Comprehensive Analysis of Software Effort Estimation Techniques: Evolving Trends, Key Challenges, and Prospective Directions

Sahar Alturki
Department of Software Engineering,
College of Computer and Information Sciences, King Saud University,
Riyadh, Saudi Arabia

Fazal E-amin
Department of Software Engineering,
College of Computer and Information Sciences, King Saud University,
Riyadh, Saudi Arabia

ABSTRACT
Effort estimation remains a cornerstone of software project management, playing a pivotal role in project planning, resource allocation, and overall success. Over the years, its importance has only grown as software projects have become more complex and diverse. To deepen understanding in this area, this paper conducted a comprehensive review of software effort estimation techniques, analyzing 21 studies published between 2014 and 2024. This review addressed four key research questions, revealing Planning Poker as the most widely used expert-based estimation technique and Random Forest as the most frequently applied method in machine-based estimation. The findings underscore that inaccurate effort estimation is often linked to issues in requirement definition and management. Additionally, the study examines the impact of software development processes on estimation accuracy. Finally, it identified key limitations and proposed future research directions from the reviewed papers, providing actionable insights to improve effort estimation methods and practices in the field.

General Terms
Software, Effort Estimation, Analysis, Review, Trends, Challenges, Future Direction.

Keywords
Software, Effort Estimation, Analysis, Review, Trends, Challenges, Future Direction.

1. INTRODUCTION
Over the years, one of the important aspects of software project management is effort estimation. It is a critical component, as it significantly influences project planning, resource allocation, and project success. The software effort estimation (SEE) is a process of predicting the amount of effort, typically expressed in person-hours or person-months, required to develop or maintain a software project. Accurate estimations are essential to ensure that projects adhere to their budgets and timelines, ultimately leading to the delivery of high-quality software products.

Historically, SEE has encountered challenges due to the inherent complexity of software development processes, which are often subject to changing requirements and varying levels of uncertainty. Traditional estimation techniques provide frameworks for making these predictions, such as expert judgment and algorithmic models like COCOMO. However, these methods can sometimes lead to significant inaccuracies, resulting in either overestimations or underestimations of required effort.

The importance of effective SEE is underscored by studies indicating that a large percentage of software projects fail to meet their initial goals, often due to inaccurate effort predictions. For instance, a survey conducted by the Project Management Institute in 2017 noted that while 69% of projects achieved their original objectives, many faced budget overruns and delays, highlighting the critical need for improved estimation techniques [1], [2], [3]. A case study in [4] highlighted that effort estimation in distributed settings frequently required extensive discussions, which are inherently more difficult to facilitate compared to the face-to-face interactions of co-located teams. For example, in projects like EnergySoftware and PrintCo, participants expressed a strong preference for on-site planning meetings during effort estimation, as these allowed for more effective discussions and greater clarity. They also found [5] that the role of respondents, data collection approach, and type of analysis had an important influence on the reasons given for the estimation error.

This paper is organized as follows. Section 2 presents the previous related systematic and mapping literature review studies published in SEE. In Section 3, it presents the review methodology, research questions, and search strategies. Section 4 shows and discusses the results. Finally, in Section 5 it summarizes and presents the conclusions.

2. RELATED WORK
Agile software effort estimation involves predicting the time, cost, and resources required to complete tasks or projects, enabling teams to effectively plan, prioritize work, and manage scope within iterative and dynamic development environments. The following papers explore different aspects of software effort estimation in Agile Software Development (ASD) using Systematic Literature Review (SLR), and Systematic Mapping Study (SMS). In [6], the authors used a Forward Snowballing approach to analyze 312 papers and identify 24 relevant studies published between 2014 and 2017. Key findings include the continued popularity of Planning Poker as the most cited estimation technique, a significant increase in the use of AI and machine learning methods, and the identification of 10 cost drivers influencing effort estimation. These cost drivers include quality requirements, task size, integration, priority, complexity, stakeholder delays, team composition, work environment, expertise, and technical capability. The review highlights ongoing challenges in achieving accuracy and consistency with cost drivers, noting a shift toward incorporating more project- and people-related factors into estimation processes. Similarly, in [7], the authors analyzed 73 new studies published between 2013 and 2020. They confirm that expert judgment remains a key method, particularly

through techniques such as Planning Poker, while also highlighting a growing trend toward the adoption of machine learning and data-driven methods. The review identifies significant improvements in the reporting of accuracy metrics, with an increasing use of story points as the primary measure of magnitude. It also underscores the various cost factors influencing effort estimation, emphasizing the critical role of project and team dynamics. Overall, the findings point to the need for further refinement of estimation techniques and the standardization of cost factors in ASD. Also, in [8], a SMS of the literature from 2018 to 2022 identifies 25 relevant studies and highlights that the most common estimation techniques are story points (SP), Planning Poker (PP), and expert judgment (EJ). The research emphasizes the importance of accurate effort estimation for project success, noting that while machine learning techniques are increasingly applied, challenges remain, such as reliance on expert knowledge and a lack of sufficient data to generalize models.

Inaccurate software effort estimation in Agile can lead to project delays, budget overruns, and resource misallocation, often resulting from factors such as incomplete requirements, team dynamics, complex dependencies, and the inherent uncertainty in predicting effort within iterative and evolving development environments. In [9], the author presents a systematic mapping study examining the factors that impact the accuracy of software development effort estimation. It identifies thirty-two different factors categorized into four main groups: estimation process, estimator characteristics, project characteristics, and external context. The study highlights the importance of accurate effort estimation for project planning and management, emphasizing the need for researchers and practitioners to be aware of these factors and their influences. Additionally, it critiques the current state of research, noting a shift from broad overview studies to more focused investigations on specific factors, while also calling for improved definitions and methods in future research to better understand and enhance estimation accuracy. Similarly, in [10], the paper focuses on identifying the reasons behind inaccurate effort estimations in Agile software development and categorizing approaches to improve these estimations. The review highlights five main themes for inaccuracies: quality issues of available information, team-related factors, estimation practices, project management issues, and business influences. It emphasizes the importance of enhancing information quality and suggests that practitioners should consider adopting automated methods for better accuracy. Additionally, the study found that while numerous approaches have been proposed to improve estimation accuracy, many lack empirical validation and risk data leakage, urging further research to address these gaps.

3. METHODOLOGY

This review adheres to the guidelines established by Kitchenham et al. [11], encompassing three distinct stages: planning, conducting, and reporting. During the planning stage, the need for the review is identified, and a detailed protocol is developed to structure and guide the research process. The conducting stage involves systematically gathering relevant studies, selecting primary research, evaluating the quality of these studies, and extracting and synthesizing pertinent data. Finally, the reporting stage focuses on producing a comprehensive report that effectively summarizes the key findings and insights derived from the review.

3.1 Research questions

The aim of this research is to explore software effort estimation (SEE). Therefore, it refines the aim in four research questions described as follows:

RQ1: What are the most commonly used techniques and models for software effort estimation?

RQ2: What are the key factors influencing the accuracy of software effort estimation?

RQ3: How does effort estimation differ among scrum of scrum, distributed agile, and DevOps software development methodologies?

RQ4: What are the limitation and future direction in software effort estimation?

3.2 Search strategies and selection criteria

The electronic databases were chosen due to comprehensive coverage of software engineering literature. The relevant studies were collected from Four reputable academic databases, including: IEEE Xplore, ACM Digital Library, ScienceDirect, and SpringerLink. The search timeframe covered the publications from 2014 to 2024, to comprehensive coverage of relevant studies over the past twenty years. A well-constructed search string is essential for capturing the most relevant studies within the research domain. This search string is "Software Effort Estimation".

This paper established inclusion and exclusion criteria to ensure that only studies relevant to the scope and objectives of this review were considered.

3.2.1 Inclusion Criteria:

- Studies specifically addressing software effort estimation.
- Studies published between 2014 and 2024.

3.2.1 Exclusion Criteria:

- Studies categorized as training materials, editorials, article summaries, interviews, prefaces, news items, reviews, correspondence, tutorials, poster sessions, workshops, or panels.
- Studies not written in English.

To facilitate the management and streamlining of the systematic review process, this paper utilized Rayyan, a web-based application designed for this purpose. The stages of the study selection process are illustrated in Figure 1.

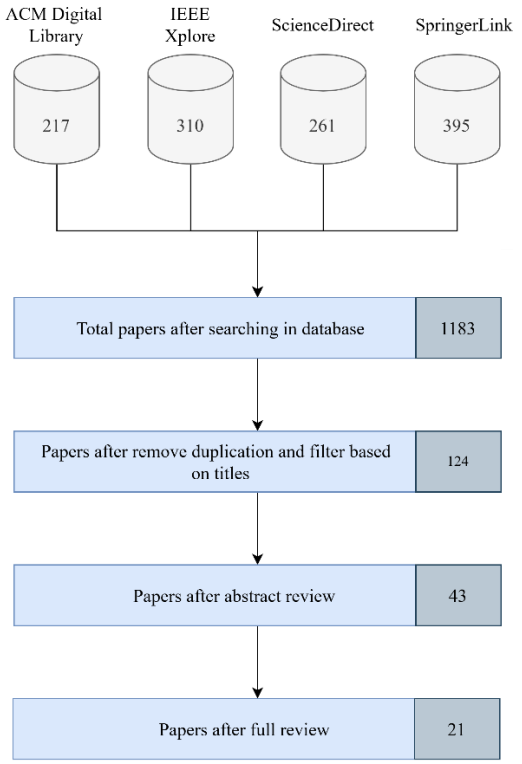

Figure 1: Search strategies and selection criteria

4. RESULT AND DISCUSSION

To address the four research questions, this paper has divided the discussion into the following four sections.

4.1 Techniques used in software effort estimation (RQ1)

The Software Effort Estimation (SEE) techniques are categorized into three groups: expert-based, machine-based, and hybrid-based approaches as present in Table 1. Among the expert-based effort estimation methods, Planning Poker (PP) is the most frequently mentioned technique in the reviewed papers, followed by Expert Judgment (EJ), Story Points (SP), Estimation by Analogy, and Delphi. Additionally, methods such as Use Case Points, Team Estimation Game, Bucket System, Dot Voting, Swimlane Sizing, Ideal Days, T-shirt Size/Dog Size, Functional Size Measures (FSM), Function Point (FP), and COSMIC Function Point (CFP) are mentioned with equivalent significance across the papers.

In contrast, among the machine-based estimation methods, Random Forest (RF) emerges as the most frequently mentioned technique in the reviewed papers. It is followed by K-nearest neighbors (KNN) and Decision Tree (DT), and subsequently by Support Vector Machine (SVM), Naive Bayes (NB), Linear Regression (LR), and Classification and Regression Trees (CART). Other techniques mentioned include Stochastic Gradient Boosting (SGB), Neural Networks (NN), Gradient Boosted Tree (GBT), Multilayer Perceptron (MLP), Support Vector Regression (SVR), Sentence-BERT (SBERT), Long Short-Term Memory (LSTM), AdaBoost, Genetic Programming (GP), Multi-Objective Genetic Programming (MOGP), Genetic Algorithms (GA), NSGAII-UO, and Random Guessing (RG).

Finally, among the hybrid-based estimation methods, COCOMO is the most frequently mentioned technique in the reviewed papers. It is followed by Linear Programming for Effort Estimation (LP4EE), Confidence Guided Effort Estimator (CoGEE), and Case-Based Reasoning (CBR). These methods combine elements of both expert judgment and machine-based approaches to enhance the accuracy and reliability of effort estimation.

4.2 The key factors influencing the accuracy in Software Effort Estimation (RQ2)

Software effort estimation (SEE) accuracy is a goal that many researchers seek to achieve, but in order to achieve it this paper need to identify the key factors that affect the accuracy of software effort estimation and lead to errors in it. The main factors that affect the accuracy of software effort estimation are requirements-related issues, where incomplete, ambiguous, or complex requirements are the most significant contributors to estimation inaccuracy. When requirements are poorly defined or subject to frequent changes, it becomes difficult for teams to accurately measure the effort required to implement. This leads to under- or overestimation, which directly impacts project schedules and budgets. This is followed by project management issues, where the experience, biases, and decision-making processes of project managers greatly impact estimation results. Inconsistent project management practices can lead to misaligned priorities and unrealistic expectations, further exacerbating estimation inaccuracy. Next come team-related issues, where challenges such as varying skill levels, distributed team structures, and limited collaboration opportunities impact the accuracy of estimates. Teams that struggle with communication or coordination are more likely to produce inaccurate effort forecasts. In addition, errors can stem from model and data issues, the use of outdated or insufficient models, and the lack of high-quality historical data undermine the reliability of machine-based estimation techniques. These issues highlight the need for better data collection practices and adaptive models. Other contributing factors include customer-related issues, such as insufficient client engagement or unclear expectations create uncertainty in determining outcomes. This often leads to inaccurate estimates and increased rework. Also, cultural-related issues, where differences in cultural perspectives and organizational silos hinder effective collaboration and understanding. Such barriers complicate estimation processes, especially in distributed environments. Finally, considerations of quality and risk factors, along with organizational issues, such as silos or inadequate communication within the organization, exacerbate the inaccuracy of the estimate. Ignoring quality considerations or underestimating risk can significantly distort estimation results. Comprehensive risk assessments and quality metrics are essential to improving estimation accuracy as shown in Figure 2.

Table 1: Categorize of Software Effort Estimation

Class	Effort Estimation Technique	Ref
Expert-Based Estimation Methods	Planning Poker	[10], [11], [12], [13], [14], [15], [17]
	Estimation by Analogy	[12], [13], [19]
	Expert Judgment	[12], [13], [16], [17], [19], [20], [21]
	Delphi	[12], [16]
	Story Points	[13], [17], [22], [23]
	Others (Use Case Points, Team Estimation Game, Bucket System, Dot Voting, Swimlane Sizing, Ideal Days, T-shirt Size/Dog Size, Functional Size Measures (FSM), Function Point (FP), COSMIC Function Point (CFP))	[17], [23]
Machine-Based Estimation Methods	Random Forest (RF)	[2], [16], [22], [24], [25], [26], [27]
	Decision Tree (DT)	[16], [22], [24], [25], [28]
	K-nearest neighbors (KNN)	[16], [25], [26], [27], [28]
	Naive Bayes (NB)	[2], [16], [26]
	Classification and Regression Trees (CART)	[26], [27], [29]
	Support Vector Machine (SVM)	[2], [16], [30]
	Linear Regression (LR)	[28], [28], [29]
	Stochastic Gradient Boosting (SGB)	[16], [22]
	Neural Networks (NN)	[2], [16]
	Gradient Boosted Tree (GBT)	[16], [25]
	Multilayer Perceptron (MLP)	[25], [28]
	Support Vector Regression (SVR)	[27], [28]
	Others (Sentence-BERT (SBERT), Long Short-Term Memory (LSTM), AdaBoost, Genetic Programming (GP), Multi-Objective Genetic Programming (MOGP), Genetic Algorithms (GA), NSGAII-UO, Random Guessing (RG))	[14], [22], [23], [24],
Hybrid-Based Estimation Methods	COCOMO	[12], [20], [21]
	Others (Linear Programming for Effort, Estimation (LP4EE), Confidence Guided Effort Estimator (CoGEE), Case-Based Reasoning (CBR))	[29], [30], [31]

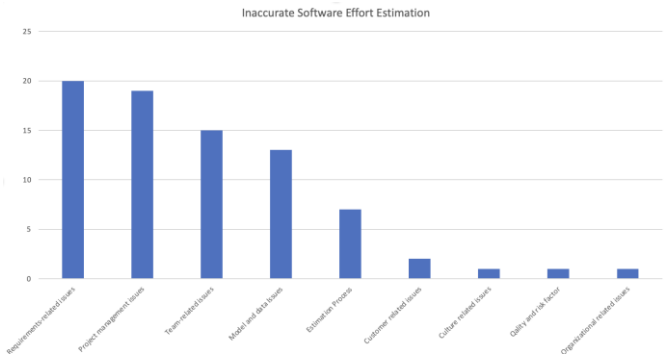

Figure 2: Key factors influence the accurate of software effort estimation

4.3 Effort estimation differ among scrum of scrum, agile, distributed agile, and DevOps software development (RQ3)

4.3.1 Estimate effort in Scrum of scrum

Scrum of Scrums meetings are conducted to facilitate communication among multiple teams and to address technical constraints and inter-team dependencies. These dependencies may sometimes require one team to prioritize and implement a low-priority feature earlier to unblock other teams. Scrum Masters from all teams participate in these meetings, which are typically held before the start of every sprint [15]. Effort estimation within the Scrum of Scrums framework is a critical component for managing large, distributed projects effectively. This process involves breaking down the product backlog into smaller, manageable sub-backlogs, enabling individual Scrum teams to estimate the effort required for their specific features. By doing so, dependencies between teams are identified and addressed early, ensuring that each team can operate autonomously while remaining aligned with the overall project objectives. The inclusion of local Product Owners (POs) in these meetings further enhances the prioritization process and effort assessment. This collaborative approach fosters improved communication and coordination among teams, resulting in more accurate effort estimations. Ultimately, the Scrum of Scrums framework not only mitigates risks associated with dependencies but also ensures better alignment, leading to improved project outcomes and streamlined delivery [32].

4.3.2 Estimate effort in DevOps

Effort estimation in DevOps is significantly affected by the dynamic nature of customer requirements, the need for rapid iterations, or sprints, communication, organizational culture, technology stack, and collaboration among teams. These factors can lead to either accurate or inaccurate estimations, which directly impact project delivery timelines and costs [20]. Traditional estimation methods often fall short due to this volatility; hence, methodologies like planning poker and advanced consensus-based techniques are employed to enhance accuracy and reduce bias among team members. By incorporating collective input and historical performance data of team members, DevOps teams can achieve more reliable estimates, which are crucial for effective sprint planning and execution. This collaborative approach not only aids in precise effort estimation but also fosters team cohesion, ultimately contributing to the success of the project [14].

4.3.3 Estimate effort in Distributed Agile

Effort estimation in distributed agile projects presents unique challenges due to the complexities introduced by the distributed

nature of teams. Global barriers, such as cultural differences, time zone disparities, language differences, and geographical distances, significantly impact the estimation process. These factors introduce unique cost drivers that are less pronounced in co-located agile teams, thereby complicating the task of providing accurate estimates. Additionally, distributed teams often encounter challenges related to communication and coordination, which further exacerbate the difficulties in effort estimation [12]. In [13] the authors highlighted that effort estimation in distributed settings frequently required extensive discussions, which are inherently more difficult to facilitate compared to the face-to-face interactions of co-located teams. However, due to the constraints of short sprint cycles, relying solely on in-person meetings was often not economically feasible. This limitation led to a dependence on teleconferencing and remote collaboration tools, which, while practical, sometimes hindered effective communication and the clarity needed for accurate estimation discussions. As a result, traditional effort estimation methods often fall short in distributed agile software development (DASD) contexts. To address these challenges, there is a need to integrate considerations for risk factors, as well as factors such as quality, novelty, and the type of work, into the estimation process to enhance accuracy and adaptability in distributed settings.

4.4 Limitation and future direction of software effort estimation (RQ4)

Based on the review, the most commonly mentioned limitation across the studies is the generalization of results, cited in 7 studies. The reviewed studies emphasize the difficulty in generalizing findings due to the diversity of software projects. This challenge limits the applicability of estimation techniques across different project contexts and domains. This is followed by concerns about the quality of historical data in 5 studies where the lack of comprehensive, high-quality datasets is a recurring issue. Studies underscore that insufficient data affects the training and validation of estimation models, making them less reliable in real-world scenarios. The complexity of software projects in 4 studies, where the increasing complexity of modern software projects, combined with the scale of distributed teams and global collaborations, introduces numerous variables that are hard to account for in estimation models. Other limitations include the maturity and experience of teams, dataset size, and the availability of real test data as presented in Figure 3. The lack of comprehensive, high-quality datasets is a recurring issue. Studies underscore that insufficient data affects the training and validation of estimation models, making them less reliable in real-world scenarios.

On the other hand, the reviewed papers highlight that 'validating the model with real project data and commercial software projects, while exploring the impact of varying user story sizes, and integrating factors such as team dynamics and challenges associated with distributed teams, can enhance the model's applicability across different project scales. Expanding the sample size will further improve the generalizability of the results is identified as a key direction for future work in 7 studies. This is followed by the development of 'comprehensive frameworks, guidelines, and training programs to enhance the estimation skills of teams,' and 'exploring more advanced machine learning techniques, including deep learning and unsupervised learning methods, to enhance prediction accuracy. Techniques such as Extreme Learning Machines and Bayesian Networks could be applied to a larger and more diverse dataset, improving the robustness and applicability of effort estimation models in agile software development—both of which are the second most frequently mentioned future work directions, as presented in Table 2.

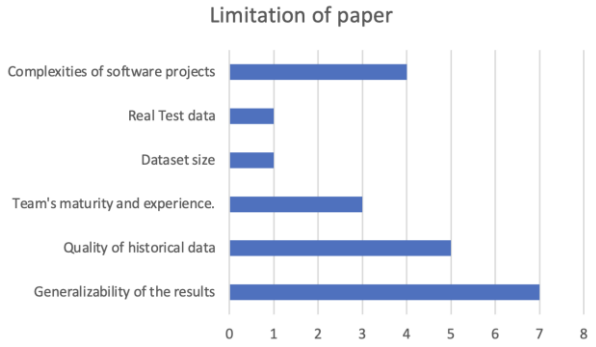

Figure 3: Limitation of the reviewed papers

5. LIMITATIONS

This study focuses on 21 papers published between 2014 and 2024. While comprehensive, this limited scope may overlook other significant research contributions outside the selected timeframe or databases. Also, it does not involve primary data collection or empirical validation. Instead, it synthesizes findings from existing literature, which may limit its contribution to actionable insights or practical implementation. Additionally, it emphasizes agile methodologies, particularly Scrum and DevOps. This focus may limit the generalizability of the findings to other development paradigms or methodologies.

Despite efforts to maintain objectivity, researcher bias in interpreting and synthesizing findings from diverse sources cannot be entirely eliminated. By acknowledging these challenges and limitations, this study provides a transparent foundation for interpreting its findings and sets the stage for future research to address these gaps.

6. CONCLUSION

This paper conducted a comprehensive review of software effort estimation techniques, analyzing 21 studies published between 2014 and 2024. The review aimed to address four key research questions. It identified that the most commonly used expert-based effort estimation technique is Planning Poker, while Random Forest is the most frequently employed method in machine-based effort estimation. The findings also highlight that inaccurate effort estimation often stems from issues related to requirements. Additionally, it explored the impact of software development processes on effort estimation accuracy. The review also identified several limitations and proposed future directions for improving software effort estimation practices. These include the need for more research on hybrid techniques that combine expert-based and machine-based methods to improve accuracy. Future studies could also explore the application of newer machine learning algorithms and artificial intelligence to enhance predictive capabilities. Furthermore, addressing the challenges related to requirement uncertainty and integrating agile practices into estimation models are potential areas for development. As software development processes continue to evolve, future work should investigate how emerging methodologies, such as DevOps and continuous delivery, influence estimation practices. These advancements could provide valuable insights and contribute to the refinement of software effort estimation techniques in the years to come.

Table 2: Reviewed papers future directions

Future direction	Ref
Interviews in both co-located and distributed contexts	[12]
Enhancing effort estimation models by incorporating more sophisticated risk assessment techniques, improving the integration of quality metrics, and exploring machine learning approaches to adaptively refine estimates based on historical project data. Expanding the dataset to include a broader range of projects will further improve the accuracy and robustness of the models.	[13], [14]
Exploring the integration of emerging technologies, such as Machine Learning Operations (MLOps), into effort estimation practices could provide valuable insights for enhancing the accuracy and efficiency of these strategies.	[20]
Developing comprehensive frameworks, guidelines, and training programs to enhance the estimation skills of teams.	[13], [30]
Exploring more advanced machine learning techniques, including deep learning and unsupervised learning methods, to enhance prediction accuracy. Techniques such as Extreme Learning Machines and Bayesian Networks could be applied to a larger and more diverse dataset, improving the robustness and applicability of effort estimation models in agile software development.	[2], [22], [24]
Validating the model with real project data and commercial software projects, while exploring the impact of varying user story sizes. Additionally, integrating factors such as team dynamics and challenges associated with distributed teams can enhance the model's applicability across different project scales. Expanding the sample size will further improve the generalizability of the results.	[14], [15], [19], [21], [27], [28], [29]
Expanding the applicability of LFM to various software engineering predictive tasks beyond effort estimation, such as bug-fixing time prediction and customer ratings prediction, can broaden its utility and enhance the accuracy of forecasting in different aspects of software development.	[26]
Conducting longitudinal studies to analyze the effectiveness of different estimation techniques over time and explore the impact of factors such as team dynamics, project complexity, and evolving agile methodologies on estimation accuracy can provide valuable insights into the long-term performance and adaptability of estimation models in real-world scenarios.	[19]
Adapting the model for various estimation scenarios, such as cross-company versus within-company analyses, and expanding its application to other software engineering tasks, like predicting app ratings and bug-fixing times, will increase its versatility and improve its utility across a broader range of predictive tasks in software development.	[27]

7. REFERENCES

[1] S. A. SALIHU, K. B. SALIU, and O. A. OWOYEMI, “A Systematic Literature Review of Machine Learning and AutoML In Software Effort Estimation,” in Conference Organising Committee, 2024, p. 145.

[2] F. B. Alhamdany and L. M. Ibrahim, “Software development effort estimation techniques: A survey,” J. Educ. Sci., vol. 31, no. 1, pp. 80–92, 2022.

[3] Y. Mahmood, N. Kama, A. Azmi, A. S. Khan, and M. Ali, “Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation,” Softw. Pract. Exp., vol. 52, no. 1, pp. 39–65, 2022.

[4] M. Paasivaara, S. Durasiewicz, and C. Lassenius, “Using scrum in distributed agile development: A multiple case study,” in 2009 Fourth IEEE International Conference on Global Software Engineering, IEEE, 2009, pp. 195–204.

[5] M. Jorgensen and K. Molokken-Ostvold, “Reasons for software effort estimation error: impact of respondent role, information collection approach, and data analysis method,” IEEE Trans. Softw. Eng., vol. 30, no. 12, pp. 993–1007, 2004.

[6] E. Dantas, M. Perkusich, E. Dilorenzo, D. F. Santos, H. Almeida, and A. Perkusich, “Effort estimation in agile software development: An updated review,” Int. J. Softw. Eng. Knowl. Eng., vol. 28, no. 11n12, pp. 1811–1831, 2018.

[7] M. Fernández-Diego, E. R. Méndez, F. González-Ladrón-De-Guevara, S. Abrahão, and E. Insfran, “An update on effort estimation in agile software development: A systematic literature review,” IEEE Access, vol. 8, pp. 166768–166800, 2020.

[8] C. A. P. Rodríguez, L. M. S. Martinez, D. H. P. Ordoñez, and J. A. T. Peña, “Effort Estimation in Agile Software Development: A Systematic Map Study,” Inge Cuc, vol. 19, no. 1, pp. 22–36, 2023.

[9] D. Basten and A. Sunyaev, “A systematic mapping of factors affecting accuracy of software development effort estimation,” Commun. Assoc. Inf. Syst., vol. 34, no. 1, p. 4, 2014.

[10] J. Pasuksmit, P. Thongtanunam, and S. Karunasekera, “A Systematic Literature Review on Reasons and Approaches for Accurate Effort Estimations in Agile,” ACM Comput. Surv., 2024.

[11] S. Keele and others, “Guidelines for performing systematic literature reviews in software engineering,” Technical report, ver. 2.3 ebse technical report. ebse, 2007.

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

[13] W. Aslam, F. Ijaz, M. I. U. Lali, and W. Mehmood, “Risk Aware and Quality Enriched Effort Estimation for Mobile Applications in Distributed Agile Software Development.,” J Inf Sci Eng, vol. 33, no. 6, pp. 1481–1500, 2017.

[14] J. Angara, S. Prasad, and G. Sridevi, “DevOPs project management tools for sprint planning, estimation and execution maturity,” Cybern. Inf. Technol., vol. 20, no. 2, pp. 79–92, 2020.

[15] D. Badampudi, "Factors Affecting Efficiency of Agile Planning: A Case Study." 2012.

[16] B. Yalçıner, K. Dinçer, A. G. Karaçor, and M. Ö. Efe, "Enhancing Agile Story Point Estimation: Integrating Deep Learning, Machine Learning, and Natural Language Processing with SBERT and Gradient Boosted Trees," Appl. Sci., vol. 14, no. 16, p. 7305, 2024.

[17] R. Sandeep, M. Sánchez-Gordón, R. Colomo-Palacios, and M. Kristiansen, "Effort estimation in agile software development: a exploratory study of practitioners' perspective," in International Conference on Lean and Agile Software Development, Springer, 2022, pp. 136–149.

[18] F. Raith, I. Richter, R. Lindermeier, and G. Klinker, "Identification of inaccurate effort estimates in agile software development," in 2013 20th Asia-Pacific Software Engineering Conference (APSEC), IEEE, 2013, pp. 67–72.

[19] M. Usman, E. Mendes, and J. Börstler, "Effort estimation in agile software development: a survey on the state of the practice," in Proceedings of the 19th international conference on Evaluation and Assessment in Software Engineering, 2015, pp. 1–10.

[20] D. Meedeniya and H. Thennakoon, "Impact factors and best practices to improve effort estimation strategies and practices in devops," in Proceedings of the 11th International Conference on Information Communication and Management, 2021, pp. 11–17.

[21] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," IEEE Trans. Softw. Eng., vol. 45, no. 7, pp. 637–656, 2018.

[22] S. M. Satapathy and S. K. Rath, "Empirical assessment of machine learning models for agile software development effort estimation using story points," Innov. Syst. Softw. Eng., vol. 13, no. 2, pp. 191–200, 2017.

[23] V. Tawosi, R. Moussa, and F. Sarro, "On the relationship between story points and development effort in Agile open-source software," in Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2022, pp. 183–194.

[24] E. Rodríguez Sánchez, E. F. Vázquez Santacruz, and H. Cervantes Maceda, "Effort and Cost Estimation Using Decision Tree Techniques and Story Points in Agile Software Development," Mathematics, vol. 11, no. 6, p. 1477, 2023.

[25] A.-E. Iordan, "An Optimized LSTM Neural Network for Accurate Estimation of Software Development Effort," Mathematics, vol. 12, no. 2, p. 200, 2024.

[26] F. Sarro, R. Moussa, A. Petrozziello, and M. Harman, "Learning from mistakes: Machine learning enhanced human expert effort estimates," IEEE Trans. Softw. Eng., vol. 48, no. 6, pp. 1868–1882, 2020.

[27] F. Sarro and A. Petrozziello, "Linear programming as a baseline for software effort estimation," ACM Trans. Softw. Eng. Methodol. TOSEM, vol. 27, no. 3, pp. 1–28, 2018.

[28] M. A. Ramessur and S. D. Nagowah, "A predictive model to estimate effort in a sprint using machine learning techniques," Int. J. Inf. Technol., vol. 13, no. 3, pp. 1101–1110, 2021.

[29] F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation," in Proceedings of the 38th International Conference on Software Engineering, 2016, pp. 619–630.

[30] V. Tawosi, R. Moussa, and F. Sarro, "Agile effort estimation: Have we solved the problem yet? Insights from a replication study," IEEE Trans. Softw. Eng., vol. 49, no. 4, pp. 2677–2697, 2022.

[31] F. Sarro, F. Ferrucci, and C. Gravino, "Single and multi objective genetic programming for software development effort estimation," in Proceedings of the 27th annual ACM symposium on applied computing, 2012, pp. 1221–1226.

[32] A. M. AlMutairi and M. R. J. Qureshi, "The proposal of scaling the roles in scrum of scrums for distributed large projects," J. Inf. Technol. Comput. Sci. IJITCS, vol. 7, no. 8, pp. 68–74, 2015.