# Supervised Machine Learning (ML) Techniques to Predict Accurate Software Development Effort, Schedule, and Cost using Story Points: A Systematic Literature Survey

Prasada Rao Chatla[1], Dr. S. Rama Sree[2]
[1,2] Aditya University, Surampalem, Kakinada, Andhra Pradesh, India
[1]prasada.chatla@gmail.com
[2]ramasree_s@adityauniversity.in

***Abstract*—Accurate software effort estimation is a critical and challenging activity in Agile Software Development, as it directly impacts project cost, schedule, resource allocation, and overall success. Traditional estimation techniques such as Expert Judgment, Planning Poker, COCOMO, Function Point Analysis, and Use Case Points often struggle to deliver reliable estimates in dynamic and evolving Agile environments. With the growing availability of historical project data, supervised Machine Learning (ML) techniques have emerged as effective data driven alternatives for improving estimation accuracy and consistency. This paper presents a Systematic Literature Review (SLR) of recent supervised ML approaches applied to Story Point–based software effort estimation in Agile projects. The review analyzes regression models, ensemble learning techniques, neural networks, and emerging Natural Language Processing (NLP) and Transformer based models, including Random Forest, Gradient Boosting, Support Vector Machines, GPT2SP, and HeteroSP. The findings indicate that ensemble learning approaches consistently outperform conventional estimation methods in terms of robustness and predictive accuracy, while Transformer based models show strong potential for automated Story Point estimation using textual user story information. The study also identifies key research challenges related to dataset scarcity, estimation subjectivity, model explainability, and cross project generalization. Finally, future research directions are discussed, emphasizing Explainable AI, hybrid estimation frameworks, and intelligent Agile analytics..**

***Index Terms*—Agile Software Development, Story Points, Software Effort Estimation, Supervised Machine Learning, Ensemble Learning, Random Forest, Gradient Boosting, Natural Language Processing, Transformer Models, Systematic Literature Review.**

## I. Introduction

As a part of SDLC, Effort Estimation is an influential activity in the process of planning and bidding of the project. Without estimating Effort, Schedule, and Cost software cannot be developed and deployed in the real world. It is always important to do accurate estimation as much as possible. Poor effort estimation may lead to delays in product deployment, an increase in budget, and customer expectations may not be reached. Lots of projects fail due to poor estimations [1],[2], [10]. Accurate Estimates will help the stakeholders to establish a better Business Case. So effective Effort Estimations can be treated as a big challenge in software development. In the Conventional Model, the estimations are not accurate. They used different metrics like Kilo Lines Of Code (KLOC), Functional Points (FP), Use Case Points (UCP), and Test Case Points (TCP), etc., which are not very helpful to the development team to derive better estimations [1], [2], [3]. Currently, most IT companies are moving towards adopting Agile Software Development Models (ASDM). In comparison with the above conventional models, ASDM provides an accurate estimation of the effort, duration, and cost. In Agile teams, an Estimate is a measure of the project effort needed to carry out a given development task that is usually expressed as a User Story or Product Backlog. Effort in Agile estimation has three things, firstly the amount of work that has to be done, secondly any complexity involved in doing that task and finally it considers any uncertainties during the time of estimation [5], [6], [10]. ASDM helps to develop and deploy the project on time, within budget and as per customer expectations. We used to Develop/Write User Stories as Requirements in the Agile models. User Story doesn't take lots of time to develop, deploy, and it also helps to provide useful information about our progress and work remaining. We used to count the number of story points for each user story using Planning Poker cards (Mostly) and Expert Judgement techniques [4], [5] The conventional estimation approaches like Expert Judgement and Planning Poker cards which are used in Agile Software Development don't provide accurate estimations. As per the PMI (Project Management Institute) survey • 69% of the projects have successfully met the expected goals and reach the scope of the project and 43% of the projects weren't completed within the stipulated budget. • 48% of the projects were delivered late and 15 % of the projects failed due to Inaccurate estimation. Nowadays most of the research experts and even companies are adopting Artificial ML Techniques to estimate effort in Agile Methodologies [6], [7], [9], [11], [12].

## II. Background Work

Software effort estimation techniques can be broadly categorized into algorithmic, non algorithmic, and Machine Learning–based approaches, each addressing estimation challenges from different perspectives [1], [6], [10]. Effort is generally measured in person months or person weeks and plays a critical role in project planning and resource allocation. Algorithmic estimation techniques rely on mathematical models derived from software size and complexity metrics. Widely used algorithmic models include Function Point Analysis (FPA), COCOMO and COCOMO II, Kilo Lines of Code (KLOC), and Use Case Points (UCP). These methods assume stable requirements and predefined productivity factors, making them more effective in traditional plan driven development environments than in Agile settings [1], [2], [3].

### A. FPA

FPA was developed by Allan J Albrecht who worked at IBM in the late 1970s. The main objective of the FPA is to overcome the difficulties associated with Source Lines of Code (SLOC) as a measure of software size and to help in developing a technique to estimate the effort in the early stage of the Software Development Life Cycle (SDLC). FPA considers the End User View of the application to be developed or updated [3]. To determine VAF, we need to have 14 General

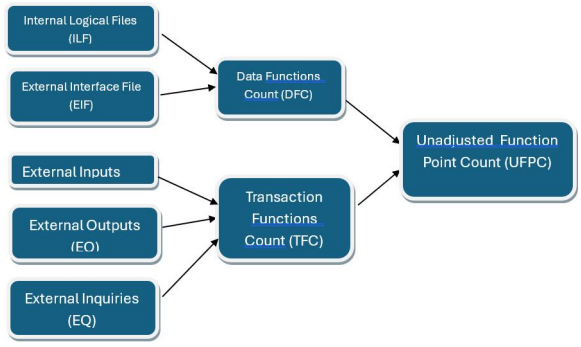


**Figure 2.1.** Steps to calculate Function Points.

Fig. 1. Steps to calculate Function Points.

System Characteristics (GSC) which are general functionality provided to the user, and the Degree of Influence (DI) of GSC would be from 0 to 5.

$$VAF = 0.65 + (0.01 TotalDI) \tag{1}$$

Calculate Final Adjusted Functional Point Count (FPC).

$$FPC = UFPCV\ AF \tag{2}$$

### B. UCP

UCP is one of the prominent software estimation techniques for estimating software size once the requirements are elicited. When companies adopt Unified Modelling Language (UML) and Rational Unified Process as a software development methodology, the UCP is more reliable and effective [3],[6] The UCP estimation method to estimate software size is based

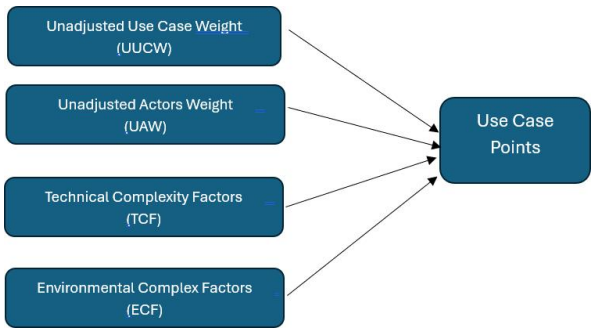


Fig. 2. Steps to Calculate Use Case Points

on the following calculations.

$$\begin{aligned} UUCW = &\sum (\text{Simple Use Cases} \times 5) \\ &+ \sum (\text{Average Use Cases} \times 10) \\ &+ \sum (\text{Complex Use Cases} \times 15) \end{aligned} \tag{3}$$

$$\begin{aligned} UAW = &\sum (\text{Simple Actors} \times 1) \\ &+ \sum (\text{Average Actors} \times 2) \\ &+ \sum (\text{Complex Actors} \times 3) \end{aligned} \tag{4}$$

$$TCF = 0.6 + (0.01 \times TF) \tag{5}$$

$$ECF = 1.4 + (-0.03 \times EF) \tag{6}$$

$$UCP = (UUCW + UAW) \times TCF \times ECF \tag{7}$$

$$\text{Total Project Estimate} = UCP \times \text{Productivity Factor (PF)} \tag{8}$$

where TF is the sum of the values calculated by 13 technical factors. where EF is the total of the values calculated by 8 environmental factors. where PF is the hours of development per UCP

### C. Story Points

The SP are determined from User Stories of Product Backlog in Agile methodologies. The User Storie, as its name suggests, addresses the description of the user's experience. User Stories consist of only one sentence and are formulated as simply as possible. User Stories (Requirements) can be changed during the project and the results should be visible for every increment. When we are writing user stories, short sentences and paragraphs should be considered. We can use only one requirement per sentence and consistent terminology should be considered. Story Points represent effort by considering development complexity, uncertainty, and risk rather than

implementation time alone [5], [6]. Agile estimation emphasizes incremental planning and team velocity, which improves adaptability but introduces variability due to differences in team experience and estimation culture [10].

In Agile projects User Stories can be measured using Story Points. The Scrum team allocates story points to user stories from the product backlog from six numbers Fibonacci Sequence (1,2,3,5,8,13,20 and 50). If any of the User Story allocated 50 or even 20 then it should be divided into smaller and more manageable user stories. Team Velocity plays an influential role in the estimation of effort, schedule, and cost in Scrum. Velocity is expressed as story points per iteration which means how quickly the development team can deliver project stories per iteration (Sprint). Non algorithmic estimation techniques primarily depend on human expertise and collective decision making. Methods such as Expert Judgment, Planning Poker, Wideband Delphi, and analogy based estimation are widely used in practice, particularly in Agile teams [4], [5]. While these techniques benefit from practitioner knowledge and collaboration, they remain subjective and susceptible to cognitive bias and inconsistencies across teams. To address the subjectivity and inconsistency of Agile estimation, recent research has increasingly focused on Machine Learning–based effort estimation approaches. Supervised Machine Learning models learn estimation patterns from historical Agile project data and have demonstrated improved predictive accuracy compared to traditional methods [6], [7], [9]. Popular ML techniques include Linear Regression, Support Vector Machines, Decision Trees, Random Forest, and Gradient Boosting models. Several empirical studies report that ensemble learning models, particularly Random Forest and Gradient Boosting, outperform standalone models by improving robustness and reducing overfitting in Story Point–based datasets [6], [7]. These findings highlight the growing importance of data driven estimation techniques in modern Agile software development.

TABLE I
AGILE PROJECT EFFORT, SCHEDULE, AND COST ESTIMATION

| Parameter | Value / Calculation |
|---|---|
| Total Project Stories | 34 |
| Total Project Story Points (SP) | 147 |
| Development Team Velocity (V) | 22 story points per iteration |
| Iteration Duration (D) | 4 weeks |
| Developer Number (per Iteration) (DN) | 5 |
| Developer Cost (per Developer per Iteration) (DC) | $7,500 |
| Estimated Iterations (I) | $I = \frac{SP}{V} = \frac{147}{22} = 6.7 \approx 7$ |
| Estimated Schedule | $I \times D = 7 \times 4 = 28$ weeks |
| Estimated Cost | $I \times DN \times DC$<br>$7 \times 5 \times 7500 = \$262{,}500$ |

## III. RESEARCH METHODOLOGY

This study adopts a Systematic Literature Review (SLR) methodology to comprehensively analyze and synthesize existing research on the application of supervised Machine Learning (ML) techniques for Story Point–based software effort estimation in Agile Software Development. The methodology is designed to ensure rigor, transparency, and reproducibility by following established SLR guidelines and incorporating principles inspired by the PRISMA framework.

### A. Research Objectives and Questions

The primary objective of this research is to systematically investigate how supervised ML techniques have been applied to improve the accuracy and reliability of software effort estimation using Story Points in Agile environments.

Based on this objective, the study addresses the following Research Questions (RQs):

- RQ1: How are supervised Machine Learning algorithms applied for predicting software development effort using Story Points in Agile projects?
- RQ2: What types of datasets and features are commonly used in Story Point–based ML effort estimation?
- RQ3: Which supervised ML techniques demonstrate superior predictive performance based on commonly used evaluation metrics?
- RQ4: What are the major challenges, limitations, and research gaps identified in existing studies?

### B. Literature Search Strategy

A systematic and comprehensive literature search was conducted across several well-established digital libraries to ensure broad coverage of high-quality and peer-reviewed studies. The following electronic databases were used:

- IEEE Xplore
- ACM Digital Library
- Scopus
- ScienceDirect (Elsevier)
- SpringerLink
- Google Scholar

The search focused on studies published between 2020 and 2025, reflecting recent advances in ML, NLP, and Transformer-based models relevant to Agile effort estimation.

Representative search strings included:

- (“software effort estimation” AND “story points” AND “machine learning”)
- (“Agile effort estimation” AND “supervised learning”)
- (“story point estimation” AND “deep learning” OR “transformer”)
- (“user stories” AND “NLP” AND “effort estimation”)

### C. Study Selection Criteria

To ensure relevance and quality, each retrieved study was evaluated using predefined inclusion and exclusion criteria.

**Inclusion Criteria:**

- Peer-reviewed journal articles or conference papers
- Studies focused on Agile Software Development
- Use of Story Points as a primary estimation metric
- Application of supervised ML techniques with empirical validation
- Clear description of datasets, methodology, and evaluation metrics

**Exclusion Criteria:**

- Studies unrelated to Agile or Story Point estimation
- Papers lacking experimental validation
- Conceptual papers without empirical evidence
- Duplicate studies or non-English publications

### *D. Study Screening and Data Extraction*

The screening process was conducted in multiple phases:

1) Title and Abstract Screening
   Studies were initially screened based on titles and abstracts to assess relevance.
2) Full Text Review
   Selected papers were reviewed in full to ensure compliance with inclusion criteria.
3) Final Selection
   A curated set of high-quality studies was selected for detailed analysis.

For each selected study, the following data were systematically extracted:

- Publication year and venue
- Supervised ML techniques used
- Dataset characteristics (size, source, features)
- Estimation inputs (Story Points, user story text, velocity, etc.)
- Evaluation metrics (MMRE, MdMRE, MAE, RMSE, PRED(n))

## IV. Machine Learning Techniques

Tom Mitchell stated in 1997 that Machine Learning (ML) techniques enable systems to learn from historical data and improve their performance through experience without being explicitly programmed for every task. ML techniques are broadly classified into two major categories: Supervised Learning and Unsupervised Learning. In addition, Hybrid Models combine Expert Judgement with Machine Learning predictions to improve estimation accuracy and reliability.

Supervised Learning techniques predict future outcomes using historical or previously labeled data. These techniques are generally categorized into:

- Regression Models, where output labels are continuous values
- Classification Models, where output labels are discrete categories

### *A. Linear Regression (LR) Models*

Linear Regression is one of the most widely used supervised learning techniques for prediction and estimation. It predicts the dependent variable $Y$ using one or more independent variables $X$. The independent variables represent input features, while the dependent variable represents the target output.

For example, Linear Regression can be used to predict:

- house prices based on area,
- software effort based on Story Points,
- or employee salary based on experience.

Linear Regression is popular because of its:

- simplicity,
- interpretability,
- and computational efficiency.

However, Linear Regression performs effectively only when relationships between variables are approximately linear.

### *B. Gradient Boosting Machines (GBM)*

Gradient Boosting Machines (GBM) were originally proposed by Friedman in 2001. GBM is a powerful ensemble-based supervised learning technique widely used for software effort estimation and predictive analytics.

The primary objective of GBM is to improve prediction accuracy by iteratively combining multiple weak prediction models into a stronger predictive model. GBM can be applied to both:

- regression problems,
- and classification problems.

Gradient Boosting techniques such as:

- XGBoost,
- LightGBM,
- and CatBoost

have demonstrated strong predictive capability in software effort estimation research.

### *C. Decision Trees*

Decision Trees are supervised learning techniques that can be used for both classification and regression problems. Decision Trees use a hierarchical tree-like structure to split data into smaller subsets based on feature conditions.

Unlike Linear Regression, Decision Trees can model nonlinear relationships between input variables and target outputs.

Decision Trees are advantageous because they provide:

- easy interpretability,
- rule-based prediction,
- and visualization capability.

However, single Decision Trees may suffer from:

- overfitting,
- instability,
- and reduced generalization performance.

### *D. Random Forest*

Random Forest is an ensemble learning technique based on multiple Decision Trees. Instead of using a single Decision Tree, Random Forest constructs multiple trees using random subsets of training data and feature selection.

The final prediction is generated by aggregating the predictions of multiple trees, thereby improving:

- prediction robustness,
- generalization capability,
- and estimation accuracy.

Random Forest is widely used in software effort estimation because it effectively handles:

- noisy datasets,
- nonlinear relationships,
- and feature interaction complexity.

*E. Support Vector Machines (SVM)*

Support Vector Machines (SVM) are powerful supervised learning models used for both classification and regression tasks. SVM constructs optimal hyperplanes that maximize separation between different classes or prediction boundaries.

SVM models are particularly effective for:

- high-dimensional datasets,
- nonlinear prediction problems,
- and complex feature relationships.

Kernel functions such as:

- linear kernel,
- polynomial kernel,
- and radial basis function (RBF)

allow SVM to model nonlinear estimation behavior effectively.

Although Deep Learning models have gained popularity in recent years, SVM continues to remain an effective technique for software analytics and predictive estimation research.

## V. Results

This section presents the answers to the Research Questions described in Section III. A total of 20 papers were initially extracted from various digital libraries. After applying the inclusion and exclusion criteria, 10 high-quality studies were selected for detailed analysis.

In [13], the authors stated that traditional estimation techniques such as Expert Opinion, Analogy, and Disaggregation often fail to provide accurate software effort estimation for Agile projects. To overcome these limitations, the study introduced supervised Machine Learning techniques including Decision Trees, Random Forest, and Gradient Boosting. The dataset consisted of 21 software projects collected from six software houses and included Story Points, Team Velocity, and Actual Effort as key parameters. Evaluation metrics such as MMRE, MdMRE, and PRED(n) were used for comparative analysis. The Stochastic Gradient Boosting (SGB) technique demonstrated superior estimation performance compared with other evaluated models.

In [15], the authors proposed a predictive model for sprint effort estimation using supervised Machine Learning techniques including Linear Regression, K-Nearest Neighbor (KNN), Polynomial Kernel methods, and Multi-Layer Perceptron (MLP). Due to confidentiality constraints, the original industrial dataset was not publicly available; however, the study collected data from 21 projects across six software houses and generated approximately 2100 records using 12 influencing factors. Experimental analysis showed that the MLP model achieved lower estimation error and improved predictive accuracy compared with other models.

In [16], the authors implemented the Na¨ıve Bayes algorithm for software effort estimation using historical project data. The SEERA dataset was used for training and validation purposes. The study compared Na¨ıve Bayes with Support Vector Machines (SVM) using the COCOMO II model. Experimental results indicated that the Na¨ıve Bayes approach achieved an estimation accuracy of 95.06%, outperforming SVM which achieved 93.45%.

In [17], the authors introduced six ensemble models for estimating software effort using Agile user stories. The study also established a dataset containing 140 user stories to support future research. Design Science Research (DSR) methodology was adopted to construct the ensemble framework. Individual models including Extra Trees, KNN, and MLP were evaluated using MAE, MSE, and RMSE metrics. The ensemble models demonstrated superior performance compared with individual prediction models.

In [18], the authors investigated early-phase software effort estimation using Natural Language Processing (NLP) techniques. The proposed approach extracted similar user stories from historical repositories to support Agile effort estimation. The study collaborated with a Danish software development company and used NLP techniques to identify related historical user stories and their corresponding development effort. The approach assists developers in estimating effort, schedule, and cost for new Agile projects.

In [19], the authors stated that Story Point estimation is often unavailable during early business-case development phases in organizations such as the Department of Homeland Security (DHS) and Department of Defense (DoD). The study analyzed Functional Stories and Issues as alternative sizing measures for early-phase effort estimation. Experimental analysis was conducted using 17 Agile projects developed between 2014 and 2021. The results demonstrated that Functional Stories and Issues are effective estimators of total software development effort.

In [20], the authors proposed hybrid estimation models combining algorithmic approaches with supervised Machine Learning techniques. Historical Agile project data were trained and tested using Decision Trees and Random Forest algorithms. Models were evaluated using 10-fold cross-validation and relative error analysis. The Bootstrap Aggregation approach demonstrated the highest prediction accuracy among the evaluated techniques.

In [11], the authors proposed the HeteroSP framework for Story Point estimation using textual descriptions of Agile project issues. The study compared HeteroSP with GPT2SP and DEEP-SP models using within-project and cross-project estimation scenarios. Experimental results reported average MAE values of 2.38, 2.61, and 2.63 respectively. The results indicated that HeteroSP achieved reliable Story Point estimation performance.

In [21], the authors investigated automated software effort prediction using Graph Neural Networks (GNN) combined with NLP techniques. Traditional approaches such as TF-IDF were compared against GNN-based text classification models. Experimental results showed that GNN-based approaches achieved approximately 80% prediction accuracy, outperforming traditional text classification techniques.

In [22], the authors proposed GPT2SP, a Transformer-based Agile Story Point estimation approach using GPT-2 pretrained language models. The study evaluated 23,313 issues

collected from 16 open-source projects and compared the proposed approach with 10 baseline estimation models in both within-project and cross-project scenarios. GPT2SP achieved a median MAE of 1.16 and demonstrated 34%–57% higher accuracy for within-project estimation and 39%–49% higher accuracy for cross-project estimation.

## VI. Research Gaps and Challenges

The Systematic Literature Review reveals considerable diversity in Story Point–based effort estimation approaches within modern Agile development environments. Despite notable progress, several critical research gaps and challenges remain unresolved.

One of the most significant challenges is the lack of large-scale, publicly available datasets based on Story Points. Most organizations consider Story Point data confidential, and existing datasets are often limited in size, noisy, and project-specific, which restricts reproducibility and generalization of Machine Learning models.

Another major challenge is model explainability. Although supervised Machine Learning models—particularly ensemble methods—demonstrate strong predictive accuracy, many operate as black-box models and fail to provide interpretable explanations for their predictions. This reduces practitioner trust and limits industrial adoption.

Cross-project estimation also remains an open research challenge. Models trained on one project or organization frequently perform poorly when applied to other projects because of domain variability, inconsistent Story Point assignment practices, and differences in team velocity.

Furthermore, the review identified inconsistencies in evaluation practices. Many studies continue to rely heavily on MMRE despite its known sensitivity to outliers and estimation bias. Limited integration of ML models with real-time Agile project management tools such as Jira and GitHub also reduces their practical applicability.

Finally, limited attention has been given to hybrid estimation frameworks that combine Expert Judgment with Machine Learning techniques and human-in-the-loop estimation approaches.

## VII. Threats to Validity

This study identifies several threats to validity associated with Story Point–based software effort estimation using Machine Learning techniques.

1) Conventional software estimation approaches such as KLOC, Function Point Analysis (FPA), Use Case Points (UCP), and Test Case Points rely on estimation techniques including Expert Judgment, Wideband Delphi, Analogy, and COCOMO models. Although large public datasets such as NASA93, PROMISE, IFPUG, and ISBSG are available, these approaches often fail to provide consistently accurate estimations.
2) Agile methodologies have become increasingly popular in modern software development because they support rapid development and deployment. Story Points derived from User Stories are commonly used as sizing metrics in Agile planning. However, techniques such as Planning Poker and Expert Judgment remain subjective and may not provide consistent estimation accuracy.
3) One of the primary limitations of Story Point estimation is the lack of publicly available datasets. Existing datasets are typically small, noisy, and project-specific, which may introduce overfitting, insufficient training data, and outlier sensitivity.
4) The selection of suitable Machine Learning models also represents a critical challenge. While regression and ensemble models generally provide improved predictive performance, researchers and industries are increasingly exploring hybrid estimation approaches that combine Expert Judgment with Machine Learning techniques.

## VIII. Conclusion and Future Scope

Software effort estimation is a critical activity for Project Managers, Developers, Testers, and Business Analysts because it supports the prediction of software effort, schedule, budget, and required resources for successful project delivery. Inaccurate estimation frequently results in project failure, schedule overruns, and budget imbalance.

Traditional software estimation models use sizing metrics such as KLOC, Function Point Analysis (FPA), Use Case Points (UCP), and Test Case Points. However, these approaches often fail to provide accurate estimation results in Agile environments. Consequently, modern Agile methodologies increasingly rely on Story Points derived from User Stories as sizing metrics.

This Systematic Literature Review demonstrates that supervised Machine Learning techniques are widely applied for Story Point–based software effort estimation. The reviewed studies investigated various models including Linear Regression, Support Vector Regression (SVR), Decision Tree Regression, Random Forest, Gradient Boosting, Deep Learning, and Transformer-based approaches.

The studies evaluated several performance metrics including MAE, MMRE, MdMRE, RMSE, and PRED(n). Experimental evidence suggests that ensemble methods such as Random Forest and Gradient Boosting generally provide superior estimation accuracy compared with conventional approaches.

Future research should focus on developing hybrid estimation frameworks that combine Expert Judgment with Machine Learning techniques. In addition, integration with Agile project management platforms such as Jira and GitHub can support continuous model retraining and real-time estimation feedback. Future work should also investigate Explainable AI (XAI), Transfer Learning, Federated Learning, and human-in-the-loop estimation models to improve estimation robustness, interpretability, and industrial applicability.

## References

[1] B. W. Boehm, *Software Engineering Economics*. Springer, 2002.
[2] P. Musilek, W. Pedrycz, N. Sun, and G. Succi, "On the sensitivity of COCOMO II software cost estimation model," in *Proc. IEEE Symposium on Software Metrics*, 2002.

[3] A. Z. Abualkishik and L. Lavazza, "IFPUG function points to COSMIC function points convertibility," *Information and Software Technology*, vol. 97, pp. 179–191, 2018.
[4] P. Faria and E. Miranda, "Expert judgment in software estimation during the bid phase," in *Proc. IWSM-MENSURA*, 2012.
[5] J. Grenning, "Planning poker or how to avoid analysis paralysis while release planning," Renaissance Software Consulting, 2002.
[6] S. M. Satapathy and S. K. Rath, "Empirical assessment of machine learning models for agile effort estimation using story points," Springer, 2017.
[7] V. Van Hai et al., "Toward improving the efficiency of software development effort estimation," *IEEE Access*, vol. 10, pp. 83249–83264, 2022.
[8] M. Rahman, T. Gonc¸alves, and H. Sarwar, "Review of existing datasets used for software effort estimation," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 7, 2023.
[9] M. A. Ramessur and S. D. Nagowah, "A predictive model to estimate effort in a sprint using machine learning techniques," *International Journal of Information Technology*, 2021.
[10] M. Ferna´ndez Diego et al., "An update on effort estimation in agile software development: A systematic literature review," IEEE, 2020.
[11] H. Phan et al., "Heterogeneous graph neural networks for software effort estimation," in *ACM/IEEE ESEM*, 2022.
[12] M. Fu and C. Tantithamthavorn, "GPT2SP: A transformer-based agile story point estimation approach," IEEE, 2023.
[13] S. M. Satapathy and S. K. Rath, "Empirical assessment of machine learning models for agile software development effort estimation using story points," Springer-Verlag London Ltd., 2017.
[14] Z. Z. K., S. K. Tipu, and S. K. Zia, "An effort estimation model for agile software development," *Advances in Computer Science Applications*, vol. 2, no. 1, pp. 314–324, 2012.
[15] M. A. Ramessur and S. D. Nagowah, "A predictive model to estimate effort in a sprint using machine learning techniques," *International Journal of Information Technology*, 2021.
[16] Q. Bushra and A. Kadam, "An improved technique for software cost estimations in agile software development using soft computing techniques," *IT in Industry*, vol. 9, no. 2, 2021.
[17] A. G. Duszkiewicz et al., "On identifying similar user stories to support agile estimation based on historical data," in *Agile-ISE 2022 International Workshop on Agile Methods for Information Systems Engineering*, Leuven, Belgium, June 2022.
[18] W. Rosa and S. Jardine, "Data-driven agile software cost estimation models for DHS and DoD," *Journal of Systems and Software*.
[19] E. Rodr´ıguez Sa´nchez, E. F. Va´zquez Santacruz, and H. Cervantes Maceda, "Effort and cost estimation using decision tree techniques and story points in agile software development," *Mathematics*, vol. 11, p. 1477, 2023.
[20] H. Phan et al., "Heterogeneous graph neural networks for software effort estimation," in *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Helsinki, Finland, Sept. 2022.
[21] H. Phan et al., "Story point level classification by text level graph neural network," in *2022 IEEE/ACM International Workshop on Natural Language-Based Software Engineering (NLBSE)*, 2022.
[22] M. Fu and C. Tantithamthavorn, "GPT2SP: A transformer-based agile story point estimation approach."