INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 7, July 2026 | ISSN: 2456-4184 | IJNRD.ORG

A Comparative Analysis of Software Development Cost Estimation Models: From Parametric Paradigms to Machine-Learning-Driven Estimation

Dr. Ambarish Kumar Patel
Head of Department, Department of Computer Application
Anjaneya University, Raipur, Chhattisgarh, India

Abstract

Objectives: Software development cost estimation (SDCE) is one of the most ill-defined problems to which forty years of relentless research have been devoted. This study aims to scrutinize the principal families of software cost estimation models and assess their comparative validity in a light of survey-type literature review. The paper cross-references the descriptive taxonomy proposed by Rajper and Shaikh [1] and evaluates the accuracy of the competing approaches using the comprehensive data sample provided by Briand, El Emam, Surmann, Wieczorek and Maxwell [2]. Methods/Analysis: A survey of the literature was carried out between 1978 and 2026 following the standard four-step protocol, including planning, conducting, extracting and reporting. The primary sources of information used in this study consist of the survey by Rajper and Shaikh [1] and the extensive project data sample collected by Briand, El Emam, Surmann, Wieczorek and Maxwell [2]. The latter reference is used to compare the accuracy of ordinary least-squares regressions, stepwise ANOVA, CART, analogy-based estimation and their combinations using five metrics, including MMRE, MdMRE and Pred(0.25). These descriptive statistics were re-calculated for verification purposes during the extraction process, and two computational errors were detected and corrected, namely, a discrepancy between the text of section 3.1 and Table 1, and a repetition of Table 4 caption. In addition to supporting the findings of Rajper and Shaikh [1] regarding the increased diversity of the contemporary estimation practices, the analysis processes the literature of the last five years in order to evaluate if the subsequent advances in deep learning and agile estimation practices would affect the conclusion. Findings: The review findings confirm that the body of estimation practice is growing primarily due to the increased complexity of the software development process, which renders any single mathematical model inadequate. However, as demonstrated by the analysis of [2], any changes in estimation accuracy achieved by varying the choice of terms in the regression equation or using alternative model families (CART, analogy) are mostly negligible, while applying estimation methods at the level of single organizations fails to deliver value in most cases. Application/Improvements: The paper argues that the potential benefits of model optimization are limited, while the improvements in organizational domain knowledge and input data mining should be prioritized, and hybrid machine-learning approaches to software cost estimation should be evaluated as data-conditioning techniques.

Keywords: Agile Effort Estimation , Analogy-Based Estimation , CART , COCOMO , Data Mining Techniques , Deep Learning , Effort Estimation , Function Points , Machine Learning Techniques , MMRE , Regression Analysis , Software Development Cost Estimation (SDCE) , SLIM , Story Points.

1. Introduction

Software Development Cost Estimation (SDCE) is the process of forecasting the exertion — expressed as human effort, elapsed schedule and monetary cost — required to build and subsequently maintain a software product or system [1], [3]. It is conventionally treated as a sub-discipline of software engineering, and it forms the foundation upon which project bidding, budgeting, staffing, planning and control decisions are erected. Where estimation is sound, the downstream project process is smoother; where it is unsound, every subsequent management decision inherits the error.

The estimation of software development cost has been an object of concentrated research attention for well over four decades, and a considerable variety of methods has been applied in order to explain development cost as a function of a large number of potentially relevant cost factors [2]. Techniques have been imported from statistics, from machine learning and from knowledge acquisition, and have been applied with varying degrees of success — although, historically, on a limited number of mostly small and medium-sized data sets. Different cost estimation techniques emerged in successive waves across the last three decades [1], and since the 1980s numerous techniques have been proposed specifically within the SDCE space, most of them concentrating on the estimation of software complexity in terms of man-effort and lines of code.

The literature contains a number of prior consolidations. Jørgensen and Shepperd [4] identified eleven distinct SDCE technique categories in their systematic review; Wen, Li, Lin, Hu and Huang [5] identified eight machine-learning

techniques applied to development effort estimation between 1991 and 2010; and Boehm, Abts and Chulani [6] organised the field into model-based, expertise-based, learning-oriented, dynamics-based, regression-based and composite families. Rajper and Shaikh [1] extended this consolidation through 2016 and demonstrated that the pattern of model evolution has not stabilised.

Yet a review that is purely descriptive cannot resolve the question that practitioners actually pose. Taxonomies enumerate what exists; they do not establish what works. Despite intense research activity, few generalisable conclusions can be drawn from the comparative literature, for three structural reasons identified by Briand et al. [2]: many evaluation studies consider only a small subset of available techniques; such studies are usually based on small data sets, which makes results vulnerable to the idiosyncrasies of the data at hand; and initial evaluations are rarely replicated, so confidence in the conclusions drawn must remain limited. Even where the same data set recurs across studies, results are often not comparable because the experimental designs differ — Briand et al. and Srinivasan and Fisher [7], for instance, both employed the COCOMO and Kemerer data sets but partitioned them differently into training and test sets.

1.1 Research Objectives

The present study is undertaken with four objectives:

- To construct a consolidated taxonomy of SDCE models spanning the parametric, expertise-based, learning-oriented, composite and data-mining paradigms, and to trace the historical evolution of the dominant COCOMO family.
- To place the descriptive taxonomy alongside quantitative, cross-validated accuracy evidence drawn from a large multi-organisational database, so that the classificatory and the evidential views of the field can be read against one another.
- To determine whether the choice of modelling technique, or the character of the underlying data, is the dominant determinant of estimation accuracy.
- To assess whether organisation-specific cost data confers a measurable advantage over pooled, multi-organisation data when standard cost factors are used.

1.2 Research Questions

Following the formulation adopted by Briand et al. [2], the empirical component of this study addresses two questions, to which a third, synthetic question is added:

RQ1. Which modelling technique performs best (a) where local, company-specific data is available, and (b) where multi-organisation data must be used?

RQ2. What are the advantages of company-specific data as compared with multi-organisation data drawn from the same application domain?

RQ3. Given the answers to RQ1 and RQ2, where should further research investment be directed — towards the modelling algorithm, or towards the measurement programme that supplies it?

The remainder of this paper is organised as follows. Section 2 reviews the related work and situates the study. Section 3 describes the review protocol and the empirical method, including the data set, the modelling techniques compared, the evaluation criteria and the cross-validation design. Section 4 presents the taxonomy of SDCE techniques. Section 5 presents and analyses the quantitative results. Section 6 discusses the findings and their implications for measurement practice. Section 7 states the threats to validity, and Section 8 concludes.

2. Related Work

Comparative evaluation of cost modelling techniques has a long lineage. During the 1980s, the widely used parametric models — Boehm's COCOMO [8], Putnam's SLIM [9], and the Function Point family of Albrecht [10] and Albrecht and Gaffney [11] — were compared using data sets of widely varying size and provenance. A recurring conclusion of that period was that these models perform poorly when applied uncalibrated to environments other than those from which they were derived [12], [13], [14].

Kemerer [12] provides the canonical illustration. Using fifteen business application projects, he compared SLIM, COCOMO, Estimacs and Function Points, and reported estimation error in terms of the Mean Magnitude of Relative

Error ranging from 85% to 772% — an interval so wide that it undermines the practical utility of uncalibrated parametric transfer. Conte, Dunsmore and Shen [15] drew on six data sets from widely differing environments and reported MMRE variation between 70% and 90% across SLIM, COCOMO and Jensen's model [16]. Their response was constructive: they proposed the COPMO model, calibrated separately to each of the six data sets, and thereby obtained an MMRE of 21%. The lesson embedded in that result is that local calibration, rather than model form, accounted for the improvement.

The 1990s admitted non-parametric techniques based on machine learning and analogy. Shepperd and Schofield [17] compared analogy-based estimation with stepwise regression across nine data sets drawn from different domains and reported that analogy outperformed stepwise regression in terms of MMRE in every case. Mukhopadhyay, Vicinanza and Prietula [18] applied Kemerer's data set and found that their analogy-based model Estor, built on case-based reasoning, outperformed COCOMO. Finnie, Wittig and Desharnais [19] compared case-based reasoning against several regression models using function points and artificial neural networks on a database of 299 projects contributed by 17 organisations; they reported better performance for CBR relative to function-point-based regression, but found that artificial neural networks in turn outperformed the CBR approach.

Srinivasan and Fisher [7] widened the comparison to include regression trees, artificial neural networks, function points, COCOMO and SLIM. Training on the 63-project COCOMO data set and testing on Kemerer's fifteen projects, they found that regression trees outperformed both COCOMO and SLIM, and that neural networks and function-point-based prediction in turn outperformed the regression trees. Briand, Basili and Thomas [20] combined the COCOMO and Kemerer data sets to compare COCOMO, stepwise regression and Optimized Set Reduction (OSR), a non-parametric machine-learning technique; OSR outperformed both comparators. Jørgensen [21] tested several variations of regression, neural networks and OSR-with-regression combinations on 100 maintenance projects, finding that two multiple regression models and a hybrid combining OSR with regression performed best, and recommending that sophisticated prediction models be used together with expert estimates in order to justify the investment they require.

More recent work has extended these comparisons into the data-mining register. Khalifelu and Gharehchopogh [22] contrasted artificial neural networks, linear regression, k-nearest neighbours and support vector regression against algorithmic models. Dejaeger, Verbeke, Martens and Baesens [23] conducted a broad comparative study of data mining techniques for effort estimation, cataloguing least median squares regression, model trees, MARS, multilayer perceptron networks, radial basis function networks, least-squares support vector machines, robust regression, OLS regression with logarithmic and Box–Cox transformations, ridge regression, case-based reasoning and CART. Hybridisation has become the characteristic move of the last decade: Molani, Ghaffari and Jafarian [24] combined a radial basis function neural network with a genetic algorithm; Dizaji and Gharehchopogh [25] hybridised ant colony optimisation with chaos optimisation; and Gharehchopogh, Ebrahimi, Maleki and Gourabi [26] combined particle swarm optimisation with fuzzy C-means clustering and learning automata.

Two limitations run through this body of work and motivate the present study. First, although parametric techniques appear in almost every comparison, the comparisons themselves remain partial: only certain techniques are evaluated in any given study. Second, replication is rare, and where the same data are reused the experimental designs diverge sufficiently to preclude direct comparison. Many studies, furthermore, rely on small data sets drawn from heterogeneous environments, which makes generalisable conclusions about model performance difficult to sustain [2].

3. Materials and Methods

3.1 Review Protocol

The literature component of this study was executed as a structured review covering the period 1978–2026, organised into the four sequential stages of planning, conducting, data extraction and reporting, following the workflow adopted by Rajper and Shaikh [1]. Candidate studies were selected on the basis of publication in established software engineering journals and conference proceedings, with priority given to primary empirical studies, systematic reviews and models that have entered common industrial use. The objective of the review was explicitly enumerative: to establish which techniques exist, how they relate to one another, and what evidence has been offered for each. The window extends approximately a decade beyond that of the anchor survey [1] in order to determine whether the deep learning turn in software analytics has altered the substantive conclusions; this material is treated separately in Section 4.6 and Section 6.5.

3.2 Data Verification Procedure

Because this study re-analyses statistics reported in the primary sources rather than re-executing the original experiments, every numeric value transcribed into Tables 4 to 8 was subjected to two consistency checks before use. First, internal arithmetic consistency was tested wherever a table permitted a derived quantity to be recovered — for example, whether the reported subset and whole-database means are mutually compatible given the reported observation counts. Second, each numeric claim made in the narrative of the source was checked against the table it cites. Two discrepancies were identified by this procedure and are documented at the point of use in Section 5; neither affects the substantive conclusions, but both would propagate silently into any study that transcribed the figures uncritically.

3.3 Empirical Data Set

The quantitative analysis reported in Section 5 is based on the Experience database, established in cooperation with sixteen Finnish companies [2]. Participating companies purchase the Experience tool [27] and pay an annual maintenance fee, in return for which they receive the tool with the embedded database, software updates and refreshed data sets. Companies may add their own data, and are given a financial incentive — a reduction in the maintenance fee for each project contributed — to donate data to the shared repository. Validity and comparability are assured procedurally: all companies collect data using the same instrument, every variable is precisely defined, and contributing companies are individually contacted in order to verify and check each submission.

At the time of the study the database comprised 206 software projects contributed by 26 companies. The projects are predominantly business applications drawn from banking, wholesale and retail, insurance, public administration and manufacturing. This domain homogeneity supports generalisation to other projects within the business application domain. Six companies contributed more than ten projects each, and one company contributed approximately one third of the entire database — a structural feature that makes it possible to compare a company-specific model against a multi-organisation model on the same test data. System size is measured in Experience Function Points, a variant of Albrecht's Function Point measure [10] in which the same five functional categories are retained but complexity weights are assessed on a five-point rather than a three-point scale.

Table 1: Variables considered in the Experience (Laturi) database [2]

Variable	Description	Scale	Values / Range / Unit
Effort	Total project effort	Ratio	Person hours (ph)
EFP	System size measured in function points	Ratio	Unadjusted Experience Function Points
BRA	Organisation type	Nominal	Banking, wholesale/retail, insurance, manufacturing, public administration
APP	Application type	Nominal	Customer service, MIS, office information systems, process control and automation, network management, transaction processing, production control and logistics, on-line and information service
HAR	Target platform	Nominal	Networked, mainframe, PC, mini computer, combined
F1–F15	Fifteen productivity factors: customer participation, development environment, staff availability, level and use of standards, methods and tools, logical complexity of the software, requirements volatility, quality requirements, efficiency requirements, installation requirements, analysis skills of staff, application experience of staff, tool skills of staff, project and team skills of staff	Ordinal	1–5 (very small – very large)

3.4 Criteria for Inclusion of Modelling Techniques

The comparative study considered a deliberate subset of the available techniques, admitted according to three criteria [2]:

Automatable. Because the accuracy evaluation employs a computationally intensive cross-validation procedure, only techniques capable of substantial automation could be considered. This criterion excludes labour-intensive procedures such as that of Kitchenham [28].

Applied in software engineering. There must exist a precedent for the technique's use within software engineering, and preferably within cost estimation specifically, so that the technique carries an initial demonstrated utility.

Interpretable. The results must be interpretable. A technique that produced difficult-to-interpret results would not constitute a useful recommendation, because project managers are in practice unlikely to apply a model they cannot understand. This criterion excludes artificial neural networks — a point of some consequence, since it means the most accurate technique reported in several prior studies is deliberately set aside on grounds of usability rather than performance.

On this basis, four base techniques were compared — ordinary least squares regression, stepwise ANOVA for unbalanced data sets, CART, and analogy-based estimation — together with two hybrid configurations, CART combined with regression and CART combined with analogy.

3.5 Modelling Techniques Applied

3.5.1 Ordinary Least Squares Regression

Multivariate least squares regression was applied by fitting the data to a specified effort model [29]. An exponential model specification was selected because linear specifications exhibited marked heteroscedasticity, violating an assumption of regression analysis. A mixed stepwise procedure selected variables with a significant influence on effort at $\alpha = 0.05$. Dummy variables handled categorical, nominally scaled variables. Ordinal variables were treated as though measured on an interval scale — a treatment justified by Spector [30], who showed that there is practically no difference between using scales with equal and unequal intervals, and by Bohrnstedt and Carter [31], who established that ordinal scales are usable as interval scales under parametric statistics.

3.5.2 Stepwise ANOVA

An analysis of variance procedure was applied to construct an effort prediction model, using the Stata tool [32] to analyse the variance of unbalanced data and fit regression estimates to models containing categorical variables. Models were built using a forward pass method resembling that of Kitchenham [28], differing in that productivity factors are treated as interval variables, interactions among independent variables are not ignored, and construction of successive one-, two- and three-variable models is less labour-intensive. The resulting model takes the multiplicative form

$$Effort = a \times EFP^{b} \times F1^{c} \times F2^{d} \times \ldots$$

where a is a constant varying with the significant class variables and F denotes a significant productivity factor at $\alpha = 0.05$, with interaction effects of class variables taken into consideration. To avoid multicollinearity, any two variables whose Spearman rank correlation coefficient exceeded ±0.75 were treated as highly correlated and were not admitted to the same model. Residual-versus-fitted plots were examined for violations of least squares assumptions; the Ramsey RESET test was used to detect omitted variables and the Cook–Weisberg test to check for heteroscedasticity.

3.5.3 Classification and Regression Trees (CART)

Regression tree models were developed using the CART algorithm [33] as implemented in the CART tool [34]. A regression tree is a collection of conditional rules of the form "if condition 1 and condition 2 … then Z", represented as a binary tree; here the dependent variable was productivity rather than effort. Each internal node specifies a condition on a project variable influencing productivity, and each branch corresponds to admissible values of that variable. Regression trees accommodate variables measured on different scale types, which is an advantage in a data set mixing ratio, nominal and ordinal measures.

Tree construction proceeds by binary recursive partitioning until a stopping criterion is met, the splitting criterion being the split that most successfully separates the projects' productivity values. The stopping criterion was set at a minimum of twenty observations for the penultimate node. Optimal trees were selected as those with minimal overall absolute deviation between actual and predicted productivity and at least ten observations in each terminal node. Median rather than mean productivity values were used as predictions in order to limit the influence of outliers. A project is classified by descending the tree from the root until a terminal node is reached, at which point effort is obtained by dividing actual size in EFP by the median productivity of that node.

3.5.4 Analogy-Based Estimation

The governing idea of analogy-based estimation is to identify the completed projects most similar to a new project. Three design decisions dominate: the choice of similarity or distance function, the selection of relevant project attributes (here, cost drivers), and the number of analogues to retrieve. The approach applied follows Shepperd and Schofield [17], [35] and the ANGEL tool [36], implemented using CBR-Works 4.0 beta [37]. The distance function is the unweighted Euclidean distance over variables normalised to the interval [0, 1]:

$$distance(P_i, P_j) = \sqrt{[\ (\Sigma_{k=1..n}\ \delta(P_{ik}, P_{jk})\ ) / n\ ]}$$

where n is the number of variables and δ is defined as the squared normalised difference for continuous variables, as 0 for categorical variables taking equal values, and as 1 for categorical variables taking unequal values. Because exhaustive search over variable combinations — as performed by ANGEL — is computationally prohibitive for the 19 variables and 119 projects involved here (of the order of 5.24×10^5 combinations), the strategy of Finnie et al. [19] was adopted instead: a two-tailed t-test was applied to all categorical variables to identify those with significant influence on productivity, and two levels were generated for each variable by merging its original levels. Prediction used both the single most similar project (Analogy-1s) and the unweighted average of the two most similar projects (Analogy-2s), a simplification justified by the finding of Shepperd and Schofield [35], [38] that accuracy is near-equivalent when more than two analogues or weighted averages are used.

3.5.5 Hybrid Configurations

Two hybrids were evaluated. CART with regression develops a regression tree and then applies regression analysis within each terminal node, so that effort is predicted by a node-specific equation rather than by a median value; a simple univariate exponential relationship between effort and system size was fitted on the observations in each terminal node. CART with analogy develops a regression tree and then applies analogy-based selection within the project subset corresponding to the terminal node reached.

3.6 Evaluation Criteria

Accuracy is assessed through the Magnitude of Relative Error (MRE) [15], defined for observation i as the absolute difference between actual and predicted effort divided by actual effort. Aggregation across N observations yields the Mean MRE (MMRE). Because MMRE is sensitive to individual predictions with excessively large relative errors, the median of the MRE values (MdMRE) is reported as a robust complement. A third criterion, Pred(l), reports the proportion of observations for which MRE is less than or equal to l; Pred(0.25) is used throughout. An implicit assumption of MRE-based evaluation is that acceptable error is proportional to project size — a ten person-month overestimate is more serious on a ten person-month project than on a hundred person-month project.

3.7 Cross-Validation Design

Computing model accuracy on the same data used to construct the model yields optimistic estimates: error is artificially depressed and does not reflect performance on unseen data [29]. Two cross-validation designs, each congruent with one of the research questions, were therefore employed.

To assess generic, multi-organisation models, test sets were formed from the projects of a single organisation, restricted to organisations contributing ten or more projects — 119 observations across six such organisations. For each, the training set was the whole database minus that organisation's projects, producing six hold-out samples. Accuracy computed this way indicates what an organisation can expect when it builds a model from an external multi-organisational data set and applies it to its own projects.

To assess local models, the 63 projects contributed by the single largest contributor — a banking organisation, hereafter the bank data — were partitioned into six randomly constructed test sets, with the remaining projects serving as the training set in each fold. Accuracy computed this way indicates what an organisation can expect when it builds a model from its own historical data.

4. A Consolidated Taxonomy of SDCE Techniques

SDCE encompasses the determination of effort in human-months, of project duration, and of total cost in monetary terms [3]. The models that perform this determination are conventionally divided into empirical models, which use data from previous projects to evaluate and estimate current ones — COCOMO [8], [39] being the archetype — and analytical

models, which estimate by formula [9], [40], [41]. Following Boehm, Abts and Chulani [6] and Rajper and Shaikh [1], the field may be organised into the families summarised in Table 2.

Table 2: Consolidated taxonomy of SDCE model families

Family	Representative techniques	Principal basis	Characteristic limitation
Model-based	SLIM, Checkpoint, PRICE-S, ESTIMACS	Proprietary parametric equations calibrated on historical project data	Proprietary formulation prevents direct comparison; poor uncalibrated transfer
Expertise-based	Delphi technique, Work Breakdown Structure	Judgement of experienced personnel and lessons from past projects	Non-repeatable; sensitive to individual bias; not automatable
Learning-oriented / ML	SVR, ANN, Bayesian networks, genetic programming, association rules, genetic algorithms, CBR, decision trees	Induction of a cost function from historical observations	Interpretability deficit; sensitivity to small and noisy training sets
Regression-based	OLS regression, stepwise ANOVA, robust and ridge regression, OLSR with Box–Cox transformation	Statistical fitting of effort to significant cost drivers	Assumption violations (heteroscedasticity, multicollinearity) require careful handling
Composite / hybrid	Bayesian calibration in COCOMO II, CART+regression, CART+analogy, ANN+GA, ACO+chaos, PSO+fuzzy C-means	Deliberate combination of two or more paradigms to offset individual weaknesses	Increased complexity and tuning burden; gains often not statistically significant
Dynamics-based	System dynamics models of the development process	Modelling of feedback and time-varying project behaviour	High modelling cost; limited empirical validation in the cost estimation literature

4.1 Model-Based Techniques

4.1.1 Putnam's Software Life-cycle Model (SLIM)

SLIM was introduced by Larry Putnam in the 1970s within the quantitative SDCE paradigm, and rests on Putnam's analysis of the software life cycle [8], [9]. The model derives from a Rayleigh distribution of project personnel level against time. The shape of the resulting curve is governed by the Manpower Buildup Index (MBI) and the Productivity Factor (PF). A practical strength of the model is that both parameters can be recorded and analysed from completed projects; where such data are unavailable, a structured questionnaire can be used to derive MBI and PF values from the reference database [6].

4.1.2 Checkpoint

Checkpoint was introduced by Capers Jones and is founded on knowledge-based software project estimation as developed in the 1980s [6], [42]. Function and feature points serve as its basic inputs, and its emphasis falls on three areas: estimation, measurement and assessment.

4.2 Expertise-Based Techniques

In the absence of empirical data, the judgement of relevant personnel and experts is elicited to predict cost on the basis of experience and lessons drawn from past projects. Two techniques dominate: the Delphi technique [8], [6], [43], which structures anonymous iterative elicitation until estimates converge, and the Work Breakdown Structure [6], [44], which decomposes the project hierarchically so that estimation occurs at a granularity where judgement is more reliable. The evident weakness of this family is non-repeatability; its equally evident strength is that it functions where no historical data exist at all, and Jørgensen [21] concluded that sophisticated prediction models are best deployed alongside expert estimates rather than instead of them.

4.3 Learning-Oriented and Hybrid Techniques

Learning-oriented techniques comprise both some of the oldest and some of the newest instruments applied to estimation [6], ranging from case studies to neural networks. Wen et al. [5] identified eight machine learning techniques applied to SDCE between 1991 and 2010: support vector regression, artificial neural networks, Bayesian networks, genetic programming, association rules, genetic algorithms, case-based reasoning and decision trees. The hybrid turn is well

represented: a radial basis function neural network coupled with a genetic algorithm [24]; ant colony optimisation combined with chaos optimisation [25]; and particle swarm optimisation hybridised with fuzzy C-means clustering and learning automata [26]. Ebrahimpour, Gharehchopogh and Khalifehlou [45] and Gharehchopogh and Pourali [46] report comparable neural–evolutionary combinations.

4.4 Composite Techniques and the COCOMO Suite

Because every technique enumerated above carries characteristic advantages and disadvantages, researchers have introduced composite techniques that incorporate two or more approaches. The Bayesian approach is the most consequential instance, since it became the basis for the development of COCOMO II [6]. COCOMO — the Constructive Cost Model — was first published by Barry Boehm [8], and although many cost estimation models for software products have been introduced, COCOMO remains the dominant one. It is conventionally explained as a three-level model:

Basic COCOMO is a single-valued static model that computes effort by treating software cost as a function of program size, where size is expressed in lines of code [6], [47].

Intermediate COCOMO computes development effort systematically as a function of program size together with a set of fifteen cost drivers [25], [48].

Detailed COCOMO computes all drivers defined by the intermediate model and additionally assesses the effect of each cost driver on each development phase [6], [48].

The suite has been extended repeatedly. From COCOMO 81 (1981) and COCOMO II (2000), the family branched into COQUALMO (1998), iDAVE (2003), COPLIMO (2003), COPSEMO (1998), COPROMO (1999), CORADMO (1999), DBA COCOMO (2004), COINCOMO (2004) and a Security Extension (2004), alongside independent estimation models such as COCOTS (2000), COSYSMO (2002), COSoSIMO (2004) and the Costing Secure System (2004) [49]. The persistence of this branching is itself an empirical finding: it indicates that no single calibration has proved adequate across the domains, delivery models and lifecycle forms that software development has successively adopted.

Table 3: Cost factors and activities covered by principal SDCE models (adapted from [5], [1]); ● = covered, ? = not identified, × = not covered

Group	Factor	Checkpoint	SLIM	ESTIMACS	PRICE-S	COCOMO II
Program attributes	Type / domain	●	●	●	●	●
	Complexity	●	●	●	●	●
	Language	●	●	?	●	●
	Reuse	●	●	?	●	●
	Required reliability	?	?	●	●	●
Computer attributes	Resource constraints	?	●	?	●	●
	Platform volatility	?	?	?	?	●
Personnel attributes	Personnel capability	●	●	●	●	●
	Personnel continuity	?	?	?	?	●
	Experience	●	●	●	●	●
Project attributes	Tools and techniques	●	●	●	●	●
	Breakage	●	●	●	●	●
	Schedule	●	●	●	●	●
	Process maturity	●	?	?	?	●
	Team cohesion	●	?	?	●	●
	Security issues	?	●	?	?	●
Activities covered	Inception	●	●	●	●	●
	Elaboration	●	●	●	●	●

Group	Factor	Checkpoint	SLIM	ESTIMACS	PRICE-S	COCOMO II
	Construction	●	●	●	●	●
	Transition and maintenance	●	●	×	●	●
Size attributes	Source instructions	●	●	×	●	●
	Function points	●	●	●	●	●
	OO-related metrics	●	●	?	●	●

4.5 Data Mining Techniques in SDCE

Although Boehm's COCOMO [8], [39], [49] remains the most widely deployed SDCE model, data mining techniques have been applied with increasing frequency in recent years. Khalifelu and Gharehchopogh [22] compared artificial neural networks, linear regression, k-nearest neighbours and support vector regression. Finnie et al. [19] applied ordinary least squares regression and case-based reasoning. Briand, Langley and Wieczorek [50] compared CBR, CART, OLSR and ANOVA. Sajadfar and Ma [51] proposed a hybrid cost estimation framework based on feature-oriented data mining. Dejaeger et al. [23] catalogued the techniques recurring across the comparative literature — least median squares regression, model trees, MARS, multilayer perceptron networks, radial basis function networks, least-squares support vector machines, robust regression, log-transformed and Box–Cox-transformed OLS regression, ridge regression, case-based reasoning and CART — drawing on studies including Chiu and Huang [52], Park and Baek [53], Kumar, Ravi, Carr and Kiran [54], Li, Xie and Goh [55], Strike, El Emam and Madhavji [56] and Li and Ruhe [57].

4.6 Developments Beyond the Anchor Review Window (2017–2026)

The taxonomy set out above terminates, in its source form, in 2016. Extending the window by a decade adds two substantive branches that the earlier classification does not accommodate, and it is necessary to establish whether either disturbs the conclusions of Section 5.

The first branch is deep learning. Where the learning-oriented family of Section 4.3 consisted principally of shallow learners applied to tabular cost-driver data, the post-2016 literature applies representation learning to unstructured project artefacts. Iordan [62] reports an optimised LSTM network for development effort estimation; Draz, Emam and Azzam [63] combine a convolutional neural network with particle swarm optimisation; Phan and Jannesari [64] apply heterogeneous graph neural networks to the problem. Metaheuristic hybridisation has continued along the trajectory already visible in 2015: Alsheikh and Munassar [65] apply grey wolf optimisation to estimation model parameters, and Ardiansyah, Ferdiana and Permanasari [66] propose a modified chaotic particle swarm variant. Watson et al. [67] provide the broader disciplinary context in their systematic review of deep learning across software engineering research, and Rossi and Fontoura [68] survey AI-based approaches to task-level effort estimation specifically.

The second branch is agile estimation, which is not a new technique so much as a new estimand: the unit of prediction shifts from whole-project effort to the story point assigned to a user story or issue. Choetkiertikul, Dam, Tran, Pham, Ghose and Menzies [69] provide the landmark contribution, combining long short-term memory with a recurrent highway network in the Deep-SE model and releasing a dataset of 23,313 issues drawn from 16 open source projects. Fu and Tantithamthavorn [70] extend the approach using a transformer architecture in GPT2SP. Tawosi and colleagues [71] subsequently released the TAWOS dataset, assembled from Jira repositories and covering 44 agile open source projects with more than half a million issues — an order-of-magnitude increase in available data over anything reported in Section 3.3. Alsaadi and Saeedi [72] address ensemble estimation for novice agile teams, and Sánchez-García et al. [73] compare statistical models against machine learning for design-stage estimation.

This material changes the scope of the taxonomy considerably. Whether it changes the answer to RQ3 is a separate question, taken up in Section 6.5.

5. Results and Analysis

5.1 Descriptive Statistics

Table 4 summarises system size in Experience Function Points and project effort in person-hours for the bank subset and for the whole cross-validation database, and adds a third pair of columns, derived in the present study, for the

remainder of the database once the bank projects are excluded. The organisational composition of the whole database is 38% banking, 27% insurance, 19% manufacturing, 9% wholesale and 7% public administration. The distributions of application type and target platform for the bank data closely track those of the whole database, which materially assists interpretation of the comparative results: differences in accuracy cannot easily be attributed to differences in project composition.

Table 4: Descriptive statistics for system size and effort. Bank and whole-database columns as reported in [2]; remainder columns derived in the present study.

Statistic	Bank data — Size (EFP)	Bank data — Effort (ph)	Whole DB — Size (EFP)	Whole DB — Effort (ph)	Remainder — Size (EFP)*	Remainder — Effort (ph)*
Minimum	48	583	48	480	—	480
Mean	671.4	8109.54	718.7	8352.52	771.9	8625.9
Maximum	3634	63694	3634	63694	—	—
Std. deviation	777.3	10453.9	679.69	9707.23	—	—
Observations	63	63	119	119	56	56

* Remainder means recovered from the reported subset and whole-database means weighted by their observation counts; dispersion statistics are not recoverable by this method and are therefore not reported.

This decomposition exposes the first of the two discrepancies noted in Section 3.2. The source states that projects from the bank exhibit, on average, higher effort than projects in the remainder of the database. The reported figures do not support this. The whole-database mean effort (8352.52 ph over 119 observations) and the bank mean (8109.54 ph over 63 observations) jointly imply a remainder mean of approximately 8625.9 ph over 56 observations — that is, the bank projects carry lower rather than higher mean effort than the projects they are compared against, by roughly 6%. The same computation gives a remainder mean size of approximately 771.9 EFP against the bank's 671.4 EFP, so the bank subset is somewhat smaller in system size as well. The correct characterisation is therefore that the bank subset is marginally smaller and less effortful on average than the rest of the database, while exhibiting greater dispersion in effort relative to its mean (a coefficient of variation of 1.29 against 1.16 for the whole database).

The direction of this correction matters for the interpretation offered later in the source, which attributes the comparatively strong showing of analogy on the bank data to the greater homogeneity of that subset. That attribution survives the correction — indeed the productivity-variance argument advanced in Section 5.3 is independent of which subset has the higher mean effort — but the descriptive claim on which it is loosely premised does not, and is corrected here.

5.2 Comparison of Modelling Techniques on the Whole Database

A note on correspondence is required before the results are presented, and constitutes the second discrepancy identified by the verification procedure of Section 3.2. The source [2] carries a caption numbering error: two distinct results tables are both captioned "Table 3", and the table that its narrative consistently refers to as Table 5 is captioned "Table 4". The intended sequence is recoverable unambiguously from the narrative, which states that the first table uses the six companies with more than ten projects as hold-out samples, the second uses randomly selected samples from the bank data, and the third applies models built on the remaining Experience data to the bank projects. The correspondence adopted here is therefore: Table 5 below = the whole-database results; Table 6 = the local bank-data results; Table 7 = the external-model-on-bank-data results. Readers consulting the original should note that its printed captions do not follow this sequence.

Table 5 reports cross-validation results over the six company hold-out samples using the entire database as training data. The pattern is unambiguous in one respect and instructive in another. Techniques not involving analogy — stepwise regression, stepwise ANOVA, CART and CART+regression — outperform every configuration involving analogy. Analogy-based configurations exhibit MdMRE values up to 22 percentage points higher than CART, which yields the best results on all three criteria simultaneously: lowest MMRE (0.52), lowest MdMRE (0.39) and highest Pred(0.25) (34%). The statistical significance of this gap is confirmed by the matched-pair Wilcoxon signed rank test [58], a non-parametric analogue of the t-test, applied to the MRE values yielded by the competing models. Among the analogy-based techniques themselves, however, no statistically significant difference in MRE is detectable.

Table 5: Average results over all hold-out samples using the entire database [2]

Criterion	Stepwise Regr.	Stepwise ANOVA	CART	CART+Regr.	Analogy-1s	Analogy-2s	CART+Analogy-1s	CART+Analogy-2s
MMRE	0.53	0.567	0.52	0.52	1.16	1.34	1.25	1.39
MdMRE	0.44	0.446	0.39	0.42	0.61	0.57	0.57	0.602
Pred(0.25)	31%	34%	34%	34%	24%	17%	24%	16%

The instructive respect concerns the non-analogy techniques taken among themselves. Simple CART appears to perform slightly better than regression, stepwise ANOVA, or CART in combination with regression — but the differences are not practically significant. The spread in MdMRE across these four techniques remains at or below five percentage points, MMRE values are closely comparable, and the differences in the underlying MRE values are not statistically significant. The choice among these four is therefore not, on this evidence, a choice that materially affects accuracy.

The weak showing of analogy stands in sharp contrast to what was found in other comparative studies using different data sets [19], [18], and since the analogy procedure applied here closely follows that of Shepperd and Schofield, the divergence warrants further investigation rather than immediate acceptance. One plausible explanation lies in the construction of the similarity function: all variables entering the similarity computation exert equal influence on the selection of the most similar project, so an irrelevant or noisy variable degrades neighbour selection as much as an informative one improves it.

5.3 Comparison on the Bank Data

Table 6 reports results on the bank data using randomly constructed hold-out samples, that is, the local-model scenario. Here all techniques display broadly similar accuracy. Variation in MdMRE is confined to seven percentage points and no statistically significant difference among the techniques' MRE values is detectable. Analogy-based techniques therefore perform relatively better in the single-organisation context than in the multi-organisation context.

Table 6: Average results over all hold-out samples using the bank data [2]

Criterion	Stepwise Regr.	Stepwise ANOVA	CART	CART+Regr.	Analogy-1s	Analogy-2s	CART+Analogy-1s	CART+Analogy-2s
MMRE	0.67	0.787	0.569	0.655	0.71	0.75	0.81	0.73
MdMRE	0.41	0.424	0.462	0.471	0.48	0.47	0.43	0.47
Pred(0.25)	22%	22%	29%	25%	14%	21%	22%	19%

This differential is itself explanatory. The bank data set exhibits much lower variance in productivity than the pooled database, and analogy is more sensitive to homogeneity than the other models are, because its prediction rests on one or two selected neighbours rather than on a fitted surface. Where productivity variance is low, selecting an inadequate most-similar project carries lesser consequences for the resulting prediction; where variance is high, the same selection error propagates directly into the estimate. Analogy-based prediction is thus more affected by outliers than techniques that average across the training set.

5.4 Local Models Versus Multi-Organisation Models

Table 7 reports the accuracy obtained when models built on the remaining Experience data — that is, on external multi-organisation data — are applied to the bank projects. Comparing Table 7 against Table 6 answers RQ2 directly. All techniques show similar MRE values across the two conditions. The largest difference in MdMRE, 13 percentage points, occurs for Analogy-1s (0.48 locally versus 0.61 externally); none of the differences is statistically significant.

Table 7: Results on the bank data using models built from the entire (external) database [2]

Criterion	Stepwise Regr.	Stepwise ANOVA	CART	CART+Regr.	Analogy-1s	Analogy-2s	CART+Analogy-1s	CART+Analogy-2s
MMRE	0.57	0.629	0.54	0.524	1.32	1.41	1.48	1.46
MdMRE	0.47	0.494	0.46	0.46	0.61	0.56	0.56	0.605
Pred(0.25)	25%	28%	27%	29%	21%	14%	24%	16%

On this analysis alone, therefore, there appears to be no advantage in developing company-specific effort estimation models when those models are built from generic cost factors and sizing measures. A structural explanation is available: the distribution of projects in terms of application domain (APP) and target platform (HAR) is similar for the single-

organisation data set and for the whole database, and these two variables were identified as important cost drivers by almost every technique compared. The bank's projects may therefore be similar in nature to the remainder of the database, and pooling costs the model little.

5.5 Selected Model Variables

Table 8 summarises how frequently each independent variable was selected across the six models generated per technique under six-fold cross-validation, considering models built on the whole database. Beyond system size (EFP), which is the dominant cost driver, organisation type (BRA) and target platform (HAR) are identified as the most influential factors on cost by all modelling techniques. Where stepwise ANOVA and stepwise regression are applied, requirements volatility (F8) emerges as a further important cost driver. Multivariate regression, ANOVA and the analogy-based approach all identify quality requirements (F9) as important in most constructed models.

Table 8: Frequency of variable selection across models (n ≤ 6 selections per variable) [2]

Technique	Variables selected (variable, frequency)
Stepwise regression	EFP 6, BRA 6, HAR 6, APP 1, F5 1, F7 3, F8 6, F9 3, F11 1
Stepwise ANOVA	EFP 6, BRA 6, HAR 6, F7 1, F8 6, F9 5
CART*	BRA 6, HAR 4
CART + regression	EFP 6, BRA 6, HAR 4
Analogy†	EFP 6, BRA 6, HAR 6, APP 6, F4 6, F9 6, F10 6, F11 6, F13 6, F14 6

* CART-only models used productivity as the dependent variable, so size in EFP does not appear in the selected independent variable list. † Covers all analogy variants and all combinations of analogy with CART.

The variable-selection evidence carries an implication that the accuracy tables alone do not. The generic factors considered — although several are demonstrably useful predictors — do not explain a large part of the effort variation present in the data. The MRE results yielded by every model are, from a practical perspective, far from satisfactory for cost estimation purposes, notwithstanding the large number of factors collected and the rigour of the data collection procedures. Standard cost factors, in short, may simply not be enough [59].

6. Discussion

6.1 The Technique Is Not the Bottleneck

The single most consequential finding of this analysis is negative. Across three evaluation criteria, two cross-validation designs and eight modelling configurations, the differences between the non-analogy techniques are neither large nor statistically significant. Simple CART models perform a little better than approaches that are in most cases considerably more complex, but the observed differences do not attain significance. The principal interest of CART consequently resides not in superior accuracy but in its simplicity of use and interpretation — a defensible basis for recommendation, but a different one from the basis usually claimed in the literature.

This finding sits uncomfortably alongside the trajectory described in Section 4. The field has, over three decades, moved from single parametric equations to fifteen-driver parametric models, to neural and evolutionary learners, and thence to multi-stage hybrids combining swarm optimisation with fuzzy clustering. If successive increments of algorithmic sophistication translated reliably into accuracy, the comparative evidence would show it. It does not. What the evidence suggests instead is that the key solution to accurate cost prediction lies not in the modelling technique itself but in the quality and adequacy of the data collection.

6.2 Implications for Measurement Programmes

If this result were confirmed by subsequent studies, it would carry serious implications for how cost data is collected and how data-driven cost models are built. To benefit genuinely from collecting organisation-specific cost data, an organisation should not merely automate the collection of generic, COCOMO-like factors; it should investigate which factors actually matter within its own setting and design a tailored, specific measurement programme accordingly [59]. The finding that company-specific models built on generic factors do not outperform generic models is, read correctly, not an argument against local data — it is an argument against local collection of non-local variables.

A second implication follows for the standing of multi-organisation repositories. Where such repositories are confined to a specific application domain and are governed by rigorous data quality assurance, they yield modelling results comparable to local cost models, while offering participating companies larger and more up-to-date project data sets than any single organisation could assemble. Common project data repositories across companies and within homogeneous application domains should therefore be regarded as not merely viable but positively beneficial. One explanation for the parity is that the principal source of heterogeneity may inhere in project characteristics themselves rather than in the organisation where the projects are executed — in which case pooling across organisations within a domain costs comparatively little.

6.3 The Interpretability Constraint

It merits emphasis that artificial neural networks were excluded from the empirical comparison on interpretability grounds, despite outperforming case-based reasoning in Finnie et al. [19] and outperforming regression trees in Srinivasan and Fisher [7]. The exclusion reflects a genuine operational constraint: a model that project managers cannot understand is a model they will not apply, and an unapplied model has no accuracy in practice. This constraint has become sharper rather than softer as the field has moved towards deep and ensemble learners, and it defines a research agenda in its own right — the development of estimation models that are simultaneously accurate and explicable, whether through inherently interpretable structures such as regression trees or through post-hoc explanation of opaque learners.

6.4 Risk Analysis as an Integral Component

Given the large number of cost factors collected, the rigour of the collection procedure, and the substantial residual uncertainty nonetheless observable in the estimates, point estimation appears to be the wrong output format for the problem. The evidence supports the position that risk analysis should be an integrated part of software cost estimation procedures [59] rather than an optional supplement. An estimate reported as a distribution, or as an interval with an attached confidence level, communicates what a point estimate conceals — namely that MMRE values in the region of 0.5 to 0.7 are typical even under favourable conditions.

6.5 Does the Post-2016 Evidence Overturn the Thesis?

A reasonable objection to the argument of Section 6.1 is that it rests on a 1999 comparison of techniques that are, by contemporary standards, elementary. If the differences between OLS regression, ANOVA, CART and analogy were small, that might reflect the narrow capability range of those four techniques rather than any general property of the estimation problem. Deep learning, on this objection, represents a discontinuity rather than another increment.

The replication evidence does not support the objection. Tawosi, Moussa and Sarro [74] revisited the deep learning approach to agile story point estimation and asked directly whether the problem has been solved, benchmarking the reported deep models against simple baselines under a consistent evaluation protocol. The pattern they report is the pattern of Section 5: the margin over unsophisticated baselines is considerably narrower than the original reports suggest, and it is sensitive to evaluation design. Mahmood, Kama, Azmi, Khan and Ali [75] reach a compatible conclusion in their systematic performance evaluation of machine learning techniques for effort estimation, finding accuracy differences across techniques that are inconsistent across datasets rather than stable in favour of any one method.

Two structural features of the newer literature reinforce rather than weaken the original finding. First, the interpretability constraint of Section 6.3 binds more tightly on transformer and graph-network models than it did on regression trees, so the trade-off between accuracy and adoptability has moved in the unfavourable direction. Second, the most consequential contribution of the agile branch is arguably not a model at all but a dataset: TAWOS [71] expands the available evidence base by roughly two orders of magnitude relative to the 206 projects analysed in Section 5. That the field's most valuable recent artefacts are measurement infrastructure rather than algorithms is precisely what the data-quality thesis predicts.

The conclusion of Section 6.1 therefore stands, with its scope enlarged: across twenty-seven years, four technique generations and two distinct estimands, the identity of the learning algorithm has remained a weaker determinant of estimation accuracy than the quality, quantity and organisational relevance of the data supplied to it.

7. Threats to Validity

Five classes of threat qualify the conclusions drawn here. First, with respect to external validity, the primary empirical evidence derives from a single database of business application projects contributed by Finnish companies; generalisation to embedded, real-time, safety-critical or contemporary cloud-native contexts is not warranted without replication. Second, with respect to construct validity, MRE-based measures are known to be biased towards underestimation and are sensitive to extreme individual predictions; MdMRE and Pred(0.25) mitigate but do not eliminate this, and the agile literature discussed in Section 4.6 has largely moved to absolute-error measures, which limits direct numerical comparison across the two eras. Third, the exclusion of artificial neural networks on interpretability grounds means the 1999 comparison is not exhaustive over the technique space described in Section 4, and the strongest learners in several prior studies are absent by construction.

Fourth, and most importantly for the present study, this is a secondary analysis: the original experiments were not re-executed, and the verification procedure of Section 3.2 can detect internal inconsistency and narrative–table mismatch but cannot detect an error common to both the narrative and the table, nor any error in the underlying computation. The two discrepancies documented in Sections 5.1 and 5.2 establish that such checking is necessary; they do not establish that it is sufficient. Fifth, the extension of the review window to 2026 was conducted to test a specific hypothesis rather than to produce an exhaustive survey of the intervening decade, and Section 4.6 should be read as an adequacy check on the thesis rather than as a systematic review in its own right.

8. Conclusion and Future Work

This study set out to reconcile the descriptive and the evidential accounts of software development cost estimation. The descriptive account, consolidated in Section 4, records continuous proliferation: model-based, expertise-based, learning-oriented, regression-based, dynamics-based and composite families, with the COCOMO suite alone branching into more than a dozen variants and extensions between 1981 and 2004. The reason for this proliferation is the changing nature of software complexity — one cannot exactly predict the cost of a whole project from a fixed formulation, and each shift in development practice invalidates a portion of the prior calibration.

The evidential account is more sober. Cross-validated comparison over 206 projects from 26 organisations shows that the considered modelling approaches do not differ substantially on MMRE, MdMRE or Pred(0.25); that simple CART models perform a little better than more complex alternatives, though not significantly so, and are chiefly attractive for their interpretability; that analogy-based procedures do not perform well when validated across organisational boundaries, plausibly because unweighted Euclidean similarity is ill-suited to heterogeneous data and because reliance on one or two analogues amplifies the influence of outliers; and that local models built on generic cost factors do not significantly outperform models built on external multi-organisation data.

Extending the review window to 2026 does not disturb this picture. The deep learning branch and the agile story-point branch add genuinely new capability and a genuinely new estimand, but replication studies conducted under consistent evaluation protocols report margins over simple baselines that are narrower than the originating papers claimed and unstable across datasets — the same pattern, on different techniques, three decades later.

A methodological contribution should also be recorded. Verification of the transcribed statistics identified two errors in the source reporting: a descriptive claim about relative effort that is contradicted by the table supporting it, and a duplicated table caption that makes the results sequence ambiguous on a casual reading. Both are corrected in Sections 5.1 and 5.2. Neither alters the substantive conclusions, but both would propagate into any subsequent study that transcribed the figures without checking them, and their existence in a well-cited paper argues for arithmetic verification as a routine step in secondary analysis.

Taken together, these accounts answer RQ3. Several factors of the software development process are inter-related, and consequently no single technique can be declared the most appropriate or suitable for SDCE in general [1]. From quantitative through empirical formulations, SDCE models may be applied alone or hybridised with robust machine learning or data mining techniques; but the marginal accuracy available from that hybridisation is small relative to the marginal accuracy available from improving what is measured and how well. The productive research question is therefore not "which learner?" but "which variables, collected how, at what granularity, and validated against what?".

Four directions for future work follow. First, direct replication of the 1999 comparative design — the same techniques, the same three criteria, the same two cross-validation schemes — on a contemporary corpus such as TAWOS, which would for the first time permit the technique-versus-data question to be tested at a scale where small effects are detectable. Second, systematic evaluation of weighted and learned similarity functions for analogy-based estimation, since the equal-influence assumption identified in Section 5.2 is a specific and remediable defect rather than a general property of the paradigm. Third, development of estimation models that satisfy the interpretability constraint without conceding accuracy, including explanation methods applicable to ensemble and deep learners. Fourth, and most consequentially, the design and empirical validation of tailored, organisation-specific measurement programmes, in order to test directly the hypothesis that data quality rather than algorithmic sophistication is the binding constraint on estimation accuracy.

Acknowledgements

The author gratefully acknowledges the Department of Computer Application, Anjaneya University, Raipur, for institutional support. The analysis presented in Sections 3 and 5 rests upon the empirical work of Briand, El Emam, Surmann, Wieczorek and Maxwell, and upon the survey of Rajper and Shaikh, whose contributions are acknowledged with thanks. The original data collection reported in [2] was made possible by Software Technology Transfer Finland (STTF).

References

[1] Rajper, S., Shaikh, Z.A. Software Development Cost Estimation: A Survey. Indian Journal of Science and Technology, vol. 9, no. 31 (August 2016). DOI: 10.17485/ijst/2016/v9i31/93058.

[2] Briand, L.C., El Emam, K., Surmann, D., Wieczorek, I., Maxwell, K.D. An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques. Proceedings of the 21st International Conference on Software Engineering, ICSE ’99, Los Angeles, CA (1999) 313–322.

[3] Leung, H., Fan, Z. Software Cost Estimation. Handbook of Software Engineering, Hong Kong Polytechnic University (2002) 1–14.

[4] Jørgensen, M., Shepperd, M. A Systematic Review of Software Development Cost Estimation Studies. IEEE Transactions on Software Engineering, vol. 33, no. 1 (2007) 33–53.

[5] Wen, J., Li, S., Lin, Z., Hu, Y., Huang, C. Systematic Literature Review of Machine Learning Based Software Development Effort Estimation Models. Information and Software Technology, vol. 54, no. 1 (2012) 41–59.

[6] Boehm, B., Abts, C., Chulani, S. Software Development Cost Estimation Approaches — A Survey. Annals of Software Engineering, vol. 10, nos. 1–4 (2000) 177–205.

[7] Srinivasan, K., Fisher, D. Machine Learning Approaches to Estimating Software Development Effort. IEEE Transactions on Software Engineering, vol. 21, no. 2 (February 1995) 126–137.

[8] Boehm, B.W. Software Engineering Economics. Prentice-Hall, Englewood Cliffs, NJ (1981).

[9] Putnam, L.H. A General Empirical Solution to the Macro Software Sizing and Estimating Problem. IEEE Transactions on Software Engineering, vol. 4, no. 4 (July 1978) 345–361.

[10] Albrecht, A.J. Measuring Application Development Productivity. SHARE/GUIDE: Proceedings of the IBM Applications Development Symposium (October 1979) 83–92.

[11] Albrecht, A.J., Gaffney, J.E. Software Function, Source Lines of Code, and Development Effort Prediction. IEEE Transactions on Software Engineering, vol. 9, no. 6 (1983) 639–648.

[12] Kemerer, C.F. An Empirical Validation of Software Cost Estimation Models. Communications of the ACM, vol. 30, no. 5 (May 1987) 416–429.

[13] Kitchenham, B.A., Taylor, N.R. Software Project Development Cost Estimation. The Journal of Systems and Software, vol. 5 (1985) 267–278.

[14] Rubin, H.A. A Comparison of Cost Estimation Tools (a panel discussion). Proceedings of the 8th International Conference on Software Engineering, IEEE Computer Society Press (1985) 174–180.

[15] Conte, S.D., Dunsmore, H.E., Shen, V.Y. Software Engineering Metrics and Models. Benjamin/Cummings Publishing Company, Inc. (1986).

[16] Jensen, R.W. A Comparison of the Jensen and COCOMO Schedule and Cost Estimation Models. Proceedings of the International Society of Parametric Analysis (1984) 96–106.

[17] Shepperd, M., Schofield, C. Estimating Software Project Effort Using Analogies. IEEE Transactions on Software Engineering, vol. 23, no. 12 (November 1997) 736–743.

[18] Mukhopadhyay, T., Vicinanza, S.S., Prietula, M.J. Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation. MIS Quarterly (June 1992) 155–171.

[19] Finnie, G.R., Wittig, G.E., Desharnais, J-M. A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models. Journal of Systems and Software, vol. 39, no. 3 (1997) 281–289.

[20] Briand, L.C., Basili, V.R., Thomas, W.M. A Pattern Recognition Approach for Software Engineering Data Analysis. IEEE Transactions on Software Engineering, vol. 18, no. 11 (1992) 931–942.

[21] Jørgensen, M. Experience with the Accuracy of Software Maintenance Task Effort Prediction Models. IEEE Transactions on Software Engineering, vol. 21, no. 8 (August 1995) 674–681.

[22] Khalifelu, Z.A., Gharehchopogh, F.S. Comparison and Evaluation of Data Mining Techniques with Algorithmic Models in Software Cost Estimation. Procedia Technology, vol. 1 (2012) 65–71.

[23] Dejaeger, K., Verbeke, W., Martens, D., Baesens, B. Data Mining Techniques for Software Effort Estimation: A Comparative Study. IEEE Transactions on Software Engineering, vol. 38, no. 2 (2012) 375–397.

[24] Molani, M., Ghaffari, A., Jafarian, A. A New Approach to Software Project Cost Estimation Using a Hybrid Model of Radial Basis Function Neural Network and Genetic Algorithm. Indian Journal of Science and Technology, vol. 7, no. 6 (2014) 838–843.

[25] Dizaji, Z.A., Gharehchopogh, F.S. A Hybrid of Ant Colony Optimization and Chaos Optimization Algorithms Approach for Software Cost Estimation. Indian Journal of Science and Technology, vol. 8, no. 2 (2015) 128–133.

[26] Gharehchopogh, F.S., Ebrahimi, L., Maleki, I., Gourabi, S.J. A Novel PSO Based Approach with Hybrid of Fuzzy C-Means and Learning Automata in Software Cost Estimation. Indian Journal of Science and Technology, vol. 7, no. 6 (2014) 795–803.

[27] Nevalainen, R., Maki, H. Laturi System Product Manual, Laturi 2.0. Finland (January 1996).

[28] Kitchenham, B. A Procedure for Analyzing Unbalanced Datasets. IEEE Transactions on Software Engineering, vol. 24, no. 4 (April 1998) 278–301.

[29] Hays, W.L. Statistics. Fifth Edition, Harcourt Brace College Publishers (1994).

[30] Spector, P. Ratings of Equal and Unequal Response Choice Intervals. The Journal of Social Psychology, vol. 112 (1980) 115–119.

[31] Bohrnstedt, G., Carter, T. Robustness in Regression Analysis. In: Costner, H. (ed.), Sociological Methodology, Chapter 5. Jossey-Bass (1971).

[32] StataCorp. Stata Statistical Software: Release 5.0. Stata Corporation, College Station, Texas (1997).

[33] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. Classification and Regression Trees. Wadsworth & Brooks/Cole Advanced Books & Software (1984).

[34] Steinberg, D., Colla, P. CART: Classification and Regression Trees — Tree Structured Nonparametric Data Analysis, Interface Documentation. Salford Systems (1995).

[35] Shepperd, M., Schofield, C., Kitchenham, B.A. Effort Estimation Using Analogy. Proceedings of the 18th International Conference on Software Engineering, ICSE-18 (1996) 170–175.

[36] ANGEL: Analogy Software Tool. Bournemouth University, Department of Computing (1996).

[37] CBR-Works 4.0 beta. Research group "Artificial Intelligence — Knowledge-Based Systems", University of Kaiserslautern.

[38] Shepperd, M., Schofield, C. Effort Estimation by Analogy: A Case Study. Presented at ESCOM 7, Wilmslow (1996).

[39] Boehm, B.W., Madachy, R., Steece, B. Software Cost Estimation with COCOMO II. Prentice Hall PTR (2000).

[40] Parr, F.N. An Alternative to the Rayleigh Curve Model for Software Development Effort. IEEE Transactions on Software Engineering, no. 6 (1980) 291–296.

[41] Cantone, G., Cimitile, A., De Carlini, U. A Comparison of Models for Software Cost Estimation and Management of Software Projects. Computer Systems: Performance and Simulation (1986) 123–140.

[42] Boehm, B.W., Valerdi, R. Achievements and Challenges in COCOMO-Based Software Resource Estimation. IEEE Software, vol. 25, no. 5 (2008) 74–83.

[43] Woudenberg, F. An Evaluation of Delphi. Technological Forecasting and Social Change, vol. 40, no. 2 (1991) 131–150.

[44] Tausworthe, R.C. The Work Breakdown Structure in Software Project Management. Journal of Systems and Software, vol. 1 (1979) 181–186.

[45] Ebrahimpour, N., Gharehchopogh, F.S., Khalifehlou, Z.A. A New Approach with Hybrid of Artificial Neural Network and Ant Colony Optimization in Software Cost Estimation. MAGNT Research Report (2015).

[46] Gharehchopogh, F.S., Pourali, A. A New Approach Based on Continuous Genetic Algorithm in Software Cost Estimation. Journal of Scientific Research and Development, vol. 2, no. 4 (2015) 87–94.

[47] Abbas, S.A., Lar, S.U., Liao, X., Naseem, R.A. Software Models, Extensions and Independent Models in COCOMO Suite: A Review. Journal of Emerging Trends in Computing and Information Sciences, vol. 3, no. 5 (2012) 1–11.

[48] Boehm, B. Making RAD Work for Your Project. Computer, vol. 32, no. 3 (1999) 113–117.

[49] Boehm, B., Valerdi, R., Lane, J., Brown, A. COCOMO Suite Methodology and Evolution. CrossTalk, vol. 18, no. 4 (2005) 20–25.

[50] Briand, L.C., Langley, T., Wieczorek, I. A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques. Proceedings of the 22nd International Conference on Software Engineering, Limerick (2000) 377–386.

[51] Sajadfar, N., Ma, Y. A Hybrid Cost Estimation Framework Based on Feature-Oriented Data Mining Approach. Advanced Engineering Informatics, vol. 29, no. 3 (2015) 633–647.

[52] Chiu, N-H., Huang, S-J. The Adjusted Analogy-Based Software Effort Estimation Based on Similarity Distances. Journal of Systems and Software, vol. 80, no. 4 (2007) 628–640.

[53] Park, H., Baek, S. An Empirical Validation of a Neural Network Model for Software Effort Estimation. Expert Systems with Applications, vol. 35, no. 3 (2008) 929–937.

[54] Kumar, K.V., Ravi, V., Carr, M., Kiran, N.R. Software Development Cost Estimation Using Wavelet Neural Networks. Journal of Systems and Software, vol. 81, no. 11 (2008) 1853–1867.

[55] Li, Y-F., Xie, M., Goh, T.N. A Study of Project Selection and Feature Weighting for Analogy Based Software Cost Estimation. Journal of Systems and Software, vol. 82, no. 2 (2009) 241–252.

[56] Strike, K., El Emam, K., Madhavji, N. Software Cost Estimation with Incomplete Data. IEEE Transactions on Software Engineering, vol. 27, no. 10 (2001) 890–908.

[57] Li, J., Ruhe, G. A Comparative Study of Attribute Weighting Heuristics for Effort Estimation by Analogy. Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, New York (2006) 66–74.

[58] Gibbons, J.D.S. Nonparametric Statistics. Series: Quantitative Applications in the Social Sciences 90, SAGE University Paper (1993).

[59] Briand, L.C., El Emam, K., Bomarius, F. A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment. Proceedings of the 20th International Conference on Software Engineering, ICSE-20 (April 1998) 390–399.

[60] Maxwell, K., Van Wassenhove, L., Dutta, S. Software Development Productivity of European Space, Military and Industrial Applications. IEEE Transactions on Software Engineering, vol. 22, no. 10 (1996).

[61] Walkerden, F., Jeffery, R. An Empirical Study of Analogy-Based Software Effort Estimation. Technical Report 98/8, University of New South Wales, Centre for Advanced Empirical Research, Australia (1998).

[62] Iordan, A.E. An Optimized LSTM Neural Network for Accurate Estimation of Software Development Effort. Mathematics, vol. 12, no. 2 (2024) 200. DOI: 10.3390/math12020200.

[63] Draz, M.M., Emam, O., Azzam, S.M. Software Cost Estimation Prediction Using a Convolutional Neural Network and Particle Swarm Optimization Algorithm. Scientific Reports, vol. 14 (2024). DOI: 10.1038/s41598-024-63025-8.

[64] Phan, H., Jannesari, A. Heterogeneous Graph Neural Networks for Software Effort Estimation. Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM (2022) 103–113.

[65] Alsheikh, N.M., Munassar, N.M. Improving Software Effort Estimation Models Using Grey Wolf Optimization Algorithm. IEEE Access, vol. 11 (2023) 143549–143579. DOI: 10.1109/ACCESS.2023.3340140.

[66] Ardiansyah, A., Ferdiana, R., Permanasari, A.E. MUCPSO: A Modified Chaotic Particle Swarm Optimization with Uniform Initialization for Optimizing Software Effort Estimation. Applied Sciences, vol. 12 (2022).

[67] Watson, C., Cooper, N., Nader Palacio, D., Moran, K., Poshyvanyk, D. A Systematic Literature Review on the Use of Deep Learning in Software Engineering Research. ACM Transactions on Software Engineering and Methodology, vol. 31, no. 2, Article 32 (March 2022). DOI: 10.1145/3485275.

[68] Rossi, B., Fontoura, L.M. AI-Based Approaches for Software Tasks Effort Estimation: A Systematic Review of Methods and Trends (2025).

[69] Choetkiertikul, M., Dam, H.K., Tran, T., Pham, T., Ghose, A., Menzies, T. A Deep Learning Model for Estimating Story Points. IEEE Transactions on Software Engineering, vol. 45, no. 7 (July 2019) 637–656. DOI: 10.1109/TSE.2018.2792473.

[70] Fu, M., Tantithamthavorn, C. GPT2SP: A Transformer-Based Agile Story Point Estimation Approach. IEEE Transactions on Software Engineering (2022).

[71] Tawosi, V., Al-Subaihin, A., Moussa, R., Sarro, F. A Versatile Dataset of Agile Open Source Software Projects (TAWOS). Proceedings of the 19th International Conference on Mining Software Repositories, MSR (2022).

[72] Alsaadi, B., Saeedi, K. Ensemble Effort Estimation for Novice Agile Teams. Information and Software Technology, vol. 170 (2024).

[73] Sánchez-García, Á.J., González-Hernández, M.S., Cortés-Verdín, K., Pérez-Arriaga, J.C. Software Estimation in the Design Stage with Statistical Models and Machine Learning: An Empirical Study. Mathematics, vol. 12, no. 7 (2024) 1058.

[74] Tawosi, V., Moussa, R., Sarro, F. Deep Learning for Agile Effort Estimation: Have We Solved the Problem Yet? arXiv preprint arXiv:2201.05401 (2022).

[75] Mahmood, Y., Kama, N., Azmi, A., Khan, A.S., Ali, M. Software Effort Estimation Accuracy Prediction of Machine Learning Techniques: A Systematic Performance Evaluation. Software: Practice and Experience, vol. 52, no. 1 (2022) 39–65.