Artificial Intelligence Review (2022) 55:5485–5516
https://doi.org/10.1007/s10462-021-10132-x

Data-driven effort estimation techniques of agile user stories: a systematic literature review

Bashaer Alsaadi[1,2] · **Kawther Saeedi**[2]

Published online: 10 January 2022

Abstract
At an early stage in the development process, a development team must obtain insight into the software being developed to establish a reliable plan. Thus, the team members should investigate, in depth, any information relating to the development. A major challenge for developers is software development effort estimation (SDEE), which refers to gauging the amount of effort needed to develop the software. In agile methodologies, a project is delivered in iterations, each of which delivers a set of requirements known as user stories. Therefore, SDEE in agile focuses on estimating a single user story's effort, not the project as a whole, as in traditional development. Among the various techniques, data-driven methods have proved effective in effort estimation, as they are unaffected by external pressure from managers. Moreover, no experts have to be available at the point when estimation is undertaken. By conducting a systematic literature review, this study presents a comprehensive overview of data-driven techniques for user story effort estimation. The results show that there has been limited work on this topic. Studies were analysed to address questions covering five main points: technique; performance evaluation method; accuracy, independent factors (effort drivers); and the characteristics of the datasets. The main performance evaluation methods are performance measures, baseline benchmarks, statistical tests, distribution of estimates, comparison against similar existing techniques and human estimation. Four types of independent factors were identified: personnel; product; process; and estimation. Furthermore, the story point was found to be the most frequently used effort metric in agile user stories.

Keywords Effort estimation · Agile · User story · Systematic literature review · Data-driven · Machine learning

✉ Bashaer Alsaadi
b.alsaadi@seu.edu.sa

[1] Information Technology Department, College of Computing and Informatics, Saudi Electronic University, Jeddah, Saudi Arabia

[2] Information Systems Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

1 Introduction

Software development effort estimation (SDEE) is a key factor in software management (Trendowicz and Jeffery 2014). It relates to estimating the effort required to complete either a whole project or a certain task in that project. Achieving a highly accurate estimate that reflects the actual workload is always a major concern for developers. Its accuracy plays a critical role in software development (Khuat and Thi My Hanh 2016)(Arora et al. 2020). Indeed, an inappropriate or unrealistic estimate may threaten a project's success. One study surveyed of more than a thousand IT developers and professionals (Rosencrance 2007), and its findings indicate that two of the three main reasons for the failure of a software project relate to effort estimation. An inaccurate estimate has detrimental consequences leading to the misallocation of resources (Nassif et al. 2019). For instance, over-assigning developers to a certain project may result in missed opportunities in other projects. By contrast, underestimation has consequences such as potentially threatening an organization's reputation (Abrahamsson et al. 2011), as the development will not receive the attention it warrants, potentially resulting in poor-quality software (Conoscenti et al. 2019).

According to Trendowicz and Jeffery (2014), SDEE methods fall into two main categories: data-driven; and expert judgment. In the former, estimators use historical data from similar completed projects containing a set of project characteristics, besides effort values, to estimate the effort needed in a current project. In the latter, experts are consulted in the estimation process to assign an appropriate value to the effort required to finish a certain function or whole project.

Data-driven methods to estimate effort, which comprise the focus of this review, do not need experts to be available each time and are unaffected by external pressure (Jørgensen et al. 2009) that comes from managers or customers. Several methods have been proposed, using advanced technologies such as artificial intelligence, machine learning, and deep learning. Development effort in a project that employs agile methodology has drivers that differ from those in a project that uses traditional software development methodology (Choetkiertikul et al. 2019). The use of incremental software development in agile methodology is now a common and mature approach. In such projects software is delivered incrementally, with each increment containing a set of tasks known as 'user stories' (Rubin 2013); thus, effort estimation in agile projects focuses on these rather than on the whole project, as in most earlier studies. There has been limited work on the former (Choetkiertikul et al. 2019), prompting this investigation into state-of-the-art techniques for a data-driven effort estimation of agile user stories.

The rest of this article is organized as follows. Section 2 presents the background of effort estimation in both software development and agile development, then discusses recent related works. Section 3 presents the methods adopted in this review. Sections 4 and 5 present, analyse, and discuss the result of the review, while section 6 concludes this review.

2 Literature review

This section first presents the background of software development effort estimation and its challenges. It briefly describes agile development effort estimation, then discusses recent related works.

2.1 Software development effort estimation methods

According to Trendowicz and Jeffery (2014), SDEE methods fall into two main categories: data-driven and expert judgment methods. The following two sections briefly describe these SDEE categories.

2.1.1 Expert-Based Effort Estimation

In expert-based methods, experts are involved in the estimation process. They are consulted to assign an appropriate value to the effort needed to finish a certain function or whole project. Experts make estimates based on their experience, as well as by analysing the information available on that project, including project requirements, project size, and available resources and tools. Estimation may be undertaken by a single expert, who may use a bottom-up strategy to estimate project effort (Trendowicz and Jeffery 2014). In this strategy, the estimator breaks down the whole project or main tasks into smaller tasks; these are then estimated separately. Finally the summation of these efforts represents the total project's estimated effort. The development effort can also be estimated by using a group of experts. Moløkken-Østvold and Jørgensen (2004) have shown that multi-expert estimation is less optimistic than that undertaken by a single expert. They found that when a group conducts a discussion, its estimation becomes more realistic and better reflects the actual effort. In-depth discussion enables the estimators to explore the project's activities in more detail. Therefore, they are more accurate in their estimation of the effort required. Multi-expert estimation may be either structured or unstructured (Trendowicz and Jeffery 2014). Unstructured group estimation means that the experts simply discuss, without following any specific approach, until they come to a final decision on the estimated effort. Meanwhile, in the structured type, the experts rely on a systematic approach and techniques.

One of the structured techniques for group estimation is Delphi (Rowe and Wright 1999), designed in the 1950s, which focuses on anonymous experts' estimation. Each provides written feedback to a coordinator, and the process undergoes several iterations until the majority reaches an agreement. Another multi-expert estimation technique widely used in agile methodology is Planning Poker (Haugen 2006). In contrast to Delphi, here the experts are involved in face-to-face discussion. Effort values are represented by cards, each with a different value. The experts privately pick a card that they feel is appropriate to the effort involved. Next, all the cards are revealed at the same time and the experts discuss the value. If there is no consensus, they re-estimate until they agree upon a specific value.

2.1.2 Data-driven effort estimation

In data-driven methods, as shown in Fig. 1, to estimate the effort required for a current project the estimators use historical data from similar completed projects. These data, besides the effort values, include a set of project characteristics. Thus the dataset is a key component in any data-driven technique. Even if the applied algorithm or method is outstanding, an accurate estimate cannot be reached if it is derived from unrealistic information and factors (Trendowicz and Jeffery 2014). Therefore, the appropriate selection of the independent factors (effort drivers) can lead to successful estimation. Data-driven estimation can use either model-based or analogy-based methods (Trendowicz and Jeffery 2014). Model-based methods use historical data as the input to a model that outputs the effort value.

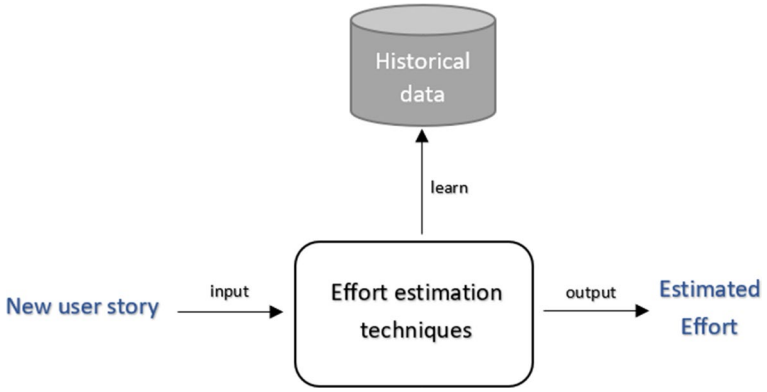

Fig. 1 Data driven effort estimation techniques

Many applications in the literature examine the use of data-driven methods for effort estimation. For example, Panda et al. (2015) investigate the effectiveness of applying neural network models in agile SDEE using historical data from six software houses. By contrast, analogy-based methods use historical data as a case base to search for the most nearly identical previous project, termed the analogue, then reuse its actual effort as an effort value in the current project. One good example is case-based reasoning (CBR), successfully adapted to SDEE (Trendowicz and Jeffery 2014). Several studies have shown the use of analogy-based methods to estimate software development efforts, such as by Phannachitta (2020).

However, in terms of intelligent approaches, machine learning (ML) is widely used in effort estimation studies (Wen et al. 2012). ML is a model-based method that teaches machines to think similarly to humans by learning from previous experience and knowledge (Dam 2019). ML has two main techniques: supervised learning and unsupervised learning (Alloghani et al. 2020). In the former, there are known input and output variables, and the machine will learn to address a mapping function between those variables to predict the output of a new input variable. Classification and regression are the two supervised learning techniques used in ML. The latter learns and discovers the hidden patterns from unlabeled data. The widely used in effort estimation is the supervised learning (Saeed et al. 2018). A brief comparison has been made below between the intelligent approaches used for effort estimation in terms of the data type: structured or unstructured data.

– *Structured data* Structured data usually comes in the form of a relational database, that contains both categorical and numerical features. These features should be carefully chosen, as the success of an effort estimation model depends on such features (Trendowicz and Jeffery 2014). One main limitation of intelligent approaches that deal with this type of data, is that the features are manually defined. Therefore, one should define and select effective features to develop an accurate model. Manual features engineering consumes more time, moreover, domain knowledge is required (Choetkiertikul et al. 2019). Furthermore, features will be designed with consideration of the context of data; therefore, they may not be suitable in other contexts (Choetkiertikul et al. 2019). Regardless of its limitation, the use of this type of data enables experts to define features that may not exist in the textual description of user stories, for example, develop-

ers' experience (Malgonde and Chari 2019), complexity (Dragicevic et al. 2017), and size (Alostad et al. 2017).

– *Unstructured data* In effort estimation, unstructured data contains only textual information about projects/user stories. Sometimes it contains programming commands. Intelligent approaches (e.g. ML) that deal with unstructured data try to predict the effort of user stories from the textual description. However, before using the estimation models as ML, the features should be extracted. Different studies manually defined a set of text-related features that contribute to effort estimation. Some of these features are: presence of keywords or tokens (Moharreri et al. 2016)(Abrahamsson et al. 2011), number of characters (Abrahamsson et al. 2011), and word or token frequency (Moharreri et al. 2016). Other studies used feature extraction methods instead of manual design, such as autoencoder (Soares 2018). With the emergence of more advanced intelligent approaches such as deep learning, there is no need to extract the features manually or before using the model (Choetkiertikul et al. 2019). Choetkiertikul et al. (2019) indicates that using a deep learning model for effort estimation overcomes the limitation of ML models in terms of feature extraction. Choetkiertikul et al. (2019) proposed a deep learning model that automatically learned the features, there is no need for experts to manually define the features. Moreover, the model can be generalized for several project contexts, as the features were not designed to suit a certain context. Despite its pros, the performance of these intelligent approaches relies heavily on the correctness of writing and describing a user story. In fact, some models perform worse when user stories are written in a short and informal language (Abrahamsson et al. 2011).

2.1.3 Software development effort estimation challenges

Developers constantly struggle to achieve the highest effort estimation accuracy. The challenges are as follows.

1. Inexperienced team members. One of the reasons for an inaccurate estimate is the fact that it is undertaken by an inexperienced team (Chongpakdee and Vatanawood 2017), that is unfamiliar with the project tasks or lacks sufficient knowledge about how to gauge the development effort (López-Martínez et al. 2018). One study aimed to investigate the difference between expert and non-expert estimation (students) using the Planning Poker technique (Mahnič and Hovelja 2012). It found that experts' estimates are more accurate.
2. External pressure. This pressure is from managers or customers and may result in either overestimation or underestimation of the required tasks' effort (Altaleb et al. 2020). For example, to satisfy the customer or the manager, effort may be involved to deliver the work at the ideal time, causing an underestimation that results in poor-quality software (Popli and Chauhan 2014)(Altaleb et al. 2020).
3. Task sequence. Some researchers believe that the sequence in which a project's tasks are estimated has an impact on estimators; thus, it affects the estimate's accuracy. Grimstad and Jorgensen (2009) argue that if there is any actual effect from the order followed, there may be an optimum sequence in which to produce the most accurate estimate. They conducted an experiment on two groups: (1) estimating small-size requirements and (2) estimating large-size requirements. Both groups then asked to estimate medium-size requirements. The results show that the group that estimated small requirements tended to underestimate the medium requirements, whereas the other group tended to overesti-

mate. A similar recent experiment involving 104 software professionals (Jørgensen and Halkjelsvik 2020) reported that large tasks are underestimated when they are tackled after small tasks, while a small task is overestimated when an estimate for a large task has come beforehand.

A possible solution to such challenges is an automated data-driven effort estimation method, for the following reasons. First, data-driven methods could eliminate or correct the bias of human-based estimation. A human may be affected by several aspects that a data-driven method is not. For example, task sequence and external pressure as discussed above. Other aspects that may increase human bias include thinking style, skill, education, and organizational role (Jorgensen and Grimstad 2012). Second, expert involvement is not required every time a team must estimate a project's effort (Trendowicz and Jeffery 2014). The data-driven technique learns and gains knowledge from previous projects (historical data); therefore, it will overcome the limitations of an inexperienced team without the need for expert involvement. Third, an automated data-driven method costs less than expert-based estimation (Trendowicz and Jeffery 2014). Finally, as compared to expert estimation, data-driven methods are more reusable and adaptable for new projects with a similar context (Trendowicz and Jeffery 2014).

2.2 Agile software development

The agile development method ensures a project's rapid production and fast delivery. As software is delivered in iterations, each iteration comprises a complete working piece (one or more user stories) of the project. Agile is a widely used Software Development Life Cycle (SDLC) (Serrador and Pinto 2015). Its popularity stems from the advantages that a team can obtain from following such a methodology. The agility of this methodology in developing complex products helps organizations deliver fast releases at a lower cost (Rubin 2013). Furthermore, with the emergence of advanced technologies and the increasing level of competition, customer requirements are constantly changing during the development life cycle. Agile can effectively respond to these changes (Matharu et al. 2015) by involving customers throughout several development phases. This involvement allows the team to continuously take customers' feedback and increase their satisfaction. There are several agile methodologies, including Scrum, Extreme Programming (XP), and Kanban (Matharu et al. 2015).

An agile team is a self-organizing team (Schwaber and Sutherland 2020), which means that they direct the whole development process. Moreover, an agile team is a cross-functional team (Schwaber and Sutherland 2020), with all the required skills and capabilities to deliver iterations by covering different development phases, including designing, building, and testing.

One of the most adopted agile methodologies is Scrum (Rubin 2013). Scrum is used for delivering complex products incrementally. Each increment or iteration is called a sprint. A sprint is a fixed-length iteration in which a cross-functional team is involved in delivering a working valuable feature or a set of features (Schwaber and Sutherland 2020). The scrum team has three roles (Schwaber and Sutherland 2020): (1) Product owner, who should ensure that all items in the product backlog are placed and prioritized correctly; (2) Scrum master, who is responsible for helping the team deliver high-value increments within the estimated time; and (3) Development team, which consists of a set of professionals,

including UI designers, testers, and programmers, who have the required skills to deliver the sprint being developed.

2.2.1 Effort estimation of agile software development

As shown in Fig. 2, software products that use agile methodologies (such as Scrum) involve a set of requirements known as user stories (Rubin 2013). Various factors decide which user story/stories should be delivered by the end of the next sprint, such as its estimated effort; however, the theory behind the method makes the process of effort estimation dissimilar from traditional SDEE. Instead of estimating the effort needed to complete the entire project, the agile team estimates the effort necessary to complete a single user story (Choetkiertikul et al. 2019). Each user story is assigned an effort value, commonly known as story point (SP). These story points can be scaled in several ways, such as the Fibonacci sequence (López-Martínez et al. 2018) (1, 2, 3, 5, 8, 13, etc.) and the T-shirt sizing technique (Rubin 2013) (XS, S, M, L, XL, etc.). Studies that have investigated effort estimation algorithms and methods for agile user stories are limited in number compared to those looking at traditional projects (Chongpakdee and Vatanawood 2017).

2.3 Related works

This section presents previous reviews of software development effort estimation techniques. A prior study in Wen et al. (2012) reviews the use of machine learning in effort estimation, including its accuracy and context. By analysing 84 primary studies, the authors found that machine learning models tend to be more accurate than non-model methods. Some reviews are of effort estimation studies that focus on a specific type of applied technique. For example, Idria et al. (2016) analyse those that build ensemble effort estimation techniques. After reviewing 24 primary studies, the authors found that such techniques overcome the limitations of individual estimation techniques, identifying two main types (homogeneous and heterogeneous). They also found that, usually, ensemble techniques outperform single techniques. A review by Dave and Dutta (2014) examined the use of the neural network in estimating software effort. They analysed 21 primary studies in terms of the model used, the dataset and the evaluation method.

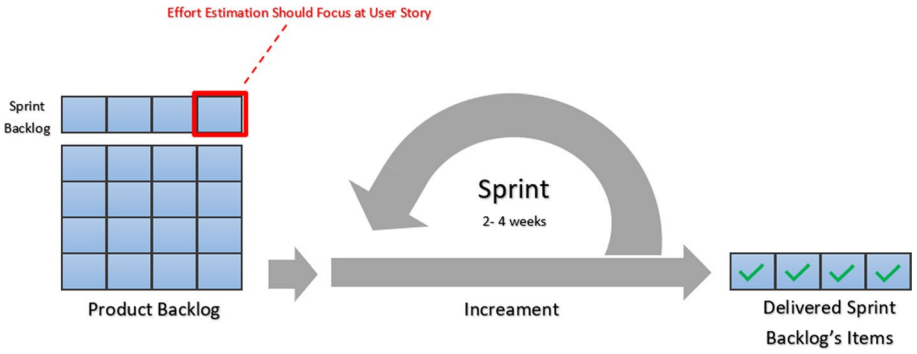

Fig. 2 Agile methodology life cycle

The finding was that increased attention is paid to neural network models for software effort estimation. They also note that most primary studies used a small dataset, such as only 63 projects.

Other reviews focus on the effort estimation of a specific development methodology. For example, Usman et al. (2014) explored effort estimation in agile development. In analysing 19 primary studies, they found that the most common techniques in agile are expert methods, estimation use case points, and Planning Poker. The most usual two-size metrics in effort estimation are use case point and story point, respectively. Similarly, the review by Arora et al. (2020) focused on effort estimation in agile development, particularly Scrum. They analysed 16 estimation techniques, overall estimation accuracy, and available datasets, finding that machine learning models outperform non-machine learning techniques in effort estimation for Scrum development. Fernández-Diego et al. (2020) conducted an updated SLR of Usman et al. (2014) work. The review included 73 new projects, these projects analysed in terms of the estimation method, the followed agile methods, effort predictors, and the used datasets. The result showed that Planning Poker is the widely used estimation method. They found that six agile methods had been used, moreover, story point is the most wieldy used effort metric. Another review (Schweighofer et al. 2016) was conducted to investigate the use of effort estimation methods for agile developments. By covering five data sources, 12 primary studies were analysed. The results showed that expert methods were widely used in effort estimation (e.g. planning poker). Furthermore, the importance of personal factors more than project factors' importance.

Although the literature contains some systematic reviews of agile development effort estimation, as presented above, these works did not focus on user stories estimation. They investigated the agile estimation methods regardless of whether it estimate a whole project or a single user story. For example, 12 out of the 17 studies covered in (Arora et al. 2020) were focused on estimating whole agile projects, not user stories. Estimating the effort of a whole agile project is unlike estimating the effort of a single user story. The difference is that user stories are usually developed during a short period, it could be even a few hours, moreover, it focuses on a specific issue or requirement. While agile projects developed during a long period, as it contains several sprints, each sprint has a duration of 20-30 days. Due to that difference, several characteristics may change, such as the effort drivers that influence the development effort, the used dataset, etc. Moreover, (Usman et al. 2014), (Fernández-Diego et al. 2020), and (Schweighofer et al. 2016) did not focus on the intelligent approaches used for effort estimation, they analysed the effort estimation methods regardless of the method type (expert-based or data-driven). While this research aims to investigate the data-driven methods used to estimate agile user stories effort.

3 Research methodology

The methodology that this article adopts is a systematic literature review (SLR). It provides a comprehensive exploration and analysis of state-of-the-art studies that help answer the research questions, following the guidelines developed by Kitchenham and Charters (2007). This article is structured by their three steps for SLR protocols: planning; conducting; and reporting.

3.1 Research questions

- *RQ1: Which data-driven techniques are used to estimate effort in agile user stories?* This question aims to identify the data-driven techniques used in estimating effort in agile user stories.
- *RQ2: How is the performance of user story effort estimation techniques measured and evaluated?* This question explores the various measures to prove the effectiveness of user story effort estimation techniques, as well as to show the accuracy of these techniques.
- *RQ3: What are the independent factors in estimating agile user stories?* This question investigates the most frequently used types of independent factors of user story effort estimation.
- *RQ4: What are the characteristics of a user stories dataset?* The final question ascertains the characteristics of user-stories datasets, including the number of user stories, and discusses the various metrics of the effort involved in developing a user story.

3.2 Search strategy

A search strategy specifies the data sources from which studies are to be extracted and defines the set of keywords to act as the search string. In this research, the search string is as follows:

("effort estimation" OR "effort prediction ") AND ("machine learning" OR model OR "Artificial intelligence" OR "deep learning") AND (Scrum OR Agile OR user stories)

Five data sources were selected: IEEE Xplore; ScienceDirect; Springer Link; Wiley; and Web of Science (ISI). This search yielded 626 studies. Table 1 shows the studies and respective data sources.

3.3 Study selection

To ensure that the primary studies are relevant to the objective of this review, the inclusion, exclusion criteria and quality assessment list are as follows:

3.3.1 Inclusion criteria

1. Studies with a primary focus on the data-driven effort estimation of agile projects.

Table 1 Initial search results according to the data source

#	Data Source	Number of studies
1	Springer link	215
2	ScienceDirect	275
3	Wiley	56
4	IEEE Xplore	27
5	Web of science (ISI)	53

2. Studies focused on the user-story level, rather than the project level.
3. Studies investigating the use of a specific estimation method, technique, or algorithm.

3.3.2 Exclusion criteria

1. Survey and review studies.
2. Studies not written in English.

3.3.3 Quality assessment checklist

Besides the inclusion and exclusion criteria, the selection process incorporates a quality assessment checklist of five criteria, some derived from Azhar et al. (2012) and Kitchenham and Charters (2007). To be selected, a study had to meet all five criteria:

1. Are the research objectives clearly defined?
2. Are the applied techniques clearly described in the paper?
3. Is the used dataset described clearly?
4. Do the researchers give enough details for the evaluation methods?
5. Is the result described and discussed clearly?

As shown in Fig. 3, the use of the keywords to search all the data sources extracted 626 articles. Checking at the subsequent level, it aimed to ensure, by reading only the abstract, that each study indeed met the inclusion and exclusion criteria. As a result, only 59 studies were included (see Appendix 1). Then, the 17 duplicated studies were removed. At the final level, the remaining studies were closely read to identify any that did not meet the defined criteria or quality assessment requirements and, thus, that did not serve the objective of this review. In this way, only 10 studies were found to be relevant. One additional level was added, which performed a Backward Snowballing approach (Wohlin 2014). This approach use the reference list of each primary study to include new revelent papers. The approach will be repeated several iterations until no new papers are founded. Using this approach, one more paper was included in this review. Thus, the final included studies have become 11 primary studies.

4 Results and analysis

This section analyses and discusses the primary studies to answer the stated research questions. Overall, as shown in Table 2, 5 out of 6 studies were journals articles, while the other were conference proceedings. Moreover, the majority had been published within the last four years (8 out of 11), half of it (4 studies) had been published in 2019. This shows that there is increasing interest in improving agile development's user-story effort estimation. However, as 2019 is the most recent year in the studies found, this does not definitely indicate that no studies were published in 2020. It indicates only that they were not extracted using the defined search terms and the data sources in this work.

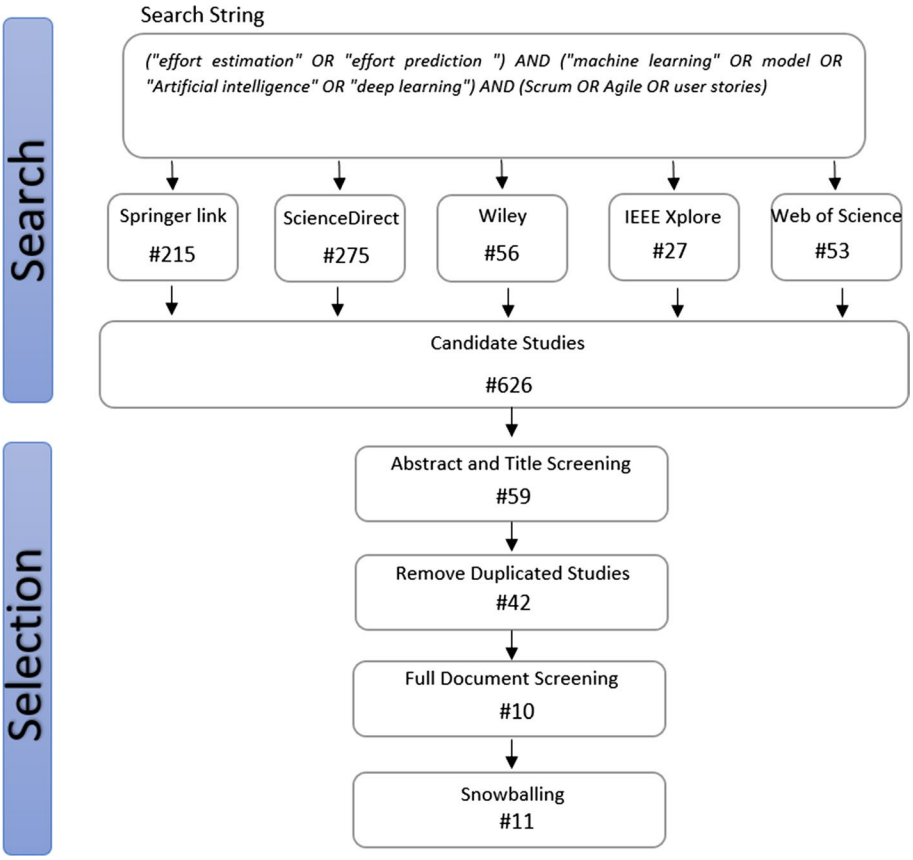

Fig. 3 SLR search process

4.1 RQ1: Which data-driven techniques are used to estimate effort in agile user stories?

The goal of the first research question was to explore which data-driven techniques were used to estimate user stories' effort. This will help software engineers and researchers get an overview of the effectiveness of such techniques in effort estimation. There are twenty techniques/algorithms were identified. Among these techniques, Decision Tree, Bayesian networks, and Support Vector Machines were frequently applied. Some techniques were used for comparison, with the authors applying different techniques to the same dataset to investigate which achieved the highest accuracy in estimating user-story effort. For example, Soares (2018) studied four types of AutoEncoders and their effectiveness in feature learning. These AutoEncoders improve the classifier's performance in estimating user stories' effort. Stacked, Variational, Denoising, and Stacked Denoising AutoEncoders were applied to six datasets to measure their accuracy and determine whether the various types of AutoEncoders produced different results. Similarly, Moharreri et al. (2016) examined the use of three classifier algorithms to estimate user stories' effort: Naive Bayes; Random Forest; and Logistic Model Tree. A comparison has been made between the three

Table 2 The selected primary studies

Study	Journal/Conference
Choetkiertikul et al. (2019)	IEEE Transactions on Software Engineering
Malgonde and Chari (2019)	Empirical Software Engineering
Dragicevic et al. (2017)	The Journal of Systems and Software
Soares (2018)	International Joint Conference on Neural Networks (IJCNN)
Scott and Pfahl (2018)	ICSSP '18: Proceedings of the 2018 International Conference on Software and System Process
Tanveer et al. (2019)	Journal of Software Evolution and Process
Hussain et al. (2010)	International Conference on Application of Natural Language to Information Systems
Moharreri et al. (2016)	2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)
Ratke et al. (2019)	2019 2nd International Conference on Computer Applications Information Security (ICCAIS)
Alostad et al. (2017)	International Journal of Advanced Computer Science and Applications(IJACSA)
Abrahamsson et al. (2011)	International Symposium on Empirical Software Engineering and Measurement

algorithms to find the one that achieves the highest accuracy in estimating the effort. In contrast, some techniques were combined to build an ensemble based model. This combination aims to achieve a more accurate estimation than an individual technique. Malgonde and Chari (2019) constructed a new ensemble model by combining seven techniques: Decision Tree; Bayesian network; Support Vector Machines; K-Nearest Neighbors; Artificial Neural Network; SRidge Regression; and Linear Regression. Each was assigned a weight that was considered by the ensemble classifier in estimating the effort. In Chongpakdee and Vatanawood (2017), two deep learning architectures were combined to predict the effort based on user-story text. The authors applied Long Short-Term Memory and Recurrent Highway Net for document and deep representations. Other studies used individual techniques to estimate the effort. Table 3 shows all the techniques identified and the studies that applied them.

4.2 RQ2: How is the performance of user story effort estimation techniques measured and evaluated?

The accuracy of effort estimation is a key factor in successful development (Khuat and Thi My Hanh 2016)(Arora et al. 2020). Therefore, this question aims to discuss how the performance and accuracy of estimation techniques are measured and evaluated. Based on the primary studies, the evaluations of user-story effort estimation techniques were grouped into six types, as follows;

1- *Performance measures* As shown in Table 4, all the selected studies used performance measures to evaluate the accuracy of data-driven estimation techniques. In fact, performance measures are considered a key evaluation technique involved in almost all other performance techniques. From the primary studies, 15 performance measures were identified for user-story effort estimation techniques. Some use a single performance measure,

Table 3 Summarization of user stories effort estimation techniques

Techniques/Algorithems	Study
Bayesian network	Malgonde and Chari (2019), Dragicevic et al. (2017), Ratke et al. (2019)
Decision Tree	Malgonde and Chari (2019), Hussain et al. (2010)
Support Vector Machines	Scott and Pfahl (2018), Malgonde and Chari (2019)
Long short-term memory and Recurrent highway network	Choetkiertikul et al. (2019)
Stacked AutoEncoders, Variational AutoEncoders, Denoising AutoEncoders, Stacked Denoising AutoEncoders	Soares (2018)
Gradient Boosted Trees	Tanveer et al. (2019)
K-Nearest Neighbors, Artificial Neural Network, SRidge Regression, Linear Regression	Malgonde and Chari (2019)
Naive Bayes, Random Forest, Logistic Model Tree, and J48 Decision Tree	Moharreri et al. (2016)
Fuzzy based Model	Alostad et al. (2017)
Stepwise Regression	Abrahamsson et al. (2011)

while others use a combination of two or more. Moreover, some studies measure the proposed model performance once by either using one dataset or averaging the results of different datasets and projects, others separate the performance results by different case studies, datasets, projects, or sprints. Furthermore, some studies measure the model performance with and without a specific feature. For example, Moharreri et al. (2016) examine the model performance with and without adding a new feature called the Planning Poker estimate, which is human estimation.

The performance measures used are Mean Absolute Error (MAE), Mean Magnitude of Relative Error (MMRE), Root Mean Squared Error (RMSE), Precision, Recall and F-measure. Other performance measures are the Median Absolute Error (MdAE), Standardized Accuracy (SA), Mean Balanced Error (MBR), Relative Absolute Error (RAE), Root Relative Squared Error (RRSE), Magnitude of Error Relative to Estimate (MER), Standard Deviation (SD), Prediction at Level k (PRED(K)) and Accuracy.

2- *Baseline benchmarks* In some studies, the estimation technique is to apply benchmarks against a specific baseline. The aim is to prove that the proposed estimation technique outperforms this baseline in terms of estimation accuracy or time. As an example, both Choetkiertikul et al. (2019) and Scott and Pfahl (2018) define three baselines, namely, Mean, Median, and Random Guessing. The authors indicate that an effective and useful estimation technique should achieve better results than the baseline of Random Guessing. In both studies, the results show that their techniques outperform these baselines.

3- *Comparison against slimier existing techniques* This evaluation technique locates the proposed estimation method in the literature, or how it fits with similar existing techniques. The authors selected the technique closest to what they propose, then conducted a comparison to show whether their proposed model achieved outstanding results. For example, both Soares (2018) and Choetkiertikul et al. (2019) compared their text-based estimation to an existing technique in the literature known as Term-Frequency Inverse Document Frequency (TF-IDF). TF-IDF is a simple technique for information retrieval; it measures the relationship and relevance between terms. The result in Soares (2018) shows no significant

Table 4 Summarization of user stories effort estimation techniques

Techniques/Algorithems	Study	Accuracy
Decision Tree	Malgonde and Chari (2019), Hussain et al. (2010)	MAE 8.44 and MBE 3.09 F-Measure 0.669
Bayesian network	Malgonde and Chari (2019), Dragicevic et al. (2017)	MAE 9.72 and MBE 5.18 MAE 0.026 and Accuracy 99.375%
Long short-term memory and Recurrent highway network	Choetkiertikul et al. (2019)	MAE 0.64-5.97, MdAE 0.53-4.93
Support Vector Machines	Scott and Pfahl (2018)	Accuracy 0.384 and MAE 1.700 and SA 69.179
Stacked AutoEncoders	Soares (2018)	F-measure(0.83±0.08,0.91±0.02,0.80±0.04, 0.88±0.01,0.71±0.06,0.85±0.03)
Variational AutoEncoders	Soares (2018)	F-measure(0.81±0.07,0.90±0.02,0.77±0.04, 0.88±0.01,0.73±0.06 ,0.81±0.03)
Denoising AutoEncoders	Soares (2018)	F-measure(0.82±0.07,0.90±0.02,0.80±0.05, 0.88±0.01,0.72±0.06, 0.81±0.02)
Stacked Denoising AutoEncoders	Soares (2018)	F-measure(0.81±0.07,0.90±0.02,0.76±0.04, 0.88±0.01,0.70±0.05, 0.78±0.02)
Ensemble-based model (Support Vector Machines, K-Nearest Neighbors, Artificial Neural Network, Ridge Regression, Linear Regression)	Malgonde and Chari (2019)	MAE 8.243, MBE 1.132, and RMSE 15.331
Gradient Boosted Trees	Tanveer et al. (2019)	MAE 0.88 and MMER 1.03 and PRED(25) 0.5
Bayesian Networks	Ratke et al. (2019)	Accuracy 81.82%
Naive Bayes (without and with estimate Planning Poker)	Moharreri et al. (2016)	MMRE 180.09% and 204.40%
J48 Decision tree (without and with Planning Poker estimate)	Moharreri et al. (2016)	MMRE 91.75% and 92.32%
Random Forest (without and with estimate Planning Poker)	Moharreri et al. (2016)	MMRE 97.33% and 108.95%
Logistic Model Tree (without and with estimate Planning Poker)	Moharreri et al. (2016)	MMRE 125.26% and 103.28%
Fuzzy based Model	Alostad et al. (2017)	(Sprint1) MMRE 0.28 and PRED (MMRE) 50% (Sprint2) MMRE 0.15 and PRED (MMRE) 60% (Sprint3) MMRE 0.09 and PRED (MMRE) 60%

Table 4 (continued)

Techniques/Algorithems	Study	Accuracy
Stepwise Regression	Abrahamsson et al. (2011)	(Case study A) MMRE 90%, mMER 67% (Case study B) MMRE 66%, mMER 47%

difference between autoencoders and TF-IDF in terms of precision and recall, while those of Choetkiertikul et al. (2019) show that their proposed method outperforms TF-IDF in terms of MAE. Malgonde and Chari (2019) compared their ensemble technique to a set of existing ensemble techniques such as Average, Extra Trees, Random Forest, Gradient Boosting, Ada Boost and Stacking. They found that their ensemble method outperforms other ensemble methods in terms of MAE, RMSE and MBE.

4- *Comparison against human estimation* Another method is comparison of the human-estimated effort (e.g. the user story's developers) to that estimated by a data-driven technique. The goal is to prove that an automated, data-driven technique is more accurate. For example, Malgonde and Chari (2019) selected a set of 10 user stories with an available effort estimate by the development team, then compared the developers' estimation accuracy to that carried out by their technique. The results showed that the proposed methods outperformed human estimation in terms of MAE, MBE, and RMSE. Similarly, Abrahamsson et al. (2011) compared the developers' estimation with their approach estimation. Two case studies involved development by different teams; in the first case study, the developers' estimation outperformed the proposed approach, while in the second the estimation by the proposed approach was within in the same range as that of the developers' estimate.

5- *Statistical tests* In this method, the authors applied a suitable standard statistical test to measure the effectiveness of their estimation technique. Malgonde and Chari (2019) conducted two types of tests, the Friedman and Wilcoxon signed tests, to compare their technique to similar methods.

6- *Distribution of estimates* Moharreri et al. (2016) proposed an evaluation method called distribution of estimates. This method aims to evaluate the suitability of an estimation model. It divides the probability distribution of a model estimation into three parts; correct estimate, overestimate and underestimate. The study indicates that in terms of an inaccurate estimate, overestimate is much preferred than underestimate. Slightly overestimate an effort is not detrimental compared to underestimate in agile software. Therefore, a model should have the highest probability of a correct estimate, and the overestimate probability should be higher than the underestimate probability.

4.3 RQ3: What are the independent factors in estimating agile user stories?

This question addresses the independent factors (effort drivers) in the estimation of the effort involved in developing a user story. From the primary studies, various sets of independent factors were found, grouped into two categories: (1) studies that estimate effort based on predefined variables, in which the work focuses on defining a set of independent factors from which the user story effort is derived (six of the 11 primary studies were in this category); (2) studies that estimate effort based on text, in which the value of the effort is derived from a written document, usually in informal natural language containing user story-related information. Such information could be the user story title, detailed description, or programming commands. Trendowicz and Jeffery (2014) identified four main types of effort factors. Accordingly, we mapped the independent factors that we found to these classifications. As the second category depends on factors in the user story title and discerption, these were not clarified further. Therefore, mapping to the Trendowicz and Jeffery classification was carried out with all studies from the first category as shown in Table 5. The four main classifications for effort factors introduced by Trendowicz and Jeffery are personnel, product, process and project. However, from what have been found

Table 5 Independent factors to estimating agile user stories effort

Factor type	Independent factor	Study
–	Textual factors (e.g. description)	Choetkiertikul et al. (2019), Chongpakdee and Vatanawood (2017), Soares (2018), Hussain et al. (2010), Ratke et al. (2019), Moharreri et al. (2016), Abrahamsson et al. (2011)
Personnel factors	Experience	Malgonde and Chari (2019), Alostad et al. (2017)
	Developers skills	Dragicevic et al. (2017)
	New task type	Dragicevic et al. (2017)
	Reputation	Scott and Pfahl (2018)
	Total work capacity (number of issues)	Scott and Pfahl (2018)
	Total work capacity (story points)	Scott and Pfahl (2018)
	Current developer workload	Scott and Pfahl (2018)
	Number of developer comments	Scott and Pfahl (2018)
Product factors	Priority	Malgonde and Chari (2019), Moharreri et al. (2016), Abrahamsson et al. (2011)
	Complexity	Dragicevic et al. (2017), Alostad et al. (2017)
	Size	Malgonde and Chari (2019), Alostad et al. (2017)
	Subtasks, Sprint	Malgonde and Chari (2019)
Process factors	Specification quality	Dragicevic et al. (2017)
Estimation factors	Estimated points from planning poker	Moharreri et al. (2016)
	Estimation accuracy	Alostad et al. (2017)

in the primary studies, a new category of effort drivers were introduced in this work which is estimation-related factors. 16 independent factors were extracted and mapped as follows:

1- *Personnel factors* These present team members' characteristics as a whole team and as individuals. Personnel factors are the most common type of effort factors used in agile user story estimation, as half of the found factors relate to the personnel characteristics. The way these factors are measured varies according to the factor itself and the study, though some studies do not indicate how they measured the personal factors. Both Malgonde and Chari (2019) and Alostad et al. (2017) introduced team *experience* as a factor to estimate user-story effort. Malgonde and Chari (2019) defined various independent factors for each developer. They found that the main split in the applied decision tree was based on the experience of the second programmer. They measured experience as the number of programming experience years. Three categories were defined for experience: [1,5], [6,10], and [15,21]. Moreover, the experience was self-reported in that study, as each developer filled in his/her data. Meanwhile, in Alostad et al. (2017) experience was measured as the number of years that developers spent in work. Based on their years of experience, they were classified into junior, intermediate, and senior. Dragicevic et al. (2017), introduced two different personnel factors. The first one is *developers skills*, this includes the experience of the developers as well as their motivation. The skills are measured once or sometimes twice a year using the Personal Capability Assessment Method (Čelar et al. 2014), which ranked each developer's skills from 1 to 5. The Personal Capability Assessment Method evaluates the personal capability of agile team members by calculating the personal points (Čelar et al. 2014). To do that, several parameters with different weights should be used. Some of these parameters are: experience; knowledge of technology; attitude towards work; and the level of learning and work habits. The second factor is *new task type*, which refers to whether the developer is familiar with this task/user story or whether it is new. Scott and Pfahl (2018) proposed five more personnel factors, indicating that developers' characteristics influence the estimation process. The first factor they proposed is *reputation*, which is measured as the ratio of the number of user stories (or issues) in that dataset that were opened and fixed by a developer to the number of user stories that were opened by the same developer plus one. Another factor is *current developer workload*, which is the number of user stories assigned to a developer at a time. In addition, new two factors were introduced. The first is *total work capacity (issues)*, which refers to the number of user stories (or issues) that the developer has achieved. The second is *total work capacity (story points)*, which refers to the total story points that the developer has achieved in that project. The last factor is the *number of comments* that the developer wrote in that project.

2- *Product factors* These are the characteristics of the software being developed. Malgonde and Chari (2019), and Abrahamsson et al. (2011), and Moharreri et al. (2016) include the *priority* of the user story in the estimation process. This reflects a ranking of the user story's importance to the customer. Besides priority, both Malgonde and Chari (2019) and Ratke et al. (2019) include another product factor, namely the *size* of the user story. Moreover, Malgonde and Chari (2019) used two other product factors: the *number of subtasks* involved in user story development and the number of *sprints* in which it should be delivered. While both Dragicevic et al. (2017) and Alostad et al. (2017) defined the *complexity* of the user story as a factor to estimate its effort, Dragicevic et al. (2017) split it into nine independent factors representing the number of simple, moderate, and high-complexity forms, functions and reports. Meanwhile, Alostad et al. (2017) used only one independent factor to represent the complexity, in which a user story should be categorized into one of five predefined categories: very easy; easy; moderate; complex; and very complex.

3- *Process factors* These are factors related to the development process, such as effectiveness and quality. From the literature, Dragicevic et al. (2017) used a process-related factor, namely, *specification quality*.

4- *Project factors* These involve the constraints on software projects, such as schedule pressure and working conditions. However, none of the primary studies include project-related factors.

5- *Estimation factors* This refers to factors related to the estimation process that has been carried out previously by the development team. Moharreri et al. (2016) used the *estimated points from planning poker* of a user story by developers as an effort driver in the data-driven model. Even if the estimated effort by developers were inaccurate, the study indicated that it may be used as a supporting feature by the algorithm to create an accurate estimation. Meanwhile, Alostad et al. (2017) introduced the *estimation accuracy* as an independent factor. This refers to the extent to which an estimation made in a previous estimation process is accurate. The accuracy values ranged from over, well, and under estimated.

4.4 RQ4: What are the characteristics of a user stories dataset?

Data play a key role in any data-driven technique, as the quality of the dataset used may affect its results. Therefore, this question aims to explore the agile user-stories datasets available for effort estimation. From the primary studies, 12 user stories datasets were extracted, as shown in Table 6. The number of user stories ranged from more than 23,000 in a large dataset to 13 in a very small dataset.

Three of the studies extracted and built user-stories datasets from issue tracking systems (e.g. JIRA) that contained open source projects, while two were the largest datasets used in user story effort estimation. Choetkiertikul et al. (2019) collected 23,313 user stories/issues from 16 projects on the JIRA issue tracking system. In this dataset, each user story had a title, description and story point. Similarly, Scott and Pfahl (2018) constructed a new dataset of 15,155 user stories from 12 projects. Each user story had text features and developer features such as reputation, current developer workload, total work capacity (number of issues), total work capacity (story points) and number of developer comments. Furthermore, Soares (2018) used an issue tracking system to build a user stories dataset from six open source projects. This dataset contains textual data on each user story and the story point.

Other studies extracted user stories datasets from software companies and houses. For example, Dragicevic et al. (2017) employed a dataset containing 160 user stories from a single software company. It had the following attributes: task id; new task type; specification quality; developer skills; working hours; and attributes for form, function and report number according to complexity (low, medium or high). In Abrahamsson et al. (2011), the authors built two datasets, each representing a different organization and different case studies. The first has 1,325 user stories from a large Italian company. From this dataset, the authors extracted only the priority, number of characters and the presence of keywords. The second has 13 user stories from a non-profit research organization. The authors extracted all the same attributes apart from priority. In Tanveer et al. (2019), the authors extracted the datasets from Insiders Technologies. The data had 210 user stories, with each user story having a total number of changes in three code metrics (LOC, Coupling Between Objects (CBO), and Weighted Methods per Class (WMC)), number of classes affected when a single user story is implemented, and effort value. Another study (Malgonde and Chari 2019) collected its dataset from the IT department

Table 6 Available user stories datasets

Data Source	Attributes	Number of user stories	Study
Issue tracking systems	Title, description, and story point	23,313 user stories	Choetkiertikul et al. (2019)
Issue tracking systems	Reputation, current developer workload, total work capacity (number of issues), total work capacity (story points), number of developer comments, and text features	15,155 user stories	Scott and Pfahl (2018)
Issue tracking systems	Textual data, and story point	–	Soares (2018)
Information technology department of a large university	Priority, number of the involved subtasks, size, sprint number, programmer 1 experience, programmer2 experience, programmer3 experience, and effort	503 user stories	Malgonde and Chari (2019)
Software company	Task ID, new task type, specification quality, form low_no, form medium_no, form high_no, function low_no, function medium_no, function high_no, report low_no, report medium_no, report high_no, developer skills, working hours	160 user stories	Dragicevic et al. (2017)
Real software project	Development team experience, task complexity, task size, and estimation accuracy	30 user stories	Alostad et al. (2017)
SAP Labs and university projects	Textual keywords	61 user stories	Hussain et al. (2010)
Software Projects	Story card ID, project, planned story points (human estimation), story card summary (text)	123 user stories	Moharreri et al. (2016)
Software Projects	User stories description (text)	–	Ratke et al. (2019)
Insiders Technologies	Total number of changes in LOC, CBO, and WMC, and number of classes affected when a single user story is implemented	210 user stories	Tanveer et al. (2019)
Software company	Number of characters, Presence of keywords, and Priority	1,325 user stories	Abrahamsson et al. (2011)
Non-profit research organization	Number of characters, Presence of keywords	13 user stories	Abrahamsson et al. (2011)

of a large university. The dataset contained 503 user stories from 24 projects. In this dataset, each user story showed its priority, number of involved subtasks, size, sprint number, programmers' experience, and effort. Hussain et al. (2010) extracted 61 textual requirements from four projects: two of them software projects from SAP Labs, and two university projects. Effort was estimated based on textual keywords. Moharreri et al. (2016) collect 123 user stories from software projects. The data contained information about story card ID, project, planned story points (human estimation), and story card summary (text). Meanwhile, Alostad et al. (2017) built a data set from real software project that involved three sprints, each with 10 user stories. However, the user story had information about development team experience, task complexity, task size, and estimation accuracy. Moreover, Ratke et al. (2019) built the data from a set of software projects. Each user story had a text description including its title. However, the authors of the study did not mention any information regarding the number of data records.

Generally, the found datasets have different factors and number of records. Moreover, it represents a variety of project types and contexts. However, two types of datasets are used for effort estimation: data constructed from the targeted company's projects, or an available dataset that was constructed from another company which is known as a cross-company dataset (Ferrucci et al. 2012). In which an estimation model is developed for company A and trained on a dataset of company B. This type of data requires that both companies' contexts should be matched (e.g. both developing open-source projects), as the effort required by both companies should be almost the same (Minku and Yao 2014). However, Ferrucci et al. (2012) indicate that effort estimation models performed worse when it uses a cross-company dataset compared to the within-company dataset. To address this issue, several studies applied filtering methods. These methods find the most similar projects to the targeted context. Some studies conducted manual filtering. For example,de Guevara et al. (2016) indicates that comparable definition of effort and size metrics is one of the used filtering methods, in which, only projects with a consistent sizing approach are selected. Scott and Pfahl (2018), Soares (2018), Alostad et al. (2017), and Moharreri et al. (2016) selected similar projects from datasets based on the effort metrics (e.g. Fibonacci sequence). Other factors are (de Guevara et al. 2016): organization type; development type; and country. Instead of manually filtering the projects, some studies used more advanced methods. In (Ferrucci et al. 2012), a k-Nearest Neighbor filtering technique had proposed. For each project in the targeted company, the technique finds the k similar projects from the cross-company dataset. This will help to have a more homogenous dataset in addition to increasing the estimation accuracy (Ferrucci et al. 2012). Minku and Yao (2014) proposed a new approach named Dynamic Cross-company Mapped Model Learning (Dycom). Dycom assumes that the target company should has some available training data, even if it is small. Moreover, the target company and the cross-company datasets should have compatible input features. The cross-company data set is divided first into several datasets based on clustering technique, projects' size, or their productivity. Several estimation models are built for the different datasets. Then, the Dycom will create a mapping function to map the estimation model to the context of the target company. The results show that using such an approach will contribute to building a more accurate estimation model than just using the cross-company dataset as it is.

4.4.1 Metrics to measure the effort of developing a user story

Development teams apply varying definitions of user stories' effort. Therefore, several metrics were employed to measure the effort of developing a user story. The goal of this

question is to discuss the most common effort metrics used in the primary studies. Three effort metrics were defined and used, as shown in Table 7. The most frequent (used by six of the studies) is story points (SP). According to Choetkiertikul et al. (2019), SP measure the relative effort of development in multiple stories; however, there are several ways in which SP can be scaled. One of these scales used in Planning Poker is the Fibonacci sequence (1, 2, 3, 5, 8, 13, etc.). Scott and Pfahl (2018), Soares (2018), Alostad et al. (2017), and Moharreri et al. (2016) stated that an SP for a user story should be expressed in one of the Fibonacci sequence numbers. Choetkiertikul et al. (2019) indicated that not all projects use this sequence in effort metrics; therefore, they include all SP at any scale. Other studies used effort metrics other than SP. For example, that used by Malgonde and Chari (2019) is person-hours, referring to the time needed to develop a complete user story. Meanwhile, Tanveer et al. (2019) used similar effort measures called person-days. In Hussain et al. (2010), the authors investigated the use of the Common Software Measurement International Consortium (COSMIC) Function Point (CFP) as a measurement for agile user stories. CFP represents the movement number of data (e.g. read, entry, exit, and write). They found that CFP helps to undertake an objective estimation method. The final two studies (Dragicevic et al. 2017) and (Abrahamsson et al. 2011) did not clearly define the effort metrics used.

5 Discussion

This section discusses the results and highlights trends in data-driven effort estimation techniques for agile user stories. Overall, after the SLR, it was observed that a limited number of studies use data-driven techniques for agile user stories effort estimation. The distribution of studies over the years shows that the field of agile user stories effort estimation is receiving increasing interest and attention, which indicates that it is an active research area. Nearly half of the studies (six studies) were published within the last three years. The effort estimation problem is highlighted broadly; however, there are limited studies on agile effort estimation, although this type of development is a widely used Software Development Life Cycle (SDLC) (Serrador and Pinto 2015) that ensures rapid production and fast delivery of high-quality software projects (Matharu et al. 2015). A study by Serrador and Pinto (2015) has statistically proved that the high level of agility in a software project has a notable impact on its success in terms of efficiency, stakeholder satisfaction and project performance. Therefore, more effort estimation studies are needed to support agile development.

Table 7 Effort metrics

Effort metrics	Study
Story Points	Choetkiertikul et al. (2019), Alostad et al. (2017), Soares (2018), Moharreri et al. (2016), Ratke et al. (2019), Scott and Pfahl (2018)
Person-Days	Tanveer et al. (2019)
Person-Hours	Malgonde and Chari (2019)
COSMIC Function Point	Hussain et al. (2010)
Not defined	Dragicevic et al. (2017), Abrahamsson et al. (2011)

In terms of data-driven techniques, discussed in the first question, 20 techniques were identified in the primary studies. Some studies use a single data-driven technique, while others apply several to either compare their performance or combine these techniques to build an ensemble estimation model. Among these techniques, only Decision Tree and Bayesian network have frequently been used, according to the literature. Moreover, it was observed that, over the years, more advanced techniques have been introduced in this field, such as deep-learning architectures. This diversity of techniques emphasizes the importance of artificial intelligence (AI) in enhancing the efficiency of software development management. However, some data-driven techniques either have not been applied yet or have been applied in a specific context by a single study, for example, ensemble effort estimation models. Kocaguneli et al. (2012) indicates that although there is no best individual effort estimation model for all contexts, the best combinations of those individual models may be found. An SLR conducted in (Idria et al. 2016) investigates the use of ensemble models for effort estimation. They found several ensemble estimation models in the literature that overcome the limitation of individual models, and their performance is more accurate than that of individual models. According to the results of this SLR, the use of ensemble effort estimation models for agile user stories is very limited. Moreover, agile user stories are written in a form of textual description. Several studies use this textual description to estimate the effort. The majority of these studies manually defined a set of text-related features (e.g., words counts). With the use of advanced intelligent approaches such as deep learning, the model can automatically learn the features (Choetkiertikul et al. 2019). In this SLR, only one study introduced the use of the deep learning model for agile user stories (Choetkiertikul et al. 2019). This creates new directions for further research to investigate additional data-driven techniques (e.g., ensemble models, deep learning, etc.) for user story effort estimation.

The techniques were evaluated using several evaluation methods. They may be grouped into six main evaluation types: performance measures; baseline benchmarks; comparison against similar existing techniques; comparison against human estimation; distribution of estimates, and statistical tests. Some studies use a single evaluation technique, while others use two or more together. The accuracy of the estimation techniques proves the effectiveness of the data-driven effort estimation of agile user stories. In fact, some of the data-driven techniques outperform human estimation. Moreover, it was observed that text-based estimation depends heavily on the correctness of writing and describing a user story: Some studies encountered difficulties due to the short and informal language of the user stories. Therefore, attempts should be made to investigate the appropriate format of a user story for effort estimation–in other words, which information should be included in the description to permit accurate estimation (e.g. product, personnel, project, process factors). Furthermore, although very few studies discuss the time taken by their proposed estimation technique, one study recorded a very fast estimation time, within two seconds. This immediate estimation confirms the usefulness of data-driven estimation techniques in the context of agile projects, as the use of such a technique during an agile meeting will not delay or disrupt it.

Several independent factors are used to estimate user-stories' effort. From the primary studies, 16 factors were identified, each mapped to one of Trendowicz and Jeffery's classes (2014). Four types of independent factors were identified: personnel; product; process; and estimation. The most common type of independent factors are personnel factors, and no project-related factors were identified in the selected studies. As presented in the literature, each study investigates differing independent factors of effort estimation. The frequency of use of any factor among studies is limited, showing that there are no standard factors that affect user-story effort in every context. Thus, it is recommended that researchers and software engineers investigate additional independent factors (effort drivers) that influence the effort to develop agile user stories.

To estimate user story effort from historical data, 12 datasets were constructed and used. These ranged from a large dataset of 23,313 user stories to a very small dataset of

only 13 user stories. These datasets were collected from different types of data sources. However, there is a need to construct more datasets in this field, as the number of datasets is limited, some are not available online, and it is hard to get such data from industry. This will encourage and create more opportunities for researchers to devise and examine new data-driven estimation methods and improve the accuracy of existing ones.

There are various effort metrics for agile user stories. The most usual is story point. This result is consistent with the results achieved by Usman et al. (2014), who conducted an SLR on agile effort estimation. They found that story point and Use Case (UC) points are the most common effort metrics; however, in our SLR, UC points are not used by the primary studies, perhaps due to the different review focus. Unlike the previous SLR by Usman et al. (2014), which covers all the effort estimation methods in agile projects, this SLR focuses on methods for user stories.

Although the selected primary studies show effective implementation of data-driven techniques in user stories effort estimation, there are few works compared to those in traditional effort estimation studies. This provides a great opportunity to conduct further research in this field. Moreover, it was observed that different studies use different effort drivers. Therefore, there is a need for researchers to study the effort drivers of agile user stories. It is not easy to obtain user stories datasets from industry. Thus, we encourage the publication of new user stories datasets for use by researchers who have an interest in this field and to help with the development of accurate effort estimation techniques for agile user stories.

This work has covered five perspectives: data-driven techniques; performance evaluation methods; estimation accuracy; effort drivers; and the dataset. However, more directions may be investigated for future meta-analyses on the selected papers including: statistical analysis of effort estimation techniques' accuracy (e.g., box plots, standard deviation, and central tendency measures); datasets limitations that affect the accuracy (e.g. unbalanced data); and analyze the estimation techniques in depth (e.g. model parameters). Moreover, there are other important directions related to agile effort estimation that could be analysed. In terms of the dataset, two types are used for effort estimation: the within-company dataset and the cross-company dataset (Choetkiertikul et al. 2019). The former is constructed from the previous company's projects, while the latter is an available dataset that was constructed from another company's projects (Ferrucci et al. 2012). The main challenge in using the cross-company dataset is to ensure that both companies work in similar contexts. To solve this issue, filtering methodscould be developed to remove irrelevant projects and select only similar ones. Some efforts have been made in this field (see Section 4.4). However, we believe that more focus and investigation could be employed for cross-company datasets. Moreover, another measurement that is used for iteration planning along with estimated effort is team velocity. Team velocity is the number of story points a team can deliver each iteration (Cohn 2005). It helps the team to determine the number of iterations needed to finish a project (Cohn2005). Velocity may be influenced by several factors, including the team itself, the project domain, and the used tools and technologies (Coelho and Basu 2012). We encourage further investigations into using AI for velocity estimation.

6 Threats to validity

This review aims to extract and analyze data-driven effort estimation techniques for agile user stories. However, there are some threats to the validity of this review. Some of these threats are related to the search process. Finding and extracting all the relevant studies to address the proposed RQs is a challenging task. The efficiency of the search process is affected by

several aspects including search string design, data sources, inclusion and exclusion criteria, and a quality assessment checklist. In this SLR, the search string was designed by defining the terms relevant to the proposed RQs; however, those terms may limit the search results. In other words, relevant studies may exist in the literature and use different terms than the selected ones; thus, they may not appear in search results and be covered by this SLR. Moreover, five popular data sources were selected; thus, this SLR does not cover relevant studies in other data sources. To that end, we applied an additional step in the search process, the Backward Snowballing approach, to find those relevant studies that did not appear in the search results.

The findings of this SLR also may be affected by the search string. A few data-driven effort estimation techniques, if any, may not be covered by this SLR. Moreover, there are some concerns related to data extraction. Each study was analyzed by two researchers to extract the data. Then, both filled out the data extraction form (see Appendix 2). However, the two researchers did not find some missing data (e.g., the number of user stories in the dataset, and effort metrics). Such missing data are represented by a dash "-" or "Not defined" in the tables. Appendix 3 shows whether each study addressed the research questions fully or partially.

7 Conclusion

Development effort estimation of user stories is a crucial element in the success of agile development. This study conducted an SLR of data-driven effort estimation techniques for agile user stories. It investigated studies that have applied such techniques from five perspectives: data-driven techniques; the various performance evaluation methods; estimation accuracy; independent factors (effort drivers) to estimate the effort; and the characteristics of the dataset. Moreover, it highlighted new directions and openings for research in this field. The review was conducted using five data sources, namely Springer Link, Science Direct, Wiley, IEEE Xplore, and Web of Science. 11 studies were selected as primary studies. It was found that data-driven techniques have proven their effectiveness in estimating agile user stories effort estimation. In fact, some techniques outperform other estimation methods (e.g. human estimation). Moreover, there is no specific method of technique evaluation; the identified evaluation methods fall into six main types: performance measures; baseline benchmarks; statistical tests; distribution of estimates; comparison against similar existing techniques; and comparison against human estimation. Furthermore, several independent factors (effort drivers) were used to estimate user story effort. There are four main types: personnel; product; process; and estimation. In terms of datasets, it was observed that few user stories datasets are available. Therefore, more should be constructed to facilitate further research in this field.

Appendix 1

See Table 8.

Table 8 Studies Passed The Abstract and Title Screening

ID	Abstract and title screening	Full document screening	Ref.	ID	Abstract and title screening	Full document screening	Ref.
S1	✓		Satapathy and Rath (2017)	S31	✓		Kaushik et al. (2022)
S2	✓		Rao et al. (2018)	S32	✓		Brezočnik et al. (2019)
S3	✓	✓	Malgonde and Chari (2019)	S33	✓	Duplicated	Choetkiertikul et al. (2019)
S4	✓		Kaushik et al. (2020a)	S34	✓	Duplicated	Malgonde and Chari (2019)
S5	✓		Bilgaiyan et al. (2018)	S35	✓		Radu (2019)
S6	✓		Bilgaiyan et al. (2019)	S36	✓	✓	Tanveer et al. (2019)
S7	✓		López-Martínez et al. (2018)	S37	✓	Duplicated	Ratke et al. (2019)
S8	✓	✓	Hussain et al. (2010)	S38	✓	Duplicated	Saini et al. (2018)
S9	✓		Wińska et al. (2021)	S39	✓	✓	Scott and Pfahl (2018)
S10	✓		Kaushik et al. (2020b)	S40	✓	✓	Soares (2018)
S11	✓		Dam (2018)	S41	✓	✓	Alostad et al. (2017)
S12	✓		Durán et al. (2020)	S42	✓		Tanveer et al. (2017)
S13	✓		Grabis et al. (2020)	S43	✓	Duplicated	Satapathy and Rath (2017)
S14	✓		Dhir et al. (2017)	S44	✓		Aslam et al. (2017)
S15	✓		Zang (2008)	S45	✓	Duplicated	Adnan and Afzal (2017)
S16	✓	✓	Dragicevic et al. (2017)	S46	✓	Duplicated	Moharreri et al. (2016)
S17	✓		Panda et al. (2015)	S47	✓	Duplicated	Dragicevic et al. (2017)
S18	✓	Duplicated	Hussain et al. (2013)	S48	✓	Duplicated	Basri et al. (2016b)
S19	✓		Saini et al. (2018)	S49	✓		Rosa et al. (2017)
S20	✓	✓	Choetkiertikul et al. (2019)	S50	✓		Basri et al. (2016a)
S21	✓		Ungan et al. (2014)	S51	✓	Duplicated	Sharma et al. (2015)
S22	✓	✓	Ratke et al. (2019)	S52	✓		Lenarduzzi et al. (2015)
S23	✓		Adnan and Afzal (2017)	S53	✓	Duplicated	Panda et al. (2015)
S24	✓	✓	Moharreri et al. (2016)	S54	✓	Duplicated	Ungan et al. (2014)
S25	✓		Basri et al. (2016b)	S55	✓	Duplicated	Zang (2008)
S26	✓		Sharma and Chaudhary (2020)	S56	✓		Pow-Sang and Imbert (2012)

Table 8 (continued)

ID	Abstract and title screening	Full document screening	Ref.	ID	Abstract and title screening	Full document screening	Ref.
S27	✓		Dewi and Sarno (2020)	S57	✓	Duplicated	Tanveer et al. (2017)
S28	✓		Sharma et al. (2015)	S58	✓	Duplicated	Tanveer et al. (2019)
S29	✓		Kompella (2013)	S59	✓		López-Martínez et al. (2017)
S30	✓	Duplicated	Kaushik et al. (2020a)				

Appendix 2

See Table 9.

Table 9 Data extraction form

General information	Title - Year - Data source
RQ1: Which data-driven techniques are used to estimate effort in agile user stories?	Data driven technique
RQ2: How is the performance of user story effort estimation techniques measured and evaluated?	Performance measures - Accuracy - Other evaluation methods
RQ3: What are the independent factors in estimating agile user stories?	Independent factors
RQ4: What are the characteristics of a user stories dataset?	Dataset source - Dataset attributes - Number of user stories - Effort metrics
Additional details	Any comments or limitations.

Appendix 3

See Table 10.

Table 10 Selected studies

Ref.	RQ1	RQ2	RQ3	RQ4
Choetkiertikul et al. (2019)	✓	✓	✓	✓
Malgonde and Chari (2019)	✓	✓	✓	✓
Dragicevic et al. (2017)	✓	✓	✓	(Partial)
Soares (2018)	✓	✓	✓	(Partial)
Scott and Pfahl (2018)	✓	✓	✓	✓
Tanveer et al. (2019)	✓	✓	✓	✓
Hussain et al. (2010)	✓	✓	✓	✓
Moharreri et al. (2016)	✓	✓	✓	✓
Ratke et al. (2019)	✓	✓	✓	(Partial)
Alostad et al. (2017)	✓	✓	✓	✓
Abrahamsson et al. (2011)	✓	✓	✓	(Partial)

Funding None.

Data availibility Not applicable.

Code availability Not applicable.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Abrahamsson P, Fronza I, Moser R, Vlasenko J, Pedrycz W (2011) Predicting development effort from user stories. In: 2011 international symposium on empirical software engineering and measurement, 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. https://doi.org/10.1109/ACCESS.2017.2771257

Alloghani M, Al-Jumeily Obe D, Mustafina J, Hussain A, Aljaaf A (2020) A systematic review on supervised and unsupervised machine learning algorithms for data science. Superv Unsuperv Learn Data Sci. https://doi.org/10.1007/978-3-030-22475-2_1

Alostad JM, Abdullah LRA, Aali LS (2017) A fuzzy based model for effort estimation in scrum projects. Int J Adv Comput Sci Appl (IJACSA)https://doi.org/10.14569/IJACSA.2017.080939

Altaleb A, Altherwi M, Gravell A (2020) An industrial investigation into effort estimation predictors for mobile app development in agile processes. In: 2020 9th international conference on industrial technology and management (ICITM), https://doi.org/10.1109/ICITM48982.2020.9080362

Arora M, Verma S, Kavita, Chopra S (2020) A systematic literature review of machine learning estimation approaches in scrum projects. https://doi.org/10.1007/978-981-15-1451-7_59

Aslam W, Ijaz F, Lali MIU, Mehmood W (2017) Risk aware and quality enriched effort estimation for mobile applications in distributed agile software development. J Inf Sci Eng 33:1481–1500

Azhar D, Mendes E, Riddle P (2012) A systematic review of web resource estimation. In: PROMISE '12: proceedings of the 8th international conference on predictive models in software engineering, https://doi.org/10.1145/2365324.2365332

Basri S, Kama N, Haneem F, Ismail SA (2016a) Predicting effort for requirement changes during software development. In: proceedings of the seventh symposium on information and communication technology, https://doi.org/10.1145/3011077.3011096

Basri S, Kama N, Sarkan HM, Adli S, Haneem F (2016b) An algorithmic-based change effort estimation model for software development. In: 2016 23rd Asia-Pacific software engineering conference (APSEC), https://doi.org/10.1109/APSEC.2016.034

Bilgaiyan S, Aditya K, Mishra S, Das M (2018) Chaos-based modified morphological genetic algorithm for software development cost estimation. In: Progress in Computing, Analytics and Networking, https://doi.org/10.1007/978-981-10-7871-2_4

Bilgaiyan S, Mishra S, Das M (2019) Effort estimation in agile software development using experimental validation of neural network models. Int J Inf Technol. https://doi.org/10.1007/s41870-018-0131-2

Brezočnik L, Fister I, Podgorelec V (2019) Solving agile software development problems with swarm intelligence algorithms. In: international conference "new technologies, development and applications", https://doi.org/10.1007/978-3-030-18072-0_35

Čelar S, Turić M, Vicković L (2014) Method for personal capability assessment in agile teams using personal points. In: 2014 22nd telecommunications forum telfor (TELFOR), https://doi.org/10.1109/TELFOR.2014.7034607

Choetkiertikul M, Dam HK, Tran T, Pham T, Ghose A, Menzies T (2019) A deep learning model for estimating story points. IEEE Trans Softw Eng. https://doi.org/10.1109/TSE.2018.2792473

Chongpakdee P, Vatanawood W (2017) Estimating user story points using document fingerprints. In: 8th IEEE international conference on software engineering and service science (ICSESS), https://doi.org/10.1109/ICSESS.2017.8342885

Coelho E, Basu A (2012) Effort estimation in agile software development using story points. Int J Appl Inf Syst (IJAIS). https://doi.org/10.5120/ijais12-450574

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

Conoscenti M, Besner V, Vetrò A, Fernández DM (2019) Combining data analytics and developers feedback for identifying reasons of inaccurate estimations in agile software development. J Syst Softw. https://doi.org/10.1016/j.jss.2019.06.075
Dam H (2019) Empowering Software Engineering with. Artif Intell. https://doi.org/10.1007/978-3-030-32242-7_3
Dam HK (2018) Empowering software engineering with artificial intelligence. In: Service Research and Innovation, https://doi.org/10.1007/978-3-030-32242-7_3
Dave VS, Dutta K (2014) Neural network based models for software effort estimation: a review. Artif Intell Rev. https://doi.org/10.1007/s10462-012-9339-x
Dewi R, Sarno R (2020). Software effort estimation using early cosmic to substitute use case weight. https://doi.org/10.1109/iSemantic50169.2020.9234227
de Guevara FGL, Fernández-Diego M, Lokan C (2016) The usage of isbsg data fields in software effort estimation: a systematic mapping study. J Syst Softw. https://doi.org/10.1016/j.jss.2015.11.040
Dhir S, Kumar D, Singh V (2017) Feedforward and feedbackward approach-based estimation model for agile software development. In: Advances in Computer and Computational Sciences, https://doi.org/10.1007/978-981-10-3770-2_7
Dragicevic S, Celar S, Turic M (2017) Bayesian network model for task effort estimation in agile software development. J Syst Softw. https://doi.org/10.1016/j.jss.2017.01.027
Durán M, Juárez-Ramírez R, Jiménez S, Tona C (2020) User story estimation based on the complexity decomposition using bayesian networks. Program Comput Softw. https://doi.org/10.1134/S0361768820080095
Fernández-Diego M, Méndez ER, González-Ladrón-De-Guevara F, Abrahão S, Insfran E (2020) An update on effort estimation in agile software development: a systematic literature review. IEEE Access. https://doi.org/10.1109/ACCESS.2020.3021664
Ferrucci F, Mendes E, Sarro F (2012). Web effort estimation: The value of cross-company data set compared to single-company data set. https://doi.org/10.1145/2365324.2365330
Grabis J, Minkēviča V, Haidabrus B, Popovs R (2020) Is team always right: Producing risk aware effort estimates in agile development. In: international conference on business informatics research, https://doi.org/10.1007/978-3-030-61140-8_7
Grimstad S, Jorgensen M (2009) Preliminary study of sequence effects in judgment-based software development work-effort estimation. IET Softw. https://doi.org/10.1049/iet-sen.2008.0110
Haugen N (2006) An empirical study of using planning poker for user story estimation. In: AGILE 2006 (AGILE'06), https://doi.org/10.1109/AGILE.2006.16
Hussain I, Kosseim L, Ormandjieva O (2010) Towards approximating cosmic functional size from user requirements in agile development processes using text mining. In: Natural Language Processing and Information Systems
Hussain I, Kosseim L, Ormandjieva O (2013) Approximation of cosmic functional size to support early effort estimation in agile. Data Knowl Eng. https://doi.org/10.1016/j.datak.2012.06.005
Idria A, Hosnia M, Abranb A (2016) Systematic literature review of ensemble effort estimation. J Syst Softw. https://doi.org/10.1016/j.jss.2016.05.016
Jorgensen M, Grimstad S (2012) Software development estimation biases: the role of interdependence. IEEE Trans Softw Eng. https://doi.org/10.1109/TSE.2011.40
Jørgensen M, Halkjelsvik T (2020) Sequence effects in the estimation of software development effort. J Syst Softw. https://doi.org/10.1016/j.jss.2019.110448
Jørgensen M, Boehm B, Rifkin S (2009) Software development effort estimation: formal models or expert judgment? IEEE Softw. https://doi.org/10.1109/MS.2009.47
Kaushik A, Tayal DK, Yadav K (2020) A comparative analysis on effort estimation for agile and non-agile software projects using dbn-alo. Arabian J Sci Eng. https://doi.org/10.1007/s13369-019-04250-6
Kaushik A, Tayal DK, Yadav K (2020b) A fuzzy approach for cost and time optimization in agile software development. In: Advanced Computing and Intelligent Engineering, https://doi.org/10.1007/978-981-15-1081-6_53
Kaushik A, Tayal DK, Yadav K (2022) The role of neural networks and metaheuristics in agile software development effort estimation. In: Research Anthology on Artificial Neural Network Applications, https://doi.org/10.4018/978-1-6684-2408-7.ch014
Khuat T, Thi My Hanh L (2016) An effort estimation approach for agile software development using fireworks algorithm optimized neural network. Int J Comput Sci Inf Secur. https://doi.org/10.1016/j.infsof.2011.09.002
Kitchenham B, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering

Kocaguneli E, Menzies T, Keung J (2012) On the value of ensemble effort estimation. IEEE Trans Softw Eng. https://doi.org/10.1109/TSE.2011.111

Kompella L (2013). Advancement of decision-making in agile projects by applying logistic regression on estimates. https://doi.org/10.1109/ICGSEW.2013.9

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, https://doi.org/10.1007/978-3-319-18612-2_9

López-Martínez J, Juárez-Ramírez R, Ramírez-Noriega A, Licea G, Navarro-Almanza R (2017) Estimating user stories' complexity and importance in scrum with bayesian networks. In: world conference on information systems and technologies, 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 Comput. https://doi.org/10.1007/s10586-017-0996-z

Mahnič V, Hovelja T (2012) On using planning poker for estimating user stories. J Syst Softw. https://doi.org/10.1016/j.jss.2012.04.005

Malgonde O, Chari K (2019) An ensemble-based model for predicting agile software development effort. Empir Softw Eng. https://doi.org/10.1007/s10664-018-9647-0

Matharu GS, Mishra A, Singh H, Upadhyay P (2015) Empirical study of agile software development methodologies: a comparative analysis. SIGSOFT Softw Eng Notes 40:1–6

Minku LL, Yao X (2014) How to make best use of cross-company data in software effort estimation? In: proceedings of the 36th international conference on software engineering, https://doi.org/10.1145/2568225.2568228

Moharreri K, Sapre AV, 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), https://doi.org/10.1109/COMPSAC.2016.85

Moløkken-Østvold K, Jørgensen M (2004) Group processes in software effort estimation. Empir Softw Eng. https://doi.org/10.1023/B:EMSE.0000039882.39206.5a

Nassif AB, Azzeh M, Idri A, Abran A (2019) Software development effort estimation using regression fuzzy models. Comput Intell Neurosci. https://doi.org/10.1155/2019/8367214

Panda A, Satapathy SM, Rath SK (2015) Empirical validation of neural network models for agile software effort estimation based on story points. Procedia Computer Science 57:772–781. https://doi.org/10.1016/j.procs.2015.07.474, 3rd international conference on recent trends in computing 2015 (ICRTC-2015)

Phannachitta P (2020) On an optimal analogy-based software effort estimation. Inf Softw Technol 125:106330. https://doi.org/10.1016/j.infsof.2020.106330

Popli R, Chauhan N (2014) Cost and effort estimation in agile software development. In: 2014 international conference on reliability optimization and information technology (ICROIT), https://doi.org/10.1109/ICROIT.2014.6798284

Pow-Sang JA, Imbert R (2012) Effort estimation in incremental software development projects using function points. In: Computer Applications for Software Engineering, Disaster Recovery, and Business Continuity, https://doi.org/10.1007/978-3-642-35267-6_61

Radu LD (2019) Effort prediction in agile software development with bayesian networks. In: ICSOFT

Rao CP, Kumar PS, Sree SR, 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, https://doi.org/10.1007/978-981-10-8228-3_20

Ratke C, Hoffmann HH, Gaspar T, Floriani PE (2019) Effort estimation using bayesian networks for agile development. In: 2019 2nd international conference on computer applications information security (ICCAIS), https://doi.org/10.1109/CAIS.2019.8769455

Rosa W, Madachy R, Clark B, Boehm B (2017) Early phase cost models for agile software processes in the us dod. In: 2017 ACM/IEEE international symposium on empirical software engineering and measurement (ESEM), https://doi.org/10.1109/ESEM.2017.10

Rosencrance L (2007) Survey: poor communication causes most it project failures

Rowe G, Wright G (1999) The delphi technique as a forecasting tool: issues and analysis. Int J Forecast 15:353–375. https://doi.org/10.1016/S0169-2070(99)00018-7

Rubin K (2013) Essential scrum: a practical guide to the most popular agile process

Saeed A, Butt WH, Kazmi F, Arif M (2018) Survey of software development effort estimation techniques. In: proceedings of the 2018 7th international conference on software and computer applications, https://doi.org/10.1145/3185089.3185140

Saini A, Ahuja L, Khatri SK (2018) Effort estimation of agile development using fuzzy logic. In: 2018 7th international conference on reliability, infocom technologies and optimization (trends and future directions) (ICRITO), https://doi.org/10.1109/ICRITO.2018.8748381

Satapathy SM, Rath SK (2017) Empirical assessment of machine learning models for agile software development effort estimation using story points. Innov Syst Softw Eng. https://doi.org/10.1007/s11334-017-0288-z

Schwaber K, Sutherland J (2020) The scrum guide™

Schweighofer T, Kline A, Pavlic L, Hericko M (2016) How is effort estimated in agile software development projects? In: proceedings of 5th Workshop Software Qual., Anal., Monitor., Improvement, Appl. (SQAMIA)

Scott E, Pfahl D (2018). Using developers' features to estimate story points. https://doi.org/10.1145/3202710.3203160

Serrador P, Pinto JK (2015) Does agile work?—A quantitative analysis of agile project success. Int J Project Manag. https://doi.org/10.1016/j.ijproman.2015.01.006

Sharma A, Chaudhary N (2020). Linear regression model for agile software development effort estimation. https://doi.org/10.1109/ICRAIE51050.2020.9358309

Sharma H, Tomar R, Dumka A, Aswal M (2015) Openecocomo: The algorithms and implementation of extended cost constructive model (e-cocomo). https://doi.org/10.1109/NGCT.2015.7375225

Soares RGF (2018) Effort estimation via text classification and autoencoders. In: 2018 international joint conference on neural networks (IJCNN), https://doi.org/10.1109/IJCNN.2018.8489030

Tanveer B, Guzmán L, Engel UM (2017) Effort estimation in agile software development: case study and improvement framework. J Softw Evol Process. https://doi.org/10.1002/smr.1862

Tanveer B, Vollmer AM, Braun S, Bin Ali N (2019) An evaluation of effort estimation supported by change impact analysis in agile software development. J Softw Evol Process. https://doi.org/10.1002/smr.2165

Trendowicz A, Jeffery R (2014) Software project effort estimation. Springer, Berlin

Ungan E, Çizmeli N, Demirörs O (2014) Comparison of functional size based estimation and story points, based on effort estimation effectiveness in scrum projects. In: 2014 40th EUROMICRO conference on software engineering and advanced applications, https://doi.org/10.1109/SEAA.2014.83

Usman M, Mendes E, Neiva F, Britto R (2014). Effort estimation in agile software development: a systematic literature review. https://doi.org/10.1145/2639490.2639503

Wen J, Li S, Lin Z, Hu Y, Huang C (2012) Systematic literature review of machine learning based software development effort estimation models. Inf Softw Technol. https://doi.org/10.1016/j.infsof.2011.09.002

Wińska E, Kot E, Dabrowski W (2021) Reducing the uncertainty of agile software development using a random forest classification algorithm. In: international conference on lean and agile software development, https://doi.org/10.1007/978-3-030-67084-9_9

Wohlin C (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. https://doi.org/10.1145/2601248.2601268

Zang J (2008) Agile estimation with monte carlo simulation. In: international conference on agile processes and extreme programming in software engineering, https://doi.org/10.1007/978-3-540-68255-4_17