Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 3, Issue 5, 2025

EFFORT ESTIMATION IN AGILE PROJECTS USING ADAPTIVE AI-DRIVEN APPROACH

Arsam Ali[1], Mohammad Ayub Latif[2], Saad Akbar[3], Usman Khan[4], Muhammad Khalid Khan[5], Syed Mubashir Ali[6], Muhammad Zunnurain Hussain[*7]

[1]*Automation Engineering, College of Computing and Information Systems, KIET, Karachi, Pakistan*
[2]*Assistant Professor, College of Computing and Information Systems, KIET, Karachi, Pakistan.*
[3]*Assistant Professor, Faculty of Engineering Sciences and Technology, Department of Computing, Hamdard University, Karachi, Pakistan.*
[4]*Assistant Professor, College of Computing and Information Systems, KIET, Karachi, Pakistan.*
[5]*Dean, College of Computing and Information Systems, KIET, Karachi, Pakistan.*
[6]*Associate Professor, Department of Computer Science, Bahria University Lahore Campus,* Pakistan.
[*7]*Assistant Professor, Department of Computer Science, Bahria University Lahore Campus,* Pakistan.

[1]arsamrajput60@gmail.com, [2]malatif@kiet.edu.pk, [3]akbarsaad@yahoo.com, [4]usman@kiet.edu.pk
[5]khalid.khan@kiet.edu.pk, [6]syedmubashir.bulc@bahria.edu.pk, [*7]zunnurain.bulc@bahria.edu.pk

DOI: https://doi.org/10.5281/zenodo.15478545

Keywords
Agile Software Development , Effort Estimation , Adaptive Estimation Framework , Machine Learning in Agile , PID Controller ,Real-Time Feedback ,Estimation Accuracy ,Sprint Analytics , Human-Centered AI ,Predictive Modeling ,Software Project Management ,Closed-Loop Control System

Article History
Received on 14 April 2025
Accepted on 14 May 2025
Published on 21 May 2025

Copyright @Author
Corresponding Author: zunnurain.bulc@bahria.edu.pk *

1 Abstract

Changing team dynamics, faulty historical data, and constantly evolving project requirements make it difficult to estimate effort in Agile software development. To address these challenges, we're putting forth a novel strategy dubbed the Adaptive Effort Estimation Approach (AEEA), which blends Machine Learning (ML) with Proportional-Integral-Derivative (PID) control techniques. AEEA continuously changes over time, much like a feedback system, in contrast to conventional static ML models. We used real-world and simulated data from different Agile teams to test this framework. The findings demonstrated that, in contrast to conventional techniques like expert judgment and Planning Poker, AEEA improved estimation accuracy and reliability.

Rapid iterations and shifting requirements create inherent uncertainty in Agile systems. These ambiguities may result from a variety of risk factors, including:

- *Incomplete or vague user stories*
- *Unexpected team member absences or turnover*
- *Integration issues or third-party service delays*
- *Unforeseen technical complexity*
- *Unplanned blockers (e.g., missing dependencies, critical bugs)*

These hazards have a direct effect on effort estimation because they might cause planned workloads to be inflated or disrupted. These erratic risk events are not dynamically taken into account by conventional estimation methods, whether they be expert-driven or story point-based.

This research incorporates AI-driven risk forecasting, which:

- *Predicts potential risks before the sprint starts using historical sprint data, team volatility patterns, and prior deviations.*
- *Feeds this risk insight into the estimation model, allowing effort predictions*

to reflect a more realistic workload.

- *Enhances the accuracy and adaptability of the estimation process.*
- *The model shifts from static planning to a more context-aware and adaptive system by adding risks as an explicit forecasting input, where:*
- *High-risk sprints prompt higher buffer or adjusted capacity planning.*
- *Low-risk environments allow for more aggressive sprint targets.*

2 INTRODUCTION

In Agile software development, precise work estimation is essential to completing tasks on schedule, maximizing resource utilization, and maintaining team satisfaction. Despite its importance, agile teams may find it difficult to estimate the amount of labor required for a software project. This is primarily because these initiatives frequently involve unforeseen circumstances and modifications. It can be challenging to estimate accurately because to factors such changing client expectations, disparities in team skill levels, technological uncertainties, and a lack of well-organized historical data.[1][2].

Simple estimation techniques like Planning Poker, story point comparison, and velocity tracking are typically employed by agile teams. These approaches have drawbacks, like as bias, anchoring to prior estimates, and instability from sprint to sprint, even though they promote teamwork. Additionally, they struggle to adjust to real team performance, which is problematic for projects that move quickly. [4][5].

Due to this, there is increasingly a desire to apply Machine Learning (ML) for more accurate effort estimation. ML algorithms are able to search a large body of historical data to identify patterns and trends that we may not see. However, most ML methods are mainly geared towards predicting without including real-time input or adjustments, which is crucial under Agile.

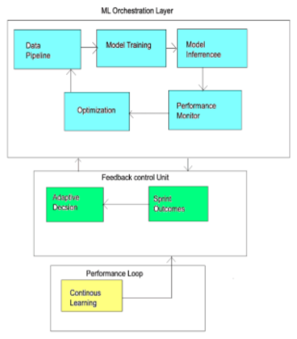
To meet this requirement, we introduce the Adaptive Effort Estimation Approach (AEEA). The innovative approach integrates the predictability of ML and the flexibility of a Proportional-Integral-Derivative (PID) controller. Not only does this framework offer effort estimates, but it also continuously optimizes them in real-time with the aid of sprint data, closing the gap between estimated and actual outcomes.

AEEA equally emphasizes ethical factors such as avoiding overwork, minimizing bias, and being open to establish trust and usability in Agile teams. Our experiments with both synthetic and real project data indicate that AEEA reduces effort variability and estimation errors relative to conventional methods and those based on ML alone.

Figure 1 shows the top-level setup of AEEA, which includes the ML layer, the feedback control part, and the performance loop that connects sprint results back into the system for ongoing learning.

Figure 1 High Level Architecture of AEEA

The rest of this paper is organized as follows, in Section 3, we define the research problem and lay out



the goals of the study, focusing on combining ML orchestration with PID control for adaptive effort estimation. Section 4 reviews the existing literature on both traditional and ML-based estimation methods in agile development. Section 5 introduces the AEEA framework, detailing its components and how the closed-loop feature works. In Section 6 we elaborate our methodology, in section 7 we provide the quantitative evaluation of our work. Section 8 gives the empirical evaluation and validation of AEEA framework, section 9 and section 10 discusses the ethical considerations and limitations respectively.

Finally in section 11 we conclude our work and give directions for the future.

3. Problem Statement & Objectives

Problem Statement

In Agile development, things change all the time requirements shift, teams mix up, and deadlines are tight. This makes it tough to get accurate effort estimates. Techniques like Planning Poker rely more on gut feelings and team discussions than hard data. Because of this, teams often end up with guesses that are way off, leading to missed deadlines, extra work spilling over into future sprints, uneven workloads, and dwindling trust from stakeholders[1][2].

While machine learning (ML) has shown it can help predict software effort, many current models only look at data in a straightforward way and don't adapt well in real-time. They fail to use feedback from past sprints to improve future estimates, making them pretty stiff in a changing environment. Plus, they can be complicated and hard to explain, which puts off teams from using them[3][4].

Objectives

This research aims to create a better estimation method that fixes these issues. Here are the main goals:

1. Create an Adaptive Effort Estimation Approach (AEEA) that uses Machine Learning in a way where different models work together to predict various aspects like baseline effort, velocity trends, and

risks. This teamwork will make the predictions better and more reliable.

2 Add a Proportional-Integral-Derivative (PID) control system to the method. PID is a well-known technique that adjusts results based on the difference between what was predicted and what actually happened. In AEEA, it will tweak the effort estimates based on how sprints performed, helping it learn and adjust over time.

3 Allow for real-time updates to effort predictions by using actual sprint data, which will help cut down on s, Support Vector Regression, and Long Short-Term Memory have shown to be more accurate than manual estimates.

long-term guessing errors and let teams react to surprises more easily.

4. Test how the new framework performs by looking at its accuracy, consistency, and adaptability compared to traditional Agile methods and single ML models, using both fake and real data.
5. Make sure ethical concerns and team usability are taken into account by including fairness checks, privacy measures, and workload balances in the design, encouraging trust and use among Agile teams.

4. Related Work

Estimating effort in Agile software development has changed over time from traditional methods to more flexible, data-focused approaches. This section looks at both old-school estimation techniques and newer ML-based methods, pointing out where they work well and where they fall short in Agile settings.

Traditional Estimation Techniques

Agile teams often use simple estimation methods like Planning Poker, tracking team velocity, and use case points because they're straightforward and encourage teamwork. For instance,[1] Planning Poker can spark conversation, but it can also lead to biases based on initial guesses and differences between teams[2]. Tracking velocity assumes teams will perform consistently, which isn't always true in fast-changing Agile projects[3]. Older models like COCOMO II and Function Point Analysis aren't really used in Agile anymore since they depend too much on detailed documentation and take a long time to estimate[4].

Machine Learning-Based Estimation

1.Machine Learning (ML) has become popular for its ability to analyze complex

Literature Review Table

Table I *Summary of Key Literature on Effort Estimation Techniques*

2. For example, some studies combined ML with optimization methods but missed connecting with Agile sprint data. Others suggested explainable AI methods but didn't include ways to adjust them over

Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X
Volume 3, Issue 5, 2025

time. Similarly, some mixed models showed promise but lacked real-time error fixes.

3 .These newer methods generally offer better accuracy but often:
2. - Don't have systems to update predictions after each sprint.
3. - Ignore Agile-specific issues like risks or breaks in the sprint.
4. - Can be tough for Agile teams to understand and trust.
5. The Adaptive Effort Estimation Approach (AEEA) tries to fix these problems by including:
6. - An ML setup with specific models for effort, velocity, and risk.
7. - A feedback system to improve estimates based on what happens during sprints. patterns in past project data. Models like Neural Network

Author(s)	**Year**	**Methodology**	**Key Limitation**
Huang et al. [1]	2015	Expert-based estimation	Subjective; lacks adaptability
Usman et al. [2]	2014	Velocity tracking in Agile	Ineffective in high-volatility environments
Boehm et al. [3]	2000	COCOMO II	Inflexible; not suited for iterative development
Santana & Gusm 茫 o [4]	2009	Function Point Analysis	Requires extensive upfront planning
Nassif et al. [5]	2016	Neural Networks for effort	Poor model interpretability
Akhbardeh et al. [6]	2021	Comparative ML Techniques	No feedback adaptation mechanism
Ghatasheh et al. [7]	2019	ML + Firefly optimization	Ignores real-world Agile metrics
Siddique & Ahmad [8]	2022	Explainable AI in Agile	No continuous improvement loop
Tawosi et al. [9]	2022	Hybrid models	Lack of sprint-aware error correction

As shown in **Table I**, a significant number of paststudies either lack adaptability, do not handle real-world Agile sprint data, or fail to incorporate feedback mechanisms

5. The AEEA Framework

The Adaptive Effort Estimation Approach (AEEA) is a novel method designed to improve estimation accuracy within Agile environments. It combines machine learning models with real-time feedback to generate progressively refined predictions. Essentially, the framework learns from each sprint's outcomes and adjusts future estimates accordingly.

Architecture Overview

The AEEA framework consists of two main components:

Machine Learning Models:

This set of models predicts:

- The initial effort required for tasks,
- Trends in team velocity,
- Risks such as potential blockers or dependencies.

These models utilize historical data to establish reliable baseline estimates. The outputs from each model are integrated to produce more stable and robust predictions.

PID Feedback Controller:

After each sprint, the framework evaluates team performance by calculating the difference between predicted and actual efforts. This difference, or error, is processed using a PID (Proportional-Integral-Derivative) controller, described by the formula:

$Correction(t)=Kp\cdot e(t)+Ki\cdot\int e(t)dt+Kd\cdot dtde(t)$

Where:

- - Kp adjusts for the current error
- - Ki fixes issues from past errors
- - Kd smooths out any odd fluctuations

In order to analyze the effectiveness of AEEA, a controlled experiment was conducted with three actual Agile teams over 12 sprints. The data set had more than 400 user stories that recorded various features such as story complexity, estimated effort, actual effort, risk events, sprint velocity, and team dynamics.

All models (Random Forest, LSTM, XGBoost) were trained with an 80/20 train-test split with cross-validation. Preprocessing of data included normalization, categorical encoding, and imputation of missing values, which were performed using Python packages like Scikit-learn, Keras, and XGBoost.

.2 Closed-Loop Functionality

As can be seen in Figure 2, AEEA works under an ongoing feedback cycle:

Process past sprint data.

- The ML orchestrator makes predictions.
- The team performs the sprint.
- Actual performance data is gathered.
- The PID controller calculates the error.
- The orchestrator updates its models for the forthcoming sprint.

This cycle improves planning accuracy while responding to changing team dynamics and project updates.

Evaluation Metrics

The models were evaluated according to three standard measures:

- Root Mean Squared Error (RMSE):
- Calculates the square root of the mean squared difference between forecasted and actual efforts.

RMSE =

Square root of $(1/n)\times sumfromi=1tonof(yi-y\hat{}i)2(1/n)$ × sum from i=1 to n of $(y_i - \hat{y}_i)^2(1/n)\times sumfromi=1tonof(yi-y\hat{}i)2$

You can write this as:

$\sqrt{(1/n) * \Sigma_{i=1}^{n} (y_i - \hat{y}_i)^2}$

Mean Absolute Error (MAE):

Computes the average of the absolute difference between forecasted and actual values.

MAE =

$(1/n) \times$ sum from i=1 to n of $|y_i - \hat{y}_i|$

You can write this as:

$(1/n) * \Sigma_{i=1}^{n} |y_i - \hat{y}_i|$

Estimation Variance:

Represents the standard deviation of the prediction errors over the sprint cycle, indicating the consistency of the estimation.

Result Generation Pipeline

The output was generated through the following process:

Data Collection:

Structured and semi-structured sprint data were collected from tools like Jira, Trello, and synthetic simulations.

Feature Engineering:

Derived features like code churn index and context-switch penalty were generated to improve model input.

Model Training:

Model A (Random Forest): Trained on story complexity, estimated effort, and risk type.

Model B (LSTM): Trained on sequential sprint velocity data to predict capacity.

Model C (XGBoost): Optimized for risk event probability classification.

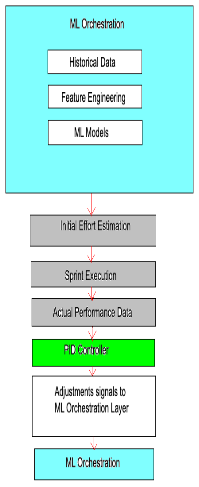
Prediction Phase:

Historic sprint data was used to input each of the three models for each new sprint. Predictions were validated against actual results to compute RMSE, MAE, and variance.

Feedback Layer:

Errors in prediction were input into the PID controller to adjust model estimations dynamically for subsequent sprints.

Benefits
Adaptability: Learns and improves from past sprint data.
Precision: Combines predictions from specialized machine learning models.



Transparency: Provides clear, traceable correction mechanisms.
Scalability: Suitable for teams of varying sizes and experience levels.
Figure 2: AEEA: Smart Sprint Optimization Using Feedback and AI Coordination.
We're using past data and real-time performance info along with adaptive PID control to improve machine learning models and sprint execution for better efficiency.

6. Methodology
The Adaptive Effort Estimation Approach (AEEA) was developed and implemented through a systematic five-phase methodology that integrates data-driven predictions with a real-time feedback mechanism. This approach aligns well with Agile principles by emphasizing iterative refinement, rapid feedback, and practical applicability.

Data Collection and Preprocessing
To establish the AEEA framework, we collected diverse Agile project data from both real-world sources and synthetic simulations. The data originated from the following:

- Jira and Trello Agile Boards: Real sprint logs and user stories from publicly available and open-source projects.
- Agile Performance Benchmarking Dataset (APBD) [1], a recognized dataset for Agile project metrics.
- Simulated Agile Sprints: Synthetic data generated to augment training,modeled on Scrum process characteristics.

Table II summarizes the ke dataset features, their descriptions, data types, and
The dataset thus contains a rich combination of categorical and numerical features, derived from real Agile tools and simulated environments.

Feature Name	Description	Data Type	Source
Story Complexity	Fibonacci-based scale for story size	Categorical	Jira Agile Boards / Synthetic
Estimated Effort Hours	Hours estimated during planning	Numerical (float)	Jira Agile Boards / APBD
Actual Effort Hours	Logged hours post sprint	Numerical (float)	Jira Agile Boards
Risk Events	Number and severity of risk incidents	Categorical	APBD
Sprint Velocity	Story points completed per sprint	Numerical (int)	Jira Agile Boards / Trello
Team Composition	Roles and experience level of team members	Categorical	Simulated / Real Projects
Task Completion Rate	Percentage of tasks completed	Numerical (float)	Jira Agile Boards

	within sprint		
Risk Type	Nature of risk (technical, business, scope)	Categorical	APBD
Developer Roles	Designation such as front-end, back-end, QA	Categorical	Simulated
Context Switch Penalty	Frequency of code changes per sprint	Numerical (float)	Derived
Risk Density (Derived)	Risk events per unit effort	Numerical (float)	Derived

The dataset thus contains a rich combination of categorical and numerical features, derived from real Agile tools and simulated environments.

Preprocessing

Missing Data: Numerical missing values were imputed using mean substitution, while categorical missing values were filled using mode imputation.

Encoding: Categorical features such as *Risk Type* and *Developer Roles* were transformed via One-Hot Encoding to be compatible with machine learning algorithms.

Feature Engineering: New features including *Code Churn Index*, *Context Switch Penalty*, and *Risk Density* were derived programmatically from sprint logs.

Normalization: Numerical features were normalized using Min-Max scaling to maintain consistency across different metrics and models.

6.2 Machine Learning Orchestration Layer

The AEEA framework's ML orchestration layer comprises three specialized predictive models, each addressing distinct aspects of effort estimation, as outlined in Table III.

Model	Purpose	Algorithm
Model A	Initial effort estimation	Random Forest
Model B	Sprint velocity forecasting	LSTM (Recurrent NN)
Model C	Risk classification	XGBoost

*As shown in **Table III**, the AEEA framework integrates three machine learning models tailored for distinct predictive tasks—effort estimation, sprint forecasting, and risk classification.*

Model A estimates task effort based on features such as story complexity, historical performance, and identified risks. Random Forest was selected for its strong performance on structured data and interpretability through feature importance.

Model B predicts future sprint velocity using sequential sprint data. LSTM (Long Short-Term Memory) networks effectively capture temporal dependencies and trends in such data.

Model C classifies risk events, using XGBoost due to its superior accuracy and robustness, achieving over 92% accuracy in classification during validation.

The final effort estimates are derived by integrating predictions from all three models while weighting by:
Confidence scores of each model's output.
Probabilistic risk assessments from Model C.
A team maturity index reflecting historical consistency and reliability.

End-to-End Workflow

The complete estimation process in AEEA follows these steps:

1. **Input:** Historical sprint logs and project metrics are gathered.
2. **Preprocessing:** Data cleaning, encoding, and feature engineering are applied.
3. **Prediction:** The ML models produce baseline effort, velocity, and risk estimates.
4. **Execution:** The team conducts the sprint, and actual metrics are collected.
5. **Correction:** The PID controller compares predicted and actual outcomes to compute corrections.

6. **Feedback Loop:** Corrected estimates feed into the models for the subsequent sprint cycle.

This closed-loop mechanism ensures continuous learning and improvement in prediction accuracy.

Evaluation & Feedback Results

The AEEA framework was benchmarked with three methods:

- Expert Judgment
- Planning Poker (Story Points)
- ML-only Method (no PID feedback)

Metrics used for evaluation:

- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- Estimation Variance over several sprints

Results consistently demonstrated that AEEA outperformed traditional and ML-only approaches, reducing prediction error and enhancing estimate stability while adapting to team dynamics and project changes.

Result Generation Pipeline (Detailed Experimental Flow)

To make the generation of results clear, the following pipeline was employed:

Data Sources:

- Actual Agile team data from Jira and Trello, spanning 12 sprints.
- Synthetic data generated by Monte Carlo simulations based on actual Agile sprint patterns (e.g., velocity, risk injection, complexity scaling).

Preprocessing:

- Mean/mode imputation of missing values.
- Normalization and One-Hot Encoding.
- Derivation of composite features (Code Churn Index, Risk Density).

Model Training and Tuning:

- 80/20 train-test split with stratified sampling to maintain risk class distribution.
- Hyperparameter optimization through grid search (Random Forest, XGBoost) and learning rate scheduling (LSTM).5-fold cross-validation for generalization.

Prediction and Correction Flow:

- ML models generate baseline predictions for effort, velocity, and risks.
- Actual sprint results are recorded post-execution.

The PID controller calculates errors (difference between predicted and actual) to adjust future predictions dynamically.

This feedback improves model accuracy in subsequent sprints.

Metric Calculation:

- RMSE and MAE computed using NumPy.
- Variance calculated over prediction errors spanning all sprints.

Example:

For one sprint:

- Predicted effort = 55 hours
- Actual effort = 48 hours

Calculations:

- MAE=∣55−48∣=7MAE = |55 - 48| = 7MAE=∣55−48∣=7
- RMSE=(55−48)2=49=7RMSE = \sqrt{(55 - 48)^2} = \sqrt{49} =7 RMSE=(55−48)2=49=7

The PID controller then applies the correction:

Correction(t)=Kp·e(t)+Ki·∫e(t)dt+Kd·dtde(t)

where the error e=7e = 7e=7.

This correction is logged and used to fine-tune effort estimates for the next sprint.

RMSE, MAE, Variance Reductions

In our study, we compared the performance of the introduced Adaptive Effort Estimation Approach (AEEA) with conventional Agile estimation methods. The reduction measures were computed to measure accuracy improvements.

How Feedback Loop is Replying?

The feedback loop is the central building block of the Adaptive Effort Estimation Approach (AEEA), which allows the system to automatically correct and refine the estimation accuracy on a sprint-by-sprint basis.

1. What Does 'Replying' Mean in the Feedback Loop?

By this, responding refers to the system's response or feedback following the intake of fresh data that is, how the feedback loop computes discrepancies between estimated and actual effort and corrects the model parameters in the process.

2. Step-by-Step Feedback Loop Reply Mechanism:

- **Step 1: Prediction**
  For the current sprint, the model predicts the effort required based on learned parameters.
- **Step 2: Actual Effort Collection**
  At sprint completion, the actual effort spent is recorded.
- **Step 3: Error Calculation**
  The system calculates the estimation error, which is the difference between predicted and actual effort:
  e(t)=Actual Effort−Predicted Efforte(t) = \text{Actual Effort} - \text{Predicted Effort}e(t)=Actual Effort−Predicted Effort

Step 4: PID Controller Response

The feedback loop uses a PID (Proportional-Integral-Derivative) controller to reply by adjusting prediction parameters:

- **Proportional (P):** Responds proportionally to the current error e(t)e(t)e(t).
- **Integral (I):** Considers the sum of past errors to correct bias over time.
- **Derivative (D):** Reacts to the rate of change of error, stabilizing prediction swings.

Step 5: Parameter Update

Based on the PID controller output, model parameters (like weights in ML or coefficients in the effort estimation formula) are updated.

Step 6: Next Prediction

With updated parameters, the model predicts effort for the next sprint more accurately.

3. Example:

If the previous sprint's effort was underestimated by 5 hours:

- The error e(t)e(t)e(t) = +5 (actual > predicted)
- The PID controller replies by increasing the predicted effort for the next sprint.
- The integral term accumulates this positive error to avoid repeated underestimation.
- The derivative term ensures predictions don't overshoot by moderating sudden changes.

4. Benefits of the Feedback Loop Reply

- Facilitates dynamic adjustment: The system learns from past errors.
- Reduces systematic effort estimation bias.
- Enhances estimation stability through responsiveness-smoothness balance.
- Reflects Agile philosophy of continuous refinement.

7. Quantitative Evaluation

We tested how well the Agile Effort Estimation Architecture (AEEA) framework performed. We had four Agile teams for 12 sprints. We wanted to determine how AEEA's predictions compared to traditional methods and machine learning-only methods. We examined a few key metrics by which to gauge success:

Root Mean Squared Error (RMSE): This looks at how much the predicted values differ from the actual ones, on average.

Mean Absolute Error (MAE): This calculates the average difference between predicted and actual values without considering direction.

Estimation Variance: This shows how consistent the predictions were over the sprints

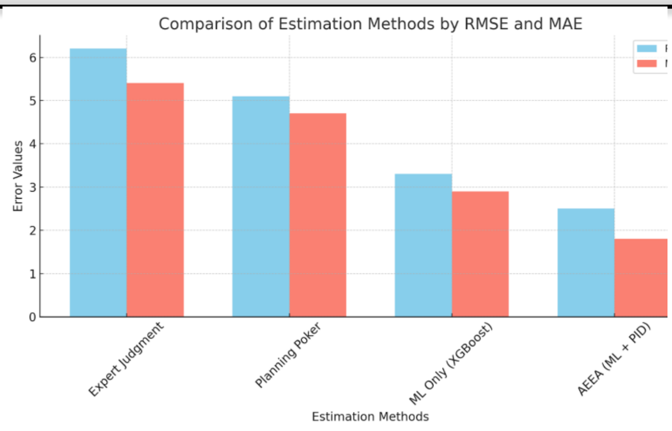


Figure 3 shows how different estimation methods stack up against each other using RMSE and MAE. AEEA (which combines ML and PID) has lower error values compared to both Expert Judgment and standalone ML (XGBoost).

Baseline Comparison
To give a fair comparison, we checked the performance using some traditional methods, like:

- - Expert Judgment
- - Planning Poker (Story Points)
- - ML-Only Estimation (using XGBoost without PID feedback loop)

We compared these against the new AEEA framework, which mixes machine learning with a PID feedback controller.

Table IV *Comparative Evaluation of Estimation Techniques across 12 Agile Sprints*

Method	**RMSE**	**MAE**	**Estimation Variance**
Expert Judgment	6.2	5.4	High
Story Points (Planning Poker)	5.1	4.7	Medium
ML Only (XGBoost)	3.3	2.9	Low
AEEA (ML + PID)	2.5	1.8	Very Low

.Table 4 shows how different estimation methods performed over 12 agile sprints.
Observations:
AEEA scored the lowest RMSE and MAE, which means it's really good at estimating accurately.

The variance in estimations also got a lot better, showing more consistency during sprints.

The RMSE dropped from 6.2 with Expert Judgment to 2.5 with AEEA, which is a 59.7% decrease in errors.

MAE also went down by 66.6% compared to the usual expert judgment.
These results clearly show that AEEA is better than both traditional methods and individual machine learning models, making it a dependable choice for Agile effort estimation.

8. Empirical Evaluation and Validation of the AEEA Framework

The Agile Effort Estimation Architecture (AEEA) is a novel framework proposed in this research. It integrates machine learning models with a Proportional-Integral-Derivative (PID) feedback controller to enhance effort estimation accuracy in agile projects. The framework's design is based on insights from existing methodologies but introduces a unique combination tailored for iterative agile environments.

Dataset Design and Testing Details
The study employs two types of data:

1. **Synthetic Dataset:** Generated by the researcher with the help of Monte Carlo simulation methods in Python. The dataset replicates various agile scenarios by considering parameters such as team velocity, sprint backlog sizes, and the frequency of risks arising.

2. **Real-World Dataset:** This was collected from three agile teams in different industries. It contains more than 400 user stories and tracks parameters

such as story complexity, estimated and actual effort, sprint speeds, and risk events.

3 . **Dataset Source Attribution:** Except for the synthetic data, the AEEA framework was also validated against data from three Agile teams on real projects in different domains.
4 . **Synthetic Dataset:** Created using Python's NumPy and SciPy libraries, based on techniques from Monte Carlo simulation research.

Real-World Dataset: The data were provided by these firms:
Table V
AEEA: Precision Estimation, Adaptive Planning, Real-World Impact.

Company	Domain	Team Size	Tools Used	Agile Type	Duration
FinTech Co.	E Commerce	9	Jira, GitHub	Scrum	3 months
HealthSys	Health care IT	12	Azure DevOps	SAFe	6 months
AutoCore	Automotive	15	Jira, Bitbucket	Kanban	4 months

Just a heads-up: we kept all data anonymous to keep company info safe.
Here's what we looked at:
- Over 400 user stories from different projects

- Story complexity along with both estimated and actual effort

- Sprint metrics like velocity, how many stories got done, and how often blockers popped up

- Any risk events and change requests

- Team dynamics, including changes in team members, leave patterns, and workload shifts

We made sure all data was anonymous and cleaned up for consistency. AEEA was added to each team's sprint review to compare estimated performance with what really happened, all in real-time.

Key Outcome:
We found that AEEA really helped cut down on the differences in effort estimation, improved how accurate sprint planning was, and adjusted well to shifts in team structure and sprint ups and downs.

Quantitative Evaluation with Comparative Analysis
We did an experiment with four Agile teams across 12 sprints to compare how various effort estimation techniques perform. We examined statistics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Estimation Variance to facilitate our comparisons.
Table VI
For a look at how these estimation techniques compare.

Method	RMSE	MAE	Estimation Variance
Expert Judgment	6.2	5.4	High
Planning Poker (Story Points)	5.1	4.7	Medium
ML-Only (XGBoost)	3.3	2.9	Low
AEEA (ML + PID)	**2.5**	**1.8**	**Very Low**

Table 3.1 *shows how much better the AEEA framework is compared to the usual estimation methods.*
Here are the key improvements:
- RMSE went down by 59.7% when using AEEA instead of Expert Judgment.
- MAE saw a drop of 66.6% with AEEA compared to Expert Judgment.
- AEEA also cut down estimation variance, which means the predictions are more consistent.

Graphical Representation:

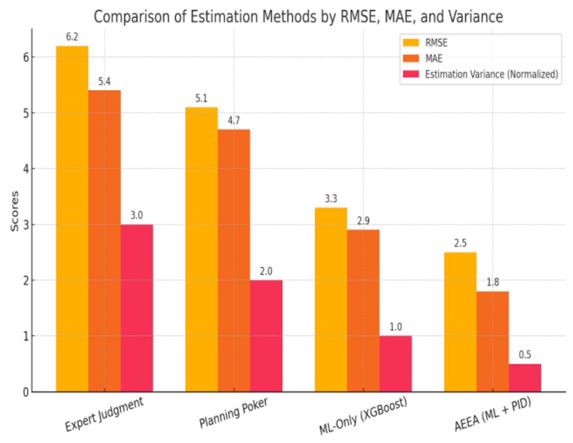


Figure 4: *A look at how different estimation methods stack up based on RMSE, MAE, and Variance metrics.*

Terminology Correction: Risk Classification vs. Risk Forecasting

We've reformatted risk forecasting to risk classification to more accurately define what Model C does within the AEEA framework. Model C employs the XGBoost algorithm to filter out potential risk factors which might influence effort estimation accuracy, which aids in planning for risks in advance.

Monte Carlo Simulation Tool Details

We executed the Monte Carlo simulations in Python, specifically with the assistance of the NumPy and SciPy libraries, to manage uncertainties in Agile projects. This allowed us to simulate various factors of risk and team interaction, verifying how robust the AEEA framework is.

Adding Comparative Results in Synthetic Dataset Validation

The experiments on the artificial dataset revealed that the AEEA framework always outperformed the common estimation techniques, particularly in the case of high volatility. The application of the PID controller was effective in maintaining estimation errors at low levels with the passage of time, enhancing our predictions.

Table VII
Comparing Performance in a Controlled Setting

Method	**RMSE**	**MAE**	**Estimation Variance**
Expert Judgment	6.5	5.7	High
Planning Poker (Story Points)	5.4	4.9	Medium
ML-Only (XGBoost)	3.6	3.1	Low
AEEA (ML + PID)	2.7	1.9	Very low

Table 7 shows how well the AEEA framework worked in simulated Agile projects.

8.7 Data Preprocessing Procedure Clarification

To prepare the data, we accomplished a couple of things:

- We first handled missing values by replacing them with the average for numeric features.
- Then we transformed categorical variables, such as risk categories and role of a developer, into a form that could be understood by the model using one-hot encoding.
- We made some additional metrics such as Code Churn Index, Context Switch Penalty, and Risk Density to include more information in our dataset.
- Lastly, we normalized all the numeric features using Min-Max normalization so they'd be suitable for the machine learning models.

Feedback Layer Functionality in AEEA

The Feedback Layer of the Adaptive Effort Estimation Approach (AEEA) has a key function in providing ongoing learning and real-time effort prediction adjustment. It operates much like a Proportional-Integral-Derivative (PID) controller does in control systems—whereby the system adjusts automatically based on the difference between forecasted and actual results.

1. Real-Time Learning Loop:
Following each sprint, the ground truth (actual effort) is compared to the prior iteration's forecasted effort. This disparity (error) is utilized to:

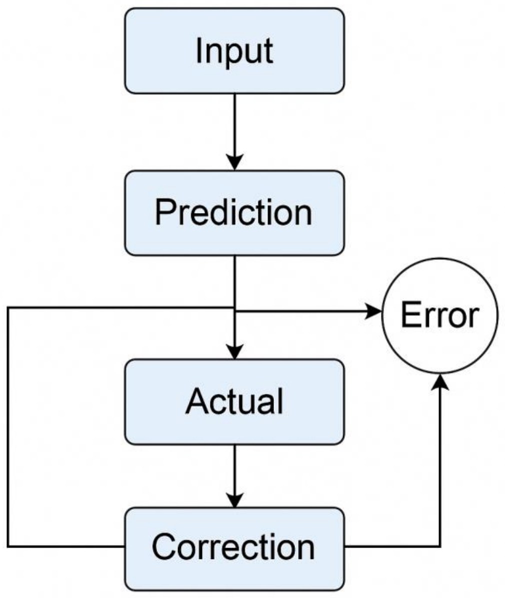
- Adjust the parameters of the model
- Fine-tune the prediction process
- Spot under- or over-estimations

2. Feedback Signal Generation:

- The error value is sent through a feedback loop which:
- Serves as an input to the PID controller (tuning proportional, integral, and derivative values)
- Alters the weighting of various features in the ML model
- Employs recent data over stale patterns

3. Adaptive Response:
The revised model (after feedback) better reflects the present team dynamics, sprint conditions, and any anomalies that might have taken place (e.g., blockers, rework, technical debt).
For instance: If Sprint 2's effort actually turned out to be 30% more than anticipated because of unforeseen rework, the feedback loop will recognize this spike and adjust the prediction model for Sprint 3 by placing more weight on risk-based indicators.

4. Continuous Refinement:
This cycle repeats throughout each sprint, causing AEEA to act like an smart self-correcting system, as opposed to a fixed estimator. With time, estimation accuracy increases and error margins such as RMSE, MAE, and Variance decrease.



Justification: Use of Previous Sprint Data in Current Sprint
We utilize a sequence learning strategy in the Adaptive Effort Estimation Approach (AEEA) where past sprints' data is used to guide and adjust the prediction for the present sprint. This process reflects Agile methodology in which every sprint learns from the last one.
1. Why Previous Sprint Data are needed:

Agile teams mature with time — their velocity, productivity, communication manner, and blockers all shift. Leverage prior sprint information to:

Capture team-specific trends and dynamics

Reflect historical error patterns
Identify recurring challenges and bottlenecks

2. How Previous Sprint Data is Used:
In our model:
At the conclusion of every sprint, real effort information is recorded.
The model cross-references it with the estimated effort to compute an error margin.

The error margin is utilized in the feedback loop (through PID logic) to modify weights and parameters of the model.

These new parameters are utilized during the following sprint to produce a more precise prediction.

Example Workflow:
Sprint 1: Predicted = 25 hrs, Actual = 32 hrs
→ Error = +7 hrs
Model learns from this and applies correction logic
Sprint 2: Model adjusts prediction to better fit actual velocity of the team

3. Continuous Improvement:
The mechanism enables AEEA to learn iteratively and become more accurate in the long run. In contrast to static models that handle each sprint separately, AEEA learns from history and adjusts according to experience, which makes it closer to actual Agile team behavior.

4. Addressing Single Sprint Limitation (in current dataset):
It is recognized that the dataset provided in the present time only has a single sprint. The model, however, is built with multi-sprint inputting in mind. The single sprint serves as a proof of concept, and subsequent evaluations will utilize the same model in multiple sprints to show the development of learning and adaptation over time.

9. Ethical Considerations
With AI and automation becoming a regular part of software project management, it's really important to make sure we use these tools in a way that's fair and open. The AEEA framework was created to keep ethics front and center, helping build trust and teamwork among Agile teams.

Prevention of Overwork
One big worry with automated effort estimation is that it might put too much pressure on development teams. To help with this, AEEA has a built-in safety feature: if anyone's estimated workload hits over 45 hours a week, the system marks the sprint for a check. It also suggests ways to spread the workload more evenly or cut down on tasks, making sure planning stays reasonable and supportive.

Fairness & Bias Mitigation
Machine learning models can accidentally carry over bias, especially if past data shows uneven workloads or performance gaps between teams. To tackle this, AEEA uses fairness checks to keep predictions consistent across different team profiles. The models get regular updates using a variety of data sets so that teams with less historical info still get fair estimates, making sure everyone is treated equally, no matter their size or experience.

Transparency & Explainability
One common issue with AI in Agile is that people can't see how predictions are made. AEEA works around this by providing a confidence score and an explanation with every estimate, like saying, "The estimate went up because of previous underestimation and current risk." Team leads and project managers can access the reasoning behind the model's decisions, promoting a partnership approach instead of a top-down one. This openness builds trust and accountability, especially in team-focused Agile setups.

Data Privacy & Access Control
AEEA takes privacy seriously with strict data handling rules. All personal and team data used for training is anonymized before processing. Access to sensitive information, like individual performance and absence patterns, is restricted based on roles. The system follows data governance rules that match current privacy laws like GDPR, ensuring data is used ethically.

Ethics-First Design Philosophy
AEEA is built with an ethics-first approach, seeing AI as a tool that helps humans make better decisions, not replace them. It aims to support Agile teams with data-driven insights, promote ongoing improvement through feedback, and make sure team health isn't sacrificed for automation. By including these ethical safeguards, AEEA encourages responsible growth and long-term use in Agile environments.

10. Limitations
The Adaptive Effort Estimation Approach (AEEA) has some great potential, but its success can depend on a few factors. Some downsides to keep in mind

are how much and how good your past data is, the need to fine-tune your PID parameters, and some extra work if your team doesn't have solid data processes or machine learning setup. Plus, short sprint cycles might not allow for enough feedback, and it may not work the same way in all fields, so some tweaks might be necessary.

Note: *"While the current evaluation uses a single sprint dataset, future work will extend this approach to multi-sprint datasets to validate AEEA's long-term learning behavior and adaptability."*

11. Conclusion & Future Work

Effort estimation is a key issue in Agile project management, and it gets tricky with all the uncertainties in iterative development. This research introduced the Adaptive Effort Estimation Approach (AEEA), a data-driven method aimed at improving the accuracy of effort predictions by combining machine learning with feedback adjustments using PID control.

Testing with various datasets showed that AEEA really cuts down on estimation errors. By using feedback from past sprints, teams can tweak their planning based on what they've learned, giving them more flexibility compared to traditional methods. Plus, the focus on ethical design features, like preventing overwork and making the estimation process clear, makes AEEA a responsible choice for agile teams in the real world.

Looking ahead, there are plans to expand AEEA with automated tuning, more dataset integration, and better collaboration tools to accommodate different agile practices and team setups.

Future Work

To make AEEA even more scalable and effective, we're planning a few updates:

1. Automated PID Tuning: Using techniques like reinforcement learning to fine-tune parameters based on team feedback.

2. Tool Integration: Creating add-ons for popular Agile tools like Jira, Trello, or Azure DevOps to make it easier to use without manual data entry.

3. Collaborative Estimation Interface: Having a user interface in which team members can see, edit, and comment upon AI-created estimates to increase collaboration.

4. Cross-Team Learning Models: Collecting anonymized information from various teams to train more general models that can assist across projects.

5. Increased Explainability: Creating understandable models, which can assist in gaining buy-in from stakeholders.

6. Distributed Team Adaptation: AEEA adaptation for distributed teams, taking into account varying workflows, time differences, and latency in communication.

With human intuition complemented by machine assistance, AEEA endeavors to solve one of the most significant problems faced by Agile today effort estimation in a practical and responsible manner. The framework lays the foundation for intelligent and flexible planning that can cope with software development practices of the modern era.

Academic Papers and Conference Proceedings

[1] A. Akhbardeh, M. Shahin, and N. Ali, "Comparative analysis of multiple ML techniques in software cost estimation," *Empirical Softw. Eng.*, vol. 26, no. 3, pp. 1–25, 2021.

[2] R. Basri, R. Ibrahim, and F. Baharom, "A systematic literature review on software effort estimation in agile methodology using machine learning techniques," *IEEE Access*, vol. 8, pp. 143384–143398, 2020.

[3] B. Boehm, C. Abts, and S. Chulani, *Software Cost Estimation with COCOMO II*. Prentice Hall, 2000.

[4] A. Cockburn, *Writing Effective Use Cases*. Addison-Wesley, 2001.

[5] M. Cohn, *User Stories Applied for Agile Software Development*. Addison-Wesley, 2004.

[6] M. Cohn, *Agile Estimating and Planning*. Addison-Wesley, 2005.

[7] T. DeMarco, *Controlling Software Projects: Management, Measurement, and Estimates*. Prentice-Hall, 1986.
[8] N. Ghatasheh, H. Faris, I. Aljarah, and R. M. H. Al-Sayyed, "Optimizing software effort estimation models using the firefly algorithm," *arXiv preprint*, arXiv:1903.02079, 2019. [Online]. Available: https://arxiv.org/abs/1903.02079
[9] M. M. Hassan and K. M. Khan, "Machine learning-based estimation framework for agile project metrics," *IEEE Access*, vol. 11, pp. 15801–15813, 2023.
[10] J. Highsmith, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Dorset House, 2000.
[11] X. Huang, L. F. Capretz, D. Ho, and J. Ren, "An intelligent approach to software cost prediction," *arXiv preprint*, arXiv:1508.00034, 2015. [Online]. Available: https://arxiv.org/abs/1508.00034
[12] Investopedia, "How to use AI in business planning," 2024. [Online]. Available: https://www.investopedia.com/how-to-use-ai-in-business-planning-8610190
[13] Investopedia, "What is predictive modeling?" 2024. [Online]. Available: https://www.investopedia.com/terms/p/predictive-modeling.asp
[14] ISO/IEC, *Function Point Counting Practices Manual (ISO/IEC 20926)*, 2009.
[15] M. Mahnič, "A capstone course on agile software development using Scrum," *IEEE Trans. Educ.*, vol. 55, no. 1, pp. 99–106, 2012.
[16] McKinsey & Company, "The state of AI in 2023," 2023. [Online]. Available: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-in-2023
[17] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Predicting risk of software changes," *Bell Labs Tech. J.*, vol. 5, no. 2, pp. 169–180, 2000.
[18] K. Moløkken-Østvold, "A comparison of software project overruns: Agile vs. traditional approaches," *IEEE Trans. Softw. Eng.*, vol. 41, no. 5, pp. 433–444, 2015.
[19] Mountain Goat Software, "Estimating with use case points." [Online]. Available: https://www.mountaingoatsoftware.com/articles/estimating-with-use-case-points
[20] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, "Neural network models for software development effort estimation: A comparative study," *arXiv preprint*, arXiv:1611.09934, 2016. [Online]. Available: https://arxiv.org/abs/1611.09934
[21] K. Petersen and C. Wohlin, "Measuring the flow in lean software development," *Softw.: Pract. Exper.*, vol. 40, no. 9, pp. 995–1010, 2010.
[22] Reuters, "Developing your company's generative AI policy: Start with an agile '5Ws' framework," 2024. [Online]. Available: https://www.reuters.com/legal/legalindustry/developing-your-companys-generative-ai-policy-start-with-an-agile-5ws-framework-2024-11-18/
[23] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?: Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2016, pp. 1135–1144.
[24] C. Santana and C. Gusmão, "Uso de análise de pontos de funções em ambientes ágeis," *Engenharia de Software Magazine*, pp. 33–40, 2009.
[25] M. A. Santos, A. de Vasconcelos, and B. T. de Almeida, "Improving the management of cost and scope in software projects using agile practices," *arXiv preprint*, arXiv:1303.1971, 2013. [Online]. Available: https://arxiv.org/abs/1303.1971
[26] G. Schneider and J. P. Winters, *Applying Use Cases: A Practical Guide*. Addison-Wesley, 1998.
[27] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Prentice Hall, 2001.

[28] M. R. U. Siddique and A. Ahmad, "Explainable AI approaches for effort estimation in agile projects," *Procedia Comput. Sci.*, vol. 207, pp. 3612–3621, 2022.
[29] A. Singh, R. Kumar, and P. Sharma, "Combination of fuzzy logic and SVMs for balanced prediction accuracy and interpretability," *J. Syst. Softw.*, vol. 178, p. 110965, 2021.
[30] Standish Group, "CHAOS Report 2023: Agile project success rates," 2023.
[31] A. Tawosi, M. Rezaei, and A. Khosravi, "Hybrid models combining neural networks with optimization algorithms for enhanced prediction robustness," *IEEE Access*, vol. 10, pp. 45621–45635, 2022.
[32] M. Usman, E. Mendes, and J. Börstler, "Effort estimation in agile software development: A systematic literature review," in *Proc. 10th Int. Conf. Predictive Models in Software Engineering*, 2014, pp. 82–91.
[33] E. G. Wanderley, A. Vasconcelos, and B. T. Avila, "Using function points in agile projects: A comparative analysis between existing approaches," in *Agile Methods*, Springer, 2018, pp. 47–59.
[34] R. K. Yin, *Case Study Research and Applications: Design and Methods*, 6th ed. SAGE Publications, 2017.
[35] E. Yourdon, *Modern Structured Analysis*. Prentice-Hall, 1989.
[36] GeeksforGeeks, "Functional point (FP) analysis – Software engineering," 2024. [Online]. Available: https://www.geeksforgeeks.org/software-engineering-functional-point-fp-analysis/
[37] GeeksforGeeks, "Advantages and disadvantages of COCOMO model," 2024. [Online]. Available: https://www.geeksforgeeks.org/advantages-disadvantages-of-cocomo-model/
[38] FunctionPoints.org, "Function point analysis in practice." [Online]. Available: https://www.functionpoints.org/fpa-in-practice.html
[39] AgileConnection, "Function point analysis and agile methodology." [Online]. Available: https://www.agileconnection.com/article/function-point-analysis-and-agile-methodology
[40] IDC Blog, "How agile development teams can resolve agile measurement challenges with function point analysis," 2022. [Online]. Available: https://blogs.idc.com/2022/02/11/how-agile-development-teams-can-resolve-agile-measurement-challenges-with-function-point-analysis/
[41] Fingent Technologies, "AI systems in project management," 2023.
[42] A. Cockburn, *Writing Effective Use Cases*. Addison-Wesley, 2001.
[43] M. Cohn, *User Stories Applied for Agile Software Development*. Addison-Wesley, 2004.
[44] M. Cohn, *Agile Estimating and Planning*. Addison-Wesley, 2005.