


RESEARCH ARTICLE

An investigation of effort distribution among development phases: A four-stage progressive software cost estimation model

Efi Papatheocharous^{1,2}  | Stamatia Bibi³ | Ioannis Stamelos⁴ | Andreas S. Andreou⁵

¹University of Cyprus, Department of Computer Science, 75 Kallipoleos Street, P.O. Box 20537, CY1678 Nicosia, Cyprus

²Swedish Institute of Computer Science, Isafjordsgatan 22/Kistagången 16, Box 1263, SE-164 29 Kista, Stockholm, Sweden

³Department of Informatics & Telecommunications Engineering, University of Western Macedonia, Kozani, Greece

⁴Department of Informatics, Aristotle University Campus, Aristotle University, PO 54124, Thessaloniki, Greece

⁵Department of Electrical Engineering/Computer Engineering and Informatics, Cyprus University of Technology, 31 Archbishop Kyprianos Street, 3036 Limassol, Cyprus

Correspondence

Efi Papatheocharous, Department of Computer Science, University of Cyprus, 75 Kallipoleos Street, P.O. Box 20537, CY1678 Nicosia, Cyprus.
Email: efi.papatheocharous@cs.ucy.ac.cy

Funding information

European Research Consortium for Informatics and Mathematics, Grant/Award Number: 246016

Abstract

Software cost estimation is a key process in project management. Estimations in the initial project phases are made with a lot of uncertainty that influences estimation accuracy which typically increases as the project progresses in time. Project data collected during the various project phases can be used in a progressive time-dependent fashion to train software cost estimation models. Our motivation is to reduce uncertainty and increase confidence based on the understanding of patterns of effort distributions in development phases of real-world projects. In this work, we study effort distributions and suggest a four-stage progressive software cost estimation model, adjusting the initial effort estimates during the development life-cycle based on newly available data. Initial estimates are reviewed on the basis of the experience gained as development progresses and as new information becomes available. The proposed model provides an early, a post-planning, a post-specifications, and a post-design estimate, while it uses industrial data from the ISBSG (R10) dataset. The results reveal emerging patterns of effort distributions and indicate that the model provides effective estimations and exhibits high explanatory value. Contributions in lessons learned and practical implications are also provided.

KEYWORDS

four-stage progressive model, project management, software cost estimation

1 | INTRODUCTION

Software cost estimation is the managerial process of estimating the effort required to build a software system according to functional and non-functional requirements, budget and timeframe specified during the project negotiation, and planning phases. Usually, the estimation process is ran at the early project phases when there is limited information available regarding the factors that influence effort. Therefore, estimates at that point include a high level of uncertainty and subjective reasoning. That is why only highly skilled and experienced managers typically perform these estimates. The estimated effort is then used to specify the resource constraints of the project.

Previous work advises the review of initial estimates on an ongoing basis during the development phases to reduce the inherent uncertainty and subjectivity of the estimates,¹ but this is hardly the case in practice. Such time-aware software cost models have been given very limited attention in research, even though they are supposed to provide early, accurate, and more meaningful estimates.² Other recent studies have compared and visualized effort allocation patterns in specific phases through systematic data analysis frameworks.³ However, no other study, to the best of our knowledge, combines these 2 aspects.

This work contributes to the state-of-the-art by studying effort distributions in various development phases of real-world projects and suggests a 4-stage progressive software cost estimation model. Estimations take into account the incremental nature of software data collection, since useful information is accumulated as projects go through various development phases. Such a procedure involves firstly an allocation of work and effort on the initialization of each development phase, and secondly, a review of these estimates by the end of each phase. The

motivation is that a significant share of development costs can be saved by careful distribution of the work among different tasks.⁴ This means that, apart from estimating the total effort required for the completion of a project, it is important to distribute it properly into the different phases, keeping in mind that faults located in the early activities require a significant amount of rework effort.

Successful software project development depends highly on allocation of the work effort among project phases. Project managers are able to make this allocation effectively when they receive updated information on the relationships between effort figures across the software development lifecycle (SDLC) stages. Such information regarding progressive activities in development becomes gradually available, ie, as the project evolves, grows, and proceeds from one phase to another, allowing the recalibration and adjustment of the total effort value. Even though some research has been made towards effort phase distributions, to the best of our knowledge, a software cost estimation model that is based on the notion of evolution of effort through experience accumulated as we proceed to subsequent project development phases and using a large number of industrial projects has not been constructed yet.

In this study, an empirical investigation is performed on the progressive adjustment of work effort estimates throughout the SDLC phases using projects from the International Software Benchmarking Standards Group (ISBSG) release R10 dataset.⁵ The aim of this work is to identify patterns of development effort distribution in software projects and develop cost models that introduce emerging opportunities that could improve project planning and controlling. Artificial neural networks (ANN) are used to approximate estimations of effort as accurately as possible, using available project information and effort values of finished project phases. The ANN were chosen as the core modeling technique due to their ability to learn, adapt, and estimate the value of an unpredictable variable reasonably well. The ANN⁶ may learn efficiently from imprecise data and allow the representation of complex problems which are difficult to model with traditional, model- or knowledge-based systems. The project samples participating in this study contain benchmark data that enclose years of practical software experience and economics that may be used for improving resource allocation and management. The models developed for each phase are considered to yield "early" effort estimates (ie, before the actual implementation phase), since they use project information that is considered available in that particular phase.

Specifically, the following research questions are investigated:

- RQ1. How is the total development effort related with each phase of the SDLC?
- RQ2. Is there a correlation between the effort values attributed to the various development phases?
- RQ3. Does updating of the estimates during the software development process lead to significant improvement in the accuracy of estimates for the total effort?
- RQ4. At a fixed point in time of the development process, how accurately can effort be approximated for the next phase?

The following sections are organized as follows: Section 2 summarizes related work on software cost models met in literature and especially effort estimation approaches and studies on development phases. Section 3 describes the research methods used, sample dataset, basic constructs used, and also the definitions and hypotheses of this work. In addition, the same section presents the data analysis and experimental procedure followed to investigate the research questions. Section 4 describes in detail the modeling technique and the performance measures used for the series of experiments conducted. Section 5 presents the experimental results and observations obtained. Section 6 involves a constructive discussion on the approach, practical lessons learned, and related threats to validity. Finally, Section 7 summarizes the conclusions and outlines future research steps.

2 | RELATED WORK

Accurate estimation of software costs is an open issue since the 1970⁷ with a long history of studies, attracting the interest of both researchers⁷ and practitioners.⁸ The software estimation systematic literature reviews in the field,^{9,10} help us classify software cost estimation approaches in 4 broad categories: (1) expert judgment methods that produce estimates based on the experience and opinion of a group of experts,¹¹⁻¹⁴ (2) parametric cost models^{7,15-19} that capture the relationship between variables to calculate the cost or duration, (3) model-based estimations derived via statistical analysis of historical data of software metrics,²⁰⁻²⁴ and (4) estimation models produced through a combination of the aforementioned techniques.¹

The application of these approaches often produces incomparable and contradictory results, a fact that suggests to tackle the problem by finding the best cases (project data) on which the models exhibit best performance. An alternative approach to improving the accuracy of software cost estimation would be to identify the best time to perform estimations and systematically include iterative feedback cycles of estimations using project data from phases. According to MacDonell,² crucial consideration should be given to time dependence when estimating the cost of software development. As our target is to provide a phase-based effort estimation model, we will focus on relevant research that reports effort distributions and cost estimates across the different phases of development. Research for estimating the effort required for different phases of software development can be classified into two categories: (1) studies adopting rules of thumb, usually at project level, (2) studies exploring effort distribution across different phases by performing data analysis.

Simple rules of thumb have been provided by practitioners: NASA's software engineering laboratory²⁵ suggested that an effort prediction interval is calculated by simple mathematical formulas according to when the estimation is performed. In the earlier days, the "40-20-40" rule,²⁶ that is, 40% of the development effort for analysis and design, 20% for programming, and 40% for integration and testing, was updated many times. Zerkowicz,²⁷ on the basis of data gathered, specified the relative effort distribution for various activities: requirements analysis 10%, specification 10%, design 15%, coding 20%, and testing 45%. Later on, Boehm²⁸ specified the effort distribution to be 60% for requirements analysis and design, 15% for implementation and 25% for testing. Moreover, Brooks²⁹ recommended the following empirical ratios of effort required to complete projects: 1/3 of project effort is allocated for planning, 1/6 for coding, 1/4 for component tests and early system test, and 1/4 for system test with all components in hand (integration testing). Ambler³⁰ reports that typical RUP projects, that is, projects developed with the rational unified process, spend approximately 10% of time in inception, 25% in elaboration, 55% in construction, and 10% in transition. In his textbook, Sommerville³¹ reports that a software system of 100 cost units distributes 15 units on specification, 15 units on design, 20 units on development activities, and 40 units on testing. The main observation for these divergent ratios is that they depend heavily on the type of software development and vary from organization to organization and even from project to project.

Several authors suggested phased-based effort estimations based on meta-data produced during development. Ohlsson and Wohlin³² studied effort during the progress of a project and used phase-based data, such as the number of requirements and flowcharts, to estimate effort for the subsequent phases. Even though the results of the study showed that the metrics used did not correlate particularly well with effort, in a holistic sense they were useful to managers in acquiring an outline for the progression of the project. Gray and MacDonell²⁰ proposed the use of fuzzy logic principles in a varying uncertainty degree throughout the software development phases. The authors suggested that a single estimation model could be used for consistency with different levels of uncertainty in the respective project phases according to the time of the estimate. The level of uncertainty in project cost estimates and how this evolves during a project was also expressed by Boehm's "cone of uncertainty."¹⁵ The cone expressed the amount of uncertainty and associated risk to the size and cost estimates decreasing with a project's development progression to completion.³³

The need to perform re-estimations of cost along development progression, especially given the relative uncertainty and subjectivity of the cost factors used in assessing effort as the project evolves, is addressed by MacDonell and Shepperd.¹ MacDonell and Shepperd¹ studied 16 projects within a single organization for 18 months. Their findings indicated that there are no standard portions of effort for particular development activities though the effort required to complete previous tasks can be valuable to predict the effort of subsequent activities. In later studies, Yiftachel et al⁴ developed an economic model for optimal allocation of resources among the development phases of requirements, design, implementation, and testing. The output software was evaluated on the basis of two levels: in the first level, the quality of the different phases artifacts (internal quality) was considered, and in the second level, the integration of all these qualities into what is consumed by the customer (external quality) was considered. MacDonell and Shepperd² used a small industrial dataset of 16 projects from a single organization and compared predictions using time-aware and leave-one-out analysis. Their results showed that failure to exploit time dependence leads to unreliable estimates of cost predictions. The authors also compared the impact of using the available data of requirements, design, and implementation phases to estimate the subsequent phase effort and observed accurate estimations for the implementation and testing phase. Yang et al³⁴ empirically studied the phase effort distribution of projects from the China Software Benchmarking Standard Group database concluding that software and team size have a consistent effect on the software code, as well as on the test, requirements, design, and transition phase distribution variations.

Some works found in literature use the ISBSG dataset to explore phase-based effort distribution. The use of the ISBSG dataset according to Fernández-Diego and Ladrón-de-Guevara³⁵ offers several advantages with respect to effort estimation, despite the diversity of its data observations. ISBSG offers a variety of up-to-date information regarding modern, relevant tools, processes, environments accompanied by process, and product data that can be used for effort estimation. Jiang et al³⁶ proposed a model to predict development effort based on the software size estimated with function points. The authors used the ISBSG dataset, generalized the average amount of the effort spent on each development phase, and provided estimates for the effort used in the building, testing, and implementation phases based only on software size. Déry and Abran³⁷ worked with a previous version of the ISBSG repository and concluded that the effort distribution reported within the various development phases needs to be handled with care, as the projects recorded seemed to cover different life cycle phases (31 different project profiles were found). In another study,³⁸ the authors used the statistical framework of Compositional Data Analysis to visualize information of software projects in the ISBSG repository that is not otherwise apparent. The analysis included software effort distributions of projects with respect to external and internal project characteristics, such as organization type, language type, and size. The work was extended in Chatzipetrou et al³ that tested whether projects in ISBSG confirm the rules of thumb suggested by literature. The results showed that only a very small number of projects are close to each of the hypothetical distributions mentioned by practitioners and scholars.

In this work, the main propositions of the aforementioned research were taken into account and also some of the limitations identified earlier were tackled. Moreover, the main contribution of the present paper is a progressive four-stage cost estimation model, which provides relatively more accurate estimations when more information gradually becomes available. The proposed model identifies the degree of inherent complexity and uncertainty of the estimates along the consecutive development phases. This work complements current literature by testing the appropriateness of a wealth of relevant data produced by each subsequent development phase with a twofold purpose: (1) predict effort of the next phase and (2) adjust and update the total effort estimate. We believe that this dual estimation will increase the probability of more efficient estimation methods with industrial application that, according to Trendowicz and Jeffery,¹⁰ is necessary.

3 | RESEARCH METHOD

In this section, we describe the research questions investigated, the dataset used, the hypotheses made, and the data analysis procedure of this study.

3.1 | Research questions

The objective of this study is to provide answers to the following research questions:

RQ1. How is the total development effort related with each phase of the SDLC?

We examine the relationship between the reported effort of each phase in completed industrial projects and the total development effort that was essentially required to perform them. We are interested in observing the level of effort differences and distribution in each phase in comparison with the total effort expenditure. In addition, we investigate the level of correlation between the total effort and the effort of each development phase. Correlation analysis identifies the relationship among project phases and the total work effort. After understanding these relationships a project manager may reallocate effort in the respective development phases to adjust the total effort expended. Moreover, we examine the differences in distributions of effort in each phase and the final total effort required for the projects selected in relation to the specific project characteristics, ie, development type, development platform, language type, and the maximum team size, ie, meaning the number of people forming the development teams.

RQ2. Is there a correlation between the effort values attributed to the various development phases?

We examine the relationship between the effort values devoted during the various phases of development. Therefore, correlation analysis is performed to identify whether the distribution of effort into the different development phases follows a certain pattern. Such correlations might indicate that a relationship exists between consecutive development phases, potentially useful in practice for prediction purposes. Additionally, comparison with previously reported studies adopting rules-of-thumb (eg, Zelkowitz,²⁷ Boehm²⁸ and Brooks²⁹) for effort allocation across different phases is also done on the basis of the findings of RQ2.

RQ3. Does updating of the estimates during the software development process lead to significant improvement in the accuracy of estimates for the total effort?

We explore whether there is significant improvement of the accuracy in the total effort estimates as the information from completed phases is inserted in a stepwise progressive cost estimation model. We investigate whether the conventional Boehm's "cone of uncertainty"¹⁵ regarding cost estimation accuracy along the SDLC is confirmed or redefined with respect to the specific empirical projects examined and software cost estimations obtained. For this reason, we consider four different estimation models used at different project development phases, that is, early, post-planning, post-specifications and post-design models. The early estimation model uses standard information available from the beginning of a software project regarding its profile and provides an estimate. The post-planning estimation model uses project attributes after planning is completed and estimates the total effort. The post-specifications model uses information from the planning and specifications phase including the relevant project, product, and environment attributes and provides an estimate adjustment of the final effort. The post-design model uses information from the planning, specification, and design phases to update the initial estimate towards a more refined one. The practical value of effort estimations after the design phase is substantially minimized during or after the stage of software implementation.

RQ4. At a fixed time of the development process, how accurately can effort be approximated for the next phase?

We examine whether it is possible to estimate the effort required to perform a subsequent phase from known project attributes identified during the planning phase of the project and the feasibility study. Again, the question of whether the effort of completed phases can be used for estimating the effort required for the subsequent phases is investigated with the four-phased estimation model at the points previously described. We are interested to explore the accuracy of such estimations as it is frequent to reallocate resources during different development phases.

3.2 | Sample dataset

The dataset is obtained from an international non-profit organization, the ISBSG. ISBSG has established and maintains a database of historic IT industry project data, embedding the practical aspect of software development, to assist and improve IT project management globally. The ISBSG repository was selected in this study to test the appropriateness of the suggested estimation models for the following reasons: (1) The ISBSG dataset contains among others a wealth of information regarding software process including software effort phase distribution data that cannot be found in most of the other publicly available datasets, such as those on PROMISE.³⁹ (2) The ISBSG repository is constantly updated,

representing relevant cutting-edge software development approaches and technologies and therefore can be used as a benchmark for estimation compared to other datasets that are older and not recently updated. (3) ISBSG is a popular dataset among researchers that attempt to test the applicability of methods for software effort estimation. Therefore, a comparison with other methods and a replication of our study would be facilitated with the use of ISBSG. (4) ISBSG is publicly available and can be used by companies that are not experienced in project data measurement and collection. A company that does not possess any private data can use ISBSG to isolate relevant projects and conduct estimations that can be further customized to match the company's reality. Therefore, ISBSG provides rich, qualitative, practical, and useful data for conducting studies as this one.

ISBSG (R10) dataset⁵ was released on January 2007 with data contributions from many industries all over the world. This has as a result the availability of a plethora of projects with different characteristics, developed following different methodologies and techniques. The data originate from 25 countries with 60% of the projects being less than 10 years old. The database consists of attributes such as the application domain, programming languages used, development techniques, resource levels, functional size, etc describing 4106 projects. These attributes can be mandatory or optional, and therefore, optional fields that are not filled-in for particular projects are excluded from the analysis.

"Summary Work Effort" is a mandatory field in ISBSG referring to the actual total effort devoted to develop a project. Since for each project data for varying sets of phases are provided, effort data can vary considerably across organizations and projects. Thus, while organizations, on the one hand, are obliged to report the total effort, on the other hand, they are optionally requested to map their own life cycle into a standard ISBSG life cycle containing the following 6 phases: Planning, Specification, Design, Build, Test and Implementation. In addition, data collectors may report a residual of the effort that has not been attributed to any of the above phases to an extra field called "Effort Unphased." If no phase breakdown is provided in the submission, this field contains the same value as the Summary Work Effort. Where phase breakdown is provided in the project data and the sum of that breakdown does not equal the Summary Work Effort, the difference is shown in this field. Projects that did not provide phase breakdown were not included in the filtered dataset used in this work. The additional field of Effort Unphased participated in the analysis to investigate the percentage of effort required for supplementary activities, such as training and negotiations. This type of effort is not attributed to the rest of the phases.

We consider that the basic phases in the projects' SDLC are essentially common in both the incremental and the traditional development methods. A typical cycle includes the phases of requirements analysis, design, implementation, testing, and sometimes installation and checkout.⁴⁰ The basic phases of the Waterfall, RUP,⁴¹ and ISBSG may overlap, as shown in Table 1, or may even be performed in an iterative manner.

In our analysis, we consider the basic SDLC phases contained in a typical cycle to build four-stage estimation models at a phase-level distinction, ie, at the planning phase, specifications phase, design phase, and build phase. However, we recognize that there are some differences regarding the sequence and timing in which the activities in each corresponding phase may be performed. Specifically, the initial phases of the traditional methods are devoted to analysis and design during which the requirements are gathered from the users and preliminary and detailed designs are produced. Then, the rest of the phases are sequentially streamlined until the product delivery. In incremental models, such as the RUP,³⁰ the sequential aspect is captured in its phases, ie, the Inception phase, Elaboration phase, Construction phase, and Transition phase. The phases for implementing, testing, and evaluating/testing the system are essentially common in both kinds of methodologies but appear with different names.

For the present research study, we specify the terminology and constructs used so that we may distinguish between the actual terms of effort and project phases as they are represented in the dataset. The term *effort* is used to provide the total effort measured in person-hours for completing a project. An *effort estimate* is defined as the most likely effort obtained from the cost estimation methodology. The 4 stages selected to perform effort estimates are (1) early estimate, before project planning; (2) post-planning, before specifications phase; (3) post-specifications estimate, before design phase; and finally, d) post-design estimate, before implementation. Our evaluation concerns only these 4 stages because, as mentioned above, after implementation is completed effort estimates are less critical with limited practical value and do not offer significant competitive advantage to project managers.

Moreover, the ISBSG refers to the term "summary work effort" as the *actual effort* recorded or the original untransformed value reported for the projects. The ISBSG also refers to *effort breakdown* as the actual person-hours distributed in the project phases, ie, planning, specification, design, build, test, and implementation. In addition, the ISBSG has made some normalization steps to make the total effort of the projects comparable for cases where effort information was not specified for all of the six available development phases. However, in this study, we did not take into consideration this normalization and we used the original project values of summary work effort that did not involve any assumptions. Thus, we consider only projects for which full effort information was submitted for all six development phases. Details on the rest of the project attributes used in this work are provided in Section 5.1.2.

TABLE 1 Mapping of software development life cycle phases

Waterfall	Requirements	Design	Implementation	Test	Installation & Maintenance	
	Inception	Elaboration				
RUP			Construction		Transition	
ISBSG	Planning	Specification	Design	Build	Test	Implementation

Abbreviations: ISBSG, International Software Benchmarking Standards Group; RUP, rational unified process.

3.3 | Assumptions

The following assumptions were made:

Assumption 1. We assume that the effort for each phase is actually measured and is not an estimate derived by the developers using the total effort expended for the whole project. We performed a simple examination of this assumption by verifying that the phased effort values do not correspond to rounded percentages of their total effort. Therefore, one may argue that such values are actually measured rather than rough percentage estimates; thus, this indicates that our assumption holds.

Assumption 2. Estimating the effort of subsequent phases using prior effort on completed phases requires the sequential development of software. Iterations or possible reviews of earlier phases are not addressed by the models proposed. To create meaningful models, we assume, therefore, that the SDLC is structured and sequential.

Assumption 3. A project manager can clearly distinguish between the activities that belong to each development phase. For each phase, a specific amount of effort recorded may be considered as the effort that was actually required to complete the activities required for each phase.

3.4 | Data analysis procedure

At a high level the analysis performed in this study involves the following actions:

1. Selection and pre-processing of the variables and projects that participated in the analysis.

Descriptive statistics for all attributes and projects were calculated. In addition, the values of each attribute were grouped together to examine the distributions of efforts for the particular project attribute (ie, Maximum Team Size). This was only performed for analysis purposes and the groups were not used in the prediction models. Since for the experimental procedure the technique of ANN was used to predict the effort value in different development phases the following transformation was required: Categorical attributes were transformed to n binary attributes, one for each category, with an indication whether an attribute belongs to that category or not.

2. Investigation of the relationship between the development effort of each SDLC phase and the total work effort.

Correlation analysis was conducted on the pairs of phased-effort values. Initially, we checked whether the effort required for the completion of each phase was normally distributed using histograms. Secondly, Spearman correlation coefficient (CC) nonparametric statistic was used as a measure of statistical dependence between 2 variables.⁴² The choice of a non-parametric test was based on the non-normal form of effort distributions. Also, two-tailed tests were performed because the nature of the relationships was unknown. The raw values X_i (independent variable) and Y_i (dependent variable) were converted to the ranked (x_i, y_i) values and if there were no tied ranks, Spearman's ρ coefficient was calculated by Equation 1.^{43,44} The sign of Spearman's correlation indicates the direction of association between the 2 variables and a zero correlation value indicates that there is no relation.

$$\rho = 1 - \frac{6\sum(x_i - y_i)^2}{n(n^2 - 1)}. \quad (1)$$

3. Use of ANN for modeling and estimating the total development effort and the effort of each subsequent phase based on information from the previous phases.

The relationship between the dependent and independent variables is modeled and captured by ANN regardless the complexity of connections, offering quick processing, training, adaptation, and validation procedures. A major advantage of ANN is the arbitrary function approximation mechanism which learns and adapts from patterns of observed data yielding overly robust models.

4. Evaluation of the estimation model using 5-fold cross validation.

The predictive power of the proposed estimation model was tested on a separate set of projects that did not participate in the model building and training process. Thus, a discrete sample of projects was excluded from the analysis and was used to evaluate the model. The rest of the project samples were used to build the ANN models using a variety of architectures. The evaluation process was performed iteratively on a set of data partitions that were complementary subsets of the initial (training) dataset. One round of cross validation involved performing the analysis on one subset (containing 60% of the data samples called training set), the validation on a second subset (containing 20% of the data samples, called validation set) and the testing on a third subset (containing 20% of the data samples, called testing set). To reduce variability multiple rounds of cross validation were performed using different partitions.

4 | MODELING TECHNIQUE

In this section, the modeling technique, that is, ANN, is described including the details of calibration of the technique. The evaluation criteria used to assess the estimation performance of the derived models are also presented.

4.1 | Artificial neural networks

Based on the neural structures of the brain, ANN comprise computational elements (called neurons) which process input records one at a time, learn by comparing the prediction of the record with the known actual value, and provide output predictions. Even though many ANN architectures are available, the feedforward multilayer perceptron⁴⁵ is currently the most popular and is especially suited for the kind of problem-solving domains that comprise very noisy, distorted, or incomplete sample data. Moreover, the backpropagation learning algorithm⁶ used is widely accepted because of its wide range of problem applicability and generally good performance.

The experiments conducted used simple structures of multilayer perceptron networks with a single hidden layer that comprises a varying number of internal neurons. Figure 1 shows the architecture of the network with the input, hidden, and output neuron layers. Each neuron uses the respective input vectors, weights, and momentum (or bias) to estimate the output as specified in Equation 2 for hidden neuron h_1^1 . The backpropagation algorithm is used to calculate derivatives of performance of mean square error with respect to the weight w_{i1}^1 , where $i = 1, \dots, n$ for each input weight and bias θ_1^1 variables.

$$h_1^1 = f \left(\sum_{i=1}^n x_i w_{i1}^1 + \theta_1^1 \right). \quad (2)$$

With backpropagation, for each input vector the networks' response is compared to the actual values and the error is gradually reduced by looking at the error changes as weights are altered. More details of the algorithm can be found in Werbos et al.⁶

Each network is initially built using a number of neurons in the input (first) layer equal to the number of attributes used as inputs in each model phase. The network is fully connected, that is, the first layer has weighted connections with every input. A range of ANN architectures was deployed, each of which contained a number of neurons in the hidden layer equal to the number of attributes used as inputs in each experiment and the number of inputs was increased by 1 until the neurons doubled the size of the input attributes. The last layer was the network output. Each subsequent layer uses the weights coming from the previous layers and adjusts them accordingly so that the accuracy performance error of the output is diminished.

The samples were separated in 5 folds of equal size; initially, the first 3 folds were used for training, the next one for validation, while the last fold was used for testing (evaluation). Using a sliding window of 1 fold, the process was repeated 5 times so that every fold was used for evaluation thus performing multiple rounds of cross validation. This also means that in each repetition the ANN used 60%, 20%, and 20% of different data samples for training, validation, and testing, covering the whole dataset.

All architectures were evaluated and the best performance was recorded, while each time the process was repeated, the layer's weights and momentum (or biases) were initialized. The weights were updated with the gradient descent with momentum weight/bias learning function. Training was performed with the scaled conjugate gradient method. This function can train any network as long as its weight, net input, and transfer functions have derivative functions.

Moreover, to avoid overfitting, training was stopped when one of the following conditions occurred: (1) The maximum number of 1000 epochs was reached; (2) performance was maximized with prediction error close to the goal, which was set to 10^{-4} ; (3) the gradient fell below the minimum performance values, which was set to 10^{-6} ; (4) the validation error increased more than 5 times the maximum validation error since the last time it decreased.

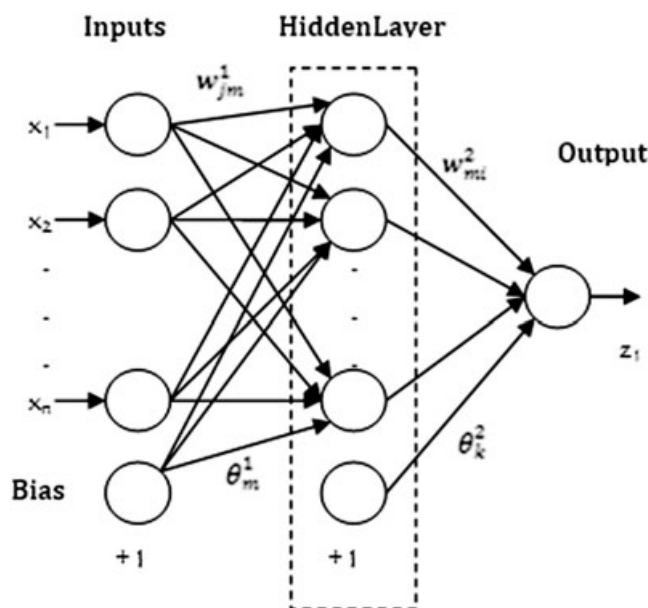


FIGURE 1 A feedforward multilayer perceptron neural network

The hyperbolic tangent sigmoid transfer function (Equation 3) was used to calculate the hidden layer's output from its net input where λ is the gain and sum is the net input received by a neuron. The output from this function is bounded in the range $[-1, +1]$ and it was selected because it does not require nonlinearity and additional scaling. Finally, the linear transfer function was used for the output layer.

$$f(x) = \frac{2}{1 + \exp(-\lambda \text{sum}_j)} - 1. \quad (3)$$

The Matlab R2009a Neural Network toolbox was used to create, train, simulate, and visualize the ANN performance. From the various ANN architectures created, we extrapolated those that yielded the lowest median of the magnitude of relative error (MdmRE) (see Section 4.2) during the simulation (testing) phase, which is one of the most popular evaluation criteria used in the area of predictive modeling.

4.2 | Evaluation criteria

Three common error metrics were considered for the evaluation of the average estimation accuracy⁴⁶ of the derived models. Each metric was used to assess the performance of the techniques used comparing the actual sample values to the predicted ones. More specifically, the magnitude of relative error (MRE), the CC, and the $\text{pred}(l)$ error measures were used. The error metrics are defined as follows:

The MRE, given in Equation 4, shows the prediction error focusing on the sample being predicted. $x_{\text{act}}(i)$ is the actual effort and $x_{\text{pred}}(i)$ the predicted effort of the i th project.

$$\text{MRE} = \left| \frac{x_{\text{act}}(i) - x_{\text{pred}}(i)}{x_{\text{act}}(i)} \right|. \quad (4)$$

The MdmRE is then reported that summarizes the MRE for all projects that participate in the test dataset.

The CC between the actual and predicted series, described by Equation 5, measures the ability of the predicted samples to follow the upwards or downwards of the original series as it evolves in the sample prediction sequence. An absolute CC value equal or near 1 is interpreted as a perfect follow up of the original series by the forecasted one. A negative CC sign indicates that the forecasting series follows the same direction of the original with negative mirroring, that is, with a rotation about the time axis.

$$\text{CC}(n) = \frac{\sum_{i=1}^n [(x_{\text{act}}(i) - \bar{x}_{\text{act},n})(x_{\text{pred}}(i) - \bar{x}_{\text{pred},n})]}{\sqrt{\left[\sum_{i=1}^n (x_{\text{act}}(i) - \bar{x}_{\text{act},n})^2 \right] \left[\sum_{i=1}^n (x_{\text{pred}}(i) - \bar{x}_{\text{pred},n})^2 \right]}}. \quad (5)$$

Equation 6 defines the $\text{pred}(l)$ metric that reveals how many data predictions k out of n (total number of data points predicted) performed well, i.e., their RE metric given in Equation 7 is lower than level l . In the experiments, parameter l was set equal to 0.25.

$$\text{pred}(l) = \frac{k}{n}, \quad (6)$$

$$\text{RE}(n) = \frac{|x_{\text{act}}(i) - x_{\text{pred}}(i)|}{x_{\text{act}}(i)}. \quad (7)$$

5 | EXPERIMENTAL RESULTS

In this section, the experimental results are recorded in detail. Section 5.1 presents the preprocessing methods applied to the data and provides the descriptive statistics of the attributes that participated in the analysis. Section 5.2 records and interprets the results of the correlation analysis performed to investigate the relationship between the total development effort and the effort devoted to each phase. Section 5.3 provides the investigation of the relationship of the effort between various development phases, and Sections 5.4 and 5.5 summarize the results of the estimation models proposed for the 4 basic phases, namely, early, post-planning, post-specification, and post-design.

5.1 | Preprocessing and statistical profile of data

5.1.1 | Data preprocessing

Preprocessing of the dataset is considered necessary to ensure the quality of the data that participate in the analysis. The dataset is therefore further analyzed to reduce the heterogeneity of the projects and maintain a subset that is meaningful for the purposes of our analysis, as appointed by recent studies on ISBSG.³⁵ For this reason, we performed the following reductions and transformations of the dataset:

TABLE 2 Descriptive statistics of the selected project attributes

Attribute Name	Category	N	%
Development type	Enhancement	29	44.62
	New development	33	50.77
	Redevelopment	3	4.62
Development platform	PC	28	43.08
	Midrange (MR)	6	9.23
	Main frame (MF)	20	30.77
	Multiplatform (Multi)	11	16.92
Language type	2GL	1	1.54
	3GL	48	73.85
	4GL	15	23.08
	ApG	1	1.54
Maximum team size	Low (<6)	23	35.38
	Medium (6-10)	20	30.77
	High (>10)	22	33.85

Abbreviation: ApG, application generator; PC, personal computer; 2GL, second generation (programming) language, 3GL, third generation (programming) language; 4GL, fourth generation (programming) language.

1. Firstly, we excluded records that did not report the total effort or the effort breakdown values to the various phases. Thus, the dataset was dramatically reduced to 101 project attributes which still are enough to provide meaningful interpretations for the subset of the projects studied.
2. Secondly, we excluded columns that contained over 40% null values. Thus, the dataset was reduced to 32 project attributes. Then, we maintained meaningful columns for effort estimation and columns with homogeneous, nonconflicting data.
3. In addition, we excluded projects with data quality rating C or D (given by ISBSG reviewers) and maintained projects with A or B rating.
4. We excluded variables that contain large portions of vague, missing (null), or multivalued records such as adjusted function points and other sizing attributes, project elapsed time, number of defects delivered, etc, as well as other variables which are not specified before project completion.
5. We maintained variables (columns) such as Development Type, Development Platform, Language Type, and Maximum Team Size together with the phased-effort values and Summary Work Effort reported for the 65 projects that remained after performing the filtering described above. These project attributes are also useful to perform comparisons among different project types, and thus, they were used in the subsequent experiments with the ANN.

5.1.2 | Data description and statistical profile

Before describing the experiments conducted, we present the statistical profile of the data sample isolated for investigating our research questions. For most of the projects in the sample there is no information recorded regarding the development technique used. However, a small percentage refers to traditional methods without providing much detail and mentions techniques such as data modeling, process modeling, event modeling, conceptual analysis, and activity modeling. Moreover, a few projects in the sample have in common the usage of the CA ERwin modeling suite* for development (known as CA Erwin Data Modeler).

The counting techniques used to measure the size of the projects range from IFPUG, to COSMIC-FFP, Mark II, NESMA, FiSMA, Fuzzy Logic, and LOC. The application types include, among others, document management, embedded systems, financial transaction processes/accounting, logistic or supply planning and control, management of licenses and permits, management or performance reporting, network management, workflow support and management, online analysis and reporting, protocols, machine control, web-based applications, etc. The primary programming languages mostly used are ASP, C/C++/C#, COBOL, COOL:GEN, Java, PL/I, Visual Basic, with architecture types deployed being Stand-alone, Client-Server or Multi-tier. Also, the projects implemented in each year are 5 in 2006, 18 in 2005, 22 in 2004, 9 in 2003, 6 in 2002, 3 in 2001, 1 in 2000, and 1 in 1994.

Table 2 presents the descriptive statistics of the project attributes that were used to investigate our research questions.

Each of the projects reports development type according to the following definitions: new development refers to the production of new software; redevelopment is recorded when new technologies are incorporated to replace or upgrade existing software; enhancement corresponds to the case where new functionality is implemented to modify or extend existing software. Each project is classified according to the development platform, as determined by the operating system used, as personal computer, midrange (MR), main frame, or multiplatform (Multi). Language type defines the type of language used for each of the projects: eg, 2GL, 3GL, 4GL, or application generator (ApG). Finally, the maximum team size represents the peak team size in number of persons in the development team that worked at any time on the project. We defined three similar sample bin sizes for groups of maximum team size that contained low (<6), medium (6-10), and high (>10) team members to group the measured data into

*<http://erwin.com/products/data-modeler>

classes and present the same statistics as for the rest categorical variables. However, the original values of Maximum Team Size were used in the experiments that follow.

5.2 | RQ1: Relationship among total effort and the effort of each development phase.

The statistics for the efforts of plan, specify, design, build, test, and implement phases, as well as the unphased effort are presented in Table 3. Lines “N,” “Max,” “Min,” “Med,” “Mean,” “Std. dev.,” “25%,” and “75%” state for each type of effort the number of observations, maximum value, minimum value, median value, mean value, standard deviation, lower and upper quartiles values, respectively.

Figure 2 depicts the established typical breakdown of effort, ie, percentage by phase for the subset of projects of ISBSG (R10). For example, the Planning effort averages 8.2% of the total project effort, Specify effort averages 7.9%, and Design effort averages 11.9%. A large effort proportion is spent on Building activities (36.8%), which appear to be more effort-intensive, while, Test and Implementation activities demand 15.5% and 5.6% of the total effort, respectively. The high amount of effort required for design and testing and the even higher amount devoted for building shows the focus of software engineers primarily on these activities.

Comparing the effort distribution of the selected ISBSG projects presented in Figure 2 with previously suggested effort distributions from various sources (shown in Table 4), we observe a significant difference in coding effort. That is, the effort reported usually in literature for the actual implementation of a project is substantially lower than the actual effort reported in the ISBSG projects. However, the same amount of effort is observed in typical RUP projects³⁰ for both coding and testing the software, even though the percentages reported do not apply in a strict sense. Requirements analysis accounts usually for approximately 10% to 20% of the effort, design for 10% to 15%, and coding for 15% to 35%. A rough range of 15% to 25% is allocated to testing activities, and finally, integration seems to consume 5% to 25% of the total effort. Time for planning and other important project management tasks, even though is reported in some cases to take a significant amount of effort (eg, 33% in Brooks²⁹) is not usually explicitly reported.

Conclusively, how much effort spent on each of the above project phases is not strictly defined throughout the literature and a consensus does not seem to exist. An interesting investigation therefore could be the identification of the particular project characteristics that influence the adjustment of the percentage spent on the particular phases observed, ie, in the RUP and ISBSG project cases.

In Figure 3 the distribution of the average effort phased difference is presented in relation to sorted quartiles of equal size. The quartiles represent the lowest 25% of the values (quartile 1), the following 25% (quartile 2), the next in ranking order 25% (quartile 3), and finally, the 25% highest values (quartile 4). The average effort proportion for each quartile is calculated by dividing the average effort of each phase with the total effort, according to Equation 8.

TABLE 3 Sample descriptive statistics for effort breakdown

	Summary Work Effort	Plan	Specify	Design	Build	Test	Implement	Unphased
N	65	65	65	65	65	65	65	65
Max	47 252	10 800	6500	5807	30 000	6966	5393	14 382
Min	172	4	4	17	70	40	2	0
Med	4294	158	250	400	1300	626	168	148
Mean	6999.71	575.63	556.23	831.77	2577.68	1085.55	390.38	982.46
Std. dev.	8451.03	1456.60	1026.80	1024.79	4686.74	1364.81	784.02	2279.87
25%	2337	64.50	98	198	570.50	267	49	0
75%	7843	488.50	669.50	1080	2661.50	1411	392.50	563

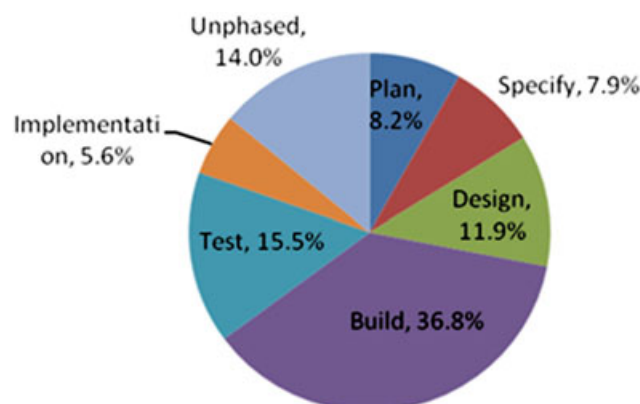


FIGURE 2 Effort average percentages distribution per phase

TABLE 4 Comparative effort phase distribution reported in literature

Study	Planning	Requirements	Specifications	Design	Coding	Testing	Integration
Zelkowitz ²⁷		Reqs (10%)	Specs (10%)	Design (15%)	Coding (20%)	Testing (45%)	
Boehm ²⁸		Reqs - Analysis - Design (60%)			Coding (15%)	Testing (25%)	
Brooks ²⁹	Planning (33%)				Coding (16%)	Testing (25%)	Integration (25%)
Pressman ²⁶		Analysis & Design (40%)			Coding (20%)	Testing & Integration (40%)	
Ambler ³⁰		Inception (10%)	Elaboration (25%)		Construction (55%)		Transition (10%)
Sommerville ³¹		Specs (15%)	Design (15%)		Development (20%)	Testing (40%)	
This study	Plan (8.2%)	Specify (7.9%)		Design (11.9%)	Build (36.8%)	Test (15.5%)	Implement (5.6%)

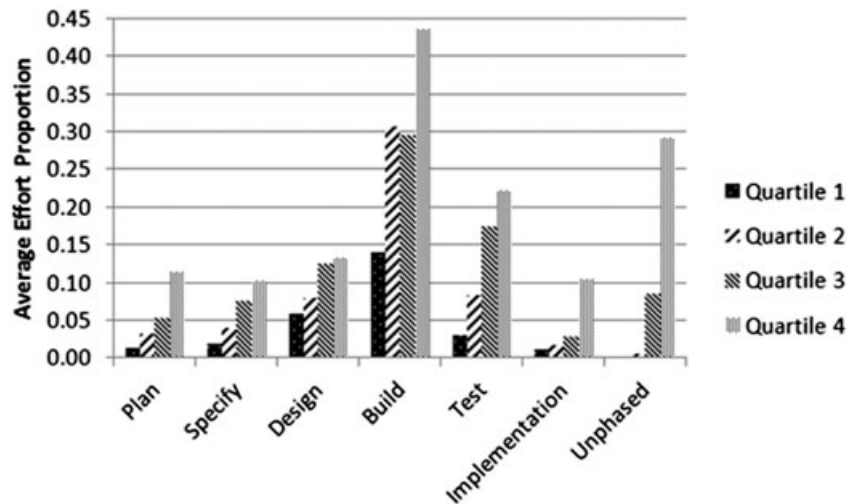


FIGURE 3 Comparison of effort phase distribution over data quartiles

$$\forall \text{project phase/category } \text{AverageEffort Proportion} = \frac{\text{AveragePhasedEffort}}{\text{SummaryWorkEffort}} \tag{8}$$

A high amount of effort value appears usually in most development phases in the third and fourth quartiles. Extreme differences between the values of effort appear in the phases after the design phase. This shows that a large distribution of higher effort values appears in the phases of Build, Test, Implementation, and Unphased indicating volatile effort in the intensive development phases, whereas in the initial development phases of Plan, Specify, and Design, Effort values are normally distributed. For the specific projects the observed irregular distribution of effort indicates that extreme contingencies may occur right after the design phase of development.

Table 5 presents the results of Spearman's two-tailed CCs analysis.⁴⁷ Considering a reasonably sized dataset Hopkins calls correlation value smaller than 0.1 trivial, 0.1 to 0.3 minor, 0.3 to 0.5 moderate, 0.5 to 0.7 large, 0.7 to 0.9 very large, and 0.9 to 1.0 almost perfect.⁴⁸ From Table 5, we can assume, as expected, that there is considerable correlation between the effort required for all phases and the total effort. Very large correlation of Summary Work Effort is found with the effort of Plan, Design, and Build phases, while large correlation is observed with Specify, Test, and Implementation. Moderate correlation exists between Unphased effort and Summary Work Effort. The above findings suggest that the value of the effort required for completing planning, design, and building activities has a direct effect on the total effort spent for the development of the project. Also, the value of the total effort required is relatively highly related to the effort values required for specifications gathering, testing, and installation/setup.

Moreover, we compared the differences regarding phased-effort values against the total work effort for each attribute in the dataset by plotting the ratio of average values of effort reported in each phase for the respective groups and their average Summary Work Effort using Equation 8.

TABLE 5 Spearman two-tailed correlation coefficients ρ of the phased-effort values and total effort (Summary Work Effort)

Phase	ρ (P value)
Plan	0.715 (.000)
Specify	0.693 (.000)
Design	0.827 (.000)
Build	0.828 (.000)
Test	0.633 (.000)
Implement	0.603 (.000)
Unphased	0.328 (.008)

Therefore, Figure 4 shows the median effort ratios of each phase compared to the total effort for the (1) 3 development types, (2) 4 development platforms, (3) 4 language types, and (4) 3 groups of maximum team sizes.

Figure 4 provides useful information regarding the different ratios of effort spent for each project phase within the selected projects of ISBSG and their respective characteristics. More specifically, it is clear from Figure 4A that the average effort spent for the design phase in relation to the total effort is rather similar for all 3 Development Types. Also, the effort ratio difference for New Developments is higher in Plan, Specify, and Design phases, as expected, due to the high effort required to perform the activities of those phases in new software developments rather than software Enhancements or Re-developments. Moreover, higher effort is required for the Enhancement type in Build and Test phases and for Re-developments in Test and Implementation phases. It is noticeable that the costly phase of Build is considerably high in relation to the total effort for all 3 Development Types and also, in New Developments a large ratio of the average effort is categorized in Unphased effort. Such extreme values may be attributed to efforts not included in any of the classic development phases for new developments, e.g., training effort on new business domains.

In Figure 4B, the average effort spent on various phases across Development Platforms is low in relation to the total effort except for the Build and Test phases. In Build phase, effort is concentrated in higher values especially for PC Development Platforms, followed by Main Frame, then Multi and finally, MR platforms. Moreover, MR platform testing effort is also high and a large proportion of Unphased effort is attributed to effort sourcing from Multi.

In Figure 4C, the effort expended for the phases Specify and Design on average for the projects using the four language types is similar in relation to the total effort. A small average effort difference is observed in the planning phase of 2GL and 3GL Language Types compared to 4GL and ApG, while the opposite is observed for the Build and Unphased efforts, ie, there is a higher difference between 4GL and ApG compared to 3GL and even more compared to 2GL Language Types. Moreover, less average effort difference is observed for testing software using 4GL Languages, whereas considerably high difference is noticed for implementing using 2GL. Therefore, the extremely higher effort differences for the Build phase of 3GL, 4GL, and ApG Language Type projects indicate the severity of the language type selection while the considerably high difference in the Implementation phase of 2GL reflects the higher level of effort needs for installation and configuration in these cases.

Finally, Figure 4D shows that different team sizes present high differences among the effort required for the Plan, Build, and Test phases. Medium and large-sized teams present higher average effort differences for the Unphased phase. Moreover, for the Test phase, low-sized teams present considerably higher differences than the rest of the team size types, and in the Build phase, the highest extreme effort proportions are observed for all team sizes and especially for large teams, which may be attributed to the increasing difficulty in communication and version control for those cases.

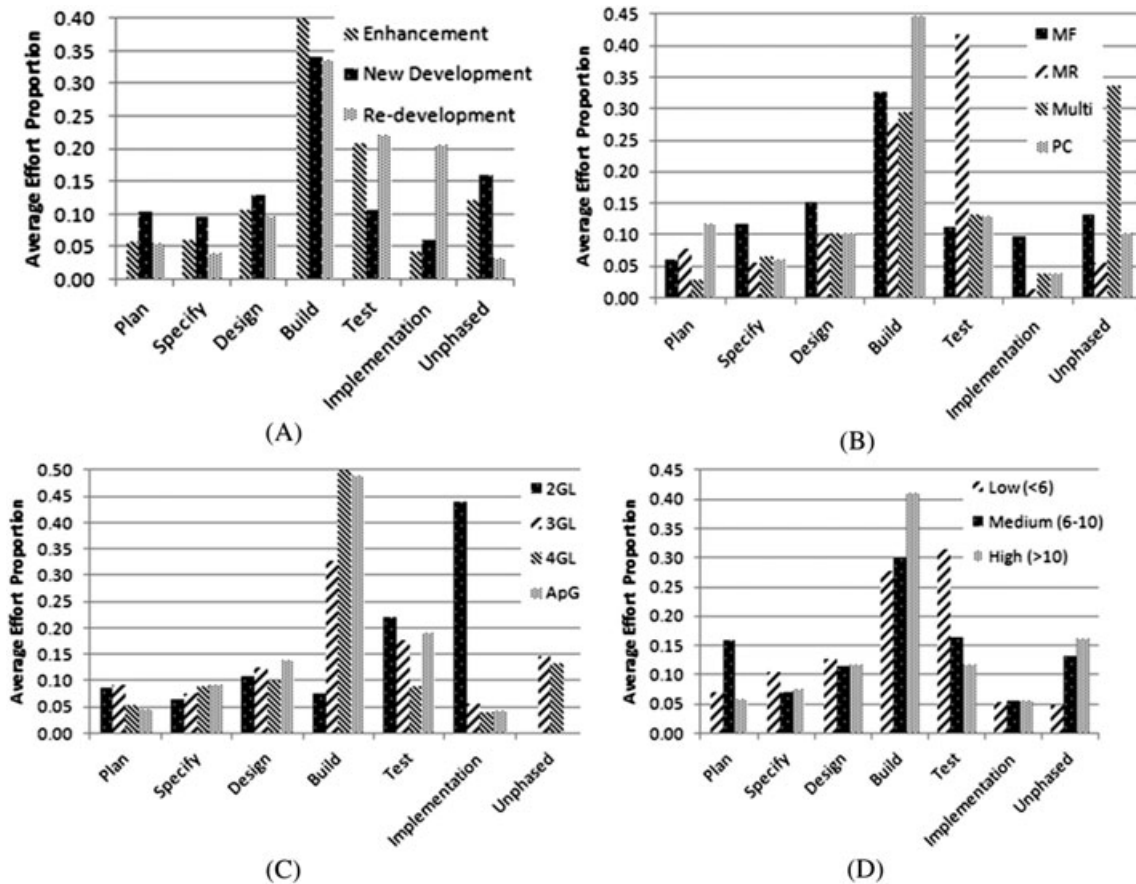


FIGURE 4 Average effort difference across A, Development Type; B, Development Platform; C, Language Type; and D, groups of Maximum Team Size. MF, main frame; MR, midrange; Multi, multiplatform; PC, personal computer

Overall, Figure 4 shows that for the attributes Development Type, Development Platform, Language Type, and groups of Maximum Team sizes, the ratio of effort reported in the Build phase is quite higher than the effort spent in the rest of the phases. This means that estimating accurately the effort for that particular phase is of paramount importance for project managers as it corresponds to the most dominating effort ratio. Thus, any deviation, even the smallest, between the estimate and the actual effort value may cause severe losses to the project.

Figures 5A to 5G show for each development phase the frequencies of the ratio percentage effort values in relation to the Summary Work Effort. For most of the phases, the distribution is negatively skewed, except in the cases of Design and Build phases where the frequency

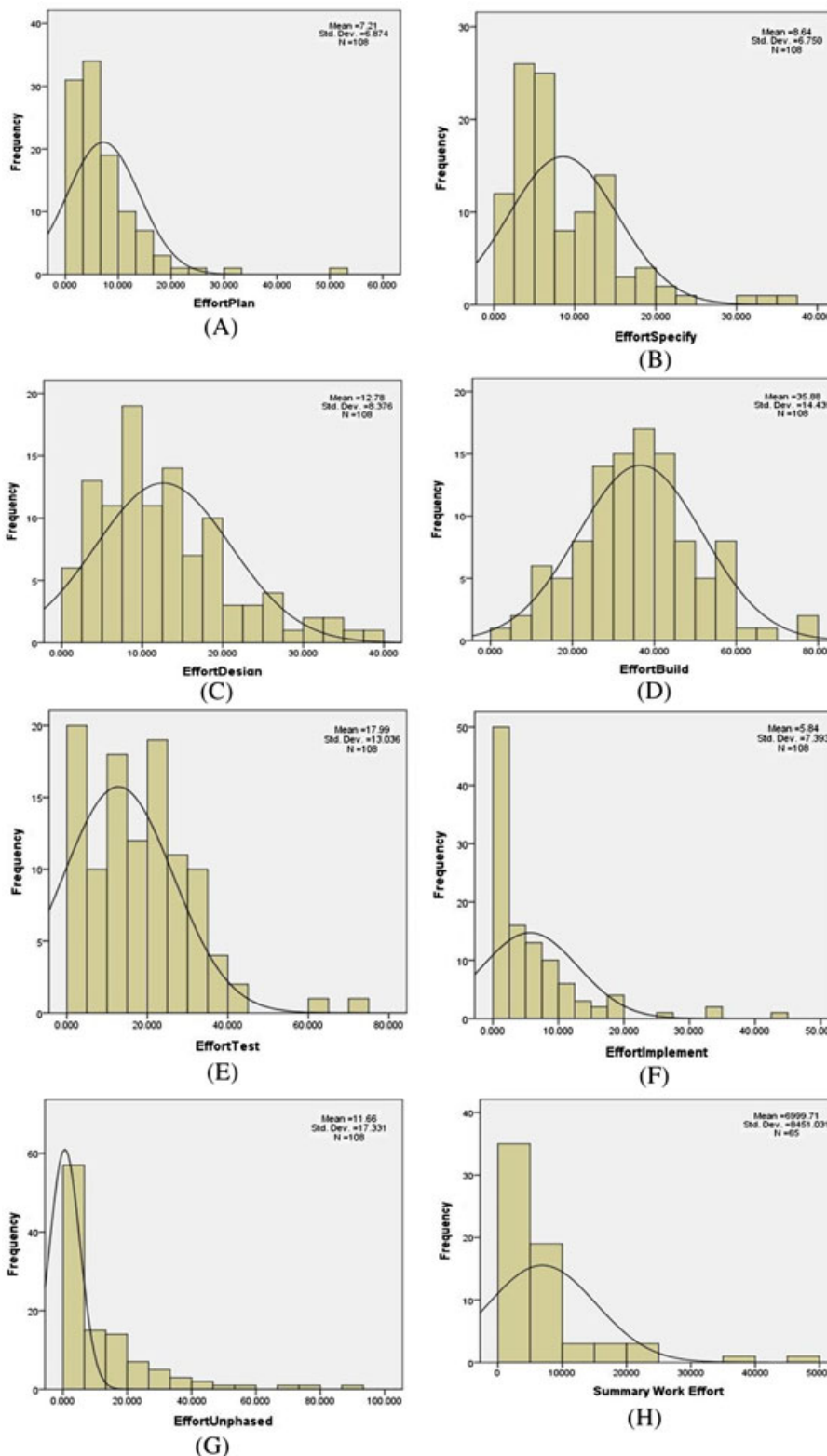


FIGURE 5 A to G, Effort percentage frequencies of various software development life cycle phases and H, summary work effort

distribution histogram appears to be normally distributed. This means that the contribution of each phase to the total effort in the latter 2 cases is more stable, as it is more concentrated around the mean and has smaller variance than in the former two.

5.3 | RQ2: relationship among the effort values of development phases

In this section, we investigate the relationship of effort values reported among different development phases. Figure 6 shows the median effort ratios of each phase for the (1) 3 Development Types, (2) 4 Development Platforms, (3) 4 Language Types, and (4) 3 groups of Maximum Team Sizes. In Figure 6A, we observe that development effort on average is higher for New Developments, lower for Enhancements, and even lower for Re-developments in most of the phases. This observation was expected since in New Developments, and to some extent in software Enhancements, usually more creativity, innovation and overall effort is required compared to cases where an upgrade of existing software is carried out, let alone the increased complexity and uncertainty observed in the production of new systems. But when it comes to the Build and Test phases this picture radically changes with Enhancement medians appearing higher for the projects participating in the analysis. The figures show that in some cases the costs of enhancing software may be prohibiting, and thus, it may be worthwhile to redevelop software since it requires on average less effort. In Figure 6B, the average values of effort for the various Development Platforms is more profound in the build and especially in the Test phase, with MR platforms requiring much more effort on testing than the rest of the platforms. In Figure 6C, 4GL Language Types exceed by far the rest in the Build phase and 2GL in the Implementation phase. In Figure 6D, as expected, a lot of effort is expended for projects that involve a lot of people in nearly all the development phases but especially in the building phase.

Table 6 presents the correlation between the efforts required for the different development phases. We can observe very large positive correlation between the pairs of Specify—Design and Design—Build effort. Large positive correlation of the effort is found between the pairs of Plan-Specify, Plan-Design, Plan-Build, Specify-Build, Specify-Implement, Design-Test, and design-implement. In general, we observe that build and design efforts are particularly highly dependent on the design and specify efforts respectively. Also, Planning effort is highly correlated with Specification, Design, and Build effort; specify effort is highly correlated with Build and Implementation, and Design effort is highly correlated with Testing and Implementation effort. The presence of large to very large correlations between the initial development phases, ie, Plan, Specify, and Design, indicates that if we include this information in an effort prediction model then we may have a relatively good chance of improving estimation accuracy. This investigation is addressed in Sections 5.4 and 5.5.

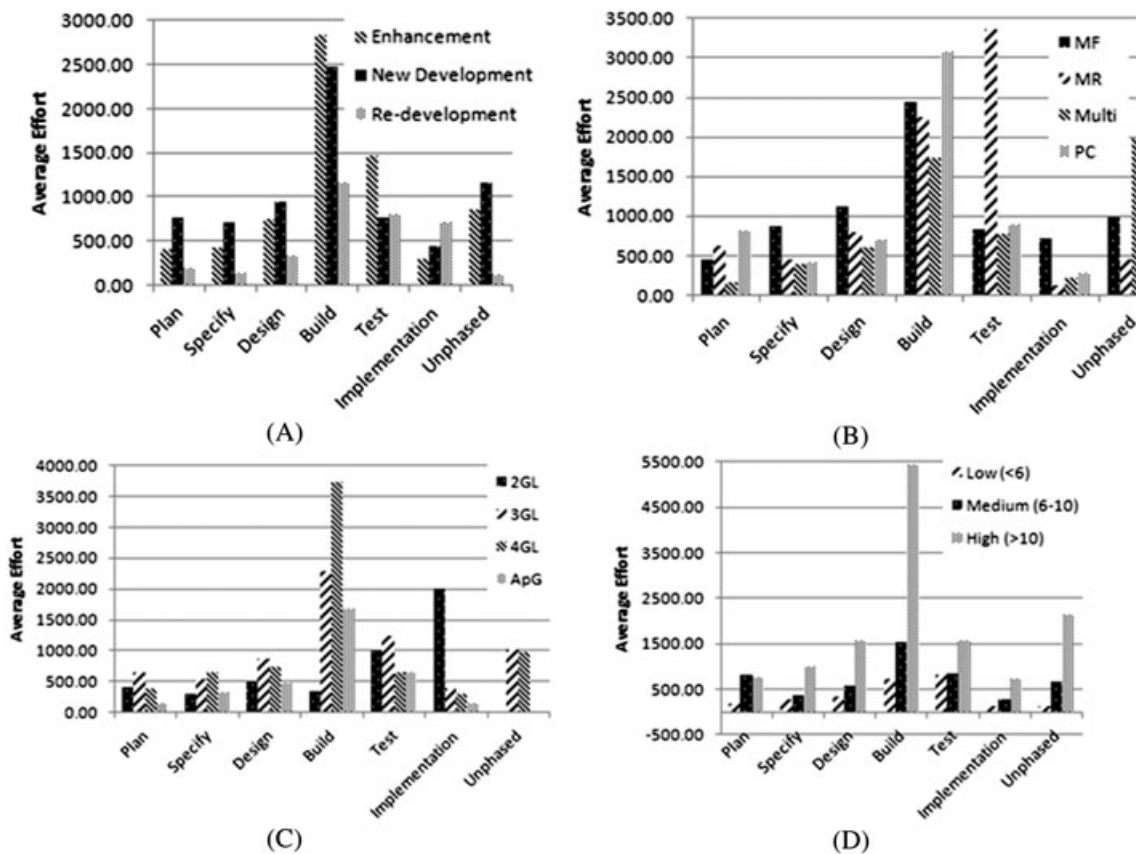


FIGURE 6 Average effort phase distribution across A, development type; B, development platform; C, language type; and D, groups of maximum team size. MF, main frame; MR, midrange; Multi, multiplatform; PC, personal computer

TABLE 6 Spearman 2-tailed correlation coefficients between the phased efforts

ρ (P value)	Plan	Specify	Design	Build	Test	Implement	Unphased
Plan	1	0.600 (0.000)	0.542 (0.000)	0.625 (0.000)	0.421 (0.000)	0.481 (0.000)	0.276 (0.026)
Specify		1	0.764 (0.000)	0.564 (0.000)	0.428 (0.000)	0.583 (0.000)	-0.005 (0.970)
Design			1	0.751 (0.000)	0.581 (0.000)	0.577 (0.000)	0.103 (0.413)
Build				1	0.493 (0.000)	0.482 (0.000)	0.265 (0.033)
Test					1	0.482 (0.000)	-0.009 (0.944)
Implement						1	-0.023 (0.856)
Unphased							1

5.4 | RQ3: updating the estimates leads to significant improvement in the accuracy?

In this section, we investigate how ANN can model and predict the value of summary work effort using the effort in 4 project phases: (1) early, using only the available project attributes derived during the project initiation; (2) postplanning, given that planning is complete, using the available project attributes to readjust the estimate; (3) postspecifications, given that the planning and specifications phases are completed and the actual phase efforts are available, then they are added as an input to the model to review the total effort estimate; and finally, (4) postdesign, given that the previous phases up to the design phase are complete and the actual effort values are known, the model is updated to readjust the total effort estimate.

Table 7 indicates the results obtained from the best performing ANN topologies during the testing phase of 5-fold cross validations. In fact, the figures correspond to the best ANN topologies obtained from the set of topologies examined with respect to their median prediction for MdMRE across all testing folds. As expected, the accuracy error in the estimates is decreased as we precede from the early to the postdesign models.

Contrary to most prior work that focuses on predictions for entire project development, this work derives predictions on the basis of the information available during projects that offers the potential to estimators of recalibrating effort estimations during projects' progression and for the remaining phases. From the results in Table 7, we can conclude that adding more information to the projects progressively as effort phases are completed (especially the initial project phases of planning and specifications), results to estimation accuracy increase of the MdMRE, CC, BRE, and $Pred(0.25)$ metrics. Even though the ANN initially achieve a mediocre accuracy degree in most cases, the accuracy of the total effort estimate after the design phase is significantly increased in comparison to the initially obtained (early) estimate. The overall performance increase may be considered significant, as we progress in the phases and add more information to the cost estimation model, especially considering that after the completion of the specifications phase the project is considered to be at the early stages of development and effort estimations are still very important for efficient resource allocation. Finally, since cross validation was used in our experiments, these model architectures generalize well the new, unseen data, and the models do not overfit the training data.

5.5 | RQ4: how accurately can effort be approximated for the next phase?

This section follows a similar reasoning as in RQ3. We use the same inputs and produce the same estimation models for the phases of (1) early, (2) postplanning, (3) postspecification, and (4) postdesign, but the target output is now the effort of the subsequent phase. Therefore, in (1), planning effort is estimated; in (2), specifications effort is estimated; in (3), design effort is estimated; and in (4), build effort is estimated.

Table 8 shows the testing performance results obtained from the best ANN produced for each topology over a 5 cross validation process. Again, the figures correspond to the best ANN topologies with respect to their median prediction (MdMRE) in testing across all folds.

TABLE 7 Artificial neural networks performance results of estimation of summary work effort progressively along development phases

Estimation Model	Architecture	MdMRE	CC	BRE	$Pred(0.25)$
Early	'12-21-1'	0.92	0.21	28.44	0.05
Post-planning	'13-24-1'	0.86	0.55	16.60	0.08
Post-specs	'14-26-1'	0.78	0.34	4.28	0.17
Post-design	'15-16-1'	0.60	0.63	1.09	0.26

Abbreviations: CC, correlation coefficient; MdMRE, median of the magnitude of relative error.

TABLE 8 Artificial neural networks performance results of effort estimation of subsequent phases progressively along development

Estimation Model	Architecture	MdMRE	CC	BRE	$Pred(0.25)$
Early	'12-19-1'	0.96	0.20	106.94	0.03
Post-planning	'13-23-1'	0.98	0.05	137.68	0.03
Post-specs	'14-24-1'	0.92	0.31	26.30	0.05
Post-design	'15-20-1'	0.84	0.43	6.72	0.08

Abbreviations: CC, correlation coefficient; MdMRE, median of the magnitude of relative error.

From the results in Table 8, we observe that ANN perform relatively better for the Post-specifications and Post-design models in comparison to the first (early and Post-planning) models of the MdMRE metric. Thus, estimating the effort required for the phase of planning and specify seems to be a harder task than estimating the effort for design and build. Despite the existence of very large correlations between the efforts among subsequent phases, the single hidden layer ANN models proposed are insufficient or cannot approximate easily the effort of the subsequent phase using only the prior's phase information in any of the models. These high effort prediction errors observed may be attributed to the highly complex and volatile development techniques and environments in today's software organizations which seem to be difficult to be captured by these models.

5.6 | Statistical significance

The statistical significance of all the results was tested using SPSS for Windows and the Mann-Whitney U test based on the *re* values (Tables 9 and 10). The statistical test shows in most cases that the estimations are improved as more information is added to the models.

6 | DISCUSSION

6.1 | Commenting on the results

The results presented regarding RQ1 revealed that the effort of specific development phases, ie, plan, design, and build, is closely related with the total development effort. Especially, the high correlations of design and build phases (>0.8) indicate that accurate and reliable prediction of summary work effort can be achieved by exploiting information that is available at the Post-design or Post-build stage.

Regarding RQ2, the analysis identified a close relation among the portions of the effort attributed to each development phase and the subsequent one, especially to the plan, specify, design, and build phases, and the effort of the next in order phase. Strong correlations were revealed among the design and build effort values and among the specify and design effort values. The results also appointed a strong relation of specify, design, and build with the planning effort values, indicating that a postplanning prediction model for the effort of subsequent phases is highly likely to succeed. Another interesting observation is that the test and implementation efforts are highly dependent on design efforts. Therefore, a postdesign effort prediction model is also likely to be successful.

The aforementioned observations indicate that a postdesign prediction model is more likely to be of both practical value and high effectiveness in software cost estimation. This was partly confirmed through RQ3 and RQ4, where several ANN architectures were used over a 5-fold cross-validation process. Overall, the ANN developed for RQ3 did not yield impressive results, for accuracy, something which indicates that the modeling attempt for improving total effort estimates by updating the model with information along the project progression is not an easy task. Therefore, even as the project evolves, it is a tedious task to reduce the high level of uncertainty included in the estimation process. Nevertheless, some improvement in prediction error is observed as the development proceeds and more information is available, indicating that the models created based on phased development effort are contributing for improved software cost estimations. This improvement is proven through statistical tests. However, this work did not investigate postbuilding stage predictions since build effort is the most costly phase of development (and thus, it would not be of particular interest if it was used as an input), and the practical value of the model would be significantly reduced.

Overall, the ANN developed to address RQ4 yielded worse results compared to those of RQ3, for accuracy, except for the postspecifications and postdesign models. This shows that the information of project development progression regarding effort may not effectively assist in predicting the effort of the subsequent phase, at least for the sample projects selected and the ANN used. Perhaps more complex ANN structures are required for modeling data with such discontinuities.

TABLE 9 Mann-Whitney signed-rank test results for the artificial neural networks experiments for the estimation of Summary Work Effort progressively along development phases

Test	Mean Rank	U	W	Z	p
Early vs Post-planning	Early > Post-planning	1848.5	3993.5	-1.229	0.219
Post-planning vs Post-specs	Post-planning > Post-specs	1950	4095	-0.757	0.449
Post-specs vs Post-design	Post-specs > Post-design	1634	3779	-2.228	0.026

TABLE 10 Mann-Whitney signed-rank test results for the artificial neural networks experiments for the estimation of subsequent phases progressively along development

Test	Mean rank	U	W	Z	p
Early vs Post-planning	Early < Post-planning	1628.5	3773.5	-2.254	0.024
Post-planning vs Post-specs	Post-planning > Post-specs	1028	3173	-5.05	0
Post-specs vs Post-design	Post-specs > Post-design	1684	3829	-1.995	0.046

Figure 7 presents the average performance error of the best performing ANN topology plot regarding MdmRE of the ANN during testing for estimating the total work effort. The benchmark line of Boehm's cone of uncertainty¹⁵ is also plotted showing at multiple points (phases) of sequential development the values of uncertainty regarding effort and cost estimation. Comparing the theoretical estimate of Boehm, we may infer that (1) the values of effort that project managers may anticipate according to theory, or assume as accurate, are 2.5 times higher than real (actual) values; and (2) along the project's progression, ie, as more information becomes available, this error concerning the effort value is decreased and becomes nearly constant at 2 times the estimate (1 positive and 1 negative).

Our experiments have shown that for the specific real-life projects in the ISBSG dataset; the error of effort estimation from the models proposed is lowered to the 1/3 compared to the degree proposed by Boehm¹⁵ at the initial stages of development, with this difference being smoothly decreased as development progresses in a less steep manner until the postdesign phase. The above indicates that the initial risk of estimating project costs in our models is significantly decreased compared to Boehm's cone estimates. However, over the last phases the risk of estimation appears to be more smoothed out over time in Boehm's cone case whereas in our models the uncertainty of estimation remains at similar levels. Therefore, in our models there is a high likelihood of improved estimates along project progression since variance is lowered (the estimation points appear to be flattened horizontally). This lower variance of estimates of uncertainty reflect a deeper knowledge and understanding of the project parameters, effort values that together contribute to a better estimation performance compared to Boehm's cone uncertainty estimates. Thus, the less the uncertainty, the less variance, and thus, tighter distribution is yielded as expected. Clearly our experimental results (regarding RQ3) indicated that adding more information as the project proceeds, produces increased certainty on project parameters, produces a tighter effort distribution and thus lowers the overall effort uncertainty.

6.2 | Implications for practitioners and researchers

Understanding the contribution of our laboratory research experiments to researchers and practitioners is imperative, according to Wieringa and Daneva.⁴⁹ In our study, we have achieved some conclusions, summarized next, under controlled conditions but using real software projects. Our intention is to create effect in the real world, and thus, we summarize implications to practitioners and research next.

The investigations related to RQ1 have shown that it is possible to accurately estimate the overall effort spent for software projects (named summary work effort in the dataset used) based on information available at the postdesign or postbuild stages. This means that practitioners who can estimate accurately the effort needed for the design or build phases of their projects are able, using for instance the technique of ANN, as shown in this paper, to make relatively accurate estimations of the total effort required to perform the project.

The study of RQ2 also provided knowledge about the relation between efforts attributed to various phases, which makes it feasible to estimate the effort required for a subsequent phase. For instance, our results have shown that knowing the effort spent on the specify phase may enable practitioners estimate the effort required for the design phase. Accordingly, the same observation is made for the design and build phases, and also for the planning and specify and design and build, thus indicating that postplanning prediction models can be constructed to aid practitioners. Also, results have shown that design efforts are related to the effort spent for the test and implementation phases. This indicates that practitioners may also make use of postdesign effort prediction models to improve project management.

In general, from a practitioner's point of view, the availability of the type of knowledge mentioned above helps to identify the most important and costly phases and thus assists in directing efforts towards reducing their overall cost, by prioritizing only the aspects that are decisive for the success of the project.

Updating model estimates during the software development process, using the ANN that investigated RQ3 and RQ4, can lead to significant improvement in cost estimation activities for both the summary work effort and the individual project phases. Inherently, the estimation is relieved

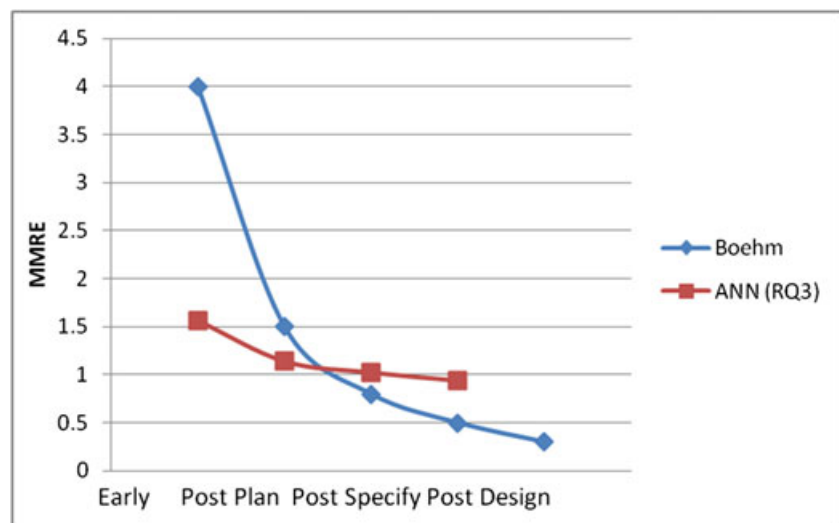


FIGURE 7 Reproduction of Boehm's cone of uncertainty with artificial neural networks (ANN) effort predictions using MMRE during the testing phase

from uncertainty as the understanding of the percentages allocation of effort in the project phases and their relation can encourage paradigm shift to better effort allocation by project managers. Promising results for cost estimation are achieved with the proposed technique as more information becomes available, which also minimizes the initial project uncertainty for effort; such estimations can be very useful to project managers as they can mitigate project risks better.

There are many lessons learned from conducting this study, which are useful to practitioners and researchers. First of all, effort distributions are useful guides for understanding costs through the progression of software projects. To make use of the ISBSG dataset, however, researchers and practitioners need to systematically analyze the available figures and make sure that meaningful filtered data are used. For the purpose of this study, we performed a careful, extensive, and rigorous data filtering to isolate only the relevant to the problem data. We believe that the replication of the study on data collected from companies that are the same size, adopt process-oriented development, and operate in the same business sector, as those contributing to ISBSG will produce the same results. Researchers may use our data analysis procedure to realize effective use of the ISBSG data. Secondly, progressive cost estimations according to the evolution of phases reduce the uncertainty reported in previous studies.¹⁵ Software engineers are therefore suggested to record, maintain, and use development effort data for each SDLC phase, as well as their contribution to the total work effort, while performing their projects. Such activity can contribute to minimizing uncertainty and increasing the quality of cost estimations. Tracking of progress and percentages of effort cost estimates may offer improved prediction accuracy and control over resource allocation for software projects. Prediction models using partial phase data, for instance, can be used in practice to estimate the summary work effort required to complete projects. This was demonstrated with ANN using the ISBSG dataset but to achieve further validation of this outcome and move outside the experimental domain considering more realistic conditions, more rigorous empirical studies need to be performed in the future in close collaboration with the software industry.

6.3 | Threats to validity

In this section, we discuss the threats to validity of our case study, based on the categorization described by Runeson et al.⁵⁰ Regarding external validity of the work the following threats exist:

Data representativeness: This work builds upon observations and measures of empirical results obtained from a specific dataset and the conclusions obtained may not apply in different contexts. The ISBSG dataset represents the most productive projects in the industry rather than the industry standards³⁵ because the participating organizations are carefully selected as pioneers in the field. However, the ISBSG dataset contains a wealth of software development data that come from a broad industry cross section which differ in size, effort, platform, language, and development techniques. Therefore, it covers a wide range of possible project attributes, application domains, and development environments that can be considered to represent the needs of a variety of different software development houses. We believe that the ISBSG dataset contains multiorganizational; cross-sector data can mitigate the impact of the experimentation in this single dataset and context. Still a wider application to more datasets, including single company datasets, could further generalize the results and reveal useful insights regarding phase-based effort estimation. Application of the method to single company relevant datasets is expected to verify the results of our analysis since the model is derived from a variety of representative projects adopting process-oriented development.

Degree of variability: The ISBSG dataset engaged in this study, suffers from 2 limitations: (1) the combination of data coming from heterogeneous sources, and (2) the missing values. The projects included in the experiments vary considerably in terms of characteristics. For example, effort distributions in relation to a set of project attributes, eg, functional size, implementation year, development type, language type, resource level and maximum team size, show that such variations exist, but their investigation may offer considerable assistance in revealing observations regarding the properties of the data sample under consideration. To minimize the impact of heterogeneity and diversity, this study adopted all the relevant data preparation processes as described thoroughly in Section 5.1, including list-wise deletion for minimizing missing values, obtaining finally an appropriate data subset for the experimentation. Additionally, software cost estimation techniques, such as ANN, applied in this study are based on the assumption that the participating projects are independent, a condition that is satisfied by the ISBSG repository, which is a large cross-sector dataset. Nevertheless, quality and consistency issues for the projects within the dataset are initially resolved by the ISBSG's quality reviewers that have assessed the quality of the submitted data values.

Software life cycles and effort mapping: The ISBSG has asked data contributors to map their project phases to the basic effort breakdown of planning, specification, design, build, test, and implementation. However, the various SDLC adopted by organizations might not be easily mapped to this basic phased breakdown or some activities performed might not belong to any of these phases. Moreover, clear distinctions on when a phase starts and ends, as well as which exact activities should be measured within the phases do not exist and this may result in measurement errors. Such errors, however, do not appear in the data, as verified by the quality assessments done by the ISBSG reviewers. In addition, to resolve the issue of activities that do not fall into specific phases ISBSG has provided the field of unphased effort. A major prerequisite, however, entails that cost estimations of RQ3 and RQ4 may be answered if there is a clear identification that a project phase has successfully been concluded and that the exact effort measurement is known. Moreover, future effort estimations from the specific project development point requires that subsequent effort will not include effort allocated to phases that have already been completed (previous project phases) and more particularly that the development is sequential.

Regarding reliability, based on the fact that our study is purely quantitative and the developed protocol (see Section 5) is thoroughly described, we believe that the process is easily replicable by other researchers. Concerning construct validity, the construction of the ANN model includes by nature several parameters that need to be customized, such as the architecture type, the validation method, the number of layers, etc. In our analysis, a range of ANN architectures with varying number of neurons in the hidden layer was deployed, each of which was validated to ensure construct validity. Still this fact provides an interesting direction for future studies that could additionally include sensitivity analysis regarding the parameters of the model.

7 | CONCLUSIONS

In this study, effort is considered as a dynamic and adjustable element which can be modeled and regulated along project evolution according to the specific phase of the estimation. Prior work does not consider the dynamic nature of software development while estimating project effort whereas the estimation models may be optimistic because of the lack of availability of the whole dataset or project information. The main research goal is to investigate whether the identification of distribution patterns in consecutive development phases located in recent industrial projects may assist in increasing project managers' awareness on the project evolution. Subsequently, from the managerial perspective, this will lead to improving project staffing, controlling, and reducing the associated risks. Other benefits constituting possible outcomes of this work include effortless return on investment, improved resource scheduling, reduced uncertainty, and enhanced decision making.

In this work, we analyzed effort allocation among the classical project development phases and argue that understanding the relationships between them may result in better project resource allocation. We presented empirical effort analysis on phased effort of projects from the ISBSG (R10) data repository in 4 successive phases proposing the following models: (1) early, used project attributes to estimate the total effort and the effort required for the planning phase; (2) postplanning, used project attributes and considered that the planning phase was completed and reviewed the total effort estimate and also estimated effort of the subsequent (specifications) phase; (3) postspecification, considered that the specifications phase was completed and used the actual planning and specifications effort to review the total effort estimate and also estimate the effort required for the design phase; and finally, (4) postdesign model, the design phase was completed and the actual design effort was used to adjust the total effort estimate and approximate the effort for the building phase. Moreover, we identified the correlations between total effort and phased-effort to investigate the possibility for project managers to build early cost estimates and refine them at later phases of the project.

Regarding the 4-stage effort models used, the very high correlations found especially between plan, design, and build and the total effort, ie, implied that substantial influence exists between the "early" phases of development and the actual effort expended. In addition, the observations obtained concerning the average ratio difference of phased-effort and total effort distributions helped in assessing the symmetry of the phases' probability distribution while identifying their connection with significant project attributes, ie, development type, development platform, language type, and groups of maximum team size.

In addition, the evaluation of between-phases correlations showed that build and design efforts are highly dependent on design and specification efforts, respectively, and thus, an estimation model predicting effort for subsequent phases would be worth investigating. Other correlations identified include: plan with specify, design, and build effort; specify with build and implementation effort; and finally, design was found highly correlated with testing and implementation efforts.

Prediction results showed that the four-stage effort estimation models proposed increased significantly the accuracy of the total development effort estimations, as more information was added to the models and as the projects advance in progress, from very low initial accuracy levels to higher in accuracy predictions. The updates on effort estimates and the reassessments were performed in four sequential and fixed development points, ie, before planning, before specifications were recorded, before design was completed, and before the actual building (implementation) of the software. We selected to update estimates before these phases as they occur relatively early in the SDLC and also produce useful information regarding the complexity of the application (for example, approximations of size metrics). Additionally, the results of the four stage model for phased-effort estimations suggested that it is safer to estimate the total effort that will probably be required for the project from the design phase, that is, after the specifications phase is performed.

Using the effort from the completed phases of the development life cycle for approximating the subsequent phased-efforts the models showed that design and Build efforts only can be estimated better than the earlier phases. Regarding the above observation, further investigation needs to be performed on the complexity and intricacy of the estimation task. Finally, we concluded that further research is needed on readjusting a model for estimating the subsequent phase effort. The creation of models for dynamically predicting phased effort can be considered as a new emerging opportunity to understand effort phased allocations and create more reliable models satisfying practitioners' needs.

Future work involves extending the experimental studies using datasets that might contain information regarding effort values for tasks related with specific development phases, even though such data, outside the ISBSG dataset, with work breakdown data are hard to find. Such work is expected to confirm the accuracy of the models on different datasets and will also enable qualitative interpretation of the results, as well as provide better understanding on more dataset cases. Additionally, a wider application to more datasets, including single company datasets could further prove generalizability and reveal useful insights regarding phase based effort estimation in specific companies, industries, domains, etc. Moreover, an automated, self-adaptive cost estimation framework for recalibrating effort estimations in progressive stages of development is also one of our future research steps, which will enable significant improvements in managing subsequent project tasks and resources.

Finally, our future plan is to conduct empirical studies, to scale our framework outside the experimental/lab environment and aim to generalize.⁴⁹ One way to accomplish this is to evaluate correlations of effort between project phases of on-going projects in industry, even if this means that the models will be developed on specific, biased and even partially incomplete information. Effort data allocated for the ongoing project activities may be used by our models to predict the effort required for upcoming phases or activities. Practitioners may assess whether this information is applicable to their particular case, ie, if it is generalizable or perhaps needs to be adapted. This information can be given to practitioners to act and perform corrective measures to prevent erroneous or risky effort allocations and improve project management. In addition, feedback may be collected for our framework from practitioners, to improve our method, iterate, and redesign the models, and thus enable us to better scale up to practice.

REFERENCES

- MacDonell SG, Shepperd MJ. Using prior-phase effort Records for re-estimation during software projects. Proceedings of the 9th International Symposium on Software Metrics. IEEE Computer Society: Washington, 2003:1-13.
- MacDonell, SG, Shepperd M. Data accumulation and software effort prediction. ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). ACM: New York, NY, USA, 2010:31:1-31:4. <https://doi.org/10.1145/1852786.1852828>.
- Chatzipetrou P, Papatheocharous E, Angelis L, Andreou AS. A multivariate statistical framework for the analysis of software effort phase distribution. *Inf Softw Technol*. 2015;59:149-169.
- Yiftachel P, Peled D, Hadar I, Goldwasser D. Resource allocation among development phases: an economic approach, international workshop on economics driven software engineering research (EDSER). ACM: Shanghai China, 2006:43-48.
- International Software Benchmarking Standards Group. The benchmark release 10, 2008. <http://www.isbsg.org> [June 2016].
- Werbos PJ. Beyond regression: new tools for prediction and analysis in the behavioral sciences. PhD thesis, Harvard University, Cambridge, MA, USA, 1974.
- Albrecht AJ. Measuring application development productivity. Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium. IBM Corporation: Monterey California, 1979:83-92.
- Nassif A, Ho D, Capretz LF. Towards an early software estimation using log-linear regression and a multilayer perceptron model. *J Syst Softw*. 2013;86(1):144-160.
- Kitchenham B, Brereton OP, Budgen D, Turner M, Bailey J, Linkman S. Systematic literature reviews in software engineering—a systematic literature review. *Inf Softw Technol*. 2009;51(1):7-15.
- Trendowicz A, Jeffery R. *Software Project Effort Estimation*. Foundations and Best Practice Guidelines for Success, Constructive Cost Model-COCOMO; 2014:277-293.
- Jørgensen M, Shepperd MA. Systematic review of software development cost estimation studies. *IEEE Trans Softw Eng*. 2007;33(1):33-53. <https://doi.org/10.1016/j.infsof.2011.09.002>
- Jørgensen M. Forecasting of software development work effort: evidence on expert judgement and formal models. *Int J Forecast*. 2007;23(3):449-462. <https://doi.org/10.1016/j.ijforecast.2007.05.008>
- Moløkken-Østfold K, Haugen NC, Benestad HC. Using planning poker for combining expert estimates in software projects. *J Syst Softw*. 2008;81(12):2106-2117. <https://doi.org/10.1016/j.jss.2008.03.058>
- Heemstra FJ. Software cost estimation. *Inf Softw Technol*. 1992;34(10):627-639.
- Boehm BW. *Software Engineering Economics*. Upper Saddle River, New Jersey: Prentice Hall PTR; 1981.
- Boehm BW, Abts C, Brown A, et al. *Software Cost Estimation with COCOMO II*. New Jersey: Pearson Publishing; 2000.
- Putnam LH. A general empirical solution to the macro software sizing and estimating problem. *IEEE Trans Softw Eng*. 1978;4(4):345-361. <https://doi.org/10.1109/TSE.1978.231521>
- Brundick B (General Editor). Parametric cost estimating handbook. <http://cost.jsc.nasa.gov/pcehg.html> [August 2011]; <https://archive.is/Bzqgr> [June 2016].
- <http://dec.bournemouth.ac.uk/ESERG/ANGEL/> [March 2011].
- Gray A, MacDonell, SG. Fuzzy logic for software metric models throughout the development life-cycle. Proceedings of the 1999 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS). IEEE Computer Society Press: New York USA, 1999:258-262.
- Heiat A. Comparison of artificial neural network and regression models for estimating software development effort. *Inf Softw Technol*. 2002;44:911-922. [https://doi.org/10.1016/S0950-5849\(02\)00128-3](https://doi.org/10.1016/S0950-5849(02)00128-3)
- Tronto IFDB, Silva JDSD, Sant'Anna N. An investigation of artificial neural networks based prediction systems in software project management. *J Syst Softw*. 2008;81:356-367. <https://doi.org/10.1016/j.jss.2007.05.011>
- Li JZ, Ruhe G. Analysis of attribute weighting heuristic for analogy-based software effort estimation method AQUA+. *Empir Softw Eng*. 2008;13(1):63-96. <https://doi.org/10.1007/s10664-007-9054-4>
- Li JZ, Ruhe G, Al-Emran A, Richter M. A flexible method for software effort estimation by analogy. *Empir Softw Eng*. 2007;12(1):65-106. <https://doi.org/10.1007/s10664-006-7552-4>
- Software Engineering Laboratory SEL-84-101. *Manager's Handbook for Software Development*. Goddard Space Flight Center, Greenbelt, Maryland: NASA; 1990.
- Pressman RS. *Software Engineering: A Practitioner's Approach*. 5th ed. New York: McGraw-Hill; 2000:172.
- Zelkowitz MV. Perspectives on software engineering. *ACM Comput Surv*. 1978;10(2):197-216. <https://doi.org/10.1145/356725.356731>
- Boehm BW. Industrial software metrics top 10 list. *IEEE Softw*. 1987;4(5):84-85.
- Brooks JFP. *The Mythical Man-Month*. USA: Addison-Wesley Pearson Education; 1995.

30. Ambler SW. A Manager's introduction to the rational unified process (RUP), December 2005. <http://www.ambyssoft.com/downloads/managersIntroToRUP.pdf> [June 2016].
31. Sommerville I. *Software Engineering (Update) 08 Edition*. England: Pearson Education Limited; 2007.
32. Ohlsson MC, Wohlin C. An empirical study of effort estimation during project execution. Proceedings of the 6th International Software Metrics Symposium. IEEE: Boca Raton FL, 1999:91-98.
33. Little T. Schedule estimation and uncertainty surrounding the cone of uncertainty. *IEEE Softw*. 2006;23:48-54. <https://doi.org/10.1109/MS.2006.82>
34. Yang Y, He M, Li M, Wang Q, Boehm BW. Phase distribution of software development effort. Proceedings of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). ACM-IEEE: Kaiserslautern Germany, 2008:61-69.
35. Fernández-Diego M, Ladrón-de-Guevara FG. Potential and limitations of the ISBSG dataset in enhancing software engineering research: A mapping review. *Inf Softw Technol*. 2014;56(6):527-544.
36. Jiang Z, Naudé P, Jiang B. The effects of software size on development effort and software quality. *Int J Comput Inf Sci Eng*. 2007;1(4):230-234.
37. Déry D, Abran A. Investigation of the effort data consistency in the ISBSG repository. 15th International Workshop on Software Measurement (IWSM). Shaker-Verlag: Montreal Canada, 2005:123-136.
38. Chatzipetrou P, Papatheocharous E, Angelis L, Andreou AS. An investigation of software effort phase distribution using compositional data analysis. *IEEE*, 2012; 367-375. <https://doi.org/10.1109/SEAA.2012.50>.
39. Menzies T, Krishna R, Pryor D. The promise repository of empirical software engineering data. North Carolina State University, Department of Computer Science <http://openscience.us/repo> [June 2016].
40. IEEE standard 610.12-1990 glossary of software engineering terminology.
41. Kruchten P. *The Rational Unified Process: An Introduction*. Boston MA, USA: Addison-Wesley Longman Publishing Co. Inc.; 2003.
42. Spearman C. The proof and measurement of association between two things. *Am J Psychol*. 1904;15:72-101.
43. Myers JL, Arnold DW. *Research Design and Statistical Analysis*. 2nd ed. New Jersey: Lawrence Erlbaum Associates Inc.; 2003:508.
44. Maritz JS. *Distribution-Free Statistical Methods*. Chapman & Hall, London; 1981:217.
45. McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activity. *Bull Math Biol*. 1943;5(4):115-133.
46. Conte SD, Dunsmore HF, Shen VY. *Software Engineering Metrics and Models*. Redwood City CA, USA: Benjamin-Cummins Pub Co; 1986.
47. Gibbons JD. *Nonparametric Methods for Quantitative Analysis*. Columbus Ohio: American Sciences Press; 1985.
48. Hopkins WG. A new view of statistics. <http://sportsoci.org/resource/stats> [October 2011].
49. Wieringa R, Daneva M. Six strategies for generalizing software engineering theories. *Sci Comput Program* 2015; 101: 136-152. ISSN 0167-6423.
50. Runeson P, Host M, Rainer A, Regnell B. *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons; 2012.
51. Dennis A, Wixom BH, Tegarden D. *Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach*. 2nd ed. New Jersey: John Wiley & Sons Inc.; 2005.
52. Jones C. *Estimating Software Costs*. New York: McGraw-Hill; 1998.
53. Rubin HA. Macroestimation of software development parameters: the ESTIMACS systems. SOFTAIR Conference on Software Development Tools, Techniques, and Alternatives. IEEE Press: New York, 1983:109-118.

How to cite this article: Papatheocharous E, Bibi S, Stamelos I, Andreou AS. An investigation of effort distribution among development phases: A four-stage progressive software cost estimation model. *J Softw Evol Proc*. 2017;e1881. <https://doi.org/10.1002/smr.1881>