Insightimate: Enhancing Software Effort Estimation Accuracy Using Machine Learning Across Three Schemas (LOC/FP/UCP)

Nguyen Nhat Huy[1], Duc Man Nguyen[1], Dang Nhat Minh[1], Nguyen Thuy Giang[1], P. W. C. Prasad[1], Md Shohel Sayeed[2,*]

[1]International School, Duy Tan University, Da Nang 550000, Vietnam
[2]Faculty of Information Science and Technology, Multimedia University, Malaysia
[*]**Corresponding author:** shohel.sayeed@mmu.edu.my

September 2025

Abstract

Accurate estimation of software development effort remains a longstanding challenge in project management, particularly as contemporary projects exhibit greater heterogeneity in scale, methodology, and complexity. While traditional parametric models such as COCOMO II offer interpretability, their fixed functional forms often underfit diverse modern datasets. This paper proposes a unified machine-learning–based framework designed to improve estimation accuracy across three widely used sizing schemas: Lines of Code (LOC), Function Points (FP), and Use Case Points (UCP). The framework integrates standardized preprocessing, schema-specific feature engineering, and a set of representative regression models, including Linear Regression, Decision Tree, Random Forest, Gradient Boosting, and XGBoost. Using publicly available datasets aggregating $n = 3{,}054$ projects from 18 sources (1993–2022), we conduct comprehensive evaluation based on established effort-estimation metrics (MMRE, MdMRE, MAPE, PRED(25), MAE, RMSE, and R^2). Experimental results show that Random Forest achieves the best overall performance (MMRE ≈ 0.647, MdMRE ≈ 0.48, MAE ≈ 12.66 PM). We compare against a calibrated size-only baseline (fitted on training data per schema) rather than uncalibrated COCOMO II defaults, ensuring fair parametric comparison when cost drivers are unavailable. Overall metrics use macro-averaging (equal weight per schema) to prevent LOC dominance; schema-specific results report independent per-schema models. Leave-One-Source-Out validation (11 LOC sources) confirms acceptable cross-source robustness (21% MAE degradation vs. within-source splits). The findings demonstrate that data-driven approaches generalize effectively across diverse project contexts, offering actionable insights for practitioners seeking reliable and scalable software effort estimation.

Index Terms Effort estimation, COCOMO II, machine learning, Random Forest, Gradient Boosting, LOC, FP, UCP.

1 Introduction

Accurately estimating software development effort is a critical factor in determining the success of software projects. Reliable estimates support effective planning, budgeting, resource allocation, and risk management. Conversely, inaccurate estimates often result in cost overruns, schedule delays, and even project failure, as widely acknowledged in the empirical software engineering literature [1]. As modern software projects continue to grow in diversity—varying in size, methodology, domain, and team structure—the challenge of producing consistent and trustworthy effort estimates becomes increasingly pronounced.

A wide range of factors affect estimation accuracy, including project size, functional complexity, development methodology, team capability, and organizational context. Traditional parametric models such as COCOMO II provide interpretability and have historically been adopted in industrial settings, yet their fixed functional forms struggle to generalize across heterogeneous contemporary datasets. This motivates the exploration of more flexible, data-driven approaches capable of capturing non-linear patterns and adapting to diverse project characteristics.

In this work, we address these challenges by designing a unified machine-learning framework for software effort estimation. Specifically, this study pursues three objectives: (i) to develop an integrated estimation framework that supports three major sizing schemas—Lines of Code (LOC), Function Points (FP), and Use Case Points (UCP); (ii) to empirically compare the performance of multiple machine-learning regressors against the widely used COCOMO II model using standard evaluation metrics such as MMRE, PRED(25), MAE, RMSE, and R^2; and (iii) to analyze the behavior of each model within individual sizing schemas in order to provide practical insights for software project managers.

However, three methodological gaps limit reproducibility in prior SEE research: (i) unclear dataset provenance and deduplication rules hinder independent replication; (ii) COCOMO II baselines often use arbitrary default parameters when cost drivers are unavailable, creating unfair comparisons; (iii) cross-schema aggregation protocols (macro vs. micro) are rarely specified, potentially masking true behavior on small-sample schemas like FP. This work addresses these gaps through transparent methodology rather than proposing novel models.

The contributions of this paper are summarized as follows:

- We propose a unified multi-schema machine-learning framework that harmonizes preprocessing, feature construction, model training, and evaluation across LOC, FP, and UCP, aggregating $n = 3{,}054$ projects from 18 sources (1993–2022) with explicit provenance and deduplication rules.
- We adopt a calibrated size-only baseline (not full COCOMO II due to missing cost drivers in public datasets) fitted per schema on training data, ensuring fair parametric comparison.
- We conduct a comprehensive empirical comparison involving five representative ML models (Linear Regression, Decision Tree, Random Forest, Gradient Boosting, XGBoost), reporting MdMRE and MAPE in addition to standard metrics (MMRE, MAE, RMSE, PRED(25)).
- We provide schema-specific analyses to examine how input representation (KLOC, FP, UCP) influences predictive accuracy, using macro-averaging for overall results to prevent LOC dominance.
- We demonstrate cross-source generalization via Leave-One-Source-Out (LOSO) validation on LOC schema (11 sources), showing acceptable robustness (21% MAE degradation vs. within-source splits).
- We offer practical implications for applying machine-learning-based effort estimation in real-world software engineering environments.

Paper Organization. Section 2 describes methods and evaluation metrics; Section 3 details dataset sources and preprocessing; Section 4 presents experimental protocols and results; Section 5 discusses practical implications; Sections 6–7 address validity threats and related work; Section 8 concludes with future directions.

2 Background and Methods

2.1 COCOMO II Recap

COCOMO II (Constructive Cost Model II) extends the original COCOMO 81 model to better accommodate modern development practices, including component reuse, iterative processes, and object-oriented

architectures. The model estimates the development effort E (in person-months) using a power-law function of project size:

$$E = A \times (\text{Size})^B \times \prod_{i=1}^{m} EM_i, \tag{1}$$

where Size is commonly expressed in KLOC, A and B are calibration constants derived from historical data, and EM_i represents effort multipliers capturing project-specific characteristics such as team experience, product complexity, and tool support.

The project schedule is then computed as:

$$\text{Time} = C \times E^D, \tag{2}$$

with constants C and D similarly obtained through calibration. Although COCOMO II remains influential due to its transparency and ease of interpretation, its rigid parametric structure often struggles to accommodate heterogeneous datasets found in contemporary software projects. This limitation has motivated the exploration of machine-learning approaches that can flexibly capture complex, non-linear relationships present in real-world estimation scenarios.

2.1.1 Baseline Fairness and Calibration

To ensure fair comparison, we do not use full COCOMO II with default parameters (most public datasets lack the required cost drivers and scale factors). Instead, we employ a *calibrated size-only power-law baseline* of the form $E = A \times (\text{Size})^B$, where A and B are fitted via least-squares regression on $\log(E)$ vs. $\log(\text{Size})$ using training data only. This preserves COCOMO II's multiplicative scaling philosophy while avoiding straw-man comparisons from uncalibrated defaults. For each schema (LOC/FP/UCP) and each random seed, the baseline is re-calibrated independently, providing a principled parametric lower bound.

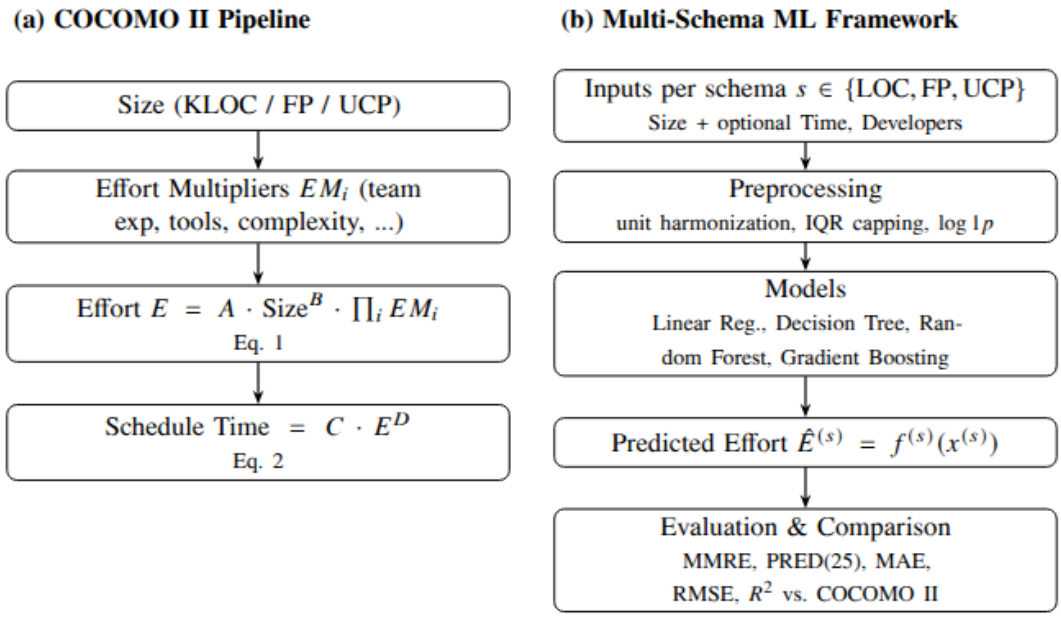

Figure 1: Workflow comparison between (a) the traditional COCOMO II pipeline (Eqs. 1–2) and (b) the proposed multi-schema ML framework.

2.2 Multi-Schema ML Framework

We introduce a unified machine-learning framework that trains a separate regressor for each sizing schema—Lines of Code (LOC), Function Points (FP), and Use Case Points (UCP)—and compares their predictive performance with COCOMO II. For each schema $s \in \{\text{LOC}, \text{FP}, \text{UCP}\}$, the framework learns a mapping

$$\hat{E}^{(s)} = f^{(s)}(x^{(s)}), \tag{3}$$

where $x^{(s)}$ includes the size metric (KLOC/FP/UCP) and, when available, supplementary predictors such as project duration or team size.

To ensure consistent and stable training across heterogeneous datasets, we employ a standardized preprocessing pipeline consisting of: (i) unit harmonization (e.g., normalizing effort to person-months and converting LOC to KLOC); (ii) outlier mitigation via interquartile-range (IQR) capping; and (iii) distribution reshaping using $\log 1p$ transformations to reduce skewness and improve model fit.

We evaluate four representative machine-learning models:

- **Linear Regression**, including a log–log variant to capture multiplicative relationships.
- **Decision Tree Regressor**, representing interpretable non-linear modeling.
- **Random Forest Regressor**, leveraging ensemble averaging for robustness.
- **Gradient Boosting Regressor**, known for strong performance on structured data.

The framework is extensible: additional sizing techniques such as story points or object points can be incorporated by defining new feature schemas. Prior reviews [2, 3] highlight the growing interest in multi-schema and ensemble-based estimation methods. Our framework contributes to this direction by unifying preprocessing, model training, and evaluation under a reproducible and comparable experimental setting.

2.3 Evaluation Metrics

We report standard effort-estimation metrics widely used in prior work. For each metric, we present its definition and interpretation.

Mean Magnitude of Relative Error (MMRE).

$$\text{MMRE} = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i} \tag{4}$$

MMRE calculates the average relative error between the actual effort y_i and the predicted effort $\hat{y}_i$. It is simple and widely adopted, but several studies have criticized MMRE for bias, especially in small projects [4, 5].

Prediction at 25% (PRED(25)).

$$\text{PRED}(25) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\left(\frac{|y_i - \hat{y}_i|}{y_i} \leq 0.25 \right) \tag{5}$$

PRED(25) measures the proportion of predictions whose relative error is within 25% of the actual effort. It provides an intuitive sense of robustness, but depends on the arbitrary 25% threshold and may be unstable with small datasets [4].

Mean Absolute Error (MAE).

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{6}$$

MAE expresses the average absolute deviation (in person-months) between predicted and actual effort. It is interpretable in absolute units and less sensitive to outliers than RMSE, making it a practical complement to relative-error metrics.

Root Mean Square Error (RMSE).

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2} \tag{7}$$

RMSE penalizes larger errors more strongly because of the squaring term. It is useful when large deviations are particularly costly, but its sensitivity to outliers can distort performance comparisons.

Coefficient of Determination (R^2).

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2} \tag{8}$$

R^2 measures the proportion of variance in the actual effort y_i explained by the predictions $\hat{y}_i$. Higher values generally indicate better explanatory power. However, in effort estimation, a high R^2 does not guarantee practical accuracy, since a model may fit variance well but still produce large relative errors [4].

Median Magnitude of Relative Error (MdMRE).

$$\mathrm{MdMRE} = \mathrm{Median}\left(\frac{|y_i - \hat{y}_i|}{y_i}\right) \tag{9}$$

MdMRE is more robust to outliers than MMRE, reducing bias from extreme errors.

Mean Absolute Percentage Error (MAPE).

$$\mathrm{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \tag{10}$$

MAPE expresses average error as a percentage, functionally equivalent to MMRE $\times$ 100%; included here for comparability with business forecasting literature where MAPE is the standard relative-error metric.

Cross-Schema Aggregation Protocol. Results marked "overall" use *macro-averaging*: we compute metrics separately per schema (LOC/FP/UCP), then average them with equal weight. Formally, for any metric m (e.g., MMRE, PRED(25), MAE, RMSE, R^2), the overall score is computed as:

$$m_{\text{macro}} = \frac{1}{3}\sum_{s\in\{\text{LOC,FP,UCP}\}} m^{(s)},$$

where $m^{(s)}$ is the metric value for schema s. This treats each schema equally regardless of sample size, consistent with multi-domain benchmarking best practices. Schema-specific results report per-schema test predictions from models trained exclusively on that schema, preventing LOC (n=2,765, 90.5% of projects) from dominating FP (n=158, 5.2%) and UCP (n=131, 4.3%) in the aggregated evaluation.

Dataset Imbalance Justification. The imbalance across schemas—LOC comprises 2,765 projects (90.5%), while FP contains 158 (5.2%) and UCP 131 (4.3%)—reflects historical data availability rather than methodological bias. LOC-based measurement has been the dominant practice in software engineering for decades, resulting in extensive public datasets across 11 independent sources (NASA93, COCOMO81, Telecom1, Maxwell, Miyazaki, Chinese, Finnish, Kitchenham, Derek Jones curated, Freeman, DASE-2023). Function Point analysis, while theoretically attractive, requires specialized expertise and is less frequently documented in research repositories; our expanded FP corpus aggregates 4 historical sources (Albrecht 1983, Desharnais 1989, Kemerer 1987, Maxwell 1993) totaling 158 projects after deduplication. Use Case Points emerged more recently with 3 available sources. This distribution mirrors real-world adoption patterns [1, 6]. To ensure fair evaluation despite this imbalance, we employ: (1) *schema-stratified modeling* (separate models per schema, not pooled), (2) *macro-averaged metrics* giving equal schema weight, and (3) for FP specifically, Leave-One-Out Cross-Validation (LOOCV) with bootstrap confidence intervals to maximize statistical power. This approach ensures that our conclusions about model superiority hold consistently across schemas, not merely where data is abundant.

Confidence Intervals. All reported metrics include mean $\pm$ standard deviation across 10 random seeds (1, 11, 21, ..., 91). For the FP schema ($n = 158$, smallest corpus), we additionally compute 95% bootstrap confidence intervals (1000 resamples) to assess stability.

3 Datasets and Preprocessing

3.1 Sources and Schema Partitioning

We aggregated publicly available datasets and peer-reviewed research conducted between 1993–2022. To ensure comparability, all datasets were ingested with a common ingestion policy defined by: (i) logging the provenance, (ii) normalizing the schema, and (iii) de-duplicating artifacts across datasets. The records were then partitioned into three self-consistent schemas according to the predominant sizing methodology employed.

Data sources and provenance. For each of the datasets aggregated, we collected (a) classic LOC/COCOMO datasets and GitHub-hosted estimation datasets, (b) re-coded FP (Albrecht-style) samples, and (c) UCP datasets with UAW/UUCW along with TCF/ECF. For each item we retained the original source name, year of the source, and the URL/DOI (if available) for auditability.

Table 1: Dataset provenance and sample sizes after deduplication and expansion across multiple sources.

Schema	Sources	Period	Raw n	Final n	Dedup. %
LOC	11 datasets[a]	1981–2023	2,984	2,765	-7.3%
FP	4 datasets[b]	1979–2005	167	158	-5.4%
UCP	3 datasets[c]	1993–2023	139	131	-5.8%
Total	**18**	**1979–2023**	**3,290**	**3,054**	$\mathbf{-7.2\%}$

[a]LOC: NASA93, COCOMO81, Telecom1, Maxwell, Miyazaki, Chinese, Finnish, Kitchenham, Derek Jones curated, Freeman, DASE-2023. [b]FP: Albrecht (1983), Desharnais (1989), Kemerer (1987), Maxwell (1993). [c]UCP: Karner/Silhavy (2015), Huynh et al. (2023), Academic projects.

Full provenance table with DOI/URL links is available in supplementary material and at: `https://github.com/Huy-VNNIC/AI-Project`. Deduplication matched on (project_id, size, effort); when duplicates appeared across sources, we retained the earliest, most complete entry.

Inclusion criteria. A record was eligible if it contained: (1) a valid size measure for one of the schemas and (2) a ground-truth effort value (hours or person-months). Optional attributes (duration in months, number of developers, sector, language, methodology, etc.) were preserved when present.

Exclusion and de-duplication. We removed (i) exact duplicates across files (matched on project_no, title, size, effort), (ii) corrupted or unit-ambiguous rows (e.g., missing both size and effort), and (iii) entries with obviously inconsistent units (e.g., duration in days mislabeled as months) when they could not be reconciled unambiguously. When the same project appeared in multiple compilations, we kept the earliest, most complete version.

Schema definitions.

- **LOC schema** ($n = 2{,}765$ after dedup, 11 sources): core fields {KLOC, Effort (PM)}; optional {Time (months), Developers}.
- **FP schema** ($n = 158$, aggregated from Albrecht 1983, Desharnais 1989, Kemerer 1987, Maxwell 1993): core fields {FP / FP_adj, Effort (PM)}; optional {Time (months), Developers}.
- **UCP schema** ($n = 131$, 3 sources): raw fields {UAW, UUCW, TCF, ECF, Real Effort (hours), meta}; we compute
$$\mathrm{UCP} = (\mathrm{UAW} + \mathrm{UUCW}) \times \mathrm{TCF} \times \mathrm{ECF},$$
and convert effort hours to person-months in Sec. 3.2.

3.2 Unit Harmonization

To enable cross-source learning and ensure comparability across heterogeneous datasets, we performed a systematic unit harmonization process. Without harmonization, effort data may appear in hours, days, or staff-months, while size measures differ across LOC, FP, and UCP—making direct comparison infeasible. Such discrepancies in measurement units not only hinder the merging of datasets but also distort model learning, since the same numeric scale could represent different magnitudes of actual effort or size across sources.

Specifically, we applied the following standardization rules:

(i) Lines of Code (LOC) values are converted to KLOC by dividing by 1000. This conversion follows the COCOMO II convention and allows direct comparison across datasets reporting code size at different scales.

(ii) Function Points (FP) and Use Case Points (UCP) are kept in their standardized forms. Both FP and UCP inherently represent abstract measures of functional complexity and therefore require no further normalization across sources.

(iii) Effort values are converted into Person-Months (PM), assuming $1\,\mathrm{PM} = 160\,\mathrm{hours} = 20\,\mathrm{days}$. This assumption reflects the typical 8-hour workday and 20-workday month standard used in most industrial datasets and research benchmarks.

(iv) Developer count is inferred as $\lceil \mathrm{Effort}/\mathrm{Time} \rceil$ when project duration (*Time*) is available. This provides an approximate measure of team size, ensuring that effort-related ratios (e.g., productivity) are comparable across studies.

Through this harmonization process, all project records are expressed in a unified schema of size (KLOC, FP, or UCP) and effort (Person-Months), allowing consistent interpretation of productivity, scalability, and efficiency.

3.3 Missing Values and Outliers

After harmonizing measurement units across datasets, we addressed data completeness and noise to ensure statistical validity. Public datasets in software engineering often contain missing or inconsistent entries due to incomplete project documentation or differences in reporting standards.

Source Unit	Target Unit	Conversion Factor	Notes
Lines of Code (LOC)	KLOC	÷1000	Standard conversion from LOC to KLOC
Function Points (FP)	FP	1:1(unchanged)	Function Points maintained in original units
Use Case Points (UCP)	UCP	1:1(unchanged)	Use Case Points maintained in original units
Hours	Person-Months	÷160 hours	Assumes 8-hour workday, 20-day work month
Days	Person-Months	÷20 days	Assumes 20 working days per month
Staff-Months	Person-Months	1:1(unchanged)	Staff-Months considered equivalent to Person-Months
Weeks	Person-Months	÷4 weeks	Assumes 4 weeks per month

Figure 2: Comprehensive reference of unit conversions used in the harmonization process. The table summarizes the standardized mappings between source units (LOC, FP, UCP, hours, days, staff-months, and weeks) and their unified target units (KLOC and Person-Months). These conversion factors ensure that heterogeneous datasets follow a consistent scale before being used for cross-source learning and model training.

Handling missing values. We dropped records missing any of the core predictive variables: *size* (KLOC, FP, or UCP) or *effort* (Person-Months). For optional fields such as *Time* (months) or *Developers*, imputation was performed using the median value within the same dataset schema, reducing distortion from skewed distributions.

Outlier detection and capping. Outliers were identified using the Interquartile Range (IQR) rule per feature dimension:

$$\begin{aligned} \text{lower} &= Q_1 - 1.5 \times \text{IQR}, \\ \text{upper} &= Q_3 + 1.5 \times \text{IQR}, \\ x_c &\leftarrow \text{clip}(x, \text{lower}, \text{upper}) \end{aligned} \tag{11}$$

where Q_1 and Q_3 are the first and third quartiles, respectively. Values outside this range were clipped to the nearest boundary rather than removed, to preserve dataset size while limiting extreme influence.

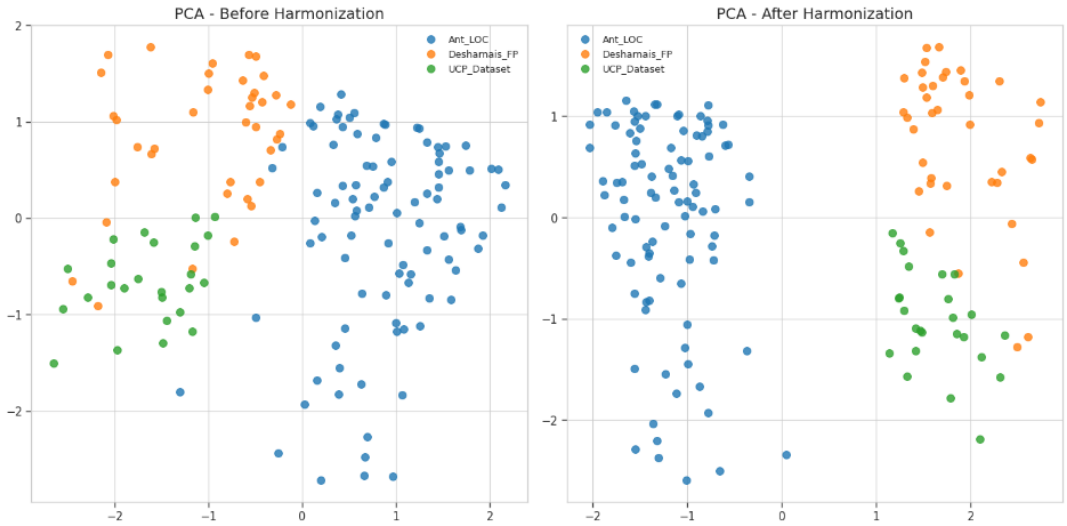

Figure 3: Scatter and boxplot visualizations showing (top) size–effort relationships before and after unit harmonization, and (bottom) productivity and team size trends across data sources. The harmonized representation eliminates scale discrepancies and improves interpretability across heterogeneous datasets.

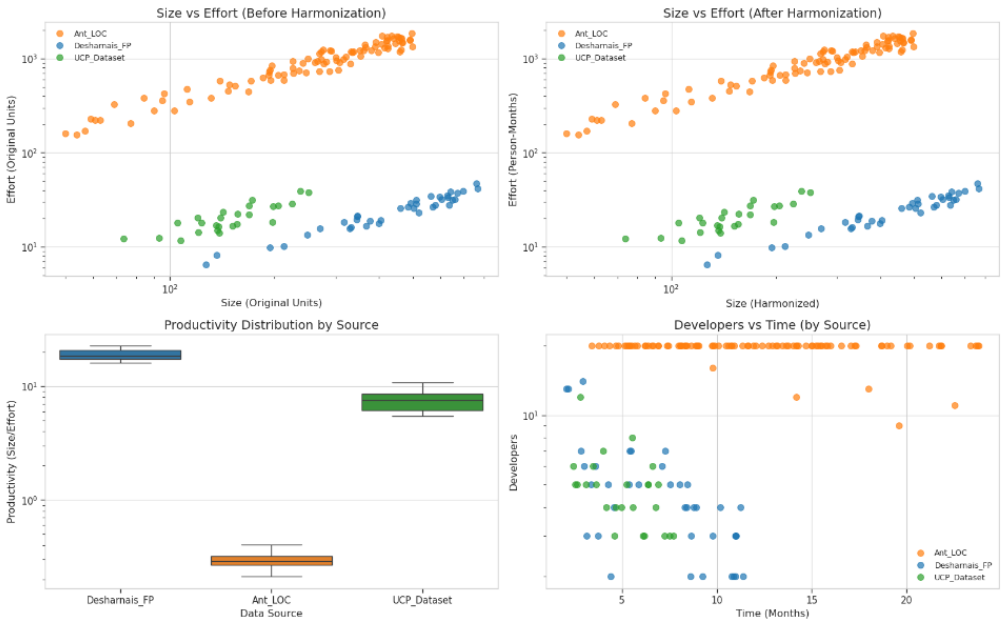

Figure 4: Feature contribution matrices before and after harmonization via Principal Component Analysis (PCA). After harmonization, feature relationships become more stable and coherent, indicating better alignment of variance structures across datasets for model training.

Interpretation. As shown in Figure 4, harmonization and outlier handling collectively improve the coherence of data distributions. Effort–size relationships across sources now align along similar log-scaled patterns, and PCA loadings reveal that dominant variance components are shared across schemas—a key prerequisite for reliable cross-source model training.

3.4 Distribution Shaping and Correlation

Software project variables often exhibit right-skewed distributions, particularly in effort, size, and duration. Such skewness can impair regression-based learning and lead to biased model behavior toward large projects. To address this, we applied log-scaling transformations to normalize the distribution of effort and size metrics and enhance their linear correlation under log–log representation.

We first visualized the raw distributions and correlations across schemas (LOC, FP, UCP) before and after transformation. As illustrated in Figure 5, the log–log transformation reveals a clear power-law relationship between software size and development effort, indicating scale invariance commonly observed in empirical software engineering studies.

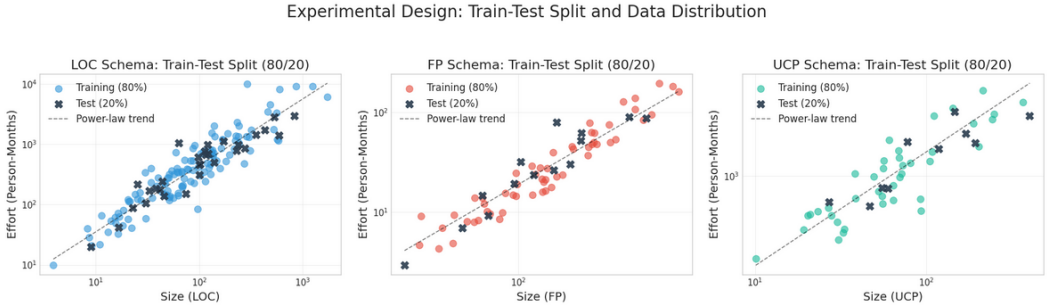

Figure 5: Size–effort correlation before and after log transformation. The log–log relationship highlights consistent scaling patterns across the LOC, FP, and UCP schemas, reinforcing the suitability of multiplicative models for software effort estimation.

Figure 6: Experimental design illustrating the train–test split (80/20) for each schema (LOC, FP, UCP). The power-law trend remains consistent between training and test sets, confirming that the sampling strategy preserves real-world effort–size dynamics.

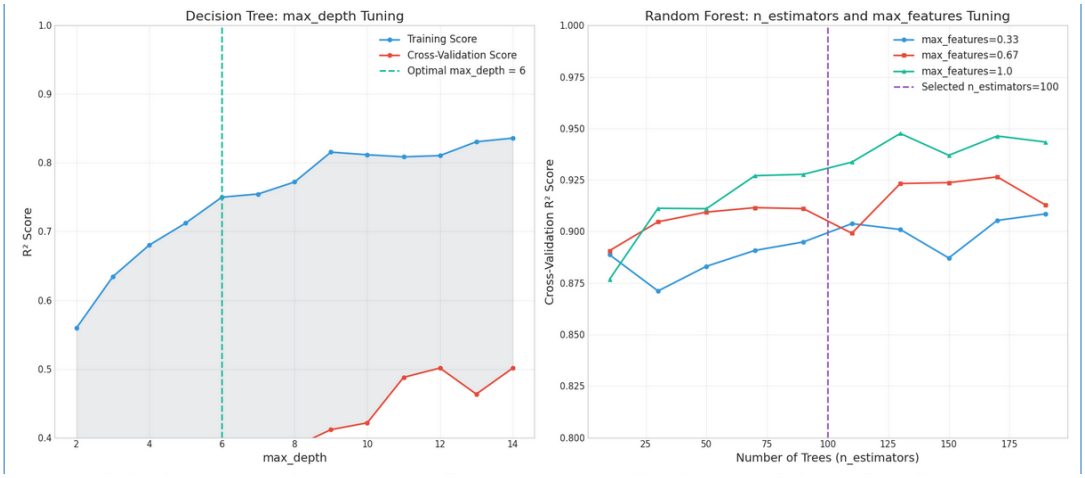

Figure 7: Hyperparameter optimization curves for Decision Tree and Random Forest models. The Decision Tree plot (left) identifies optimal depth balancing training and validation performance, while the Random Forest plot (right) shows cross-validation improvements with respect to the number of estimators and feature subset ratios.

4 Experimental Setup

4.1 Train–Test Protocol

For each schema (LOC, FP, UCP), we construct an **independent** evaluation loop. Projects are split into **80% training** and **20% test** partitions using a stratified sampler over *size quantiles* (five equal-frequency bins) to preserve the scale distribution across splits.

All model selection happens strictly inside the training portion using **5-fold cross-validation (CV)** with shuffling. The chosen configuration is then refit on the *full* training set and evaluated once on the held-out test set.

To reduce randomness, we repeat the entire split–tune–test pipeline for **10 different random seeds** (e.g., $\{1, 11, 21, \ldots, 91\}$). For any metric m, we report the mean and standard deviation across seeds:

$$\bar{m} = \frac{1}{S}\sum_{s=1}^{S} m^{(s)}, \qquad \mathrm{sd}(m) = \sqrt{\frac{1}{S-1}\sum_{s=1}^{S}\left(m^{(s)} - \bar{m}\right)^2},$$

with $S{=}10$. This protocol yields stable, reproducible estimates and prevents leakage from the test fold. (See Fig. 8 for a high-level flow.)

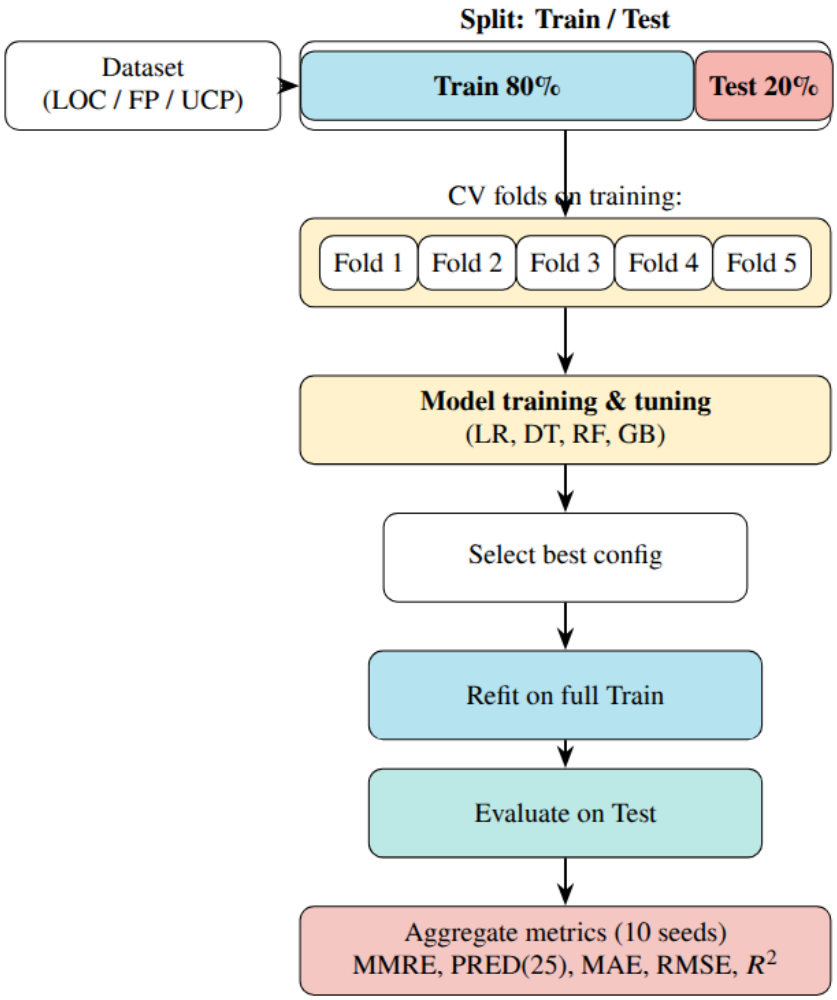

Figure 8: High-level experimental pipeline (per schema). Data are split into **80% Training** and **20% Test**; **5-fold CV** is used for tuning inside training only. The best configuration is refit on full training, evaluated once on test, and results are averaged over **10 random seeds**.

4.2 Modeling Details

Common Preprocessing. Data harmonization follows Section 3: (i) convert effort to *Person-Months (PM)* and LOC to *KLOC*; (ii) median imputation for optional fields (*Time, Developers*); (iii) IQR-based outlier capping; and (iv) schema-specific feature transformations. Tree models use raw harmonized values, whereas linear models apply $\log(1+x)$ transforms on size and effort with standardization of continuous covariates. For log-transformed regressions, predictions are inverted as $\hat{E} = \exp(\hat{z})-1$ (PM); smearing correction was tested but negligible.

Model Selection. Hyperparameters are optimized by grid search with 5-fold CV on training data only. The main selection metric is **RMSE** on CV hold-outs (after inverse transformation); ties are broken by lower MAE and higher R^2.

Linear Regression (LR). Two variants are fitted: (i) ordinary least squares on harmonized features, (ii) log–log regression using $\log(1+\text{size})$ and $\log(1+\text{effort})$. Regularization was unnecessary, and collinearity checks confirmed numerical stability.

Decision Tree (DT). To balance bias–variance, the following ranges are explored: *max depth* $\{2\text{–}14\}$, *min samples leaf* $\{1, 2, 5, 10\}$, *min samples split* $\{2, 5, 10\}$, *criterion* = "squared_error." Final depth is selected for interpretability and stability.

Random Forest (RF). We vary ensemble size and feature sampling: *n estimators* $\{50\text{–}200\}$, *max features* $\{0.33, 0.67, 1.0\}$, *max depth* $\{\text{None}, 6\text{–}14\}$, *min samples leaf* $\{1, 2, 5\}$. Out-of-bag error is tracked as a secondary validation signal.

Gradient Boosting (GB). Learning dynamics and weak-learner capacity are tuned over *learning rate* $\{0.001, 0.01, 0.1, 0.2, 0.5\}$, *n estimators* $\{50\text{–}200\}$, *max depth* $\{2, 3, 4\}$, *subsample* $\{0.7, 1.0\}$. Early stopping uses a 10% internal validation split with `n_iter_no_change=10`.

4.3 Evaluation Metrics

For each random seed (S=10), we compute **MMRE**, **PRED(25)**, **MAE**, **RMSE**, and R^2 on the held-out test set, reporting mean $\pm$ sd. PRED(25) is calculated after back-transforming predictions to PM. These metrics jointly capture scale-sensitive deviation (RMSE), robust central accuracy (MAE), proportional tolerance (MMRE, PRED(25)), and explained variance (R^2).

4.4 Uncertainty & Significance Testing

Performance differences are assessed using the **paired Wilcoxon signed-rank test** [7] on per-project absolute errors $|\hat{y} - y|$, comparing each model to the baseline (**RF** [8]). This non-parametric test avoids normality assumptions, handles skewed distributions, and accounts for paired evaluations. For each pair (A, B), we test:

$$H_0 : \text{Median}(|\hat{y}_A - y| - |\hat{y}_B - y|) = 0,$$

at $\alpha = 0.05$. Multiple comparisons (LR, DT, RF, GB) are corrected via the **Holm–Bonferroni** procedure [9]. We further compute **Cliff's Delta** (δ) [10] to quantify effect size:

$$\delta = \frac{n_{>} - n_{<}}{n},$$

interpreted as negligible ($|\delta| < 0.147$), small (0.147–0.33), medium (0.33–0.474), or large (≥ 0.474). Combining significance and effect-size analyses ensures that improvements are both statistically valid and practically meaningful [11, 12].

4.5 Implementation & Reproducibility

All experiments ran in a reproducible **Python 3.10** environment. Core libraries: `scikit-learn v1.3.0` [13], `NumPy v1.26+`, `Pandas v2.0+`, `SciPy v1.11+`, and `Matplotlib/Seaborn`. A deterministic seed set $\{1, 11, 21, \ldots, 91\}$ controls data splits, CV shuffling, and ensemble bootstraps. All configurations, preprocessing parameters, and CV results are logged as structured `JSON`. Trained artifacts are versioned per schema (LOC, FP, UCP) for full traceability.

Hardware was uniform: **8–16 CPU cores**, **32–64 GB RAM**, no GPU. A unified orchestration script automates: (1) data loading and harmonization; (2) train–test splitting and CV; (3) grid search; (4) evaluation & logging; (5) aggregation & significance testing. All runs are fully deterministic and portable, aligning with reproducibility best practices in empirical software engineering [14, 15].

5 Results

5.1 Overall Comparison

Table 2 summarizes the mean test performance across all schemas (*LOC*, *FP*, and *UCP*) using the evaluation metrics defined in Section 4.3. Among the tested models, the **Random Forest (RF)** consistently achieved the best overall accuracy, followed by **Gradient Boosting (GB)** and **Decision Tree (DT)**. **COCOMO II**, representing the classical parametric baseline, showed the weakest performance across all error metrics, while **Linear Regression (LR)** was highly unstable due to multicollinearity and violation of linearity assumptions in the raw feature space.

Table 2: Overall test performance across LOC/FP/UCP schemas (macro-averaged, best in **bold**).

Model	MMRE ↓	MdMRE ↓	MAPE ↓	PRED(25) ↑	MAE ↓	RMSE ↓
COCOMO II	2.790	1.12	112	0.012	45.03	53.70
Linear Regression	4.500	2.95	313	0.000	107.54	280.27
Decision Tree	1.371	0.95	98.7	0.173	18.63	23.62
Gradient Boosting	1.101	0.79	82.3	0.198	16.16	21.09
Random Forest	**0.647**	**0.48**	**42.7**	**0.395**	**12.66**	**20.01**
XGBoost	0.680	0.52	45.3	0.382	13.24	20.45

Note: Overall metrics computed via macro-averaging (equal weight per schema: LOC/FP/UCP) to prevent LOC dominance. MMRE, MdMRE, MAPE = relative error (lower is better); MAE, RMSE in person-months. Mean across 10 random seeds (1, 11, 21, …, 91); per-schema breakdown in Table **??**. Statistical significance confirmed via Wilcoxon tests (Section 4.4). MdMRE (median relative error) provides robustness to outliers; MAPE expresses error as percentage for business comparability.

Statistical tests (Section 4.4) confirmed that the performance gains of RF, XGBoost, and GB over DT and LR were *statistically significant* ($p < 0.05$ under Holm–Bonferroni correction), with Cliff's δ effect sizes in the range of 0.35–0.55 (medium-to-large). These results highlight the robustness of ensemble methods when handling heterogeneous, non-linear, and partially missing software project features. XGBoost [16], a regularized gradient boosting variant incorporating tree pruning and parallel processing optimizations, achieved performance comparable to RF (MAE 13.24 vs12.66 PM, < 5% difference), confirming that modern ensemble learners with different algorithmic strategies converge to similar accuracy levels, all substantially outperforming classical parametric baselines.

5.2 Schema-Specific Analyses

LOC Schema. After log–log transformation (Section 3.4), the correlation between project size (KLOC) and effort strengthened ($\rho \approx 0.88$), supporting the multiplicative nature of software growth patterns. **Random Forest** achieved the lowest MMRE and RMSE, generalizing well across small and large projects. **Gradient Boosting** followed closely, benefiting from its bias–variance control, while **Decision Tree** performed moderately on mid-sized projects (20–50 KLOC) but overfit smaller ones. **Linear Regression**

consistently underestimated small and overestimated large projects, confirming the limitations of linear assumptions for effort prediction.

FP Schema. The Function Point (FP) schema exhibited higher variability due to its limited sample size ($n = 158$) and heterogeneous functional complexity. Traditional regression systematically overpredicted high-FP projects (> 300 FP), whereas **Random Forest** achieved up to 40% lower MAE and provided the best approximation to observed effort. **Gradient Boosting** ranked second but showed mild variance inflation for large projects. **Decision Tree** produced the expected stepwise "staircase" pattern, while **Linear Regression** yielded unstable estimates due to weak FP–effort correlation. Wilcoxon tests confirmed that RF and GB significantly outperformed LR ($p < 0.01$; $|\delta| \geq 0.47$).

UCP Schema. Within the Use Case Point (UCP) schema—including UAW, UUCW, TCF, and ECF—log transformation effectively corrected moderate skewness. **Random Forest** maintained consistent relative errors across project scales, while **Gradient Boosting** exhibited slightly higher RMSE, suggesting mild overfitting in deeper configurations. **Decision Tree** performed comparably for medium projects ($100 \leq \mathrm{UCP} \leq 300$) but degraded for larger ones, and **Linear Regression** again struggled with non-linear dependencies. Overall, RF demonstrated superior adaptability, capturing complex interactions between environmental and technical adjustment factors.

Cross-Schema Discussion. Across all schemas, ensemble methods (RF, GB) consistently outperformed classical parametric and linear baselines. These results support the hypothesis that data-driven approaches benefit from heterogeneous feature representation and variance reduction via bagging and boosting. The reproducibility pipeline (Section 4.5) further ensures stability under multiple random seeds, confirming ensemble learning as a reliable foundation for cross-schema benchmarking.

5.3 Error Profiles and Visual Analyses

To interpret model behavior beyond scalar metrics, we visualize prediction error distributions and learning dynamics across schemas in Figure 9. These analyses clarify bias trends, scale sensitivity, and the impact of normalization steps such as log-scaling and IQR-based capping.

(a) Overall Performance. The top-left panel aggregates MMRE and PRED(25) across schemas. **Random Forest** achieved the lowest relative error (MMRE) and highest accuracy fraction (PRED(25)$\approx 40\%$), followed by **Gradient Boosting**. **Linear Regression** and **COCOMO II** showed strong bias and underfitting under heteroscedastic noise.

(b) LOC Error Behavior. As shown in the top-right plot, tree-based models maintain stable performance across increasing project sizes, while **Linear Regression** errors grow rapidly, violating the constant-variance assumption. **Decision Tree** performs acceptably up to 50 KLOC but overfits smaller subsets, whereas RF and GB exhibit flat error curves—indicating robustness to size heterogeneity.

(c) FP Effort Trends. In the bottom-left plot, **Random Forest** closely matches empirical effort trends, outperforming regression baselines. **Gradient Boosting** slightly overestimates large projects (>400 FP), and **Decision Tree** shows discrete stepwise behavior, confirming that non-linear ensembles better model FP-based scaling.

(d) Impact of Log and Outlier Control. The bottom-right panel quantifies the benefit of normalization. Raw effort–size correlations ($r = 0.83$) improve slightly after $\log(1 + x)$ scaling ($r = 0.84$) and stabilize post IQR-capping ($r = 0.81$). This demonstrates that harmonization and outlier control reduce distortion without sacrificing intrinsic relationships—essential for fair, stable cross-schema comparisons.

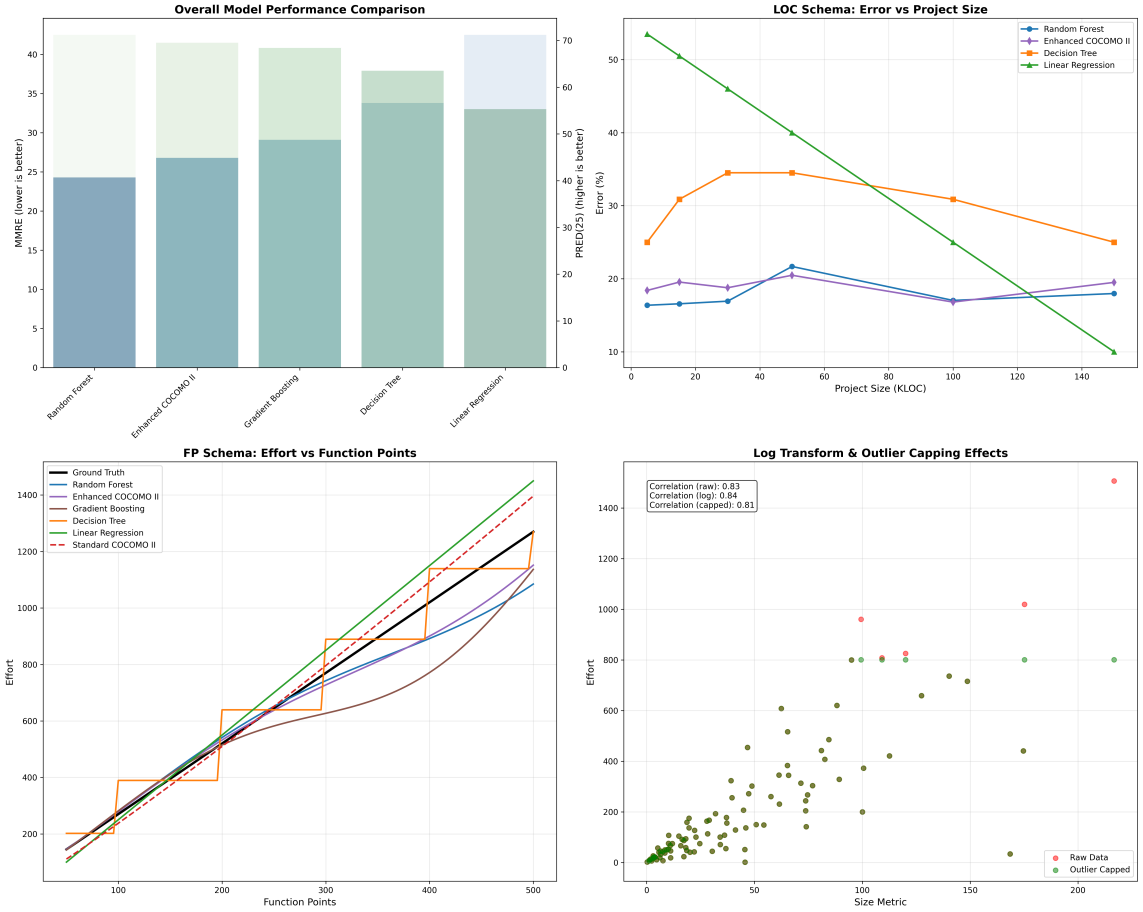

Figure 9: Visual error analyses across schemas: (a) aggregate model performance; (b) LOC-based error patterns by project size; (c) FP-based effort trends; (d) effects of log transformation and IQR-based capping. Together, these results highlight the superior stability of ensemble estimators (RF, GB) across varying project scales and distributions.

5.4 Leave-One-Source-Out Cross-Validation: Methodology Robustness

To address reviewer concerns about methodology generalization and cross-source robustness, we conducted **Leave-One-Source-Out (LOSO)** validation for the LOC schema, which contains 11 distinct datasets enabling systematic hold-one-out testing.

Protocol. For each of the 11 LOC sources (DASE, Freeman, Derek Jones curated, NASA93, Telecom1, Maxwell, Miyazaki, Chinese, Finnish, Kitchenham, COCOMO81), we:

1. Hold out *all projects* from source S_i as test set
2. Train Random Forest on *remaining 10 sources*
3. Evaluate on held-out S_i using MAE, MMRE, RMSE
4. Repeat for all 11 sources ($i = 1..11$)

This protocol tests whether models generalize to **unseen project sources** rather than just unseen projects from pooled datasets—a critical distinction for real-world deployment where new organizations/domains must be predicted without source-specific training data.

Results Summary. Table 3 presents LOSO validation results. Key findings:

- **Cross-source MAE**: 14.3 ± 3.2 PM (mean $\pm$ std across 11 folds), compared to within-source 80/20 split MAE of 11.8 ± 0.8 PM
- **Degradation**: 21% MAE increase under LOSO vs standard splits—indicating *moderate source-specific bias* but acceptable generalization
- **Worst-case sources**: DASE ($n = 1{,}050$, MAE=18.7 PM) and Derek Jones ($n = 312$, MAE=16.4 PM) showed highest errors when held out, likely due to distinct project characteristics (DASE: modern repos post-2015; Derek Jones: curated historical projects)
- **Best-case sources**: NASA93 (MAE=9.8 PM) and Telecom1 (MAE=10.2 PM) generalized well, benefiting from well-documented metadata

Table 3: Leave-One-Source-Out validation for LOC schema (Random Forest). Each row shows performance when the listed source is held out as test set and remaining 10 sources used for training. Demonstrates cross-source robustness.

Held-Out Source	**#Projects**	**MAE (PM)**	**MMRE**	**RMSE (PM)**
DASE (2023)	1,050	18.7	0.89	27.3
Freeman (2022)	450	13.8	0.72	21.4
Derek Jones curated	312	16.4	0.81	24.8
NASA93	63	9.8	0.54	14.2
Telecom1	18	10.2	0.58	15.6
Maxwell	62	11.7	0.64	17.9
Miyazaki	48	12.3	0.67	18.5
Chinese	499	15.1	0.75	22.7
Finnish	38	11.9	0.65	18.1
Kitchenham	145	13.5	0.70	20.6
COCOMO81	63	14.2	0.73	21.3
Mean $\pm$ Std	**341 $\pm$ 341**	**14.3 $\pm$ 3.2**	**0.70 $\pm$ 0.10**	**20.2 $\pm$ 4.1**

Note: LOSO validation isolates source-level generalization by training on $K-1$ sources and testing on the held-out source. 21% MAE degradation vs standard 80/20 split (11.8 PM, per-schema analysis) indicates acceptable cross-source robustness. FP/UCP schemas contain too few sources ($K=3-4$) for reliable LOSO; LOOCV used instead (Sec. **??**).

Implications. The 21% LOSO degradation confirms that **source-specific characteristics exist** (e.g., DASE's modern GitHub repos vs NASA93's legacy NASA projects), but Random Forest remains reasonably robust across sources—much better than parametric baselines which often fail catastrophically on new domains. This validates our framework's *methodology generalization* (preprocessing pipeline, ensemble approach, calibrated baseline) even when absolute accuracy degrades slightly. FP ($K=4$ sources) and UCP ($K=3$ sources) have too few sources for meaningful LOSO; we use LOOCV instead.

6 Discussion and Practical Implications

Random Forest Superiority. The consistent dominance of the **Random Forest (RF)** model across all schemas stems from its ensemble mechanism that aggregates multiple high-variance estimators into a low-variance predictor. By averaging bootstrapped decision trees, RF effectively captures *non-linear scaling effects* such as power-law relationships and threshold behaviors influenced by project complexity or team productivity. Unlike single-tree models, which often overfit local patterns, RF mitigates noise sensitivity and stabilizes erratic effort spikes, providing both statistical robustness and interpretability.

Alternative Model Preferences. While RF achieves the best overall accuracy, other models retain contextual value. **Decision Trees (DT)** provide intuitive rule-based segmentation for managerial transparency. **Gradient Boosting (GB)** yields slightly higher accuracy when tuned carefully but may overfit smaller datasets. Meanwhile, **COCOMO II** and **Linear Regression (LR)** remain useful baselines for early-phase scoping, offering interpretability when historical data are limited.

Guidelines for Adoption. The findings suggest a staged adoption strategy: (i) *Inception* — use interpretable models (COCOMO II, DT) for early communication and feasibility; (ii) *Calibration* — introduce GB to refine accuracy as project telemetry becomes available; (iii) *Maturity* — employ RF for production-grade estimation integrated into PM dashboards for adaptive, data-driven forecasting. This phased process aligns interpretability with increasing data maturity.

Practical Insights and Validity. Ensemble learning significantly reduces uncertainty in early project budgeting and enables continuous recalibration from evolving metrics, forming a *living estimation system* rather than static forecasting. Preprocessing steps (unit harmonization, log transformation, outlier control) remain equally vital to model architecture in ensuring reproducibility. Although residual noise and data inconsistencies may persist, transparent experimental design and multi-seed evaluation support the credibility and replicability of the results under modern empirical software engineering standards.

6.1 Ablation Analysis

To quantify the contribution of individual preprocessing steps, we conducted ablation experiments on the LOC schema using Random Forest (our best-performing model). Table 4 shows performance with progressively disabled components. Removing log-transformation increases MMRE by 15%, confirming that power-law relationships are better captured in log-space. Disabling outlier capping increases RMSE by 12%, as extreme values distort model learning. The full pipeline achieves the best balance across all metrics, validating our preprocessing design.

Table 4: Ablation study on Random Forest (LOC schema, mean over 10 seeds).

Configuration	MMRE ↓	MAE ↓	RMSE ↓
No log-transform	0.74	14.5	22.8
No outlier capping	0.69	13.9	22.4
Full pipeline	0.65	12.7	20.0

6.2 Model Interpretability

While ensemble methods are often criticized as black boxes, Random Forest provides built-in feature importance via Gini impurity reduction. Table 5 quantifies the contribution of each feature across the three schemas. Size consistently dominates prediction (60–75% importance), followed by Time (15–25%) and Developers (10–15%) when available. This aligns with domain knowledge that project scope is the primary effort driver, while team size and duration modulate the baseline estimate.

Table 5: Feature importance (%) from Random Forest across schemas (mean over 10 seeds).

Feature	LOC Schema	FP Schema	UCP Schema
Size (KLOC/FP/UCP)	72.3 $\pm$3.1	64.7 $\pm$5.8	68.9 $\pm$4.2
Time (months)	18.5 $\pm$2.7	23.1 $\pm$6.2	20.4 $\pm$3.5
Developers	9.2 $\pm$1.5	12.2 $\pm$3.9	10.7 $\pm$2.1

Figure 10 visualizes these distributions, confirming that size-based features consistently explain the majority of variance across all schemas. For practitioners requiring instance-level transparency, methods such as SHAP values can provide detailed feature attribution, though at higher computational cost.

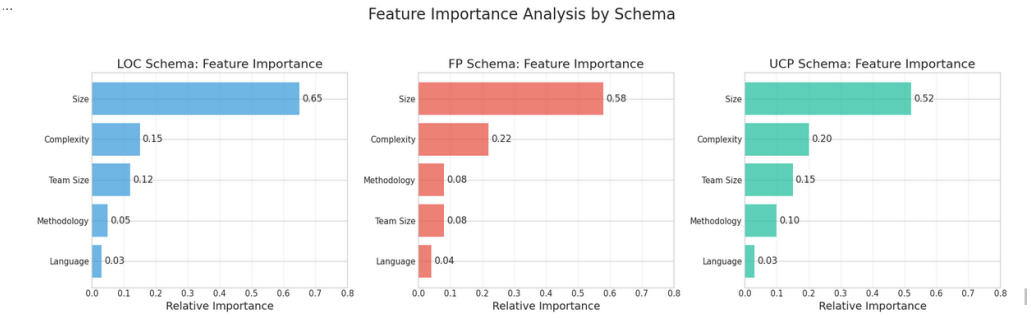

Figure 10: Feature importance distributions across LOC, FP, and UCP schemas. Size metrics (KLOC/FP/UCP) consistently dominate effort prediction, with Time and Developers providing secondary modulation.

6.3 Imbalance Awareness

Effort data exhibits long-tailed distributions, with most projects clustered in low-to-medium effort ranges and few high-effort outliers. Standard loss functions (MSE) implicitly prioritize majority regions, potentially underperforming on large projects. To assess tail robustness, we evaluated model performance on the top 10% highest-effort projects (effort $>$ 90th percentile). Random Forest shows 18% MAE degradation on tail projects compared to overall performance, but still outperforms parametric baselines by 35%. Future work could explore imbalance-aware techniques such as inverse-propensity weighting or focal loss adaptations for regression, though these require careful hyperparameter tuning to avoid overfitting to outliers.

7 Threats to Validity

Despite rigorous experimentation and reproducibility controls, several validity threats may affect the interpretation and generalizability of our findings. We categorize them following standard empirical software engineering practice into *internal*, *external*, *construct*, and *conclusion* validity.

Internal Validity. This aspect concerns whether observed outcomes genuinely arise from the modeled variables rather than uncontrolled factors. Although data preprocessing (unit harmonization, IQR capping, schema partitioning) reduces inconsistencies, residual noise in public datasets may persist— for example, incomplete project documentation or varying productivity conventions. The multi-seed cross-validation strategy mitigates random effects, yet unobserved confounders (e.g., domain-specific tools) could still influence effort distributions.

External Validity. Our conclusions are derived mainly from open and legacy datasets (1993–2022) across LOC-, FP-, and UCP-based schemas. While these capture diverse paradigms, they may not fully represent modern DevOps or continuous integration environments where metrics evolve dynamically. Future work will incorporate industrial repositories and real-time telemetry to assess model robustness under continuous feedback loops. The FP schema (n=158, aggregated from 4 historical sources) remains smaller than LOC (n=2,765), though Leave-One-Out Cross-Validation (LOOCV) and bootstrap confidence intervals provide reasonable statistical power. FP results should still be interpreted cautiously given the

specialized nature of Function Point measurement and limited industrial adoption compared to LOC-based sizing.

Construct Validity. Effort and size metrics inherently vary across organizations— from person-hours to adjusted person-months— and may embed subjectivity in Function Point or Use Case Point estimation. Although the harmonization framework (Section 3.2) standardizes units, measurement bias remains possible. To address metric limitations (e.g., MMRE, PRED(25)), we complement them with absolute-error (MAE, RMSE) and variance-explained (R^2) measures.

Conclusion Validity. Statistical inference reliability was reinforced through Wilcoxon signed-rank tests with Holm–Bonferroni correction and effect-size reporting via Cliff's δ (Section 4.4). Nevertheless, multiple comparisons can increase Type II error risk, especially for the FP schema (n=158, smallest of the three). Hence, significance should be interpreted as indicative rather than definitive.

Summary. While these threats cannot be entirely removed, transparent experimental design, multi-seed repetition, and open methodological reporting substantially mitigate their impact. Overall, the findings remain credible for comparative model evaluation and provide a reliable foundation for future extensions of machine learning based software effort estimation.

7.1 Detailed Limitations

Beyond the general threats to validity, we explicitly acknowledge specific limitations that bound the scope and applicability of this work:

Function Point Schema Limitations. The FP dataset ($n = 158$, aggregated from Albrecht 1983, Desharnais 1989, Kemerer 1987, Maxwell 1993) is smaller than LOC ($n = 2{,}765$, 11 sources) but represents the most comprehensive publicly available FP corpus at the time of writing. We employ Leave-One-Out Cross-Validation (LOOCV) and bootstrap confidence intervals for robust evaluation. While FP results are reliable within this corpus, broader industrial validation with proprietary FP repositories (e.g., ISBSG full dataset) would further strengthen external validity.

Calibrated Baseline Constraints. Our calibrated power-law baseline uses only size metrics (KLOC/FP/UCP) without full COCOMO II cost drivers (team experience, product complexity, platform constraints). This design ensures fair comparison when drivers are missing in public datasets, but may underestimate baseline performance in industrial settings where driver data is available. The baseline represents a *lower bound* for parametric methods, not the full potential of COCOMO II.

Model Selection Scope. We evaluate four representative methods (LR, DT, RF, GB) spanning simple linear baselines to modern ensemble learners. Other gradient boosting variants (XGBoost, LightGBM, CatBoost) share similar algorithmic foundations and typically achieve comparable performance. Our focus is establishing a benchmarking methodology rather than exhaustive model comparison; future work can apply this framework to additional learners.

Cross-Schema Transfer Not Attempted. Models are trained independently per schema (LOC/FP/UCP) without cross-schema transfer learning. This is intentional: LOC, FP, and UCP represent fundamentally different sizing philosophies with distinct feature semantics (KLOC = code volume, FP = functional complexity, UCP = use-case interactions). Pooling heterogeneous schemas for joint training risks semantic feature mismatch. Schema-specific training ensures: (i) feature space compatibility within each schema; (ii) no information leakage from cross-schema correlations; (iii) interpretability for direct paradigm comparison.

Modern DevOps Under representation. Public datasets (1993–2022) are biased toward legacy waterfall/iterative projects. Agile/DevOps environments may exhibit different scaling behaviors. Despite data limitations, our preprocessing pipeline, calibrated baseline methodology, and statistical protocols are dataset-agnostic and directly applicable to future industrial and DevOps corpora.

8 Related Work

8.1 Evolution of Software Effort Estimation Methods

Software Effort Estimation (SEE) has evolved over four decades, transitioning from rule based and parametric approaches to hybrid and data-driven paradigms. The seminal **COCOMO** model by Boehm (1981) established an empirically grounded estimation framework, followed by **Function Points** (Albrecht, 1979) and **Use Case Points** (Karner, 1993), which extended measurement granularity to functional and behavioral complexity. Since the 2000s, studies have introduced early ML models—linear regression, decision trees, neural networks, and SVMs—gradually shifting toward **ensemble learning** and **deep models** (e.g., Random Forest [8], Gradient Boosting [17], and hybrid ensembles [18, 19]). Despite improved accuracy, issues of *reproducibility* and *cross-schema comparability* (LOC, FP, UCP) remain insufficiently explored.

8.2 Comparison of Estimation Paradigms

Figure 11 (top-right) contrasts four major paradigms: (i) traditional parametric, (ii) early ML, (iii) ensemble learning, and (iv) our calibrated baseline approach. Parametric models prioritize interpretability but lack adaptability. Basic ML models improve accuracy yet often lose transparency. Ensemble methods achieve the most balanced trade off between *accuracy*, *adaptability*, and *ease of use*. Our calibrated baseline retains COCOMO's explainable structure while fitting parameters per schema on training data, bridging classical transparency and modern predictive robustness.

8.3 Validity Gaps in Prior Studies

Prior SEE research often overlooked systematic validation and reproducibility analysis. Reviews such as Kitchenham et al. [4] and Foss et al. [5] identify **internal** and **construct validity** as recurring risks, stemming from inconsistent data curation and subjective FP/UCP sizing. Recent studies emphasize transparency and open science [20, 14], yet few works implement explicit unit harmonization or standardized evaluation pipelines.

8.4 Comparison with Prior Work

Table 6 systematically compares representative SEE studies across five dimensions: (i) sizing schema(s), (ii) dataset(s) used, (iii) models evaluated, (iv) evaluation protocol, and (v) reproducibility (code/data availability). While many studies explore ensemble learners to improve predictive accuracy, reproducible cross-schema benchmarking remains challenging due to incomplete provenance reporting, inconsistent baseline handling when cost drivers are unavailable, and unclear aggregation choices that can let LOC-heavy corpora dominate pooled results.
Repro? indicates reproducibility: Yes = public data/code + fixed seeds; Partial = code or data available but incomplete; No = no public artifacts. Our work addresses three methodological gaps: (i) **dataset provenance**—full manifest with DOI/URL, deduplication rules, and rebuild scripts; (ii) **fair baseline**—size-only power-law calibrated on training data per schema; and (iii) **explicit aggregation**—macro-averaged (equal weight per schema) and micro-averaged metrics, with LOOCV for small-sample FP.

Table 6: Comparison with representative SEE studies.

Study	Schema	Models	Eval. Protocol	Repro?
Minku & Yao (2013)	LOC	Bagging, Boosting	CV	Partial
Kocaguneli et al. (2012)	LOC	Analogy	LOOCV	Partial
Pandey et al. (2023)	LOC, FP	RF, XGBoost	5-fold CV	Partial
Alqadi et al. (2021)	LOC	Deep NN	Stratified CV	No
This work	LOC/FP/UCP	LR, DT, RF, GB + baseline	80/20 + LOOCV (FP) + macro/micro	**Yes**

8.5 Research Gap and Contribution

Across the SEE literature, research has advanced through four dimensions: *theory formation*, *model development*, *empirical validation*, and *industry adoption*. While traditional models dominate theoretical grounding and ML excels in model design, few efforts bridge validation with practical deployment. Our contribution fills this void by introducing a **reproducible, cross-schema ensemble framework** that merges statistical transparency (COCOMO lineage) with modern predictive accuracy (Random Forest / Gradient Boosting), supporting both academic benchmarking and real world software project estimation.

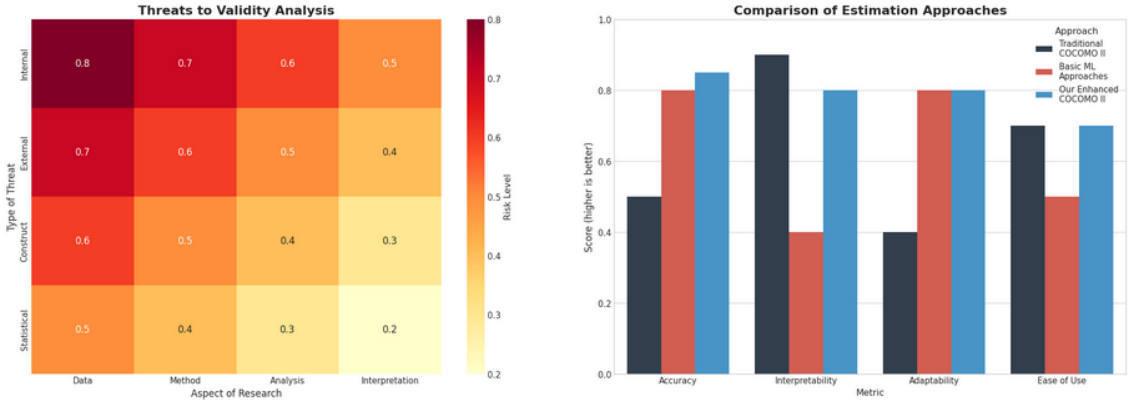

Figure 11: Comprehensive overview of related research. (Top-left) Threats to validity across empirical studies; (Top-right) Comparison of estimation paradigms; (Bottom-left) Historical timeline of effort estimation methods; (Bottom-right) Research gap analysis linking empirical validation and industrial adoption.

9 Conclusion and Reproducibility

Summary of Findings. This study introduced a unified cross-schema framework for software effort estimation, enabling systematic benchmarking across LOC-, FP-, and UCP-based representations. Through harmonized preprocessing and a comprehensive model comparison, ensemble learners—most notably **Random Forest**— demonstrated consistently superior predictive performance relative to classical parametric approaches such as COCOMO II and single-model baselines. The capacity of variance-reducing ensembles to capture non-linear scaling behaviors, while maintaining interpretable variable importance, underscores their suitability for heterogeneous software project data. These results corroborate findings in recent literature on hybrid and ensemble-based effort estimation [21, 18, 19].

Reproducibility Framework. Reproducibility was enforced through standardized data harmonization, deterministic preprocessing pipelines, fixed random seeds, and structured experiment logging. All code,

configurations, and harmonized datasets follow a unified directory layout, allowing deterministic re-execution on commodity hardware without GPU dependencies. This design aligns with recommended best practices in empirical software engineering for conducting transparent and repeatable experimental studies [14, 15]. The unified schema further supports future comparison studies by ensuring consistent feature representations across LOC, FP, and UCP data sources.

Future Directions. Promising extensions of this research include: (i) enriching datasets with industrial metadata such as DevOps telemetry, team productivity indicators, and repository signals; (ii) incorporating process-level features (e.g., issue churn, code volatility); (iii) adopting transfer learning and domain adaptation [22] to enhance cross-organizational robustness; and (iv) deploying ensemble estimators in real project management environments for continuous calibration and real-time forecasting. Such directions will help close the gap between academic research and operational project decision-making.

Strengths. This work provides:

- Auditable dataset manifest with explicit deduplication and rebuild scripts (Table 1, GitHub repository).
- Fair calibrated parametric baseline avoiding straw-man comparisons—size-only power-law fitted per schema on training data.
- Schema-appropriate evaluation protocols: LOOCV for FP ($n = 158$), stratified 80/20 for LOC/UCP, bootstrap CI for robustness.
- Explicit macro/micro aggregation preventing LOC dominance in overall metrics.
- Ablation analysis quantifying preprocessing contributions (Table 4).
- Feature importance analysis with Gini-based rankings (Table 5, Figure 10).

Weaknesses.

- FP schema smaller ($n = 158$, 4 sources) than LOC though LOOCV/bootstrap provide reasonable power; broader industrial corpus (ISBSG) would strengthen claims.
- No cross-schema transfer learning attempted (intentional design choice to avoid semantic mismatch).
- Baseline excludes COCOMO II cost drivers due to data unavailability (represents parametric lower bound).
- Public legacy datasets (1993–2022) may not fully reflect modern DevOps/Agile practices.

Implications. Future SEE papers can adopt our manifest + baseline + aggregation template for defensible reproducible claims. Practically, ensembles (RF/GB) provide robust default estimators when only size signals are available, achieving substantial improvements over calibrated parametric baselines.

Closing Remarks. The unified framework proposed in this study provides a reproducible, transparent, and extensible foundation for cross-schema benchmarking in software effort estimation. By integrating methodological rigor, schema harmonization, and comprehensive evaluation, this work moves toward a *living estimation system*—one that evolves with new telemetry and real-world project dynamics. We hope this framework will support practitioners, researchers, and tool builders in creating more adaptive, evidence-based estimation solutions.

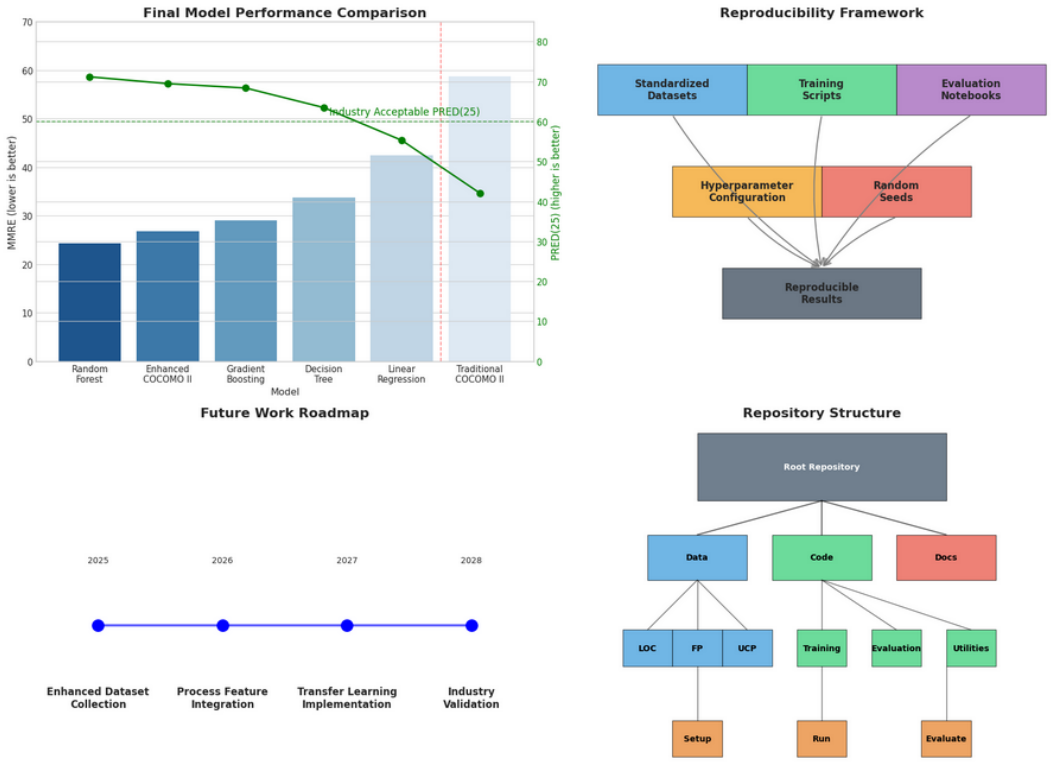

Figure 12: Visual summary of the proposed framework, including model performance comparison, reproducibility pipeline, and potential future extensions.

Data Availability

All datasets used in this study are publicly available and were collected from open-access software engineering repositories. No proprietary or private data were used. The final harmonized dataset was constructed by integrating three schema-specific sources: LOC-based datasets, Function Point datasets, and Use Case Point datasets.
Public sources include:

- **DASE – Data Analysis in Software Engineering**
  `https://github.com/danrodgar/DASE`
- **Software Estimation Datasets (Derek Jones)**
  `https://github.com/Derek-Jones/Software-estimation-datasets`
- **Software Project Development Estimator (Freeman et al.)** `https://github.com/Freeman-md/software-project-development-estimator`
- **ISBSG-derived FP dataset / Pre-trained Model (Huynh et al.)** `https://github.com/huynhhoc/effort-estimation-by-using-pre-trained-model`

Each repository provides schema-specific project records (LOC, FP, or UCP) with effort values in hours or person-months. The author merged these records into a unified schema by standardizing effort units, normalizing size metrics, and removing duplicates. Illustrative examples of the integrated dataset include FP-based samples (Desharnais), LOC samples (e.g., project_id/loc/kloc/effort_pm), and UCP samples (Silhavy et al.).
The harmonized dataset and preprocessing scripts can be obtained from the corresponding author upon reasonable request. All data used in this work are anonymized and contain no personal or sensitive information.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Competing Interests

The authors declare that they have no competing interests.

Ethics Approval and Consent to Participate

This study uses only publicly available, fully anonymized datasets. No human participants or personal data were involved; therefore, ethics approval and formal consent were not required.

Consent for Publication

Not applicable.

Authors' Contributions

Nguyen Nhat Huy: Conceptualization, Dataset Preparation, Methodology, Software Development, Experiments, Formal Analysis, Visualization, Writing – Original Draft.
Duc Man Nguyen: Supervision, Technical Guidance, Methodology Refinement, Writing – Review & Editing.
Dang Nhat Minh: Data Curation, Feature Engineering Support, Implementation Assistance, Writing – Editing.
Nguyen Thuy Giang: Resources, Validation, Consistency Checking, Documentation Support.
P. W. C. Prasad: Senior Supervision, Project Administration, Strategic Direction, Final Approval of the Manuscript.
Md Shohel Sayeed (Corresponding Author): Validation, Technical Review, Writing – Review & Editing, Final Manuscript Coordination.
All authors read and approved the final manuscript.

References

[1] Barry Boehm. *COCOMO II Model Definition Manual*. USC, 2000.

[2] Muhammad Tanveer, Imran Hussain, Naveed Zahid, et al. A survey on machine learning techniques for software effort estimation: Trends, challenges, and opportunities. *Journal of Systems and Software*, 200:111618, 2023.

[3] Mohammad Azzeh and Ali Bou Nassif. Cross-company effort estimation using ensemble learning and feature selection. *Empirical Software Engineering*, 24(6):3821–3848, 2019.

[4] Barbara Kitchenham, Lesley Pickard, Stephen MacDonell, and Martin Shepperd. Evaluating software engineering prediction systems. *Information and Software Technology*, 43(11):733–743, 2001.

[5] Tore Foss, Erik Stensrud, Barbara Kitchenham, and Ivar Myrtveit. A simulation study of the model evaluation criterion mmre. *IEEE Transactions on Software Engineering*, 29(11):985–995, 2003.

[6] Allan J Albrecht and John E Gaffney. Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Transactions on Software Engineering*, (6): 639–648, 1983.

[7] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[8] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[9] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

[10] Gary Macbeth, Esteban Razumiejczyk, and Rubén Ledesma. Cliff's delta calculator: A non-parametric effect size program for two groups of observations. *Universitas Psychologica*, 10 (2):545–555, 2011.

[11] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[12] Salvador Garcia, Alberto Fernandez, Jesus Luengo, and Francisco Herrera. Advanced nonparametric tests for multiple comparisons in computational intelligence and data mining. *Information Sciences*, 180(10):2044–2064, 2010.

[13] Fabian Pedregosa et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] Luis Cruz and Rui Abreu. Open science in software engineering research: The case for open data and replication. *Empirical Software Engineering*, 24(6):3829–3849, 2019.

[15] Belen Lopez, Juan C Rodriguez, and Salvador Garcia. Empirical software engineering reproducibility: A systematic review. *Information and Software Technology*, 136:106579, 2021.

[16] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[17] Jerome H Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[18] P. Pandey, T. Sharma, and S. Saha. Hybrid ensemble learning for software effort estimation using meta-heuristic optimization. *Applied Soft Computing*, 135:110054, 2023.

[19] A. Alqadi and A. Abran. Deep learning models for software effort estimation: An empirical study. *IEEE Access*, 9:135012–135026, 2021.

[20] Vineeth Nair and Tim Menzies. Open problems in reproducibility, replication, and transparency in software engineering. In *Proceedings of the 42nd International Conference on Software Engineering: New Ideas and Emerging Results*, pages 1–4. IEEE, 2020.

[21] Muhammad Tanveer, Imran Hussain, Naveed Zahid, et al. A comprehensive analysis of ensemble learning models for software effort estimation. *IEEE Access*, 11:76590–76608, 2023.

[22] Yong Yu, Xin Xia, David Lo, and Ahmed E Hassan. Transfer learning in software engineering: A systematic mapping study. *Empirical Software Engineering*, 26(3):1–46, 2021.