Judgement method implemented in Agile is based on Planning Poker. Planning Poker is a group of estimation technique recommended for agile software development. The group discussion in this technique is assumed to provide higher accuracy and reduce excessive optimism which is the characteristic of expert judgment-based methods, although there is little empirical evidence about it. Mahnič and Hovelja (2012) conducted a research to fill the gaps by comparing the estimation effort based on the same user stories of the beginner groups and expert groups. The comparing result showed that the people involved in the group estimation process had the expertise, therefore the optimism bias arising from group discussion was diminished.

Lenarduzzi et al. (2015) stated that the estimation efforts conducted by the developers had higher accuracy than those achieved through measuring the functional size. Besides, estimation accuracy in Scrum cannot improve with the help of SiFP and IFPUG Function Points because of weak predictive power.

The two research results above showed that the expertise of professionals in software development companies is the primary aspect that influences the accuracy of estimation through Planning Poker. Furthermore, López-Mart´ınez et al. (2017a) strengthened the statement by conducting a research that proved the existence of dissent between Scrum experts and beginners on the following factors, namely experience, time, effort, priority, and user stories value. From the five factors, experience and time have differences in opinions, while effort and priority have similar opinions. For the user stories value factor, many people doubted considering it as necessary, and some did not apply this factor in the planning phase. In line with this, the industry is likely not to consider this factor.

Although former research showed the existence of different opinions between Scrum experts and beginners, Chatzipetrou et al. (2018) showed that beginners were more engaged with the Planning Poker than estimates. During the Planning Poker activity, the beginners are more interested and involved. This showed that estimation with Planning Poker is a fun activity.

The user story is measured with relative value known as Story points (SP) that are commonly used as the base of calculation in the Planning Poker. Furthermore, Zahraoui and Idrissi (2015) improved the accuracy of effort and time estimates by proposing Adjusted Story Point (ASP) measure instead of SP to calculate the total effort of a scrum project. ASP used three adjustment factors, which are Priority Factor (PF), Story Size Factor (SF), and Complexity Factor (CF). Unfortunately, the use of ASP is not yet applicable to reel scrum projects, therefore, further research is needed to improve the proper values of Story Point Adjustment Factors (SPF).

Although Planning Poker has a lot of benefits, the result was relying on observation experts. To overcome these issues, López-Martínez et al. (2017b) proposed a new model to improve Planning Poker by estimating the complexity and importance of user stories using Bayesian Network. Even though it showed good results primarily to facilitate newbie developers in deciding when to estimate user stories, this model still needed expert knowledge to build a Bayesian Network.

López-Martínez et al. (2018) validated the model that was built on previous studies using the Bayesian network that considered user stories based on its complexity and the level of importance. The validation result showed that effort estimation from professionals had a higher degree of correlation than beginner's estimates. This indicated that factors in the real-world application were also considered in the proposed model. Despite showing promising results, the model needs to be tested on a whole real project. Besides, it needs a mobile application development to ease its implementation.

Tanveer et al. (2017a) proposed an innovative hybrid method that incorporated expert knowledge and changes analysis to impact the process of estimation. This hybrid method was furnished by Tanveer et al. (2018) with a prototype tool built based on the previous framework. That hybrid method can improve the effectiveness of effort estimates.

Moharreri et al. (2016) complemented manual PP with an automated estimation through extracted Agile story cards and their actual effort time. The Auto-Estimate was developed by comparing alternate methods like Naive Bayes, Random Forest, J48, and Logistic Model Tree (LMT), whereas the better result was from the combination of J48+PP, J48, and LMT+PP.

Algorithmic Method. The algorithmic effort estimation method in Agile software development consists of the COSMIC, phase-wise algorithm, Function Point (FP), and Use Case Point (UCP). This part explains the relevant articles that implement the algorithmic estimation method.

COSMIC was proposed by Desharnais et al. (2011) to improve the guess estimate in Agile Project Management (APM). A new procedure was proposed that was built by consolidating COSMIC measurement method at the micro-level (User Stories) and the quality of the documentation for functional analysis deployment. Their study showed that this approach could help the planner to clearly understand why the global effort changed by the time.

The Phase-wise algorithm to compute the estimation effort is proposed by Choudhari and Suman (2012a; 2012b) which conducted an empirical study through a questionnaire to determine efforts of maintenance to compute phase-wise effort estimation. Their postulate provided more realistic results and worthwhile in estimating maintenance effort, especially in extreme programming based on maintenance environments. Therefore, refinement is still needed on the proposed technique.

Function Point (FP) is a part of algorithmic effort estimation classification use by Kang et al. (2010) as an addition to the story point for Agile projects systematic estimation. FP is used to minimize fluctuations of value estimation that is caused by relative values from the user story. Based on a comparison with traditional methods, the addition story point with FP showed better performance.

Excellent performance of FP was applied by Silas et al. (2017) to improve cost estimation in Agile development by proposing the hybridization of Class Responsibility Collaborators models with FP. The process can be boosted through the hybridization adoption of Class Responsibility Collaborators models with FP.

The Use Case Point (UCP) is an effort estimation method that is included in the Algorithmic classification. Khatri et al. (2016) implemented UCP, as an estimation method for Agile development in early stage by emphasizing on main complexity factors like technical and environmental. In this context, the use of the UCP can adequately estimate effort and improve the accuracy based on environment and technical factors under agile development.

UCP can also collaborate with the Scrum Framework by connecting between User Stories on the Product Backlog and Use Case on UCP as performed by Yuliansyah et al. (2018). In their study, UCP adjusted User Stories on Product Backlog by adding the transactions attribute on a user story, and thereby transforming it to Functionality.

Popli and Chauhan (2013) proposed algorithmic estimation method by considering various factors to increase estimation accuracy of release date, cost, effort, and duration, especially for the project Scrum. The algorithm used Sprint-Point Based Framework for Agile and involved two factors, which are project and people-related factors that are proven to be effective and feasible. Although these factors show strong influence on the value of sprints point, in the future, other factors can be added that most affect the estimation.

Machine Learning Method. Most of the relevant articles have the objective of improving accuracy through machine learning as a crucial issue in Agile effort estimation. Some involve the application of machine learning to resolve the metric size of the user story, which is commonly used as the base of estimation in Agile.

To improve the accuracy of Agile effort estimation, a combination of Neural Network was proposed by Panda et al. (2015a; 2015b) and Rao et al. (2018). While Malgonde and Chari (2018) applied predictive algorithms with ensemble-based approaches that consisted of Support Vector Machine (SVM), Artificial Neural Networks (ANN), K-Nearest Neighbors (KNN), and Decision Trees (DT).

Beside Neural Network, the accuracy of Agile effort estimation is also improved using metaheuristic algorithms. Khuat and Le (2017) devised a hybrid model to enhance accuracy in Agile estimation by applying Particle swarm optimization (PSO) and artificial bee colony (ABC). Meanwhile, a novel formula for Agile software effort estimation based on velocity and story points is built from two metaheuristic algorithms mentioned before. However, this hybrid model needs further research on its implementation.

The accuracy in Agile software estimation is also improved using the ontology model. Adnan and Afzal (2017) proposed the model to build a knowledge base by saving significant tacit knowledge during project development. Various agents of the estimation system access the existing knowledge base and autonomously perform a suitable estimate for the success of future projects.

Commonly, Planning Poker (PP) is one of the effort estimations in Agile, and it is based on a user story that is measured by relative values. Some studies overcome this problem by proposing models that extracted keywords from user stories automatically.

Abrahamsson et al. (2011) proposed an Agile effort prediction model that was based on user stories using keyword extraction tools. Meanwhile, Kowalska and Ochodek (2014) proposed a new approach through Semantic Web technologies. However, Choetkiertikul et al. (2018) introduced a novel combination of two robust deep learning architectures, which were long short-term memory and recurrent highway network to estimate story points.

Chongpakdee and Vatanawood (2017) applied document fingerprints to retrieve the similar issues from the public repository of project management assets. In addition to the extraction of the keyword on a user story, Dragicevic et al. (2017) proposed to implement the Bayesian Network model.

To handle frequent requirement changes in Agile software development, two studies had been conducted to resolve this issue. Bilgaiyan et al. (2018) applied Artificial neural networks (ANNs) in Agile effort estimation. Furthermore, the ANN-feedforward back-propagation neural network and Elman neural network are used to keep track, maintain, and estimate the whole product. Meanwhile, Soares (2018) proposed to embed autoencoders in automatic software development effort estimation as text classification.

Even though all those studies provided many advantages, the models still need more massive datasets and other features because user stories are not only written in English, and sometimes influenced by developers' demographics, story criticality, and other systems and framework aspects.

Improved Agile software estimation was conducted by proposing a model based on support vector regression (SVR) that is optimized by the grid search method (GS). In fact, empirical evaluation through the leave-one-out cross-validation method against 21 historical Agile projects demonstrated that this model increases the performance of the SVR technique (Zakrani et al., 2018).

Statistic Method. To make the effort estimation method from the conventional life cycle in accordance with Agile development, Garg and Gupta (2015) proposed a new model by implementing Principal Component Analysis (PCA) to ascertain the key attributes of the development cost. Their study found that the proposed methodology was suitable for Agile projects.

After knowing software effort estimation methods and techniques applied in Agile, the next step is associating software effort estimation techniques with the agile characteristics.

Table 8 shows Planning Poker is a software effort estimation technique that fulfillment the Agile characteristics. Although the Planning Poker suitable for the Agile characteristics, there needs to be further developed to overcome some weaknesses such as a high level of reliance on experts and estimation based mostly on guesses or experience.

Table 8
Mapping software effort estimation method, technique, and characteristic of Agile

Method and Technique	Agile Characteristic				
	C1	C2	C3	C4	C5
EXPERT JUDGEMENT					
Planning Poker	√	√	√	√	√
ALGORITHMIC					
Phase wise	√		√		
COSMIC	√		√		
FP	√		√		
UCP	√		√		
MACHINE LEARNING					
Document Fingerprints					
Bayesian Network					
Classification					
Ontology					
Neural Network Families					
Fuzzy Families					
Metaheuristic Algorithm (ABC & PSO)			√		√
STATISTIC					
PCA					
Impact Analysis					

Note:
C1: Involving users actively, C2: Empower the team to make decisions, C3: capture requirements in lightweight and visual, C4: focus on frequent delivery by developing small product, C5: incremental and iterate

Approach for Estimating Effort in Agile. The implementation of software effort estimation could be achieved through two approaches, which are Non-hybrid and Hybrid. The non-hybrid approach applies a single effort estimation method, whereas the hybrid implements a combination of several methods. Half of the relevant articles used the non-hybrid approach, 36.84% implemented the hybrid, while the rest did not mention the used approach.

The non-hybrid approach consists of effort estimation methods such as Planning Poker, Phase Wise, COSMIC, Function Points, Use Case Point, Document Fingerprints, Bayesian Network, Text Classification, Ontology Model, and Principal Component Analysis. Meanwhile, the Hybrid approach contains method combinations between Expert Judgement and Statistic, Expert Judgement and Machine Learning, and Machine Learning and others. Most of the hybrid approach is a mix of machine learning techniques. Table 9 showed the methods and approach applied in Agile.

Attributes that Affect Estimating Effort in Agile. Volatility and change of customer requirement in Agile software development (ASD) is a difficult and challenging task in estimation effort. Generally, estimation in ASD is mainly based on user story (US) that measures by story points (Zahraoui & Idrissi, 2015). The US is commonly estimated using group processes that improve accuracy compared to individual process (Moløkken-Østvold & Jørgensen, 2004).

Most relevant articles (84.21%) explained the attributes used on effort estimation, and 15.79% did not specifically mention the attributes. Based on the relevant articles, effort in Agile Software Development can be measured by user story, Use Case with sizing method story points, Use Case and function points. Nevertheless, the attribute of being used in the effort estimation can vary significantly.

According to Table 10, there are twenty-four attributes used in effort estimation. The Agile estimation classification that is most employed is Statistic, followed by Expert Judgment and Machine Learning. These attributes are grouped by three criteria, namely the frequency of use, implementation on the effort classification, the recent frequency of use (last three years). Figure 5 shows the mapping of attributes based on those criteria.

The top five highest frequency attributes are Complexity, Story Points, Experience, Size, User Story, Effort, and Time. The implementation of attributes on the Agile effort classification showed that Complexity is implemented on all estimation classification. Whereas, Experience, Function Point, Size, Task, Time, User Story, Weight are attributes applied to three estimation classification. In the last three years, the most attribute that was used in Agile is Complexity, followed by Experience, Size, Effort, and Time. Also, the attributes that fulfilled all criteria are Complexity, Experience, Size, and Time.

Table 9
Types of software effort estimation method and approach

Approach	Method	Technique	Author
Non-hybrid	EJ	Planning Poker	(Chatzipetrou et al., 2018), (López-Mart´ınez et al., 2017a), (Lenarduzzi et al., 2015), (Mahnič & Hovelja, 2012), (Zahraoui & Idrissi, 2015)
	A	Phase wise	(Choudhari & Suman, 2012a), (Choudhari & Suman, 2012b)
	A	COSMIC	(Desharnais et al., 2011)
	A	FP	(Kang et al., 2010)
	A	UCP	(Yuliansyah et al., 2018)
	ML	Document Fingerprints	(Chongpakdee & Vatanawood, 2017)
	ML	Bayesian Network	(Dragicevic et al., 2017)
	ML	Classification	(Soares, 2018)
	ML	Ontology	(Adnan & Afzal, 2017), (Kowalska & Ochodek, 2014)
	St	PCA	(Garg & Gupta, 2015)
Hybrid	EJ, St	EJ and Impact Analysis	(Tanveer et al., 2017b), (Tanveer et al., 2018)
	EJ, ML	Planning Poker and Machine learning (J48, LMT, Bayesian Network)	(Moharreri et al., 2016), (López-Martínez et al., 2017b), (López-Martínez et al., 2018)
	ML & ML	ABC and PSO	(Khuat & Le, 2017)
	ML & ML	Adaptive Neuro-Fuzzy Modelling, Generalized Regression Neural Network and Radial Basis Function Networks (RBFNs)	(Rao et al., 2018)
	ML & ML	Combination of Neural Networks	(Bilgaiyan et al., 2018), (Panda et al., 2015a), (Panda et al., 2015b)
	ML & ML	Combination two deep learning	(Choetkiertikul et al., 2018)
	ML & ML	Combination of Machine Learning such as SVM, SVR, ANN, KNN, and DT.	(Abrahamsson et al., 2011), (Malgonde & Chari, 2018), (Zakrani et al., 2018)

Table 10
Effort Estimation Attributes

No.	Attribute	SEE Method				Frequency of use									
		EJ	A	ML	St	2010	2011	2012	2013	2014	2015	2016	2017	2018	Total
1	Application Type				▨						1				1
2	Architecture				▨						1				1
3	Complexity	▨	▨	▨	▨					3	1	3	5	2	14
4	Dependencies				▨							1	1	1	3
5	Effort	▨		▨						1	2		2	2	7
6	Environmental		▨											1	1
7	Experience	▨		▨	▨							2	4	2	8
8	Function Point	▨	▨		▨		1				2				3
9	Impact	▨			▨							1	1		2
10	Knowledge	▨			▨							1	1		2
11	Maturity				▨						3				3
12	Platform				▨						3				3
13	Previously Estimates	▨			▨							1	1		2
14	Priority	▨		▨									1	2	3
15	Size	▨		▨	▨	1					1		1	5	8
16	Skill			▨									1		1
17	Story Points			▨						5	2		1	2	10
18	Sprint Points			▨					3					1	4
19	Task	▨		▨	▨					1				2	3

Table 10 *(Continued)*

No.	Attribute	SEE Method				Frequency of use									
		EJ	A	ML	St	2010	2011	2012	2013	2014	2015	2016	2017	2018	Total
20	Time	■	■	■			1		1				3	1	6
21	Use Case Point		■									1		1	2
22	User Story	■	■	■			1	3		1	1	1	1		8
23	Velocity			■					1		2		1	1	5
24	Weight	■	■		■									2	3

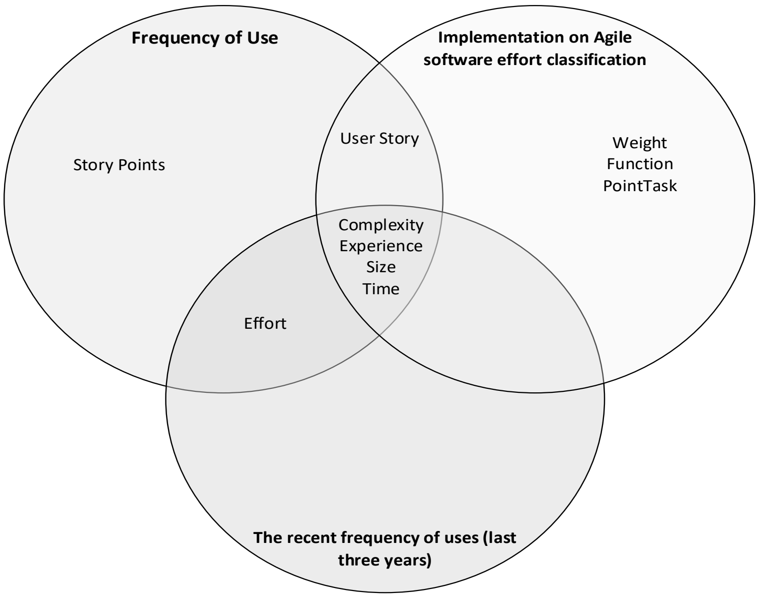
Frequency of Use
Implementation on Agile software effort classification
Story Points
User Story
Weight
Function
PointTask
Complexity
Experience
Size
Time
Effort
The recent frequency of uses (last three years)

Figure 5. Mapping of Agile software effort estimation attributes

Complexity attribute is interpreted in different aspects, and most study understands it as the complexity of the project (Popli & Chauhan, 2014b; Garg & Gupta, 2015; Tanveer et al., 2016; Tanveer et al., 2017b; Bilgaiyan et al., 2018). Some research used it to represent technical complexity (Hamouda, 2014; Khatri et al., 2016; Yuliansyah et al., 2018). Also, environmental complexity is a part of the attribute that applies in two research (Hamouda, 2014; Khatri et al., 2016). In fact, the attribute is considered to represent form, function, report, and requirements complexities (Dragicevic et al., 2017).

Many studies used story point attribute that represent itself (Popli & Chauhan, 2014a; Hamouda, 2014; Panda et al., 2015a; Panda et al., 2015b; Khuat & Le, 2017; Zakrani et al., 2018; Rao et al., 2018). Meanwhile, others apply this to show baseline story-points, estimated story-points (ESP), and unadjusted value of story-points (Popli & Chauhan, 2014b).

Experience attribute in most studies is used to measure the implementation experience of the developer or programmer (Tanveer et al., 2016; Tanveer et al., 2017b; López-Mart´ınez et al., 2017a; López-Martínez et al., 2017b; López-Martínez et al., 2018; Malgonde & Chari, 2018). Other studies assess this attribute based on developer's experience in making estimation (Tanveer et al., 2016; Tanveer et al., 2017b).

Some studies apply size attribute that represents the value of the user story (López-Martínez et al., 2017b; López-Martínez et al., 2018), the code metrics for each affected class (Tanveer et al., 2018), and the team size (Garg & Gupta, 2015). However, other studies do not detail the size attribute (Kang et al., 2010; Bilgaiyan et al., 2018; Malgonde & Chari, 2018; Tanveer et al., 2018).

Effort estimation in Agile software development is mainly based on the user story attribute (Choudhari & Suman, 2012a; Choudhari & Suman, 2012b; Mahnič & Hovelja, 2012; Popli & Chauhan, 2014a; Zahraoui & Idrissi, 2015; Chongpakdee & Vatanawood, 2017; Choetkiertikul et al., 2018). In Abrahamsson et al. (2011), the user story was elaborated into keywords, namely length, and priority.

Effort can be assumed as effort per person (Popli & Chauhan, 2014a) or actual effort (Panda et al., 2015a; Panda et al., 2015b). However, the attribute also represents itself without supplement (López-Martínez et al., 2017a; López-Martínez et al., 2017b; López-Martínez et al., 2018; Malgonde & Chari, 2018).

Commonly, the attribute of time refers to the time it takes to complete a project (Popli & Chauhan, 2013; López-Martínez et al., 2017a; López-Martínez et al., 2017b; López-Martínez et al., 2018). In some studies, it is explicitly explained as Person-hours (Desharnais et al., 2011) or Working Hours (Dragicevic et al., 2017).

CONCLUSION

Machine learning is the most common effort estimation method used in Agile software development, followed by Expert judgment and Algorithmic. However, machine learning has limitation in its implementation because it needs very large dataset that source from expert's knowledge.

The implementation approach of the estimation effort in Agile software development does not differ between the non-hybrid and hybrid. The non-hybrid approach is applied on 50.00% of relevant articles and the hybrid is applied on 36.84%.

Twenty-four attributes are involved in Agile effort estimation, and those with the top five highest frequency of use are Complexity, Story Points, Experience, Size, User Story, Effort, and Time. Furthermore, the attribute that is implemented on all classification of the estimation is Complexity. Those with the highest frequency of use in the last three years are Complexity, Experience, Size, Effort, and Time. In addition, those that fulfilled all criteria are Complexity, Experience, Size, and Time.

ACKNOWLEDGEMENT

Pantjawati Sudarmaningtyas would like to thank her supervisor and her institution for providing support to this research, especially for Universitas Dinamika, which had provided financial support. Pantjawati Sudarmaningtyas is also grateful to colleagues and peer reviewers who saved her from many mistakes through comments, feedback, and constructive criticism. Pantjawati Sudarmaningtyas hope this research can give benefits and contribute to further research.

REFERENCES

Abrahamsson, P., Fronza, I., Moser, R., Vlasenko, J., & Pedrycz, W. (2011, September 22-23). Predicting development effort from user stories. In *2011 International Symposium on Empirical Software Engineering and Measurement* (pp. 400-403). Banff, Canada. https://doi.org/10.1109/ESEM.2011.58

Adnan, M., & Afzal, M. (2017). Ontology based multiagent effort estimation system for scrum agile method. *IEEE Access*, *5*, 25993-26005. https://doi.org/10.1109/ACCESS.2017.2771257

Bilgaiyan, S., Mishra, S., & Das, M. (2018). Effort estimation in agile software development using experimental validation of neural network models. *International Journal of Information Technology, 11*, 569-573. https://doi.org/10.1007/s41870-018-0131-2

Bloch, M., Blumberg, S., & Laartz, J. (2012). *Delivering large scale IT.pdf*. Retrieved October 23, 2017, from https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value

Boehm, B. W. (1984). Software engineering economics. *IEEE Transactions on Software Engineering*, *SE-10*(1), 4-21. https://doi.org/10.1109/TSE.1984.5010193

Bourque, P., & Fairley, R. E. (2014). *Guide to the software engineering body of knowledge*. IEEE Computer Society Press. https://doi.org/10.1234/12345678

Chatzipetrou, P., Ouriques, R., & Gonzalez-Huerta, J. (2018). Approaching the relative estimation concept with planning poker. In *Proceedings of the 7th Computer Science Education Research Conference (CSERC '18)* (pp. 21-25). Association for Computing Machinery. https://doi.org/10.1145/3289406.3289409

Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., & Menzies, T. (2018). A deep learning model for estimating story points. *IEEE Transactions on Software Engineering, 45*(7), 637-656.

Chongpakdee, P., & Vatanawood, W. (2017, November 24-26). Estimating user story points using document fingerprints. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, (pp. 149-152). Beijing, China. https://doi.org/10.1109/ICSESS.2017.8342885

Choudhari, J., & Suman, U. (2012a). Phase wise effort estimation for software maintenance: An extended SMEEM model. In *Proceedings of the CUBE International Information Technology Conference* (pp. 397-402). Association for Computing Machinery. https://doi.org/10.1145/2381716.2381790

Choudhari, J., & Suman, U. (2012b). Story Points based effort estimation model for software maintenance. *Procedia Technology*, *4*, 761-765. https://doi.org/https://doi.org/10.1016/j.protcy.2012.05.124

Desharnais, J. M., Buglione, L., & Kocatürk, B. (2011). Using the COSMIC method to estimate agile user stories. In *Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement* (pp. 68-73). Association for Computing Machinery. https://doi.org/10.1145/2181101.2181117

Dragicevic, S., Celar, S., & Turic, M. (2017). Bayesian network model for task effort estimation in agile software development. *Journal of Systems and Software*, *127*, 109-119. https://doi.org/10.1016/j.jss.2017.01.027

Garg, S., & Gupta, D. (2015, March 3-5). PCA based cost estimation model for agile software development projects. In *2015 International Conference on Industrial Engineering and Operations Management (IEOM)* (pp. 1-7). Dubai, United Arab Emirates. https://doi.org/10.1109/IEOM.2015.7228109

Hamouda, A. E. D. (2014, July 28-August 1). Using agile story points as an estimation technique in CMMI organizations. In *2014 Agile Conference* (pp. 16-23). Kissimmee, FL, USA. https://doi.org/10.1109/AGILE.2014.11

Institute, P. M. (2017). *A guide to the project management body of knowledge (PMBOK® Guide)-Sixth edition*. Project Management Institute, Inc.

Jørgensen, M. (2016). The use of precision of software development effort estimates to communicate uncertainty. In *8th International Conference on Software Quality Days (SWQD)* (pp. 156-168). Springer. https://doi.org/10.1007/978-3-319-27033-3_11

Kang, S., Choi, O., & Baik, J. (2010, August 18-20). Model-based dynamic cost estimation and tracking method for agile software development. In *2010 IEEE/ACIS 9th International Conference on Computer and Information Science* (pp. 743-748). Yamagata, Japan. https://doi.org/10.1109/ICIS.2010.126

Khatri, S. K., Malhotra, S., & Johri, P. (2016, September 7-9). Use case point estimation technique in software development. In *2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)* (pp. 123-128). Noida, India. https://doi.org/10.1109/ICRITO.2016.7784938

Khuat, T. T., & Le, M. H. (2017). A novel hybrid ABC-PSO algorithm for effort estimation of software projects using agile methodologies. *Journal of Intelligent Systems*, *27*(3), 489-506. https://doi.org/10.1515/jisys-2016-0294

Kitchenham, B., & Charters, S. (2007). *Guidelines for performing systematic literature reviews in software engineering* (Version 2.3). Department of Computer Science, Keele University. https://doi.org/10.1145/1134285.1134500

Kowalska, J., & Ochodek, M. (2014). Supporting analogy-based effort estimation with the use of ontologies. *E-Informatica Software Engineering Journal*, *8*(1), 53-64. https://doi.org/10.5277/e-Inf140104

López-Martínez, J., Ram´ırez-Noriega, A., Ju´arez-Ram´ırez, R., Licea, G., & Mart´ınez-Ram´ırez, Y. (2017a). Analysis of planning poker factors between university and enterprise. In *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)* (pp. 54-60). Conference Publishing Services. https://doi.org/10.1109/CONISOFT.2017.00014

López-Martínez, J., Juárez-Ramírez, R., Ramírez-Noriega, A., Licea, G., & Navarro-Almanza, R. (2017b). Estimating user stories' complexity and importance in scrum with Bayesian networks. In *World Conference on Information Systems and Technologies* (pp. 205-214). Springer. https://doi.org/10.1007/978-3-319-56535-4_21

López-Martínez, J., Ramírez-Noriega, A., Juárez-Ramírez, R., Licea, G., & Jiménez, S. (2018). User stories complexity estimation using Bayesian networks for inexperienced developers. *Cluster Computing*, *21*(1), 715-728. https://doi.org/10.1007/s10586-017-0996-z

Lenarduzzi, V., Lunesu, I., Matta, M., & Taibi, D. (2015). Functional size measures and effort estimation in agile development: A replicated study. In *International Conference on Agile Software Development* (pp. 105-116). Springer. https://doi.org/10.1007/978-3-319-18612-2_9

Mahnič, V., & Hovelja, T. (2012). On using planning poker for estimating user stories. *Journal of Systems and Software*, *85*(9), 2086-2095. https://doi.org/10.1016/j.jss.2012.04.005

Malgonde, O., & Chari, K. (2018). An ensemble-based model for predicting agile software development effort. *Empirical Software Engineering, 24*, 1017-1055. https://doi.org/10.1007/s10664-018-9647-0

Mersino, A. (2018). *Agile project success rates are 2x higher than traditional projects (2019)*. Retrieved January 27, 2020, from https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/

Moharreri, K., Sapre, A. V., Ramanathan, J., & Ramnath, R. (2016). Cost-effective supervised learning models for software effort estimation in agile environments. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)* (pp. 135-140). Conference Publishing Services. https://doi.org/10.1109/COMPSAC.2016.85

Moløkken-Østvold, K., & Jørgensen, M. (2004). Group processes in software effort estimation. *Empirical Software Engineering*, *9*(4), 315-334. https://doi.org/10.1023/B:EMSE.0000039882.39206.5a

Panda, A., Satapathy, S. M., & Rath, S. K. (2015a). Empirical validation of neural network models for agile software effort estimation based on story points. *Procedia Computer Science*, *57*, 772-781. https://doi.org/https://doi.org/10.1016/j.procs.2015.07.474

Panda, A., Satapathy, S. M., & Rath, S. K. (2015b). Neural network models for agile software effort estimation based on story points. *Proceedings of the International Conference on Advances in Computing, Control and Networking*, *57*(57), 26-30. https://doi.org/10.15224/978-1-63248-038-5-06

Popli, R., & Chauhan, N. (2013, March 9-10). A sprint-point based estimation technique in Scrum. In *2013 International Conference on Information Systems and Computer Networks* (pp. 98-103). Mathura, India. https://doi.org/10.1109/ICISCON.2013.6524182

Popli, R., & Chauhan, N. (2014a, February 6-8). Cost and effort estimation in agile software development. In *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)* (pp. 57-61). Faridabad, India. https://doi.org/10.1109/ICROIT.2014.6798284

Popli, R., & Chauhan, N. (2014b, March 1-2). Estimation in agile environment using resistance factors. In *2014 International Conference on Information Systems and Computer Networks (ISCON)* (pp. 60-65). Mathura, India. https://doi.org/10.1109/ICISCON.2014.6965219

Project-Management.com. (2019). *10 key principles of agile software development*. Retrieved January 11, 2020, from Project Management.com website: https://project-management.com/10-key-principles-of-agile-software-development/

Rao, C. P., Kumar, P. S., Sree, S. R., & Devi, J. (2018). An agile effort estimation based on story points using machine learning techniques. In *Proceedings of the Second International Conference on Computational Intelligence and Informatics* (pp. 209-219). Springer. https://doi.org/10.1007/978-981-10-8228-3_20

Silas, F. A., Yusuf, M., & Bijik, A. H. (2017). Hybridization of class responsibility collaborators model (HCRCM) with function point to enhance project estimation cost in agile software development. *Circulation in Computer Science*, *2*(6), 20-24. https://doi.org/10.22632/ccs-2017-252-32

Soares, R. G. F. (2018, July 8-13). Effort estimation via text classification and autoencoders. In *2018 International Joint Conference on Neural Networks (IJCNN)* (pp. 01-08). Rio de Janeiro, Brazil. https://doi.org/10.1109/IJCNN.2018.8489030

Srivastava, B., & Wadhwa, M. (2013). Relative analysis of software cost and effort estimation techniques. *International Journal of Computer Science and Engineering (IJCSE)*, *2*(3), 53-68.

Tanveer, B., Vollmer, A. M., & Engel, U. M. (2017a). Utilizing change impact analysis for effort estimation in agile development. In *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 430-434). Conference Publishing Services. https://doi.org/10.1109/SEAA.2017.64

Tanveer, B., Guzman, L., & Engel, U. M. (2017b). Effort estimation in agile software development: Case study and improvement framework. *Journal of Software-Evolution and Process*, *29*(11), 1-14. https://doi.org/10.1002/smr.1862

Tanveer, B., Guzmán, L., & Engel, U. M. (2016). Understanding and improving effort estimation in agile software development. In *Proceedings of the International Workshop on Software and Systems Process - ICSSP '16* (pp. 41-50). Association for Computing Machinery. https://doi.org/10.1145/2904354.2904373

Tanveer, B., Vollmer, A. M., & Braun, S. (2018). A hybrid methodology for effort estimation in agile development: An industrial evaluation. In *Proceedings of the 2018 International Conference on Software and System Process* (pp. 21-30). Association for Computing Machinery. https://doi.org/10.1145/3202710.3203152

Trendowicz, A., & Jeffery, R. (2014). *Software project effort estimation: Foundation and best practice guidelines for success*. Springer. https://doi.org/10.1007/978-3-319-03629-8

VersionOne.com. (2017). 11th Annual state of agile report. *VersionOne Agile Annual Report*, 1-16. https://doi.org/10.1093/jicru/ndl025

Yuliansyah, H., Qudsiah, S. N., Zahrotun, L., & Arfiani, I. (2018). Implementation of use case point as software effort estimation in scrum framework. *IOP Conference Series: Materials Science and Engineering*, *403*(1), 1-10. https://doi.org/10.1088/1757-899X/403/1/012085

Zahraoui, H., & Idrissi, M. A. J. (2015, October 20-21). Adjusting story points calculation in scrum effort & time estimation. In *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)* (pp. 1-8). Rabat, Morocco. https://doi.org/10.1109/SITA.2015.7358400

Zakrani, A., Najm, A., & Marzak, A. (2018, October 21-27). Support vector regression based on grid-search method for agile software effort prediction. In *2018 IEEE 5th International Congress on Information Science and Technology (CiSt)* (pp. 1-6). Marrakech, Morocco. https://doi.org/10.1109/CIST.2018.8596370