IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 June 2025, accepted 28 June 2025, date of publication 4 July 2025, date of current version 10 July 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3586081

RESEARCH ARTICLE

Efficient Software Development Effort Estimation Approaches for Improving Scalability in the Training Phase

HO LE THI KIM NHUNG, PETR SILHAVY, AND RADEK SILHAVY
Faculty of Applied Informatics, Tomas Bata University in Zlín, 760 01 Zlín, Czech Republic
Corresponding author: Radek Silhavy (rsilhavy@utb.cz)

This work was supported by the Faculty of Applied Informatics, Tomas Bata University, Zlín, under Project RO30246061025/2102.

ABSTRACT Effective software effort estimation is essential for project management, but faces scalability challenges with large datasets. While clustering can address this complexity, standard methods often rely on random initial centers, leading to inconsistent and less precise results. This randomness frequently overlooks critical contextual factors, such as industry or domain-specific characteristics, which can impair cluster quality and the accuracy of effort predictions. To overcome these issues, this study introduces the Contextual Initial Cluster Centroids (CICC), a novel methodology designed to optimize initial centroid selection. Unlike approaches that depend on randomness or are computationally intensive, CICC uses parallel processing of Jaccard similarity and a refined neighbor-finding technique, K-Reciprocal Nearest Neighbors (KRNN), to identify the most relevant and similar projects as initial centers. This deterministic approach ensures clusters are built around meaningful, context-rich representatives, reducing computations and improving scalability. Experiments on public software project datasets show that CICC significantly outperforms existing techniques. It achieves higher cluster quality, measured by metrics like the Global Silhouette Index, and provides more accurate effort estimates, indicated by lower MAE and higher PRED values. By establishing a more robust and efficient foundation for effort estimation, CICC offers considerable potential to optimize project planning and resource allocation in large-scale software development.

INDEX TERMS Neighbor-based collaborative filtering, similarity measures, software development effort estimation, system scalability.

I. INTRODUCTION

Accurate estimation of the effort required for software development, referred to as Software Development Effort Estimation (SDEE), is essential for effectively managing software projects. This estimation process assesses the workload needed to complete a project, typically measured in person-hours or person-months, providing critical insights into budget requirements and anticipated completion timelines [1], [2]. The intricacy of this task is influenced by various factors, such as project scope, the complexity of requirements, the experience of the team, and the technologies employed [3], [4]. Due to these variables, achieving accurate estimation is a significant challenge, yet it plays a pivotal role in determining the success of the software development life cycle (SDLC).

The software engineering community has focused on developing more sophisticated predictive models to address these estimation challenges. In recent years, many approaches integrating machine learning, especially those leveraging historical project data, have shown significant promise in improving estimation accuracy [5], [6]. These efforts include diverse strategies such as leveraging advanced textual embeddings, sophisticated ensemble models, and novel feature integrations [7], [8], [9]. While these advanced strategies represent important directions in SDEE, a foundational and compelling approach remains similarity-based estimation, especially when rich historical project data is available.

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Liu.

The core rationale for this approach is that the effort required for a new project can be reliably predicted by examining the actual effort from completed projects with similar characteristics, such as requirements, scope, or technical artifacts. This principle is effectively implemented using techniques like collaborative filtering (a form of neighbor-based effort estimation, as detailed in Section II), which systematically analyzes historical data to identify analogous projects [10]. This approach provides a robust foundation for more accurate and data-driven effort estimation by aggregating effort data from a group of carefully selected similar projects.

A. PROBLEM FORMULATION

Accurately identifying these similar projects is critical to improving the accuracy of the estimation as it directly affects the quality of the estimation process. This difficulty is evident during the training phase when many pairwise comparisons cause scalability and performance limitations. Improving the system's scalability is critical for addressing this issue. A very effective solution is the cluster of projects, which limits the similarity computations to individual clusters and thus reduces the overall computational load [11], [12]. The method effectively reduces the number of necessary pairwise comparisons by utilizing clustering of projects, thus improving systems' scalability for evaluating software effort.

However, the success of clustering strongly depends on the quality of the process. Inadequately constructed clusters can reduce the accuracy of future estimates [13]. Although previous research on SDEE has focused mainly on constructing project representation vectors for cluster formation, less attention has been paid to selecting suitable projects to initialize the cluster formation process [14], [15], [16], [17]. Standard practices, such as random selection of initial centers in the technique, often lead to non-deterministic and inconsistent estimation results [18]. This randomness can lead to suboptimal groupings that must capture important project characteristics, such as industry or domain-specific characteristics. Consequently, poor clustering of results can affect the accuracy of effort estimates.

B. OBJECTIVES OF THE STUDY

This paper aims to enhance the accuracy of effort estimation using the neighbor-based collaborative filtering method. Specifically, our contributions are as follows:

- We introduce a method for selecting key projects to serve as the initial centroids for clustering, termed Contextual Initial Cluster Centroids (CICC). A significant advantage of our approach is that the project selection process is grounded in two fundamental pieces of information: the industry sector and the programming language. This focused methodology ensures high flexibility and accuracy in the system, facilitating scalability improvements during the SDEE training phase.
- In addition to reducing the number of project pairs that require similarity computation through clustering, we aim to decrease computation time further using parallel computing platforms to address inefficiencies in neighbor-based collaborative filtering for effort estimation. We significantly speed up these computations by minimizing the number of project pairs needing similarity calculations and employing parallel processing. Specifically, we redesigned the Jaccard similarity metric to considerably reduce calculation time during the training stage.

We conducted experiments to compare our initialization method for project clustering in effort estimation systems with the random initialization method and eight other relevant initialization methods in previous studies. These experiments are performed on two standard datasets. The results indicate that CICC, based on industry sectors and programming languages, enhances project clustering quality and improves system scalability by reducing the number of project pairs requiring similarity computations during the training phase. Statistical tests reinforce these findings.

TABLE 1. The symbols used in the proposed approaches.

Notation	Description
$\mathcal{P}$	Set of projects $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$
$\text{sim}(p, g)$	Similarity measure between project p and project g
I_p, I_g	Sets of attributes for projects p and g
$\mathcal{N}_p^k$	Set of the top k projects that are similar to project p
k	Number of selected neighbors
m	Number of observations
$\bar{p}, \bar{g}$	Mean values of attributes for projects p and g
$\hat{s}_p$	The estimated project size of p
s_g	The observed project size of g
$w_{(p,g)}$	The weight indicating the influence of neighbor g on p
$s(x_i)$	The silhouette coefficient for x_i
$a(x_i)$	Average distance from x_i to other points in its cluster
$b(x_i)$	Min average distance from x_i to points in neighboring clusters
$N_{negative}$	The number of data points with $s(x_i) < 0$
$N_{positive}$	The number of data points with $s(x_i) \geq 0$
Y	The Jaccard similarity matrix

C. RESEARCH QUESTIONS AND HYPOTHESES

In light of the objectives outlined above, the research question that needs to be answered:

- **RQ1: Compared to other initialization techniques across different datasets, how does the CICC method affect clustering quality when evaluated using validity measures such as the Global Silhouette Index (GSI) and the Silhouette Validity Ratio (SVR)?**
  This question aims to assess the impact of the CICC method on clustering effectiveness and its ability to minimize misclassifications, particularly in comparison to existing methods used for initialization in SDEE.
- **RQ2: How does the CICC method perform in estimation accuracy compared to existing methods?**

This question investigates the influence of the centroid initialization method on effort estimation accuracy, clarifying whether improvements in clustering quality result in enhanced estimation performance.

Based on these RQs, a t-test was conducted to evaluate the estimation errors, testing the null hypothesis that the mean estimation errors μ_{CICC} and μ_{other} are equal. Statistical analysis confirms that our approach significantly improves scalability and accuracy in software effort estimation. The following statistical hypotheses were tested:

- H_0: $\mu_{CICC} = \mu_{other}$ – There is no difference in estimation ability between CICC and existing approaches; the mean estimation error values are the same.
- H_1: $\mu_{CICC} < \mu_{other}$ – There is a difference in estimation capability between CICC and existing approaches, with statistically significant evidence suggesting that CICC provides better accuracy and lower estimation error than the existing methods.
- H_2: $\mu_{CICC} > \mu_{other}$ – There is a difference in estimation capability between CICC and existing approaches, with statistically significant evidence indicating that existing methods provide better accuracy and lower estimation error than CICC.

D. PAPER ORGANIZATION

The rest of the paper is structured as follows. Section II provides background on memory-based collaborative filtering for effort estimation. Section III discusses related work on enhancing scalability through project clustering approaches in SDEE and variants of K-means addressing initialization problems. Section IV summarizes an idea for improving these methods. Section V details the proposed approaches. Section VI presents the experiments conducted. Section VII provides concluding remarks. In these final sections, we address potential threats to validity and outline directions for future improvements. Table 1 lists the notations used in this study.

II. BACKGROUND

A. MEMORY-BASED COLLABORATIVE FILTERING FOR EFFORT ESTIMATION

Memory-based Collaborative Filtering, also known as neighbor-based collaborative filtering (NCF) for effort estimation, is a widely used technique for estimating unknown outcomes by exploiting the patterns and similarities found in existing data [19], [20]. In the context of SDEE, NCF utilizes historical project data to predict the effort required for new projects. By identifying and comparing similar projects using various criteria, NCF produces accurate effort estimates and enhances project management and resource allocation - an essential aspect of software development [21], [22]. This estimation process is typically applied during the early phases of SDLC, such as project planning, feasibility analysis, or initial resource allocation, where early insights into potential effort are crucial.

The methodological approach is divided into three steps, in which historical project data is used to estimate the unknown effort for new software projects using collaborative filtering. These steps include the calculation of project similarities, the selection of neighbors, and the aggregation of effort estimates.

- **Training phase**: The primary objective is to compute the similarities between each pair of projects within the dataset. A similarity measure is defined to assess the similarity between a target project p and each project $g \in \mathcal{P}$ (where $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$), denoted as $\text{sim}_(p, g)$ for $g = 1, \ldots, m$. This measure is applied to determine how similar two projects are based on their feature vectors.
- **Estimation phase**: In this phase, the most similar projects are selected from the dataset to create a set of neighbors for the target project p. This selection process is driven by similarity scores, ensuring that the selected neighbor projects closely match p in terms of their characteristics. To estimate the project size of p, the top k neighbors with the most similar characteristics to p are determined, denoted as N_p^k. This procedure evaluates the similarity between the project p and each other project g. The sizes of the neighbors in N_p^k are then aggregated to estimate the project size of p. Figure 1 illustrates the estimation of the size of the project p. Due to the similarities of the project pairs, the neighbor set size ($k = 2$) for p_1 includes p_2 and p_4, resulting in $N_{p_1}^2 = \{p_2; p_4\}$. Consequently, the size of p_1 is calculated as the aggregation of the sizes of p_2 and p_4. These feature vectors are composed of quantifiable project attributes detailed for our study in Section VI-A.

1) PROJECT SIMILARITY COMPUTATION

The accuracy of similarity measures between software projects is crucial for accurate effort estimation based on neighbors. For this reason, selecting appropriate similarity measures impacts the effectiveness of neighbor sets, which are essential for extracting insights from historical data [23], [24]. A general approach to measuring the similarity between two projects is to focus only on the features with which both projects have interacted. For example, the Pearson Correlation Coefficient (PCC), the Mean Square Deviation (MSD), and the Jaccard index (JAC) are often used to assess the similarity of projects.

The PCC [11] (1) quantifies the linear correlation between the attribute values of two projects and is therefore suitable for evaluating the linear relationship between project attributes. The MSD [25] (2) assesses dissimilarity by calculating the squared differences between the corresponding characteristics of two projects. It thus provides a measure of dissimilarity based on attribute values. The JAC [26] (3) measures similarity by comparing the intersection and union of project attributes. It proves to be more effective than PCC and MSD in terms of definite or less precise attribute

Compute the similarity of each project pair

sim	p_1	p_2	p_3	p_4
p_1	1	0.652	0.168	0.548
p_2	0.652	1	0.660	0.126
p_3	0.168	0.660	1	0.168
p_4	0.548	0.126	0.168	1

Training phase

Estimating unknow size: $\hat{s}_{p_1}$

- The top ($k = 2$) neighbors that have the most similar interests to p_1: $\mathbb{N}_{p_1} = \{p_2; p_4\}$
- Aggregate the neighbors' sizes: s_{p_2} and $s_{p_4} \rightarrow \hat{s}_{p_1}$

Prediction phase

FIGURE 1. Training and estimating phases in memory-based collaborative filtering for SDEE.

data.

$$\mathbf{sim}_{(p,g)} = \mathbf{PCC}_{(p,g)} = \frac{\sum_{i=1}^{n}(p_i - \bar{p})(g_i - \bar{g})}{\sqrt{\sum_{i=1}^{n}(p_i - \bar{p})^2}\sqrt{\sum_{i=1}^{n}(g_i - \bar{g})^2}} \tag{1}$$

$$\mathbf{sim}_{(p,g)} = \mathbf{MSD}_{(p,g)} = \frac{1}{n}\sum_{i=1}^{n}(p_i - g_i)^2 \tag{2}$$

$$\mathbf{sim}_{(p,g)} = \mathbf{JAC}_{(p,g)} = \frac{|I_p \cap I_g|}{|I_p \cup I_g|} \tag{3}$$

where p_i and g_i are attribute values of project p and g, $\bar{p}$ and $\bar{g}$ are their respective means, n is the number of attributes, and I_p, I_g are sets of attributes of project p and g.

Several studies have proposed improved versions of these metrics. One notable approach is the integration of multiple similarity measures. For example, one study combined MSD with JAC to develop a metric for quantity (i.e., JAC) and attributes (i.e., MSD). Traditional methods for measuring similarity typically focus on the two projects being compared. However, the characteristics of a project can also be illuminated by its relationship to other projects. Therefore, some studies have refined the existing similarity measures. For example, a proposed method enhances the similarity score between two projects if each is included in the other's neighbor set, thereby reflecting their reciprocal relationship [27]. Based on the similarity within each project's neighbor groups, [28] states that the similarity between two projects is ascertained. In [29], the similarity between two projects is adjusted by aggregating their similarities with a third project.

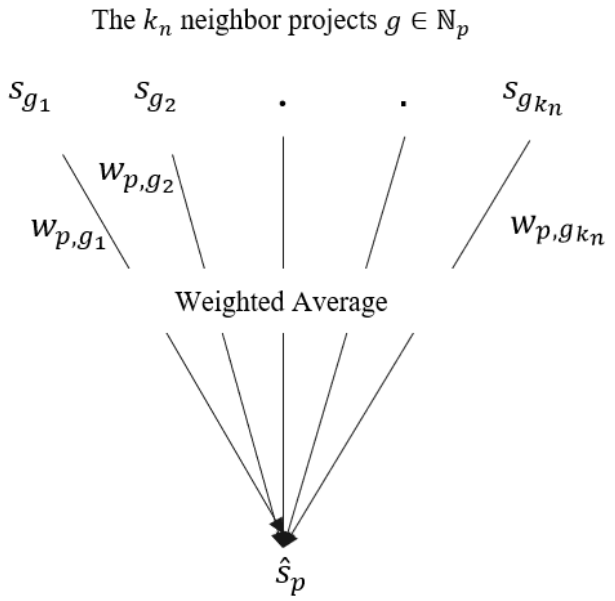

FIGURE 2. Project size estimation for p based on the weighted average of sizes from neighboring projects in set N_p^k.

2) NEIGHBOR PROJECT SIZE AGGREGATION

To determine the size estimate for the project p, a set of neighbors N_p^k consisting of k projects is selected based on their similarity to p. As shown in Figure 2, the project sizes of these neighboring projects are aggregated using a weighted average function, as follows:

$$\hat{s}_p = \sum_{g \in N_p^k} w_{(p,g)} \cdot s_g \tag{4}$$

where $\hat{s}_p$ represents the estimated project size of p, s_g is the observed size of the neighbor $g \in N_p^k$, and $w_{(p,g)}$ is the weight indicating the influence of neighbor g on project p.

Some research proposed that the weighting should be based on how similar a neighbor is to the target project. The greater the similarity, the more reliably its attributes reflect those of the target project. Consequently, the weighting between a target project p and a neighbor g corresponds to their similarity $sim_{(p,g)}$. This method ensures that the estimate reflects the relative influence of each related project [22], [23], [29], as follows:

$$\hat{s}_p = \frac{\sum_{g \in N_p^k} \mathrm{sim}_{(p,g)} \cdot s_g}{\sum_{g \in N_p^k} \mathrm{sim}_{(p,g)}} \tag{5}$$

Despite its effectiveness in producing accurate estimates, its lack of scalability limits the foundational NCF approach. The main bottleneck is the similarity computation phase,

which involves comparing every project with every other project in the dataset. In a system with m projects, this results in a computational complexity of at least $O(m^2 \cdot f)$, where f represents the minimum number of standard metrics or attributes between two projects. As m increases by thousands or more, this quadratic complexity increases dramatically, leading to prohibitively long processing times and resource constraints that make the method impractical for large-scale industrial datasets [30]. To address this challenge, the research community has focused on solutions such as project clustering to reduce computational load, as discussed in section III-A.

B. STEPWISE REGRESSION

Stepwise regression (StepR) [13], [31], [32], [33] was utilized in the size estimation experiments to identify the optimal combination of independent variables that significantly influence the dependent variable for inclusion in a predictive model. The StepR approach can be summarized as follows:

1) Construct an initial model: Specify the variables to be included in the initial model.
2) Define the model constraints: Establish the desired level of complexity for the final model, including options for linear, quadratic, and interaction terms.
3) Set a control threshold: Determine the criteria for adding or removing variables during the modeling process.
4) Evaluate the model: Reassess the model after each variable addition or removal to gauge its performance.
5) Iterate until convergence: Continue the StepR process until no further improvements in model fit are observed.

StepR generates multiple models, each representing a multiple linear regression based on independent variables. The algebraic form is represented as follows:

$$y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_m X_{im} + \epsilon_i \quad (6)$$

In this equation, y_i represents the dependent variable, while $X_{i1}, X_{i2}, \ldots, X_{im}$ denote the predictor variables. The coefficients $\beta_1, \ldots, \beta_m$, represent the regression coefficients, with β_0 as the intercept and ϵ_i as the residual term.

III. RELATED WORK

A. ENHANCING SCALABILITY: PROJECT CLUSTERING APPROACHES IN SDEE

The scalability challenge inherent in NCF arises from its computationally expensive pairwise similarity calculations, which have led the research community to explore clustering-based approaches [12]. The core idea is to partition the dataset into smaller groups, or clusters, so that similarity calculations are confined only to projects within the same cluster, as illustrated in Figure 3. This reduces the complexity from $O(m^2 \cdot f)$ to approximately $O(p^2 \cdot f)$, where the cluster size p is substantially smaller than the total number of projects m. The effectiveness of this approach hinges on the quality of these clusters, so for the final effort estimation to be accurate, projects within a cluster must be genuinely similar. Consequently, a significant body of research has aimed to apply and enhance various clustering methodologies within the SDEE process.

Various clustering methodologies have been explored to boost estimation accuracy. For instance, combining fuzzy clustering and analogy methods has demonstrated improved estimation accuracy [34]. In addition, challenges associated with analogy-based estimation have been addressed through general fuzzy C-means approaches that incorporate various factors [35]. Idri et al. [36] reviewed over 60 papers published between 1990 and 2012, highlighting efforts to eliminate irrelevant data points, often utilizing clustering techniques to refine datasets and discern subgroups. Azzeh et al. [37] addressed the challenge of identifying nearby projects by bisecting k-medoids.

Moreover, advanced techniques such as hybrid models combining support vector machines (SVM) and radial basis neural networks (RBNN) have been developed to enhance classification and estimation [38]. Applying clustering for equation generation and evaluating improved expectation maximization (EM) methods has yielded promising results [39]. In [15], the authors asserted that K-means typically outperforms hierarchical approaches, underscoring the significance of selecting an appropriate clustering type and distance metric for optimal results. In [40], the authors also utilized K-means, in conjunction with k-nearest neighbors, for fabric defect detection, further demonstrating the utility of these techniques in diverse pattern recognition tasks.

Clustering facilitates the evaluation of project attributes based on similarity, enabling effective comparisons between new projects and those in similar clusters. Various strategies, including moving windows and particle swarm optimization algorithms, have demonstrated the utility of clustering in effort estimation [41]. The effectiveness of moving windows in subset selection and the relevance of past projects with similar completion times for model generation have been highlighted [42]. Research in [43] suggests that threshold clustering can positively influence estimation accuracy, while a categorical variable segmentation (CVS) model [31] has been proposed for segmenting datasets based on relative project size. Furthermore, [44] indicates that incorporating product delivery rates as an estimation parameter can enhance the accuracy of software delivery models.

However, previous research on SDEE has predominantly focused on developing methods for constructing representation vectors of projects for clustering. These studies aim to group projects with similar characteristics to facilitate accurate effort estimation. Despite significant progress in SDEE, a critical aspect has been largely overlooked: the systematic selection of appropriate projects to initialize the clustering process. Most existing methods rely on the random selection of projects as the initial centroids in the clustering algorithm. While this approach simplifies implementation, it introduces non-deterministic outcomes, as the choice of initial centroids heavily influences the final clusters formed. Consequently, this variability in centroid selection often leads to inconsistent estimation results, limiting these

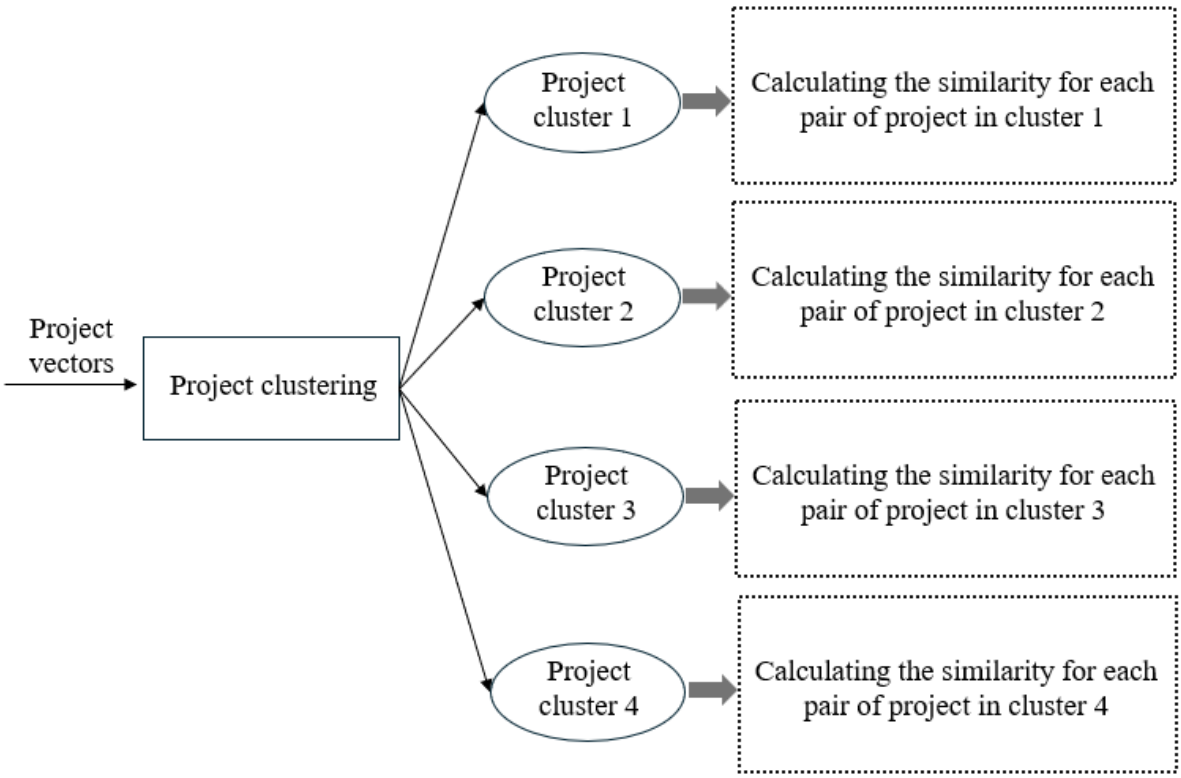

FIGURE 3. Applying project clustering in the training phase.

methods' practical applicability in SDEE. Addressing this gap necessitates approaches to selecting initial centroids, ensuring that clustering methods produce consistent and meaningful project groupings.

The following subsection reviews the literature on initialization processes for selecting initial cluster centers.

B. VARIANTS OF K-MEANS ADDRESSING INITIALIZATION PROBLEMS

The goal of our study stems from the need to enhance the scalability of memory-based collaborative filtering within software development effort estimation. To achieve this, we employed K-means for project clustering. K-means remains popular due to its simplicity, efficiency, and well-understood limitations, making it a practical option for numerous applications [45], [46]. Additionally, its local fine-tuning capability makes it highly effective and adaptable, often serving as a core component of more advanced clustering algorithms [47], [48]. However, K-means is notably sensitive to the choice of initial cluster centroids, which can significantly influence the quality of clustering results [18]. Random selection of initial centroids often leads to inconsistent cluster structures, suboptimal solutions, and increased iterations, highlighting the importance of addressing this initialization challenge [49].

Recognizing the impact of initialization on clustering outcomes, researchers have proposed numerous strategies to improve the initial selection of centroids for K-means. These strategies aim to achieve better starting points for clustering, leading to more stable and efficient outcomes. For example, the cluster centroid initialization algorithm (CCIA) selects the initial centers based on the similarity of the data samples. This method ensures that the chosen centers represent the data distribution, which improves the quality of clustering [50]. The initial cluster centroids are based on the randomized seeding technique (ICCRS), which probabilistically selects initial centroids based on the data distribution. The competitive algorithm $\Phi(\log k)$ improves the speed and accuracy of K-means clustering by optimizing centroid placement from the outset [51]. Arai and Barakbah [52] proposed a hybrid approach, initial cluster centroids based on hybrid K-means and hierarchical clustering (ICCHK), which combines K-means with hierarchical clustering. This method first applies K-means with random centroids, and the resulting clusters are refined using hierarchical techniques. This approach is particularly practical for datasets with high dimensionality or many clusters.

The method initial cluster centroids based on the cohesion and rough set neighborhood (ICCRN) [53] calculates the centroids by analyzing the cohesion and coupling degrees within the neighborhoods of data objects. This approach improves the clustering performance and ensures the formation of more cohesive cluster structures. Similarly, the initial cluster centroids based on the axis and coefficient of

Deviation (ICCACD) approach identify primary axes with minimal correlation but high deviation coefficients. Centroids are then computed along these axes, enabling distinct cluster formation, particularly in datasets with complex attribute relationships [18]. Initial cluster centroids based on data normalization (ICCDN) address datasets with mixed positive and negative attribute values by normalizing data objects and calculating distances from a common origin to determine centroids. This method is highly effective in ensuring consistency across diverse datasets [54]. Initial cluster centroids based on binary search (ICCBS) iteratively determine attribute minima and maxima through binary search, optimizing centroid selection to ensure centroids are well-distributed and aligned with the dataset's attribute ranges [55].

The initial cluster centroids based on the hybrid distance (ICCHD) method combine Euclidean distance with density-based metrics, prioritizing dense regions of the dataset for centroid selection. This approach enhances clustering performance by focusing on regions with high data density [56]. Additionally, weighted k-means assigns weights to centroids based on cluster variance, ensuring that clusters of varying sizes are treated appropriately. This technique enhances the robustness of K-means clustering, particularly for datasets with non-uniform cluster distributions [57]. Li [58] proposed a proximity-based method that leverages nearest neighbor pair assumptions to initialize centroids in two-cluster datasets, ensuring better alignment of centroids with the data distribution and improving clustering outcomes.

The variety of initialization strategies reviewed above underscores the critical importance of selecting effective initial centroids in K-means clustering. However, our literature review, particularly in the SDEE context, reveals two primary gaps that motivate our research. First, most existing applications rely on methods sensitive to random initialization, which can lead to inconsistent and unreliable clustering outcomes. Second, these conventional approaches often fail to account for rich contextual information; they typically overlook domain-specific features, such as a project's industry sector or programming language, that could guide a more meaningful and deterministic clustering process, highlighting the need for more refined strategies.

IV. MOTIVATION

Our research is directly motivated by the need to address the gaps identified in the related work, specifically the reliance on non-deterministic random initialization and the failure of existing methods to leverage rich contextual data for clustering in SDEE. To address these challenges, we were driven to develop a new technique that fundamentally improves how initial cluster centroids are determined.

The core of our improvement centers on moving away from the random initialization standard and instead developing a deterministic, context-aware approach. We argue that leveraging two critical data characteristics, industry sector and programming language, the selected initial centroids will be more relevant and representative of the problem domain. These characteristics are particularly valuable in software, offering essential insights into a project's nature that are often overlooked [17], [31], [59], [60]. This approach is designed to produce more stable and meaningful clusters, enhancing the precision of effort estimation.

Secondly, we also address the computational overhead of similarity measures. We introduce a design for the parallel processing of the Jaccard similarity metric, aiming to streamline computations further and enhance processing speed within the clustering framework. In Section V, we will detail the full methodology that arises from these motivations.

V. OUR PROPOSED METHODS

A. CONTEXTUAL INITIAL CLUSTER CENTROIDS

The proposed CICC methodology offers a distinct, structured approach to address common limitations in centroid initialization, such as reliance on random selection or overlooking crucial contextual information. Unlike conventional methods that apply statistical measures globally across the entire dataset, refine initially random centroids without specific contextual grounding, or lack dedicated mechanisms to prioritize key projects within distinct operational contexts, CICC's novelty lies in its structured, context-driven process. This process has two main stages:

- Contextual pre-segmentation: CICC's foundational step is partitioning the dataset into homogenous subgroups. This is achieved by leveraging key domain-specific contextual attributes that characterize the projects.
- Focused analysis *within* subgroups: Deterministic analysis, including similarity assessment and spectral methods, is performed *within* these contextually relevant subgroups to identify representative projects that inform initial centroid placement.

The CICC's context-first strategy thus generates initial centroids that are more aligned with the data's intrinsic structure, leading to more meaningful clusters and more accurate effort estimations. Let $\mathcal{P}$ represent the dataset of m data objects. We define subsets of $\mathcal{P}$ corresponding to each unique combination of industry sector (IS) and programming language (PL). This is formally expressed as:

$$\mathcal{P}_{ij} = \{x \in \mathcal{P} \mid x[\text{IS}] = \text{IS}_i \wedge x[\text{PL}] = \text{PL}_j\} \tag{7}$$

where, $\mathcal{P}_{ij}$ represents the subset of projects that belong to a specific IS_i and PL_j. For each $\mathcal{P}_{ij}$, we calculate the Jaccard similarity matrix Y to quatify the relationship between projects.

$$Y_{p,g} = JAC_(p, g) \tag{8}$$

In the subsequent step, we perform spectral analysis of the similarity matrix Y, more precisely, of the eigenvalues and eigenvectors from solving the characteristic polynomial:

$$\det(Y - \lambda I) = 0 \tag{9}$$

where λ represents the eigenvalues, and I denotes the identity matrix. The resulting eigenvalues indicate the variance

captured by the corresponding eigenvectors. Among the calculated eigenvalues, we focus on identifying the largest eigenvalue $\lambda_{\max}$ and its corresponding eigenvector $v_{\max}$. The eigenvector corresponding to $\lambda_{\max}$ illustrates the primary direction of the variance in the dataset:

$$Yv_{\max} = \lambda_{\max} v_{\max} \tag{10}$$

A key component of the CICC method is the calculation of the reference index $R(i)$, which quantitatively assesses the importance of each project within its specific context:

$$R(i) = N(i) \times D(i) \tag{11}$$

where $N(i)$ is the normalized squared component of the dominant direction vector $v_{\max}$, $D(i)$ is derived from the mean distances to the k nearest neighbours. The density of each project is effectively represented as the inverse of this mean distance. This formulation provides a measure of the local density of each project, where a smaller mean distance indicates a higher density of neighboring projects.

$$D(i) = \frac{1}{\frac{1}{k}\sum_{j=1}^{k} d(p, g)} \tag{12}$$

The $R(i)$ plays an important role in the clustering process, ensuring that projects with significant contextual relevance are prioritized. The projects are subsequently sorted according to their $R(i)$ values. The dataset is then divided into k clusters according to these sorted reference indices, ensuring each cluster contains projects exhibiting substantial contextual similarities. For each cluster C_j, the initial centroids μ_j are calculated based on the characteristics of the projects within that cluster. The centroids are derived as the weighted average of the project attributes, taking into account the density of the projects, which is assessed using the reference index and the distances to the most important neighbours. This approach ensures that the centroids represent the distribution of projects in their respective clusters. The algorithm 1 outlines the initialization of cluster centroids in K-means clustering according to the CICC approach.

Algorithm 1 Initialization of CICC

Input:

- $\mathcal{P}$: a dataset containing m data objects
- k: the number of clusters

Output:

- C: a set of clusters obtained from $\mathcal{P}$
- initial centroids (μ_j) for each cluster

procedure CalculateCICC($\mathcal{P}$, k)
    1. Identify unique combinations:
        • Define the subset $\mathcal{P}_{ij}$ using (7)
        • If $|\mathcal{P}_{ij}| < k$, continue to the next combination
    2. Compute similarity matrix and variance analysis:
        For each subset $\mathcal{P}_{ij}$:
            • Construct similarity matrix Y using (8)
            • Calculate eigenvalues and eigenvectors from matrix Y using (9)
            • Identify the eigenvector $v_{\max}$ associated with the largest eigenvalue $\lambda_{\max}$ using (10)
    3. Calculate reference index $R(i)$:
        For each project x_i in X_{ij}, compute $R(i)$ using (11).
    4. Calculate project density:
        For each subset $\mathcal{P}_{ij}$, calculate $D(i)$ using (12)
    5. Sort and partition data:
        Order projects according to $R(i)$ and segment the dataset into k clusters
    6. Compute initial centroids:
        For each partition C_j, compute centroid μ_j
    7. Aggregate and select final centroids:
        Compile centroids across all partitions and select k centroids for subsequent similarity calculations
    Return C, μ_j
end procedure

B. IMPROVING THE SCALABILITY

The proposal presented in Section V-A has significantly enhanced the quality of project clustering, ultimately reducing the number of project pairs that need to be considered in system training for similarity computation. This section investigates the Jaccard similarity formula to further reduce training time by implementing parallel computation mechanisms.

The Jaccard similarity between two projects (p) and (g) can be mathematically expressed as follows:

$$\begin{aligned} \mathrm{JAC}(p, g) &= \frac{h_{p,g}}{f_{p,g} - h_{p,g}} \\ h_{p,g} &= |I_p \cap I_g| \\ f_{p,g} &= |I_p| + |I_g| \end{aligned} \tag{13}$$

To compute Jaccard similarity efficiently, we implemented parallel processing. This approach enables simultaneous computations for multiple project pairs, significantly optimizing performance. The process is executed as follows: For each cluster C_i, the Jaccard similarity $\mathrm{JAC}_{(p,g)}$ between all unique pairs of projects (p, g) is computed. These pairwise calculations were performed concurrently by leveraging Python's multiprocessing library, which distributes the workload across available CPU cores.

Once the similarity scores are computed, we apply an adaptive thresholding mechanism. The threshold t is derived from the distribution of Jaccard similarity scores. By retaining only those project pairs with $JAC_(p, g) > t$, we filter out less significant pairs that could hinder performance and obscure valuable insights. The detailed process of parallel similarity calculation for Jaccard similarity is present in algorithm 2.

C. ADVANCED NEIGHBOR SELECTION

To enhance the initialization of centroids for K-means clustering, we employed the KRNN method [27].

Algorithm 2 Parallel Similarity Calculation for Jaccard Similarity

Input:

- $\mathcal{P}$: a dataset containing m data objects
- C: a set of clusters obtained from $\mathcal{P}$

Output:

- A list of project pairs with their Jaccard similarity scores

procedure CalculateJaccardSimilarity($\mathcal{P}$, C)
    1. Initialize results: Results $\leftarrow \emptyset$
    2. For each cluster $C_i \in C$ do:
        For each unique pair (p, g) in C_i:
            • Calculate $JAC(p, g)$ using (13)
            • Temp $\leftarrow$ Temp $\cup$ $\{(p, g, JAC(p, g))\}$
        Apply adaptive thresholding:
            • Calculate threshold $t = \text{Percentile}_{75}(\text{Temp})$
                For each entry in Temp:
                    If $JAC(p, g) > t$:
                    Results $\leftarrow$ Results $\cup$ $\{(p, g, JAC(p, g))\}$
    3. Return Results
end procedure

This strategy significantly improves clustering outcomes by ensuring mutual recognition among neighbors. Specifically, if a target project p considers another project g as a neighbor, then project g must also recognize project p as one of its neighbors. This mutual acknowledgment reinforces the selection of initial neighbors based on Jaccard similarity values.

First, the nearest neighbors for the target project p are determined by calculating their Jaccard similarity values with all projects g that belong to the set $\mathcal{P} = p_1, p_2, \ldots, p_m$. After selecting the initial nearest neighbors based on Jaccard similarity, the KRNN mechanism refines this selection by checking whether each identified neighbor g includes the target project p in its top k nearest neighbors. For each project g recognized as a neighbor, the following process is implemented:

1) Initial neighbor calculation: Determine the initial set of neighbors for the project p based on the calculated Jaccard similarities.
2) Reciprocal neighbor verification: For each neighbor g, assess whether the project p is included in g's top k nearest neighbors. If this condition holds, g is confirmed as a valid neighbor of p.
3) Handling insufficient neighbors: When fewer than k neighbors are identified through the KRNN process, the initially identified nearest neighbors are retained to ensure the requisite count is met.

This approach generates a list of neighbors characterized by significant Jaccard similarity and mutual recognition, thereby improving the neighborhood's integrity surrounding the target project p. By confirming these mutual relationships, the CICC methodology enhances the initialization of centroids for K-means clustering and increases the accuracy of subsequent analyses and estimates. Furthermore, the KRNN refinement allows for dynamic adjustments in neighbor selection based on the dataset's distribution, enabling the algorithm to remain robust across varying contexts.

VI. EXPERIMENTAL DESIGN

A. DATASETS AND DATA PREPARATION

The practical applicability of our proposed methods is evaluated using recognized benchmark datasets: the International Software Benchmarking Standards Group (ISBSG) dataset (denoted as fpa_isbsg) [61] and the China dataset (denoted as fpa_china) [62]. Research on SDEE relies heavily on historical data [63], as it remains valuable for understanding patterns and trends applicable to modern contexts, provided it is correctly recorded and analyzed. These datasets possess large-scale and detailed documentation, providing a robust foundation for analysis and offering insights into software project complexities across time, thus maintaining their relevance. Firstly, their widespread use in prior SDEE studies provides an essential baseline, enabling direct and meaningful comparison of our CICC methodology's performance against a large body of existing work. Secondly, these datasets are publicly available (PROMISE repository for fpa_china) or accessible via academic programs (fpa_isbsg), which promotes transparency and facilitates the reproducibility of our findings by other researchers. Thirdly, despite their historical nature (with projects typically completed some years before the dataset release versions used), they encompass a diverse range of project types, sizes, and characteristics, making them a valuable testbed for evaluating fundamental improvements introduced by new estimation-enhancing techniques like CICC.

To ensure data quality and prepare the datasets for analysis, the following initial data cleaning and filtering steps were performed:

1) Initial filtering: the raw fpa_isbsg dataset was refined based on its "Data Quality Rating", retaining only projects with a rating of "A" or "B" to establish a baseline of data reliability.
2) Handling of missing values: Following the initial filtering, records with missing values for the dependent variable (functional size), any core independent variables, or essential contextual attributes were removed from both datasets.
3) Thresholding for categorical groups: Finally, contextual categories containing fewer than 10 projects were excluded. This common heuristic ensures a balance between data retention and the statistical stability of groups, preventing sparse or unrepresentative categories from unduly influencing subsequent analysis.

After completing all data cleaning and filtering steps, the final datasets used for our experiments consist of the fpa_isbsg set with 1,641 projects and the fpa_china set with 499 projects. Table 2 presents the descriptive statistics for these final datasets. In all datasets, the dependent variable is the functional size, maintained in its original measurement

TABLE 2. Descriptive statistics for the final prepared datasets.

Dataset	Sizing Method	Instances	Features	Mean	Median	Min	Max
fpa_isbsg	Functional Point	1,641	58	5,122	2,419	8	186,203
fpa_china	Functional Point	499	14	487	215	9	17,518

scale for natural consistency. The independent variables used in sizing estimation are External Inputs (EI), External Outputs (EO), External Queries (EQ), Internal Logical Files (ILF), and External Interface Files (EIF). These quantitative metrics offer a detailed overview of the functional complexity of projects, aligning with established software engineering practices.

The contextual attributes utilized for our clustering approach are Industry Sector (IS) and Programming Language (PL) for the fpa_isbsg dataset, and Resource Type (RT) for the fpa_china dataset. Our approach ensures that the selected clustering projects are more relevant and represent the problem domain. These variables classify projects into pertinent groups, offering contextual insights for more nuanced analysis. The IS categorizes projects by application area, PL by technology stack, and RT provides insights into resource allocation patterns.

In all instances where multiple independent variables are present, Min-Max normalization is applied to ensure consistent influence across variables [3], [4]. Since variables often exhibit diverse ranges, this variation can adversely affect learning. By applying (14), the variables are scaled and standardized from their original range $(x_{\min}, x_{\max})$ to a new range $(\text{New}_{\min}, \text{New}_{\max})$.

$$x'j = \frac{x_j - x_{\min}}{x_{\max} - x_{\min}} \times (\text{New}_{\max} - \text{New}_{\min}) + \text{New}_{\min} \quad (14)$$

where $x'j$ is normalized value of x_j.

B. EXPERIMENT SETUPS

As depicted in Figure 4, we explore various approaches to project centroid initialization from the literature, all of which have been implemented with K-means clustering. These methodologies are summarized in Table 3. Our primary objective is to evaluate the proposed initialization technique introduced in Section V, comparing it against the traditionally used random initialization method employed in prior studies on SDEE and the other approaches covered.

Once the clusters are established, we apply the StepR method for neighbor-based size estimation. This approach utilizes regression and feature selection to assess the relationships between project size and various predictors. Each target project p identifies its k-nearest neighbors N_p^k based on Jaccard similarity values. Jaccard similarity calculations use parallel computing mechanisms to efficiently compute project similarities and neighbor sets.

In implementing CICC, we enhance efficiency by applying Jaccard similarity calculations using parallel computing mechanisms, which reduces training time. Additionally, the KRNN approach is utilized to refine these similarity measures. This added validation step ensures bidirectional neighbor recognition; if project p recognizes project g as a neighbor, the reverse must also hold, enhancing neighbor selection accuracy.

The final size estimation $\hat{s}_p$ for the target project p is computed by aggregating the products of the similarity scores and the sizes of each neighboring project, effectively consolidating the contributions of significant neighbors. This approach ensures that the size estimation for the project p accurately reflects the contextual influences from its neighboring projects.

The experimental datasets were methodically divided into 80% for training and 20% for testing purposes. To ensure robust and reliable comparisons, we conducted experiments across five different random splits of the training and testing data, averaging the results to account for variability and randomness within the datasets.

For consistency across all evaluations, the number of neighbors (k) was fixed at 30. Choosing an appropriate k is critical, as too few neighbors can increase sensitivity to outliers. At the same time, too many may dilute the influence of the most similar projects, potentially leading to inaccurate estimates [10]. This selection was based on a preliminary empirical analysis for this study, which assessed different neighbor sizes ranging from 15 to 45. This analysis revealed that values between 25 and 35 yielded consistently low and stable estimation errors. Within this empirically identified optimal range, $k = 30$ offered a robust balance, effectively mitigating the risks of outlier sensitivity (from too few neighbors) and noise introduction (from too many). Therefore, fixing k at this representative value ensures a stable foundation for the size estimation process and enhances the overall reliability of our findings by ensuring each target project interacts with a defined and reasonably sized set of neighbors. By maintaining a consistent number of neighbors and using the StepR method for estimating project sizes, we aim to produce reliable size estimations that accurately incorporate how neighboring projects can influence each other. This approach enhances the credibility of our clustering analysis.

Another key parameter in our experimental setup is the number of clusters for the K-means algorithm. Selecting an appropriate number of clusters is a critical aspect of k-means clustering. While various state-of-the-art methods exist for determining an ‘‘optimal’’ number of clusters in practice, such as the Elbow method and Silhouette analysis, these often yield data-dependent and sometimes subjective results [16]. In this study, our primary focus is on the performance

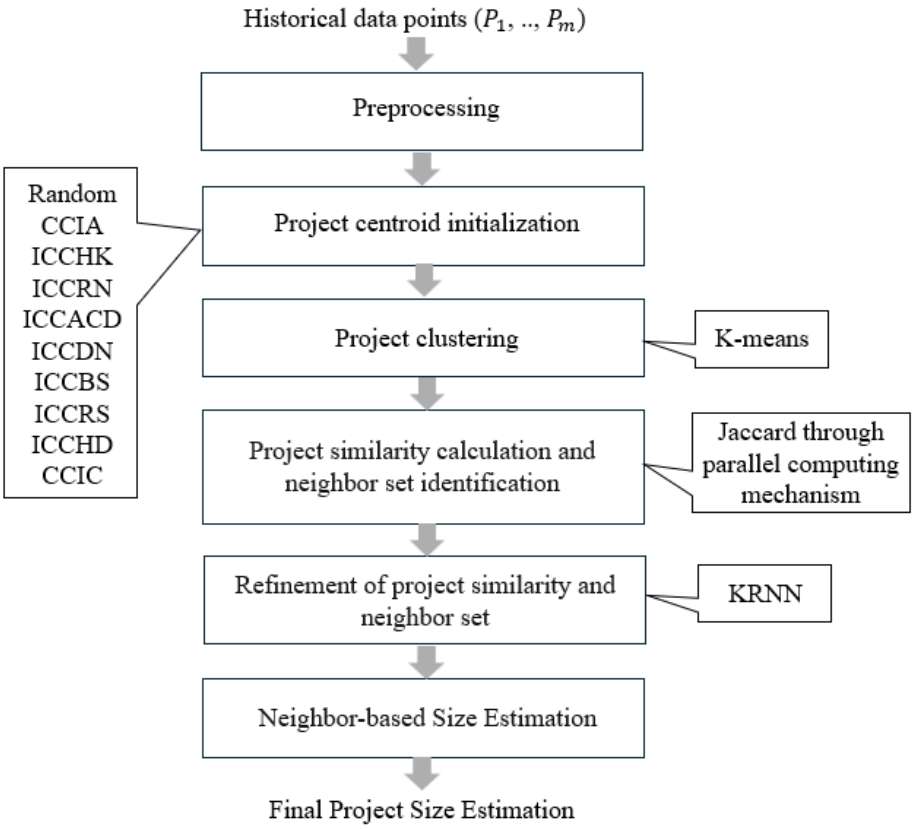

FIGURE 4. Experimental setups.

and robustness of different centroid initialization techniques. Therefore, instead of aiming to identify a single optimal number of clusters for each dataset, we evaluate the methods using a predefined range of cluster counts, specifically varying the number of clusters from 4 to 10. This approach allows us to observe trends and assess how sensitive each initialization method, including our proposed CICC, is to variations in the number of clusters.

C. EVALUATION CRITERIA

1) VALIDITY MEASURE

Validity measures in clustering analysis evaluate the quality and effectiveness of clustering algorithms. They provide insights into how well a clustering model has grouped data points, capturing the degree of similarity within clusters and the degree of dissimilarity between different clusters [64]. Among these measures, GSI and SVR are used in this study.

The GSI (15) quantifies clustering quality by providing a single numerical value derived from individual silhouette coefficients. The GSI ranges from -1 to $+1$, where a value close to $+1$ indicates that most points are well-clustered with distinct and separate clusters. In contrast, values around 0 suggest overlapping clusters, while negative values imply that points may have been misallocated to clusters.

$$GSI = \frac{1}{m}\sum_{i=1}^{m} s(x_i)$$
$$s(x_i) = \frac{b(x_i) - a(x_i)}{\max(a(x_i), b(x_i))} \tag{15}$$

where $s(x_i)$ is the silhouette coefficient for a data point x_i, $a(x_i)$ is the average distance from the point x_i to all other points in the same cluster, reflecting the cohesion of that cluster. $b(x_i)$ is the minimum average distance from x_i to points in any neighboring cluster, indicating how well-separated the cluster is from others.

The SVR (16) explicitly highlights the proportion of poorly clustered points relative to the total data points. Specifically, the SVR takes into account two key metrics: $N_{negative}$, which represents the number of data points with negative silhouette values ($s(x_i) < 0$); these points are closer to neighboring cluster points than to those in their cluster, indicating potential misallocations. The second metric, $N_{positive}$, reflects the number of data points with positive or zero silhouette values ($s(x_i) \geq 0$), representing points that are effectively clustered within their respective groups and thus contribute positively to overall clustering quality. A higher SVR indicates a more significant proportion of data points that

TABLE 3. The characteristics of the initial cluster centroid approaches used in the experiments.

Ref.	Initial centroids computing process	Key benefits	Abbr.
[50]	Calculates initial centroids based on the similarity of data samples to ensure they represent the underlying data distribution.	Improves clustering quality by ensuring centroids are representative of data distribution.	CCIA
[51]	Divides the dataset into small, random subsamples. K-means is applied to each subset, and the configuration with the minimum error is used as the initial centroid for the full dataset.	Enhances the likelihood of identifying high-quality centroids, improving clustering accuracy.	ICCRS
[52]	Applies K-means with random centroids, then refines the results using hierarchical clustering.	Effective for high-dimensional datasets or datasets with many clusters.	ICCHK
[53]	Calculates centroids by analyzing cohesion and coupling degrees within neighborhoods of data objects, considering multiple norms.	Ensures cohesive cluster structures and enhances overall clustering performance.	ICCRN
[18]	Identifies primary axes with minimal correlation and high deviation coefficients, then computes centroids along these axes.	Produces distinct and effective clusters, particularly in datasets with complex attribute relationships.	ICCACD
[54]	Normalizes data objects and calculates distances from a common origin to determine initial centroids.	Ensures consistency across datasets with mixed positive and negative attribute values.	ICCDN
[55]	Iteratively determines attribute minima and maxima through binary search to compute centroids.	Aligns centroids with dataset attribute ranges and ensures they are well-distributed.	ICCBS
[56]	Combines Euclidean distance with density-based metrics to prioritize dense regions of the dataset for centroid selection.	Focuses on high-density regions, enhancing clustering performance.	ICCHD

may be misallocated to their clusters, serving as a metric for identifying the shortcomings of the clustering model. By analyzing the relationship between $N_{negative}$ and $N_{positive}$, it becomes possible to diagnose specific weaknesses in the achieved clustering. A high $N_{negative}$ count relative to the total suggests issues with cluster separation and effective grouping.

$$SVR = \frac{N_{negative}}{N_{negative} + N_{positive}} \tag{16}$$

2) ACCURACY MEASURE

The accuracy of the experimental effort methods is assessed using Mean Absolute Error (MAE, (17)) [14] and Percentage of Prediction with $x\%$ ($PRED(x)$, (18)) [65]. All these criteria have demonstrated their effectiveness [3]. Traditional metrics, such as Magnitude of Relative Errors (MRE) and similar criteria, are not utilized in this context because they tend to favor higher estimates [66].

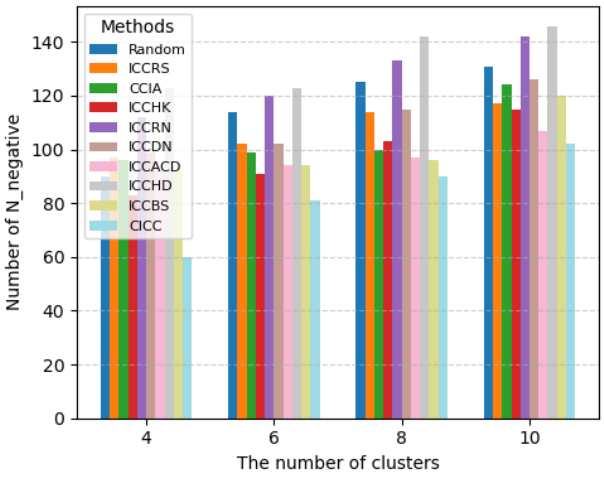

FIGURE 5. Comparison of $N_{negative}$ per clusters for experimental methods in the *fpa_isbsg* dataset.

$$MAE = \frac{1}{2}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{17}$$

$$PRED(x) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} 1 & \text{if } \dfrac{|y_i - \hat{y}_i|}{y_i} \leq x \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

VII. RESULT AND DISCUSSION

A. COMPARISON OF EXISTING AND PROPOSED APPROACHES THROUGH VALIDITY MEASURE

The comparison of GSI and SVR for the existing and proposed methods is presented here, with a focus on the negative values $N_{negative}$ specifically for the datasets *fpa_isbsg* and *fpa_china*. Figures 5 and 6 illustrate the $N_{negative}$ obtained from existing methods (Random, CCIA, ICCRS, ICCHK, ICCRN, ICCACD, ICCDN, ICCBS, and ICCHD) alongside the proposed CICC method for both datasets. The CICC method demonstrates superiority over the existing methods, as evidenced by its lower $N_{negative}$ values. As clusters increase, the $N_{negative}$ values for other methods generally rise. Methods such as ICCRN, ICCHD, and ICCBS frequently exhibit higher negative values, indicating difficulties in distinguishing complex boundaries. In contrast, CICC maintains robust performance and confirms its ability to form well-defined clusters. Additionally, the proposed CICC method shows more positive values $N_{positive}$ compared to the existing methods, indicating that most data objects are appropriately positioned within their respective clusters.

A comparison of GSI and SVR values is detailed in Table 4. The CICC method demonstrates superior index

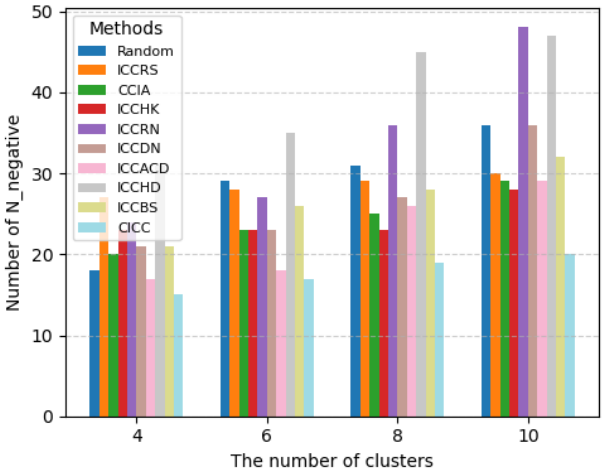

FIGURE 6. Comparison of $N_{negative}$ per clusters for experimental methods in the *fpa_china* dataset.

TABLE 4. The average GSI and SVR results of experimental methods. The best results are highlighted in bold.

	fpa_isbsg dataset		*fpa_china* dataset	
Methods	GSI	SVR	GSI	SVR
Random	0.227	0.132	0.242	0.118
ICCRS	0.323	0.123	0.238	0.109
CCIA	0.332	0.126	0.285	0.108
ICCHK	0.346	0.132	0.276	0.092
ICCRN	0.309	0.146	0.232	0.130
ICCDN	0.355	0.127	0.279	0.103
ICCACD	0.342	0.111	0.270	0.086
ICCHD	0.311	0.139	0.215	0.152
ICCBS	0.345	0.116	0.271	0.285
CICC	**0.517**	**0.097**	**0.478**	**0.075**

values across all datasets, with a high GSI and the lowest SVR. These results suggest that the proposed method yields predominantly positive DBCS values, indicating that most data objects are accurately classified within their respective clusters. The DBCS value for a data object i is defined as $DBCS(i) = b(x_i) - a(x_i)$. The metrics $N_{negative}$ and $N_{positive}$ represent the counts of data objects with negative and positive DBCS values, respectively.

Minimizing the SVR, which ranges from 0 to 1, is crucial for assessing the effectiveness of clustering methods. A lower count of negative DBCS values is an indirect indicator of clustering accuracy, as data objects with negative values are typically located at the boundaries of clusters. It is also important to note that different clustering algorithms can yield varying cluster structures for the same dataset. Consequently, the number of data objects situated within boundary regions will differ, with the count of negative elements providing critical insight into the overall effectiveness of

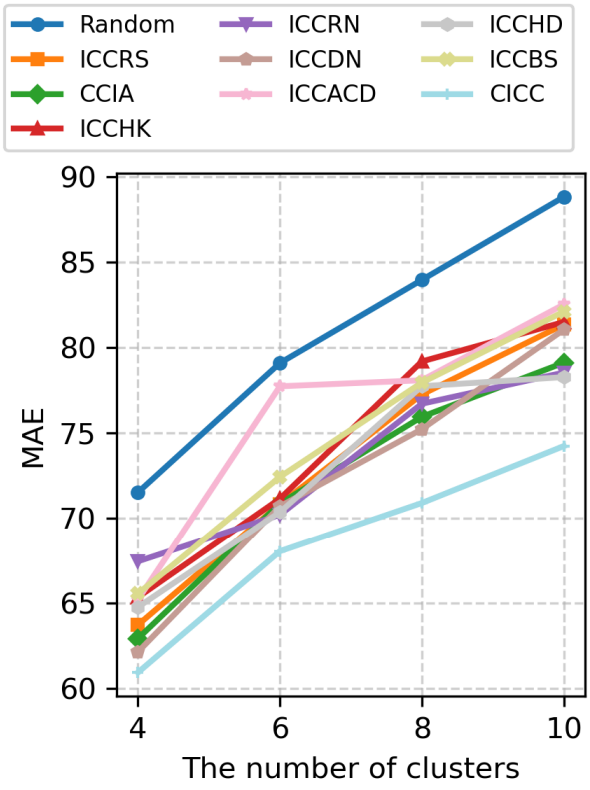

FIGURE 7. MAE results of experimental methods in the *fpa_isbsg* dataset.

each clustering method. Thus, the CICC achieves the lowest SVR, underscoring its potential as a more effective clustering solution than existing methods.

B. COMPARISON OF EXISTING AND PROPOSED APPROACHES THROUGH ACCURACY MEASURE

Next, we analyze the estimation accuracy achieved by applying different centroid initialization methods, which are evaluated using MAE and PRED (0.25) on the *fpa_isbsg* and *fpa_china* datasets. This comparative analysis highlights the superior performance of our CICC method over existing methods. Specifically, Figures 7 and 8 illustrate that our method yields better MAE results as clusters increase from 4 to 10. The effectiveness of CICC comes from its ability to cluster contextual data characteristics systematically. This produces better estimation accuracy than existing methods, particularly as the number of clusters increases. Notably, the estimation accuracy of CICC tends to decline more slowly with the addition of clusters, indicating robust performance across varying contexts. Moreover, it is crucial to acknowledge that as the number of clusters increases, the number of project pairs requiring similarity computations for each cluster can decrease. This can lead to significant savings in computational costs, as fewer comparisons are needed within more focused, smaller clusters.

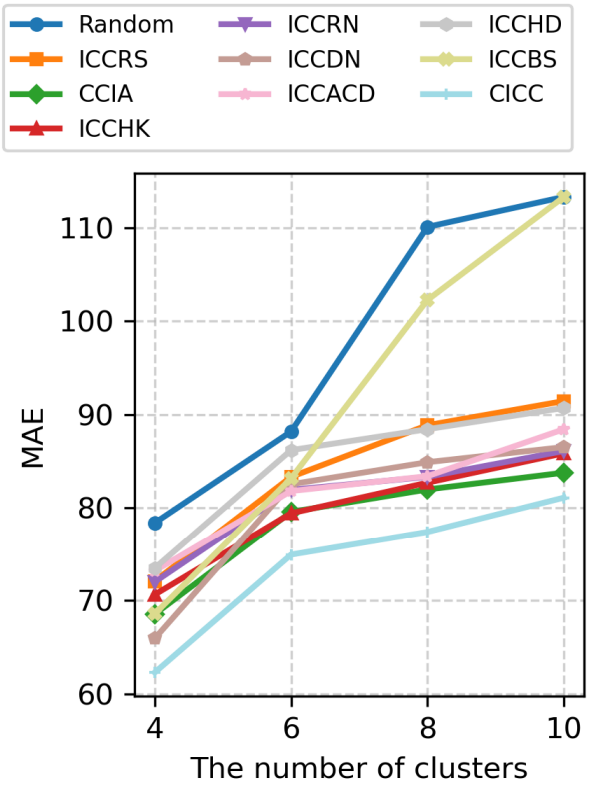

FIGURE 8. MAE results of experimental methods in the *fpa_china* dataset.

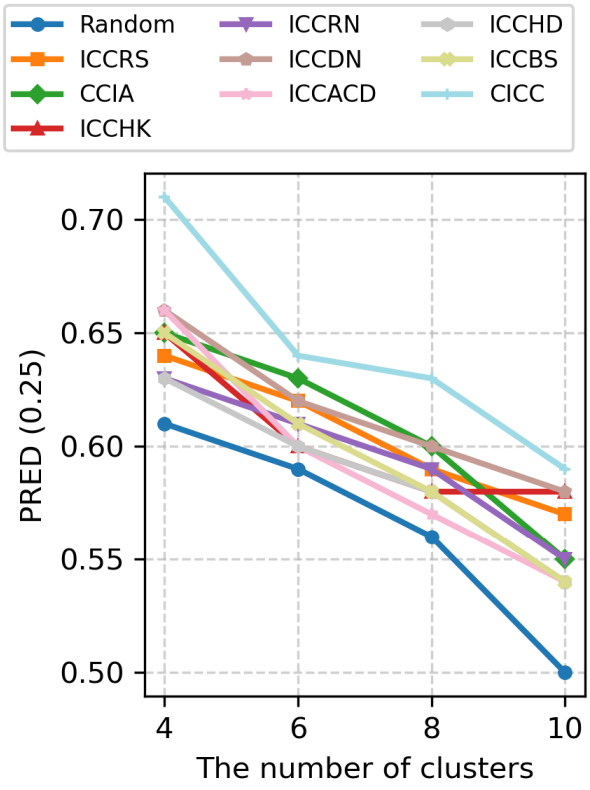

FIGURE 9. PRED (0.25) results of experimental methods in the *fpa_isbsg* dataset.

In the *fpa_isbsg* dataset, CICC consistently achieved the lowest MAE values across various cluster sizes, demonstrating its effectiveness in initial centroid selection. Specifically, at $k = 4$, CICC recorded an MAE of 60.95, outperforming Random by 15% (MAE of 71.53), CCIA by 4.4% (63.74), and ICCRS by 3.2% (62.94). Similarly, CICC outperformed ICCRN, ICCDN, ICCBS, and ICCHD with improvements of 1.9%, 6.3%, 7.0%, and 3.1%, respectively. This trend continued at $k = 6$ and $k = 8$, with CICC achieving MAEs of 68.06 and 70.91, respectively. It outperformed CCIA and ICCRS by approximately 4.1% and 4.0% at $k = 6$ and 8.2% and 6.7% at $k = 8$. At k, CICC achieved an MAE of 74.24, which is significantly lower than Random's MAE of 88.83, corresponding to an improvement of around 17%.

Similarly, in the *fpa_china* dataset, CICC demonstrated its superiority by consistently achieving the lowest MAE values. At $k = 4$, CICC recorded an MAE of 62.29, outperforming Random (78.37) by approximately 20.5%. It also surpassed CCIA and ICCRS, which had MAEs of 68.54 and 72.09, by about 9.1% and 13.6%, respectively. This trend persisted at $k = 6$, where CICC achieved an MAE of 74.94, outperforming CCIA and ICCRS by approximately 5.8% and 10.1%, respectively. At $k = 8$, CICC maintained its lead with an MAE of 77.39. At $k = 10$, CICC achieved an MAE of 81.08, 28.5% lower than Random's MAE of 113.32, and notable improvements over other methods.

Figures 9 and 10 present the results from various experimental methods using PRED (0.25). PRED (0.25) measures the proportion of predictions that fall within a 25% margin of error relative to the actual values, providing insight into the estimation accuracy of each method. The proposed CICC method consistently demonstrates superior performance across most cluster sizes in both datasets, indicating a high proportion of accurate estimates. While the PRED (0.25) scores generally remain strong for the CICC method, the complexity of the clustering task can present challenges for all methods. As the number of clusters increases, each cluster becomes more specialized and may represent a smaller subset of the data. This specialization can make it more challenging for the estimation methods to maintain accuracy, particularly if the data points within each cluster become more diverse or less representative of the overall dataset.

Next, we consider the number of project pairs requiring similarity computation, which is crucial for the cost of the memory-based collaborative filtering training phase for effort estimation. Table 5 shows that the CICC requires fewer project pairs for similarity computation than other methods. This indicates that CICC facilitates the creation of more evenly distributed project clusters, thereby avoiding the concentration of multiple projects within specific clusters.

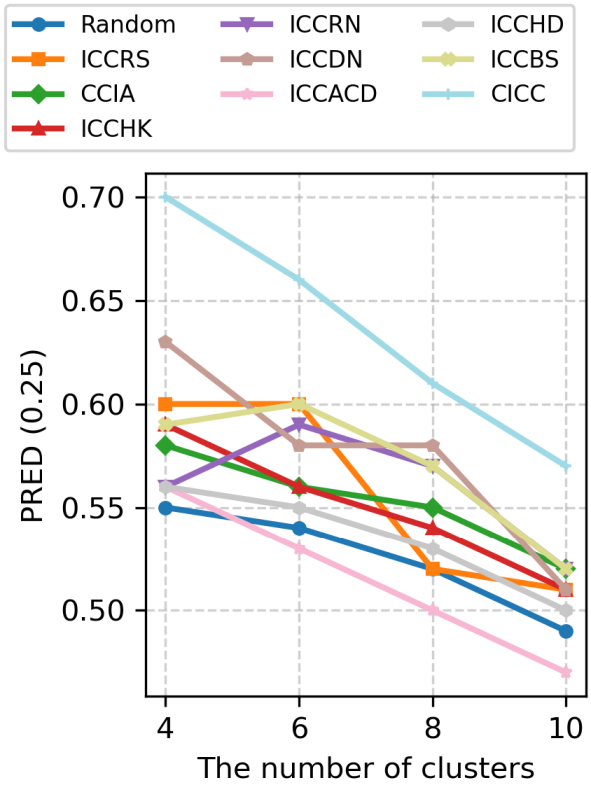

FIGURE 10. PRED (0.25) results of experimental methods in the *fpa_china* dataset.

TABLE 5. The number of project pairs required similarity computation. The best results are highlighted in bold.

The number of project pairs for similarity computation	The number of clusters			
	4	6	8	10
	fpa_isbsg dataset			
Random	158,268	120,541	79,571	67,743
ICCRS	160,913	117,987	72,154	64,909
CCIA	169,735	132,082	100,185	70,442
ICCHK	153,593	124,558	85,906	69,935
ICCRN	153,973	119,754	82,660	54,861
ICCDN	153,854	120,338	85,785	73,254
ICCACD	168,970	132,837	102,244	75,138
ICCHD	152,689	119,886	73,779	60,243
ICCBS	161,888	120,098	96,768	72,311
CICC	**126,269**	**100,648**	**64,521**	**51,278**
	fpa_china dataset			
Random	12,767	9,232	7,181	5,139
ICCRS	14,478	9,957	7,221	5,901
CCIA	15,043	11,068	9,276	7,211
ICCHK	13,328	12,686	8,275	7,785
ICCRN	12,391	9,405	6,160	5,266
ICCDN	14,108	10,959	7,830	7,655
ICCACD	14,549	12,586	9,956	8,737
ICCHD	13,900	9,871	6,582	4,719
ICCBS	15,036	12,363	7,778	6,309
CICC	**7,925**	**5,627**	**4,650**	**4,017**

The CICC enhances efficiency in computing Jaccard similarity through strategic parallel processing. This approach enables concurrent calculations of multiple project pairs and optimizes performance by leveraging multi-threading and distributed computing resources. For each cluster C_i, we compute Jaccard similarity JAC(p, g) for all unique pairs (p, g). An adaptive thresholding mechanism filters out less significant pairs, retaining only those with JAC$(p, g) > t$, where t is derived from the similarity score distribution. This approach prevents bottlenecks and highlights essential insights. As a result, fewer project pairs need evaluation in neighborhood-based collaborative filtering, leading to lower computational costs and shorter training times.

Finally, we aim to provide more convincing conclusions about the effectiveness of the proposed CICC through statistical analysis. A t-test was used to compare the proposed method against existing approaches based on their estimation errors. This test determines whether one method is statistically superior to another based on p-values and sample means. A p-value of less than 0.05 and a lower sample mean indicate statistical significance, confirming that one method outperforms another. Furthermore, to assess the practical importance of these performance differences, we employed Cohen's d [67]. We interpret Cohen's d values using standard benchmarks: $d \approx 0.2$ indicates a small effect, $d \approx 0.5$ a medium effect, and $d \approx 0.8$ a large effect. Consequently, a statistically significant p-value ($p < 0.05$) combined with a medium to large Cohen's d provides strong evidence for CICC's superior and practically meaningful performance. The results, presented in Tables 6 and 7, consistently show that CICC achieves statistically significant improvements accompanied by noteworthy effect sizes across the compared methods.

C. INTERPRETING CICC'S METHODOLOGICAL ADVANTAGES

The consistent superior performance of CICC across cluster validity, estimation accuracy, and computational efficiency, as confirmed by statistical testing, can be directly attributed to its novel approach to centroid initialization that prioritizes contextual understanding from the outset, as detailed in Section V-A. Briefly, this approach involves two key stages: first, an initial partitioning of projects based on key contextual attributes (IS, PL, or RT) into homogeneous subgroups; second, deterministic analytical steps (including similarity assessment and spectral methods) are performed *within* these subgroups. This fundamental distinction from traditional approaches that operate globally or rely on random seeding

TABLE 6. The t-test and effect size results on estimation errors of the proposed CICC method compared with other methods in the *fpa_isbsg* dataset. Note: $A \gg B$ means that A is statistically superior to B.

Pairs of methods	Sample mean	Cohen's d	p-value	Statistical conclusion
CICC vs. Random	68.539 vs. 80.858	1.874	0.0004	CICC >> Random
CICC vs. ICCRS	68.539 vs. 73.300	0.704	0.0128	CICC >> ICCRS
CICC vs. CCIA	68.539 vs. 72.223	0.576	0.0083	CICC >> CCIA
CICC vs. ICCHK	68.539 vs. 74.305	0.873	0.0087	CICC >> ICCHK
CICC vs. ICCRN	68.539 vs. 73.235	0.860	0.0082	CICC >> ICCRN
CICC vs. ICCDN	68.539 vs. 72.288	0.542	0.0263	CICC >> ICCDN
CICC vs. ICCACD	68.539 vs. 75.897	1.113	0.0038	CICC >> ICCACD
CICC vs. ICCHD	68.539 vs. 72.794	0.702	0.0101	CICC >> ICCHD
CICC vs. ICCBS	68.539 vs. 74.518	0.926	0.0031	CICC >> ICCBS

TABLE 7. The t-test and effect size results on estimation errors of the proposed CICC method compared with other methods in the *fpa_china* dataset. Note: $A \gg B$ means that A is statistically superior to B.

Pairs of methods	Sample mean	Cohen's d	p-value	Statistical conclusion
CICC vs. Random	73.930 vs. 97.495	1.772	0.0099	CICC >> Random
CICC vs. ICCRS	73.930 vs. 83.926	1.194	0.0003	CICC >> ICCRS
CICC vs. CCIA	73.930 vs. 78.459	0.602	0.0042	CICC >> CCIA
CICC vs. ICCHK	73.930 vs. 79.647	0.773	0.0038	CICC >> ICCHK
CICC vs. ICCRN	73.930 vs. 80.835	0.957	0.0032	CICC >> ICCRN
CICC vs. ICCDN	73.930 vs. 79.968	0.682	0.0038	CICC >> ICCDN
CICC vs. ICCACD	73.930 vs. 81.701	1.064	0.0027	CICC >> ICCACD
CICC vs. ICCHD	73.930 vs. 84.705	1.359	0.0000	CICC >> ICCHD
CICC vs. ICCBS	73.930 vs. 91.827	1.178	0.0332	CICC >> ICCBS

underpins the observed performance gains, particularly in heterogeneous datasets where project characteristics can vary widely.

Many existing initialization methods operate on the entire dataset or attempt to refine randomly chosen centroids, which can be less effective in such diverse environments. Specifically, while probabilistic methods like ICCRS aim to mitigate risks associated with pure random selection, CICC's deterministic process, initiated *within* already contextually refined subgroups, inherently enables better alignment of initial centroids with specific project types from the outset. Global statistical or hierarchical techniques, such as those employed by CCIA and ICCHK, analyze the dataset. In contrast, CICC applies its analytical techniques (including density-aware refinement and spectral analysis to identify variance patterns) *within* each context-specific subgroup. This allows CICC to discover nuanced patterns relevant to distinct operational contexts, rather than relying on average patterns that might obscure these vital distinctions across a heterogeneous dataset.

Other approaches like ICCDN (origin distance normalization), ICCACD (global feature correlations), or ICCHD (global spatial distance and density measures) typically identify important features or dense regions across the entire dataset without the primary step of contextual pre-segmentation. CICC integrates spectral and density insights strategically after the dataset is segmented by context. This ensures that the ''importance'' of a project for centroid consideration is evaluated relative to its direct contextual peers, leading to more relevant centroid placement. Even methods like ICCBS, which focus on an even distribution of centroids based on feature ranges, may overlook the distinct project populations that CICC uncovers by first respecting fundamental contextual boundaries within a larger, diverse dataset.

Overall, CICC's approach fundamentally differs because it embeds and prioritizes context at the beginning of the initialization process. The initial centroids are thus not only statistically well-placed but are also highly representative of distinct operational contexts, a critical advantage when dealing with heterogeneous collections of software projects. This leads to more meaningful clusters from the outset, which directly enhances cluster validity (as seen with GSI and SVR improvements) and results in more accurate effort estimations. Moreover, CICC's advantages were not limited to overall performance but also extended to consistency across the tested range of cluster counts from 4 to 10. While compared methods exhibited more pronounced fluctuations or sharper degradations in MAE as the number of clusters varied (an effect particularly evident on both datasets),

CICC's estimation accuracy generally demonstrated greater stability. This suggests that CICC's context-aware initialization contributes to more robust clustering structures that are less sensitive to the precise selection of clusters, a valuable characteristic when the ''optimal'' number of clusters is uncertain or varies.

VIII. THREATS TO VALIDITY

In this section, we outline the potential threats to the validity of our study, categorized into three main areas: internal, external, and construct validity.

A threat to internal validity arises from our reliance on specific datasets *fpa_isbsg* and *fpa_china* to evaluate the proposed method and other methods. These datasets were utilized because they are publicly available, except for the ISBSG dataset, which requires registration in an academic program for access. Although these datasets are reputable in SDEE, variations in data quality or inherent biases could affect the broader applicability of our findings. Inherent biases refer to systematic errors in data collection, measurement, or analysis that may distort the results, leading to conclusions that do not accurately reflect the actual situation.

Regarding external validity, a key consideration is the generalizability of our results to current software development environments. While these established datasets demonstrate the effectiveness of CICC, particularly in centroid initialization within well-understood historical data contexts, the projects within them were completed years prior to the datasets' collection. As software development practices, tools, and project complexities continually evolve, results derived from these datasets may not fully reflect contemporary industry paradigms. Therefore, to ensure the broader applicability of our approach in current and future software development practices, further validation on more recent and diverse datasets is recommended. This future work will help confirm the robustness and relevance of CICC across the rapidly changing landscape of software engineering. Additionally, given the specific contexts represented in these datasets, such as particular industries or project types, the observed effectiveness of CICC may vary when applied to domains or datasets with distinct characteristics.

Concerns regarding construct validity may arise from the metrics selected to evaluate clustering quality and estimation accuracy. We carefully applied validity and accuracy measures to assess the existing and proposed methods, as presented in Section VI-C. This research aimed to form the most accurate conclusions regarding the methods. As a result, we can cautiously conclude that the experimental results of this study have potential generalizability.

IX. CONCLUSION

This study presented CICC, a novel methodology significantly enhancing neighbor-based software effort estimation. The methodology achieves this by partitioning projects based on key contextual attributes to guide a deterministic cluster initialization process. This ensures more relevant project groupings from the outset, leading to improved similarity measures, superior clustering quality, and more accurate effort estimation outcomes. Furthermore, CICC's design incorporates elements like parallel processing, contributing to notable gains in computational efficiency. Our experimental evaluations robustly validated CICC's superiority in these key aspects when compared against nine existing initialization techniques across two benchmark datasets.

From a practical standpoint, the CICC methodology offers software project managers distinct advantages for real-world estimation. CICC's core mechanism of first partitioning projects by key contextual attributes (such as industry sector and programming language) before further analytical steps allows managers to establish more reliable and homogenous initial project groupings for analogy-based or neighbor-based effort estimation. This tailored grouping, arising directly from CICC's design, leads to more accurate baseline estimates early in the project lifecycle. Specifically, when faced with a new project, a manager can apply CICC to quickly identify a small, highly relevant cluster of past projects that are not only feature-similar but also explicitly from the same industry and built with similar technology, significantly improving the precision of analogous estimates. Furthermore, CICC's deterministic process for centroid initialization, a direct contrast to methods relying on random selection, ensures that effort estimation processes become more repeatable and consistent over time. This reliability aids substantially in better resource planning, budget allocation, and risk management. Finally, the efficiency incorporated into CICC's design, such as its application of parallel processing for similarity calculations within these contextually defined clusters, means that these more accurate and consistent estimations can be achieved effectively even with large historical datasets without prohibitive delays.

Through a comprehensive analysis, we addressed and contributed to the following RQs:

- **RQ1: Compared to other initialization techniques across different datasets, how does the CICC method affect clustering quality when evaluated using validity measures such as GSI and SVR?** The results from our experiments indicate that the CICC method significantly improves clustering quality when evaluated against traditional initialization techniques across various datasets. The GSI consistently showed higher values for clusters initialized using the CICC method, suggesting that the projects are better grouped with distinct separations from other clusters. Additionally, the SVR revealed a lower proportion of poorly clustered points, confirming that the CICC method enhances the relevance and accuracy of selected projects. These findings indicate that the CICC method offers a robust framework for optimizing clustering quality in the context of effort estimation.
- **RQ2: How does the CICC method perform in estimation accuracy compared to existing methods?**

The comparative analysis of estimation accuracy demonstrates that the CICC method outperforms existing techniques in terms of precision and reliability. By utilizing contextually relevant projects and ensuring that initial centroids are effectively determined, the CICC method leads to more accurate effort estimates. Evaluation metrics such as MAE and other standard measures indicate significant improvements in estimation accuracy, validating the effectiveness of our approach. The results underscore the potential of the CICC method to contribute to more reliable estimation practices in software development and engineering.

X. FUTURE WORK

While the CICC methodology, as demonstrated, significantly enhances centroid initialization, leading to improved clustering and estimation accuracy, its computational characteristics and scalability present avenues for further investigation. The CICC process involves operations such as similarity matrix construction, spectral analysis, and density estimation (via k-nearest neighbors) within each context-defined subgroup ($\mathcal{P}_{ij}$). These steps could introduce computational demands for datasets with very large individual contextual subgroups or an extremely high number of such distinct subgroups. Therefore, future research could explore optimizing these internal CICC computations, specifically, by investigating more scalable approximations for spectral analysis or density estimation within very large subgroups, or by developing strategies for efficiently managing a vast number of smaller contextual segments. Addressing these aspects would further bolster CICC's efficiency and applicability in increasingly large-scale SDEE scenarios.

Following these potential refinements to CICC's core, our primary direction for subsequent research is to enhance the project similarity measures used for neighbor selection after clustering. The current Jaccard index, though a standard approach [11], [68], primarily captures attribute overlap. We aim to develop a more nuanced metric that better reflects one project's multi-faceted ''usefulness'' in estimating another.

To address this, we will focus on refining similarity metrics by explicitly incorporating this concept of ''usefulness.'' In this context, we interpret ''usefulness'' as how much complementary information two software projects provide each other when considered as a pair across the full spectrum of their defined characteristics, reflecting the richness and diversity of their combined feature sets. To operationalize this, we envision an enhanced similarity approach that moves beyond simple attribute overlap. This will involve integrating several distinct facets: Firstly, contextual factor similarity will quantify high-level commonalities by directly overlapping key descriptive contextual categorical attributes, providing a foundational understanding of shared operational contexts. Secondly, functional size profile similarity will assess the alignment in the detailed breakdown of functional components between projects, potentially using vector similarity measures like cosine similarity on their respective functional profiles. Thirdly, development duration similarity will incorporate a comparison of project development durations.

We hypothesize that such a multi-factor similarity approach will lead to a more precise identification of truly analogous projects and, consequently, more accurate effort estimations. Finally, ongoing validation of CICC and any new similarity measures on larger, more diverse, and contemporary datasets will ensure their robustness and broad applicability in evolving software development contexts.

REFERENCES

[1] B. Boehm, C. Abts, and S. Chulani, ''Software development cost estimation approaches—A survey,'' *Ann. Softw. Eng.*, vol. 10, nos. 1–4, pp. 177–205, Nov. 2000.

[2] R. Silhavy, P. Silhavy, and Z. Prokopova, ''Algorithmic optimisation method for improving use case points estimation,'' *PLoS ONE*, vol. 10, no. 11, Nov. 2015, Art. no. e0141887.

[3] H. L. T. K. Nhung, V. V. Hai, P. Silhavy, Z. Prokopova, and R. Silhavy, ''Incorporating statistical and machine learning techniques into the optimization of correction factors for software development effort estimation,'' *J. Softw., Evol. Process*, vol. 36, no. 5, p. e2611, May 2024.

[4] H. L. T. K. Nhung, V. V. Hai, R. Silhavy, Z. Prokopova, and P. Silhavy, ''Parametric software effort estimation based on optimizing correction factors and multiple linear regression,'' *IEEE Access*, vol. 10, pp. 2963–2986, 2022.

[5] A. Saeed, W. H. Butt, F. Kazmi, and M. Arif, ''Survey of software development effort estimation techniques,'' in *Proc. 7th Int. Conf. Softw. Comput. Appl.*, New York, NY, USA, Feb. 2018, pp. 82–86.

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

[7] M. Hassanali, M. Soltanaghaei, T. J. Gandomani, and F. Z. Boroujeni, ''Exploring stacking methods for software effort estimation with hyperparameter tuning,'' *Cluster Comput.*, vol. 28, no. 4, p. 241, Aug. 2025.

[8] I. Atoum and A. A. Otoom, ''Enhancing software effort estimation with pre-trained word embeddings: A small-dataset solution for accurate story point prediction,'' *Electronics*, vol. 13, no. 23, p. 4843, Dec. 2024.

[9] Y. Li, Z. Ren, Z. Wang, L. Yang, L. Dong, C. Zhong, and H. Zhang, ''Fine-SE: Integrating semantic features and expert features for software effort estimation,'' in *Proc. IEEE/ACM 46th Int. Conf. Softw. Eng.*, New York, NY, USA, Feb. 2024, pp. 1–12.

[10] H. L. T. K. Nhung, P. Šilhavý, and R. Šilhavý, ''Enhancing software effort estimation through influencers-based project similarity measurement,'' *Proc. Comput. Sci.*, vol. 246, pp. 3256–3264, Feb. 2024.

[11] F. Fkih, ''Similarity measures for collaborative filtering-based recommender systems: Review and experimental comparison,'' *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 9, pp. 7645–7669, Oct. 2022.

[12] C. C. Aggarwal, *Neighborhood-Based Collaborative Filtering*. Cham, Switzerland: Springer, 2016, pp. 29–70.

[13] R. Silhavy, P. Silhavy, and Z. Prokopova, ''Evaluating subset selection methods for use case points estimation,'' *Inf. Softw. Technol.*, vol. 97, pp. 1–9, May 2018.

[14] Y. Alqasrawi, M. Azzeh, and Y. Elsheikh, ''Locally weighted regression with different kernel smoothers for software effort estimation,'' *Sci. Comput. Program.*, vol. 214, Feb. 2022, Art. no. 102744.

[15] Z. Prokopova, R. Silhavy, and P. Silhavy, ''The effects of clustering to software size estimation for the use case points methods,'' in *Software Engineering Trends and Techniques in Intelligent Systems*. Cham, Switzerland: Springer, 2017, pp. 479–490.

[16] V. V. Hai, H. L. T. K. Nhung, Z. Prokopová, R. Šilhavý, and P. Šilhavý, ''Analyzing the effectiveness of the Gaussian mixture model clustering algorithm in software enhancement effort estimation,'' *Intell. Inf. Database Syst.*, vol. 132, no. 1, pp. 255–268, Mar. 2022.

[17] V. V. Hai, H. L. T. Kim Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, ''Toward improving the efficiency of software development effort estimation via clustering analysis,'' *IEEE Access*, vol. 10, pp. 83249–83264, 2022.

[18] M. Erisoglu, N. Calis, and S. Sakallioglu, "A new algorithm for initial cluster centers in k-means algorithm," *Pattern Recognit. Lett.*, vol. 32, no. 14, pp. 1701–1705, Oct. 2011.

[19] J. S. Breese, D. Heckerman, and C. M. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. 14th Conf. Uncertainty Artif. Intell.*, San Francisco, CA, USA, 2013, p. 43.

[20] P. Parhi, A. Pal, and M. Aggarwal, "A survey of methods of collaborative filtering techniques," in *Proc. Int. Conf. Inventive Syst. Control (ICISC)*, Jan. 2017, pp. 1–7.

[21] Q. Zhang, "Analysis and prediction model of college sports learning performance based on big data collaborative filtering," in *Proc. 8th Int. Conf. Inf. Syst. Eng. (ICISE)*, Jun. 2023, pp. 346–349.

[22] M. V. Kosti, N. Mittas, and L. Angelis, "Alternative methods using similarities in software effort estimation," in *Proc. 8th Int. Conf. Predictive Models Softw. Eng.*, New York, NY, USA, Sep. 2012, pp. 59–68.

[23] P. Phannachitta, "Robust comparison of similarity measures in analogy based software effort estimation," in *Proc. 11th Int. Conf. Softw., Knowl., Inf. Manage. Appl. (SKIMA)*, Dec. 2017, pp. 1–7.

[24] A. Kaushik, P. Kaur, N. Choudhary, and Priyanka, "Stacking regularization in analogy-based software effort estimation," *Soft Comput.*, vol. 26, no. 3, pp. 1197–1216, Feb. 2022.

[25] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, New York, NY, USA, Aug. 1999, pp. 230–237.

[26] G. Koutrika, B. Bercovitz, and H. Garcia-Molina, "FlexRecs: Expressing and combining flexible recommendations," in *Proc. ACM SIGMOD Int. Conf. Manage. data*, Georgia, Jun. 2009, pp. 745–758.

[27] W. Cai, W. Pan, J. Liu, Z. Chen, and Z. Ming, "K-reciprocal nearest neighbors algorithm for one-class collaborative filtering," *Neurocomputing*, vol. 381, pp. 207–216, Mar. 2020.

[28] X. Sun and L. Zhang, "Multi-order nearest neighbor prediction for recommendation systems," *Digit. Signal Process.*, vol. 127, Jul. 2022, Art. no. 103540.

[29] D. Margaris and C. Vassilakis, "Improving collaborative filtering's rating prediction coverage in sparse datasets by exploiting the 'friend of a friend' concept," *Int. J. Big Data Intell.*, vol. 7, no. 1, pp. 47–57, 2020.

[30] M. Nilashi, O. Ibrahim, and K. Bagherifard, "A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques," *Expert Syst. Appl.*, vol. 92, pp. 507–520, Feb. 2018.

[31] P. Silhavy, R. Silhavy, and Z. Prokopova, "Categorical variable segmentation model for software development effort estimation," *IEEE Access*, vol. 7, pp. 9618–9626, 2019.

[32] P. Silhavy, R. Silhavy, and Z. Prokopova, "Spectral clustering effect in software development effort estimation," *Symmetry*, vol. 13, no. 11, p. 2119, Nov. 2021.

[33] C. Lokan and E. Mendes, "Investigating the use of duration-based moving windows to improve software effort prediction: A replicated study," *Inf. Softw. Technol.*, vol. 56, no. 9, pp. 1063–1075, Sep. 2014.

[34] V. K. Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "Increasing the accuracy of software development effort estimation using projects clustering," *IET Softw.*, vol. 6, no. 6, pp. 461–473, Dec. 2012.

[35] F. A. Amazal and A. Idri, "Estimating software development effort using fuzzy clustering-based analogy," *J. Softw., Evol. Process*, vol. 33, no. 4, Apr. 2021, Art. no. e2324.

[36] A. Idri, F. A. Amazal, and A. Abran, "Analogy-based software development effort estimation: A systematic mapping and review," *Inf. Softw. Technol.*, vol. 58, pp. 206–230, Feb. 2015.

[37] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, "Neural network models for software development effort estimation: A comparative study," *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2369–2381, Nov. 2016.

[38] M. Azzeh and A. B. Nassif, "A hybrid model for estimating software project effort from use case points," *Appl. Soft Comput.*, vol. 49, pp. 981–989, Dec. 2016.

[39] M. Garre, J. J. Cuadrado, M. A. Sicilia, M. Charro, and D. Rodriguez, "Segmented parametric software estimation models: Using the EM algorithm with the ISBSG 8 database," in *Proc. 27th Int. Conf. Inf. Technol. Interface*, 2005, pp. 181–187.

[40] M. Kanwal, M. M. Riaz, S. S. Ali, and A. Ghafoor, "Saliency-based fabric defect detection via bag-of-words model," *Signal, Image Video Process.*, vol. 17, no. 4, pp. 1687–1693, Jun. 2023.

[41] S. Amasaki and C. Lokan, "The effect of moving windows on software effort estimation: Comparative study with CART," in *Proc. 6th Int. Workshop Empirical Softw. Eng. Pract.*, Nov. 2014, pp. 1–6.

[42] P. Silhavy and R. Silhavy, "Evaluating kernel functions in software effort estimation: A comparative study of moving window and spectral clustering models across diverse datasets," *IEEE Access*, vol. 11, pp. 126335–126351, 2023.

[43] L. L. Minku, "A novel online supervised hyperparameter tuning procedure applied to cross-company software effort estimation," *Empirical Softw. Eng.*, vol. 24, no. 5, pp. 3153–3204, Oct. 2019.

[44] E. Ventura-Molina, C. López-Martín, I. López-Yáñez, and C. Yáñez-Márquez, "A novel data analytics method for predicting the delivery speed of software enhancement projects," *Mathematics*, vol. 8, no. 11, p. 2002, Nov. 2020.

[45] T. Kinnunen, I. Sidoroff, M. Tuononen, and P. Fränti, "Comparison of clustering methods: A case study of text-independent speaker modeling," *Pattern Recognit. Lett.*, vol. 32, no. 13, pp. 1604–1617, Oct. 2011.

[46] X. Huang, L. Zhang, B. Wang, F. Li, and Z. Zhang, "Feature clustering based support vector machine recursive feature elimination for gene selection," *Int. J. Speech Technol.*, vol. 48, no. 3, pp. 594–607, Mar. 2018.

[47] K. Krishna and M. N. Murty, "Genetic K-means algorithm," *IEEE Trans. Syst., Man, Cybern.*, vol. 29, no. 3, pp. 433–439, Mar. 1999.

[48] P. Fränti, "Genetic algorithm with deterministic crossover for vector quantization," *Pattern Recognit. Lett.*, vol. 21, no. 1, pp. 61–68, Jan. 2000.

[49] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," *Expert Syst. Appl.*, vol. 40, no. 1, pp. 200–210, Jan. 2013.

[50] S. S. Khan and A. Ahmad, "Cluster center initialization algorithm for K-means clustering," *Pattern Recognit. Lett.*, vol. 25, no. 11, pp. 1293–1302, Aug. 2004.

[51] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2007, pp. 1027–1035.

[52] K. Arai and A. R. Barakbah, "Hierarchical K-means: An algorithm for centroids initialization for K-means," *Rep. Fac. Sci. Eng., Saga Univ.*, vol. 36, no. 1, pp. 25–31, Jan. 2007.

[53] F. Cao, J. Liang, and G. Jiang, "An initialization method for the K-means algorithm using neighborhood model," *Comput. Math. Appl.*, vol. 58, no. 3, pp. 474–483, Aug. 2009.

[54] M. Yedla, S. R. Pathakota, and T. M. Srinivasa, "Enhancing K-means clustering algorithm with improved initial center," *Int. J. Comput. Sci. Inf. Technol.*, vol. 1, no. 2, pp. 121–125, Jan. 2010.

[55] Y. Kumar and G. Sahoo, "A new initialization method to originate initial cluster centers for K-means algorithm," *Int. J. Adv. Sci. Technol.*, vol. 62, pp. 43–54, Jan. 2014.

[56] J. Yang, Y. Ma, X. Zhang, S. Li, and Y. Zhang, "An initialization method based on hybrid distance for K-means algorithm," *Neural Comput.*, vol. 29, no. 11, pp. 3094–3117, Nov. 2017.

[57] G. Tzortzis and A. Likas, "The MinMax K-means clustering algorithm," *Pattern Recognit.*, vol. 47, no. 7, pp. 2505–2516, Jul. 2014.

[58] C. S. Li, "Cluster center initialization method for K-means algorithm over data sets with two clusters," in *Proc. Eng.*, vol. 24, 2011, pp. 324–328.

[59] J. Liu, Q. Du, and J. Xu, "A learning-based adjustment model with genetic algorithm of function point estimation," in *Proc. IEEE 20th Int. Conf. High Perform. Comput. Commun., IEEE 16th Int. Conf. Smart City, IEEE 4th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Los Alamitos, CA, USA, Jun. 2018, pp. 51–58.

[60] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," *J. Syst. Softw.*, vol. 137, pp. 184–196, Mar. 2018.

[61] ISBSG. (Feb. 2, 2015). *ISBSG Development & Enhancement Repository Release 13*. [Online]. Available: http://isbsg.org

[62] J S. Shirabad and T. J. Menzies, "The promise repository of software engineering databases," School Inf. Technol. Eng., Univ. Ottawa, Ottawa, ON, Canada, 2005.

[63] J. Verner, J. Sampson, and N. Cerpa, "What factors lead to software project failure?" in *Proc. 2nd Int. Conf. Res. Challenges Inf. Sci.*, 2008, pp. 71–80.
[64] Z. Ansari, M. F. Azeem, W. Ahmed, and A. V. Babu, "Quantitative evaluation of performance and validity indices for clustering the web navigational sessions," 2015, *arXiv:1507.03340*.
[65] A. Idri, I. Abnane, and A. Abran, "Evaluating pred(p) and standardized accuracy criteria in software development effort estimation," *J. Softw., Evol. Process*, vol. 30, no. 4, pp. e192–5, Apr. 2018.
[66] E. Stensrud, T. Foss, B. Kitchenham, and I. Myrtveit, "An empirical validation of the relationship between the magnitude of relative error and project size," in *Proc. 8th IEEE Symp. Softw. Metrics*, Oct. 2002, pp. 3–12.
[67] A. J. Larner, "Effect size (Cohen's d) of cognitive screening instruments examined in pragmatic diagnostic accuracy studies," *Dementia Geriatric Cognit. Disorders Extra*, vol. 4, no. 2, pp. 236–241, May 2014.
[68] G. Jain, T. Mahara, and K. N. Tripathi, "A survey of similarity measures for collaborative filtering-based recommender system," in *Proc. Soft Comput., Theories Appl.*, Singapore, 2020, pp. 343–352.

HO LE THI KIM NHUNG received the B.S. and M.S. degrees in information systems from the University of Science (HCMUS), Vietnam, in 2010 and 2014, respectively, and the Ph.D. degree in engineering informatics from the Faculty of Applied Informatics, Tomas Bata University in Zlín, Zlín, Czech Republic, in 2022. From 2010 to 2018, she was a Lecturer with the Department of Information Systems, Faculty of Information Systems, University of Science (HCMUS). She is currently an Assistant Professor at the Faculty of Applied Informatics, Tomas Bata University in Zlín. Her research interests include database management systems, software engineering, and software effort estimation methods based on use case points.

PETR SILHAVY received the Ph.D. degree in engineering informatics from the Faculty of Applied Informatics, Tomas Bata University in Zlín, Zlín, Czech Republic, in 2009. He has expertise as the CTO and a Software Developer in database programming, database design, data management, and data science. He is currently an Associate Professor with the Faculty of Applied Informatics, Tomas Bata University in Zlín. He is also a Senior Researcher and an Associate Professor of system engineering and informatics with a demonstrated history of working in research and higher education. His research interests include prediction and empirical methods for software engineering.

RADEK SILHAVY received the Ph.D. degree in engineering informatics from the Faculty of Applied Informatics, Tomas Bata University in Zlín, Zlín, Czech Republic, in 2009. He is currently an Associate Professor and a Senior Researcher at the Faculty of Applied Informatics, Tomas Bata University in Zlín. He is also an Associate Professor of system engineering and informatics with a demonstrated history of working in research, higher education, project management, and software analysis. His research interests include predictive analytics for software engineering, empirical methods in software engineering, and prediction models focused on cost, size, and effort estimations in systems/software engineering. He is also involved in academic publishing as the Editor-in-Chief, an editor, or a reviewer.