Applied Nanoscience (2022) 13:2405–2417
https://doi.org/10.1007/s13204-021-02204-9

ORIGINAL ARTICLE

Evaluation of estimation in software development using deep learning-modified neural network

N. Sreekanth[1] · J. Rama Devi[2] · Kumar A. Shukla[3] · D. K. Mohanty[4] · Amedapu Srinivas[5] · G. Nageswara Rao[6] · Afaque Alam[7] · Ankur Gupta[8]

Received: 18 September 2021 / Accepted: 26 October 2021 / Published online: 6 February 2022

Abstract
For both clients and developers in the development of software, the effort estimate is vital. Lower efforts may lead to poorly created processes, low-quality and delayed budgets being improperly approved by the management. This could lead to a failure to complete the task in the defined schedule. The assessed parameter helps to share the information needed to accomplish project results successfully. The main parameters for the software projects include time, conditions, people, infrastructure, materials and funds and hazards. For this reason, large evaluations of various elements, including effort and cost, are vital to a project. If the estimate is less than necessary, the advancement of the project is also lacking in cash and time. The paper's main aim is to evaluate the software development estimate to improve current RE, MRE, MMRE and PRED methodologies.

Keywords Software · Estimation · Neutral network · Deep learning

Introduction

The development of software includes a variety of aspects that significantly affect productivity. The estimate of effort considered an important aspect of software development. The right development depends on the right estimate. Various strategies are used to estimate software development. In today's competitive environment, the accuracy of the estimate is most vital. The clients recognize gradually the value of project management and management through popularizing information engineering in software projects. Software developers are sufficiently precise in the effort and quality assurance given. Many organizations still care to precisely predict the software development effort. Reducing development costs and timetables can have a detrimental impact on usability, quality, reputation, and competitiveness of software products (Azath and Wahidabanu 2012; Cuadrado-Gallego et al. 2006). The estimate of costs and cost of software projects is a very useful tool, because software has an enormous role in the worldwide market today. However, the effort to build a software system with optimal methodology is not properly estimated. The data collected are usually not adequate for an accurate prognosis in the early stages of system development. Methods of estimating effort may be based on expert subjective evaluations or formal estimates. Formal models use the unique identifier to estimate quantitative value effort. The efficient management of a software project requires an accurate and trustworthy software effort estimate. Several methods have been built and evaluated over the past decade, using previous project data and various ideas and frameworks coupled with current methods to estimate the software efforts. When estimating the amount and effort available right from the outset, the

✉ D. K. Mohanty
dkmohanty.iitkgp@gmail.com

1 Malla Reddy Engineering College for Women, Secunderabad, Telangana, India

2 Department of CSE, PVP Siddhartha Institute of Technology, Vijayawada, India

3 Vellore Institute of Technology, Vellore, Tamil Nadu, India

4 School of Computer Engineering, Kalinga Institute of Industrial Technology, Deemed to be University, Bhubaneswar, Odisha, India

5 Department of CSE, Sreenidhi Institute of Science and Technology, Hyderabad, India

6 Department of EEE, Lakireddy Bali Reddy College of Engineering, Mylavaram, AP, India

7 Department of CSE, Bakhtiyarpur College of Engineering, Bakhtiyarpur, Bihar, India

8 Department of Computer Science and Engineering, Vaish College of Engineering, Rohtak, Haryana, India

project manager is sure about future actions because many actions made during the development are affected or rely upon initial estimations. Therefore, effort evaluation is one of the essential processes in the planning and management of projects in software. The development effort is considered one of the essential components of the software cost and is usually the most difficult task to anticipate, in particular in technical innovation. In recent years, numerous ways to help project development estimate the work to produce software for global software development projects have been proposed. Software effort estimate is an important phase in software project management as the future management of the software project is largely dependent on the exactness of the effort assessment. The critics of software cost estimation include (1) effort estimation procedure in the earlier phase of software development and (2) although several software cost estimating methods and performance measures are accessible exponentially (Morrow et al. 2014; Aroba et al. 2008; Popovic et al. 2015; Silhavy et al. 2017).

It is regarded as the very primordial phase of the software development process while being believed to be the main duty, as correct evaluations of the growth of the present project can only be carried out when its delivery accuracy and its cost management are accurate. And a more wide-ranging assessment of a software project that is presently under development will lead to a better schedule of your organization's future software projects (Urbanek et al. 2014). With an above-mentioned cause, several researchers from so many past decades have received major attention from software efforts. In other words, the estimate of the software cost is the combination of projections of a software project construction effort and calendar time. The work contains the summary of the working hours and all employees involved in the soft project development process. Only from the very beginning of the development of software projects have these organizations faced the problem of inaccurate estimates of software development and development time. The fact that unclear data on the software project will be generated at the time of its estimation procedure was a good explanation for this and this time is persistent. Any software development project manager can assess the progress, giving him/her a good track of the prospective cost management and accuracy in the delivery time, software product evaluation is the only thing which can. However, this provides the organization with a more comprehensive insight into the use of resources and will therefore help the company to plan more successfully its future projects. In this respect, several strategies for software cost estimation in the last many decades have been presented, which differ, but sadly do not meet the acceptable requirement of accuracy in software development projects, from algorithmic to non-algorithmic to dataset mining (Seo et al. 2013; Satapathy et al. 2016). There have, however, also been developing and showing gains in a different variety of artificial neural networking models. This research is based on the development of an estimating model based on hybrid of artificial neural networks and certain metaheuristic optimization techniques for accuracy in software cost estimations.

Effort estimation using deep learning-modified neural network

This is the prediction of the effort to build or manage software products in person. Estimates can be categorized in several ways. The analyses based on the size-centered estimate technique include function point analysis (FPA), class point analysis and case point analysis in the formal estimate model (UCP). UCP is used to estimate the effort required to finish the SD (i.e., estimate effort). The UCP modeling software is the basis for the estimating effort (UC). The estimated effort is achieved through the multiplication of UCP with the effort rate (ER). For the purposes of determining the total human resources required for a project, an ER is the estimate value generated based on past project statistics. CPA is simply utilized here because a design from the most important unified modeling language (UML) diagrams explicitly inferred the class point. The primary advantage with CPA over FPA is that during the coding stage, the total FP is calculated, but that the class point from the SDLC design phase is calculated (Azzeh et al. 2011). Therefore, this can be estimated at the beginning of the SDLC. The estimate of the software development work for the project offers and plans (SDEE). In additional to plans, the cost of inadequate budgets may be greater, which means that business prospects may be wasted if there are big losses due to pessimism and over-optimism. The software business is still less skilled in managing the SD budget predictions, the most onerous problem (Qi et al. 2017). This has enabled current SD firms to take various things into account (improper requirement engineering, ambiguous resource elicitation, uncertain cost, and 11 EE). In the end, this leads to the breakdown of the development phase. To combat these chaos-provoking difficulties, the SD firm has to create a counter mechanism. An excellent technique to address this challenge is to further estimate the whole process and determine if the project developed is viable and utilized exclusive with the resources available. SEE estimates the time necessary for the completion of the SD process or project. The EE is important to know how much the corresponding software value is created. The SDEE is the most convenient way to computer the software required to construct or maintain software for inputs, such as FP, size, software design, UCP and the like in person. Furthermore, the EE is also seen as the contribution to the iteration plan, investments analyses, budgets, project plans, pricing and invitation to tender. Mostly, only the formal SEE or UC or

FP models were built to estimate size or development effort, but the software SRS is not included for this calculation. Therefore, a strategy is needed which can estimate the SD effort owing to the software's demand complexity (Azath and Wahidabanu 2012; Morrow et al. 2014; Urbanek et al. 2014; Satapathy et al. 2016).

One of the crucial factors of business development is knowledge and technology. Most companies rely on software and hardware technology. The company also reflects on the investment to be made in the software. Some of the company's purchase new software while some of them themselves build new software. In this whole situation, time and money investments have a significant part to play, and the cost and time of software development must thus be estimated (Popovic et al. 2015; Silhavy et al. 2017; Araujo et al. 2017). Software cost estimation is the approach to calculate the size, effort, duration and cost of software development. The software has shown to be an essential and necessary component in the development of a software project and to reduce the cost of software. Consequently, various approaches for cost estimating have been created. In the project planning stage of the software development life cycle, the software cost estimation process begins (SDLC). The initial cost estimate phase is highly significant, because the correctness of the estimates depends on how much trustworthy data the budget and the time in which the project is completed are available to the estimator. The method of cost estimating consists of collecting the accurate raw data. If it is not finished in time, it can result in an excess of budget. The principal method is to accomplish the project in due course, using budget and resources. Resources include, for example, software, hardware, guidance, testing, etc. The estimation of software costs is separated into two classes: methods of algorithm and non-algorithm (Murillo-Morera et al. 2017; Pospieszny et al. 2017).

Related work

Azath et al. (2020)

Efficient cost estimation is the most challenging business in a software development process. The estimation of software work is a key component of cost estimation. Management takes careful account of software efforts and advantages before allocating the necessary resources to the project or to the contract order. Unfortunately, this preliminary estimate is difficult to measure, as it provides limited early-stage information about the project. The study proposes a new approach based on the rationale of software computers to estimate software work. The rules are based on the input dataset in this method. These rules are then grouped to properly estimate them. For the clustering of the dataset, we use modified fuzzy C methods. Once the clustering is completed, different rules are obtained and the input for the neural network is delivered as these rules. Here, using optimization algorithms, we are modifying the neural network. The optimizing algorithms used in this field are the ABC, modified cuckoo search and hybrid ABC–MCS algorithms. Three optimized sets of rules are thereby obtained that are utilized to estimate the effort process. Parameters, such as median absolute relative error and medium relative error, are used to examine the efficiency of our proposed technique.

Kodmelwar (2019)

Evaluation is an important part of the project process for any good project management. In the project management, it plays a significant role in executing the necessary directives. The parameter analyzed helps to share resources needed for the successful completion of the project results. Time, requirements, people, infrastructure/materials and funds and threats are key criteria in controlling the software projects. This is one reason why it is extremely essential for a project to evaluate key factors, such as the time and resources, they demand. The evaluated parameter helps to share the resources needed to properly complete the project results. Time, prerequisites, persons, infrastructure/materials and funds and danger are the important criteria that control software initiatives. This is one reason why it is incredibly fundamental to evaluate key components, such as time and resources, required for a project. If the estimate is lower than needed, then cash and time are also lacking with the advancement of the project. If the possibilities of wasting resources are overestimated, then it is therefore necessary to properly anticipate the situation in future to avoid it.

Kodmelwar et al. (2018)

Efforts are not only required in the maintenance of software but also in the development phase of software. For the estimation of effort over the years, a number of methodologies have been established. There is already soft computing and weighted algorithms for the existing methodologies. For the software company to create a software product, the right method of estimating work is important. A novel and effective paradigm for estimating the effort of software development is created to make this achievable. Java as the front-end tool and COCOMO data pack as the backend tool are used to perform the proposed method. And that's how it is done. The data set is first considered as the input. The CSA is then used for the formulation of the deep learning-modified neural network (MNN). The neural network type (NN) is the co-evolutionary network. The optimization is carried out utilizing HPSO in this step (hybrid particle swarm optimization). Finally, the parameters of performance were calculated. The results have been compared to the work already done. The work proposed had very favorable outcomes.

Altuntas and Alptekin (2017)

Software is fast-growing and becoming increasingly important worldwide. Almost all companies and institutes from many industries develop new apps and platforms with software projects. Accurate effort estimates have become, as necessary in every project, an important concern for businesses, especially project managers. Several methodologies and models for estimating software projects have been developed since the 1970s. COCOMO, a constructive method proposed in the late 1970s, was the first milestone model. A number of alternative types have followed, with function point and use case point being the most popular and usable. Following the 2000s, artificial neural networks became important particularly among the problem areas that benefit from data analysis and self-learning because of developments in technology. The estimates of effort for software development are also similar to prior project data, which could help to predict efforts in new projects. Software development effort estimates. This article therefore uses neural network methods to develop a model for software estimation. Due to an exhaustive survey, the features of the network were selected. The applicability of the methodology is proved with real-life project data from one of Turkey's biggest banks.

Yadav and Singh (2014)

Software development budget calculation is dependent on the development work and planning prediction of the software. On the basis of historical software project data sets, the software development effort and timetable may be anticipated exactly. In this paper, a model has been constructed to estimate the cost of object-oriented software development using a hidden layer feed-forward neural network (OHFNN). Utilizing the weight vector from OHFNN as an initial population for the genetic algorithm, this model has been further optimized using a genetic algorithm. Convergence was achieved by reducing the number of squared errors in each input vector and the software development effort was predicted on optimum weight vectors. On the PROMISE software engineering repository data set, the model was empirically validated. The model performance is more precise than the commonly known cost model (COCOMO).

Proposed methodology

The EE is vital for SD consumers and developers. This decreasing effort can assist managers to approve systems that exceed their budgets with weak functions and poor quality. In the time period before the tasks have been accomplished, it can finally lead to a failure. In all computer centered systems, software is considered the core of the process. This crucial component is developed using correct criteria. Each organization's main goal is to economically develop software products. The major objective is precisely the precise projected effort needed to carry out the tasks. Many past investigations have proven that projects exceed the estimates and completion time specified without accurate reporting

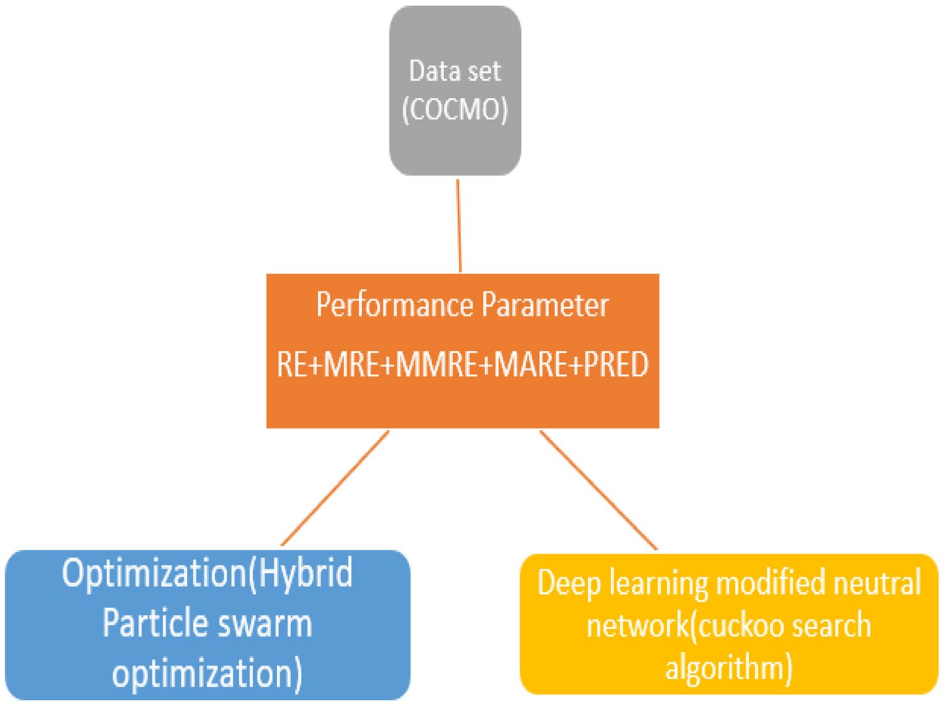
Data set (COCMO)
Performance Parameter
RE+MRE+MMRE+MARE+PRED
Optimization(Hybrid Particle swarm optimization)
Deep learning modified neutral network(cuckoo search algorithm)

Fig. 1 Architecture of the proposed method

Table 1 Effort multipliers

Acap	Analysts capability
Pcap	Programmers capability
Aexp	Application experience
modp	Modern programming practices
tool	Use of software tools
vexp	Virtual machine experience
lexp	Language experience
sced	Schedule constraints
stor	Main memory constraints
data	Data base size
time	Time constraints for cpu
turn	Turnaround time
virt	Machine volatility
cplx	Process complexity
rely	Required software reliability

and careful planning. Effective EE in an SD procedure is the most important activities of modern computer technology (Azzeh 2017; Al Yahya et al. 2010). Here is proposed a new EE framework. Neutral network is used to conduct the EE. Using a neutral network helps to estimate software efforts using lower costs and failure rates. The neutral network is utilized simultaneously with optimization to produce an enhanced result. When the neutral network weights are selected, the optimization procedure is carried out that helps to better classify the model. The optimization algorithm for the work being suggested is HPSO with genetic operators. Various model performance monitoring parameters, such as RE, MRE, MMRE, MARE and PRED, are calculated and the effort to evaluate the system efficiency compared to various existing models. To measure system efficiency, other models are also calculated. The following block diagram can further explain the system proposed (Fig. 1).

Data set specification: to enter the suggested system, the COCOMO dataset is employed. This is a publicly available repository data set. This dataset is publicly available to encourage the development of software that can be reject, repeated, verified or improved. Different EE approaches have typically been validated by the COCOMO dataset.

In it there are 60 SPs, which are determined by 17 attributes in addition to the present effort. In the COCOMO data set, the actual effort is measured per individual per month, indicating that a person has to create a given project for the whole number of months. The exactness of new methodologies remains a typical assessment, even when COCOMO's dataset has now reached 25 years. COCOMO presupposes that software dimensions are straightforward (Merlo-Schett et al. 2002; Azzeh and Nassif 2015; Bardsiri et al. 2012).

Effort multipliers

Each multiplication cost driver has a list of rating levels and a set of EMs (CD). This nominal level never changes the projected EE = 1.00 effort. In general, the projected effort is modified by nominal ratings. The anticipated costs would be raised by 10% to a higher RELY as demonstrated in the COCOMOII.2000 calibration (required software reliability). However, 26% would add a high RELY rating. The project can receive intermediate ratings and different EMs. Size driver and cost driver ratings could not be equal for each module but the same one for the dimensions and for the development programmed cost driver requires (Dan 2013). The COCOMO II model can be utilized to estimate timetables and effort on the entire project or a multi-module project. The EMs are listed in Table 1 (Dejaeger et al. 2012). The estimated values of the EMs are categorized in a series of classifications. Classes are extremely high, very high, minimum, low, very low and high productivity.

Cost drivers

CDs are utilized to capture the SD characteristics that affect the project work. It is an estimated cost driver model factor. Cost driver (in this case, person-months). The quality ratings of all COCOMO cost drivers represent the development efforts of the cost driver. This is the range between extra low and extra high. A value called EM is present at the rating level for every multiple cost driver (El Bajta 2015). This context turns a qualitative cost driver classification into a QC for use. The EM value of 1.00 is awarded a multiplying cost driver rating. If there is greater SD effort in the rating level of a multiplicity cost driver, its EM correction is > 1.0. If the rating level lowers the attempt, then the EM rating level is < 1.0.

Cuckoo search algorithm

CS is a strategy used to optimize the outcomes of the dataset. CS is a global optimization method that imitates a few cuckoo species parasitic behavior. This conduct is closely linked to its brood parasitism, in which they place their eggs in another host bird's nest. This approach enhances the likelihood of survival of their eggs. The host bird may sometimes locate such ovens, remove the unknown ovens or leave the nest and build a new one (Fig. 2).

There are also two essential components in this approach for intensifying and diversifying search. The initial is the search operator crossover that employs LFs to increase the CS algorithm's operational performance. The randomness of LFs and variance are more successful than 85j typical random walks with Brownian movements, since the range

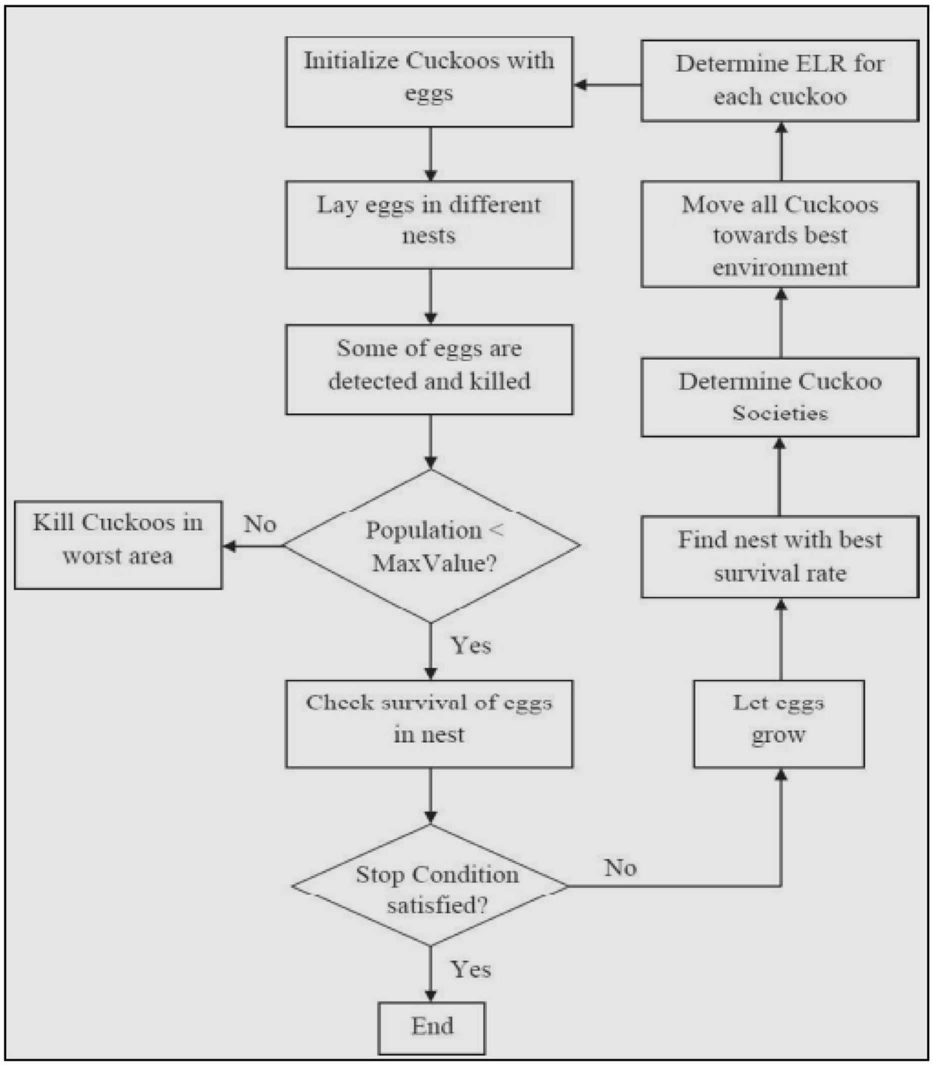
Initialize Cuckoos with eggs
Determine ELR for each cuckoo
Lay eggs in different nests
Move all Cuckoos towards best environment
Some of eggs are detected and killed
Determine Cuckoo Societies
Kill Cuckoos in worst area
No
Population < MaxValue?
Find nest with best survival rate
Yes
Check survival of eggs in nest
Let eggs grow
Stop Condition satisfied?
No
Yes
End

Fig. 2 Flow diagram of CS algorithm

of steps provided by this technique is unlimited. This CS method uses LFs to create a new solution.

Result of the experiment

The data set values are considered to be the input values for implementing the system. There are various properties for the input. A definition of the present and the suggested performance dimension is available for the proposed system. A clear examination contrasted the existing and proposed system. Total 50 examples (10, 20 and 50) were evaluated on the basis of which results were derived (Hamdan et al. 2006). In the performance evaluation, RE, MMRE, MARE, (PRED), (AE), ET, (MAE), RMSE, RRMS are measured and measurements are employed (APE). These variables analyze the offered procedures and find the results after each 10 examples have been added.

Relative error

The size of the actual value is called the error measurement relative error. The actual error is compared with the real value size:

$$\text{Relative error} = \frac{(\text{estimated error} - \text{actual error})}{\text{actual error}}.$$

Figure 3 shows the relative error variation for the current system and the deep proposed system approaches developed for that purpose. In 50 instances, the existing system is 0.3737 relative error; in contrast to the existing deep proposed system are 0.3154 less relative errors than the existing system. The relative error is about equal in 10, 20 and 30 occurrences; however, the recommended value indicates a smaller relative error value compared to the current existing system (Kad and Chopra 2012). In 40 cases, the present existing system and the proposed

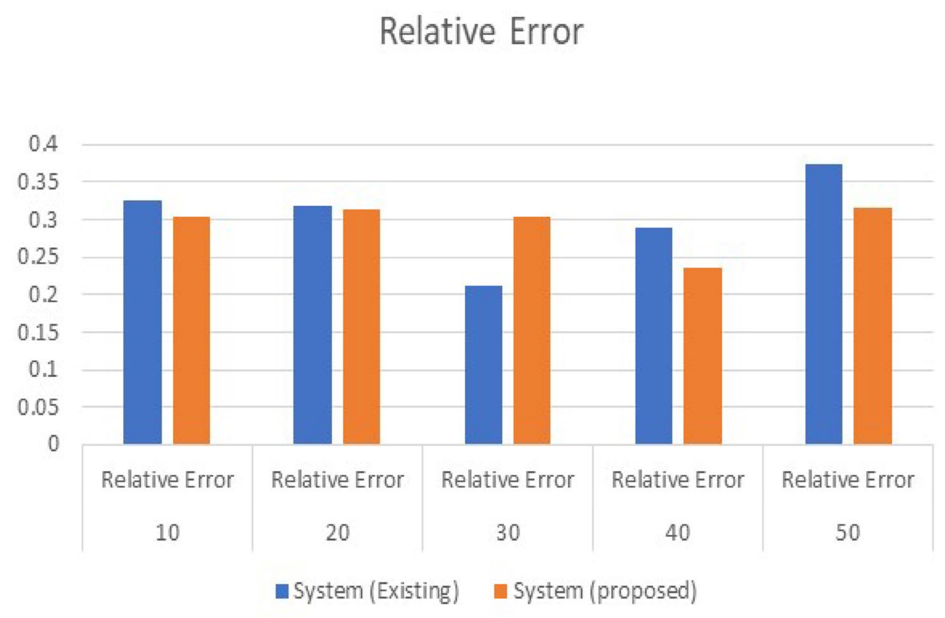

Fig. 3 Relative errors of different occurrences

deep proposed system are slightly different. Based on these assessments, the proposed deep proposed system has low RE values and for all comparable instances; the existing system has a high relative error value (Karaboga and Akay 2009).

Magnitude of relative error

Relative error magnitude is the computed performance parameter of this system. It is a key component of software analysis and it shows the level of error assessment in a single evaluation:

$$\text{Magnitude of relative error} = \frac{(\text{estimated error} - \text{actual error})}{\text{actual error}}.$$

Figure 4 compares the performance of the current existing system and the proposed deep proposed system systems with the magnitude of relative errors values. The existing SYSTEM has 0.8684 for 50 instances, but the proposed deep proposed system has 0.1658 which is lower compared to the previous magnitude of relative errors. In a lot of situations, the present and suggested systems have a huge difference in the magnitude of relative errors value, while the 104 N deep magnitude of relative errors proposed has the lesser magnitude of relative errors value (Kocaguneli et al. 2013). The diagram in the Fig. 4 confirms the effectiveness of the proposed system over the conventional existing system.

Mean magnitude of relative errors

The existing system and deep proposed system techniques are defined for mean magnitude of relative errors. Mean magnitude of relative errors is an average estimate accuracy measure and a de facto evaluation scenario to obtain SP prediction models accuracy. The mean value of RE is considered to be mean magnitude of relative errors:

$$\text{MMRE} = \frac{\sum_{a=1}^{N} \text{MRE}}{N}.$$

The present existing system and the proposed deep system focused on the mean magnitude of relative errors are compared with Fig. 5. For different cases, the values are reached from 10 to 50. Mean magnitude of relative errors is too high for the existing system ten instances, but deep proposed system's recommended mean magnitude of relative errors is 1.1471, which is significantly lower. Also, the mean magnitude of relative errors value decreases as the number of cases grows. It is stated from these conclusions that the deep proposed system suggested evinces better results (Lee et al. 2012). The excellent results of the suggested deep proposed system are shown in Fig. 5.

Mean absolute relative errors

All actual errors are considered mean absolute relative errors on average. Mean absolute relative errors are also a metric for quality software:

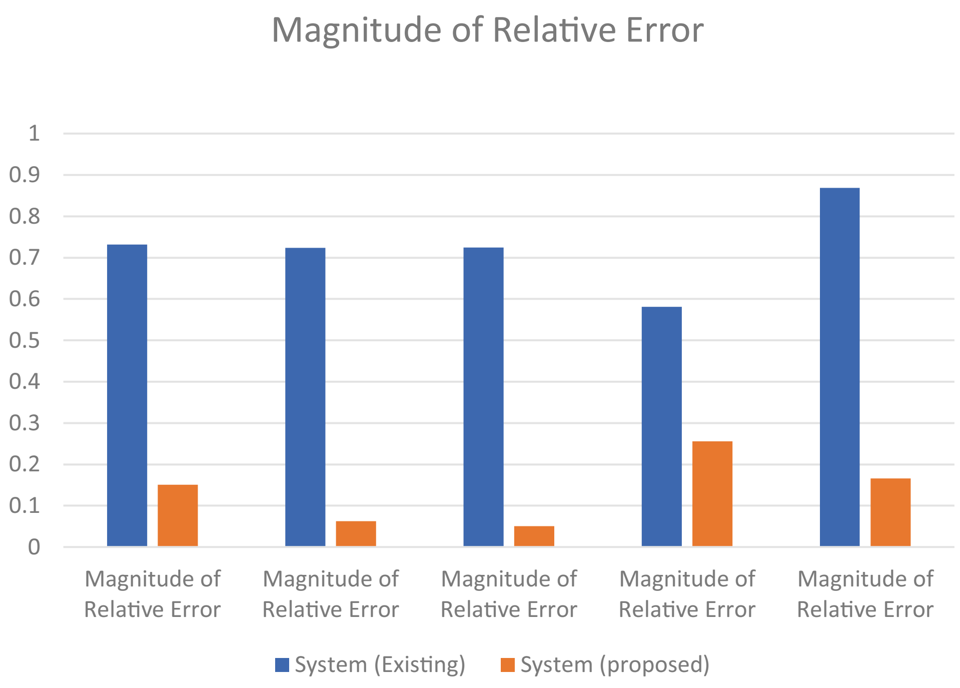

Fig. 4 Magnitude of relative errors of different occurrences

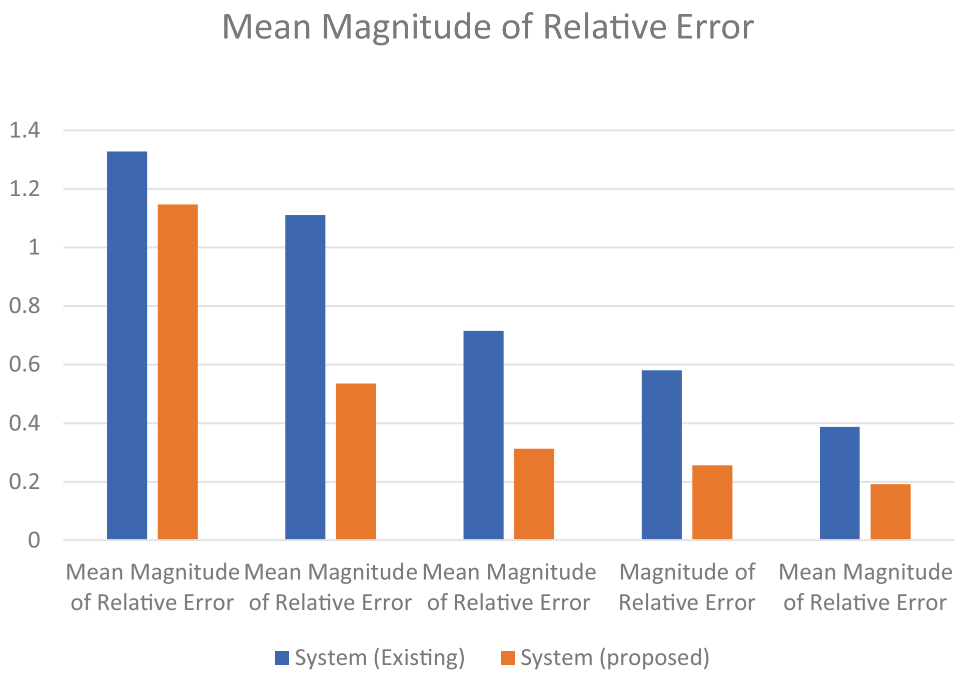

Fig. 5 Mean magnitude of relative errors of different occurrences

$$\text{MARE} := \frac{1}{N}\sum_{a=1}^{N} \text{estimated} - \text{actual}.$$

Figure 6 contrasts the performance of the current existing system and suggested deep proposed system techniques with the mean absolute relative errors parameter. In ten instances, mean absolute relative errors have been determined to have the highest variation. Mean absolute relative error for the present existing system is 2.1249 in ten occurrences, while the planned deep proposed system is 1.5514. There is a large discrepancy in mean absolute errors values in 10 and 50 cases. For 50 occurrences, for the existing system, the mean absolute relative error is 0.564; for the proposed deep proposed system, it is 0.277 and the existing system is lower (Malathi and Sridhar 2012). The mean absolute relative errors value of the

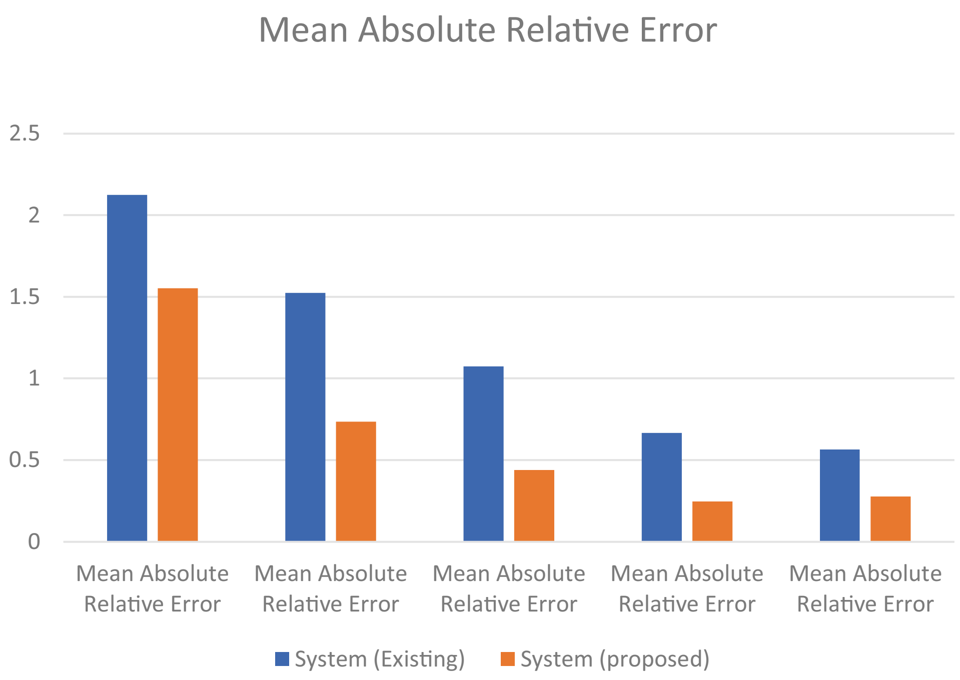

Fig. 6 Mean absolute relative errors of different occurrences

existing approach decreases sharply in Fig. 6. However, following the implementation of the proposed approach, the slope is shown to be less steep.

Mean percentage of prediction

The mean percentage of prediction percentage is a performance measure that is extremely important in software development. In contrast to traditional existing system, the mean percentage of predictions of the proposed system can be found in the following graph. The average estimated percentage value within the $n\%$ of the actual value is considered mean percentage of prediction:

$$\text{PRED} = \frac{A}{N}.$$

Figure 7 contrasts existing system performance and suggested mean percentage of prediction centric deep proposed system approaches. For ten cases, there is a sharp increase in the mean percentage of prediction value of the current existing system (Oliveira et al. 2010). But the proposed deep proposed system for the same ten cases has 1.0778 mean percentage of prediction, too low when compared to the existing system now available. The lower mean percentage of prediction value also reflects the higher efficiency of the deep proposed system suggested. The mean percentage of prediction value also decreases as the number of instances

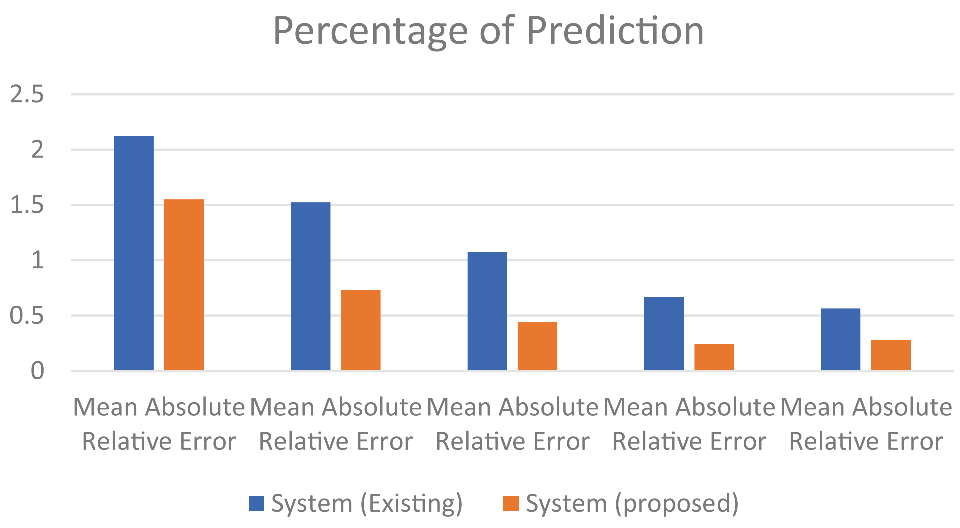

Fig. 7 Mean percentage of prediction of different occurrences

increases. The suggested deep proposed system displays high performance among all comparable cases.

Table 2 evaluates the deep-modified neutral network by contrasting it with the present neutral network centered on the above aspects. In ten cases, there is a disparity in the relative error levels between the system concerned and the system 0.0125; similarly, the difference is 0.5805, taking into consideration the size of the relative error values; their differences are 0.1808, when their mean magnitudes of the relative error are taken into account; their variance is 0.5735, if its mean relatively error values are considered. For 50 cases, there are 0.0583 differences between the current system and the proposed system in relation to the relative values of their errors; their differences in relation to their magnitude of relative errors also are 0.7026; their differences are 0.196 in relation to the mean relative values of their error. The proposed deep-modified neutral network has lower values for the existing 101 parameters than the current neutral network, and the differential values of the proposed deep-modified neutral network illustrate how different is from the existing neutral network. Similarly, for comparable parameters in remaining (20–40) cases, the deep-modified neutral network provided has lower values (Popovic et al. 2015). The evaluation findings indicate that the technology presented by the deep-modified neutral network gives superior performance (Fig. 8).

The objectives focus on the algorithm's complexity. A basic command algorithm needs less runtime. Table 3 shows the time of implementation of the traditional neutral network and the proposed Deep-modified neutral network. The suggested deep-modified neutral network requires less execution time in comparison with the processing time of the current neutral network. In ten situations, the proposed deep-modified neutral network, compared with 653 ms, is lower than the current execution time (Popovic et al. 2015). Table 3 demonstrates that the number of instances increases the execution time. Therefore, the proposed system provides improved performance. The variances in all performance measurements can be graphically represented for existing and prospective systems. The image presents a clear concept of modifications to the parameter and also confirms that the results of the suggested deep-modified neutral network were sought. The AE may not reflect its influence correctly in many circumstances because it is not related to the size of the value researched. Various

Table 2 Tables of different performance values

Occurrences	Dimensions	System (existing)	System (proposed)
10	Relative error	0.3255	0.304
	Magnitude of relative error	0.7317	0.1512
	Mean magnitude of relative error	1.3279	1.1471
	Mean absolute relative error	2.1249	1.5514
	Percentage of prediction	3.0398	1.0778
20	Relative error	0.3179	0.3133
	Magnitude of relative error	0.7237	0.0624
	Mean magnitude of relative error	1.1108	0.5353
	Mean absolute relative error	1.5231	0.7347
	Percentage of prediction	1.0173	1.0422
30	Relative error	0.2117	0.3033
	Magnitude of relative error	0.7244	0.051
	Mean magnitude of relative error	0.7146	0.3127
	Mean absolute relative error	1.0738	0.439
	Percentage of prediction	1.251	0.5872
40	Relative error	0.2898	0.2355
	Magnitude of relative error	0.5809	0.2561
	Mean magnitude of relative error	0.4827	0.1491
	Mean absolute relative error	0.6655	0.2465
	Percentage of prediction	0.8684	0.3418
50	Relative error	0.3737	0.3154
	Magnitude of relative error	0.8684	0.1658
	Mean magnitude of relative error	0.3873	0.1913
	Mean absolute relative error	0.564	0.277
	Percentage of prediction	0.7387	0.3847

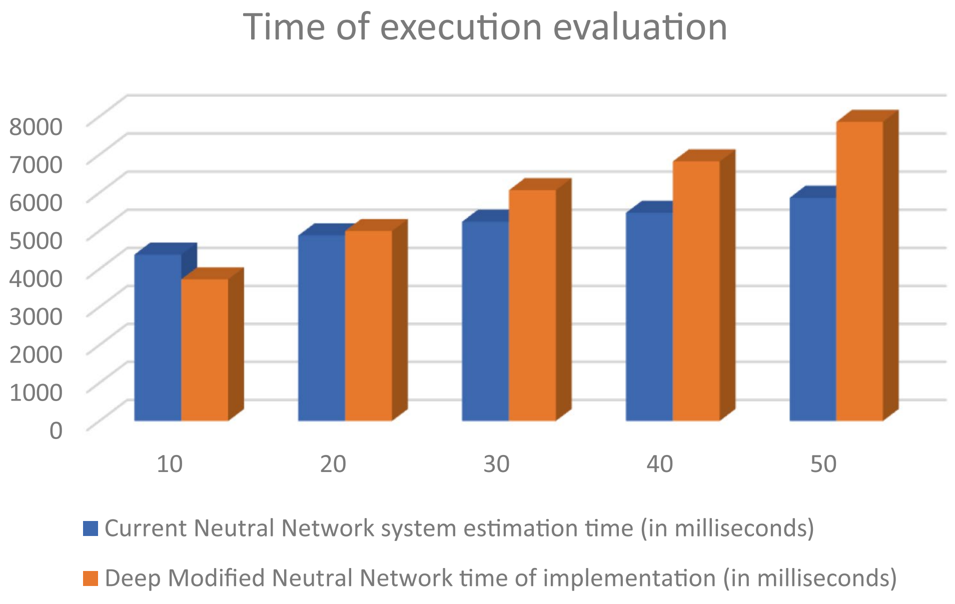

Fig. 8 Time of execution relation

Table 3 Time of execution evaluation

Dimensions	Current neutral network system estimation time (ms)	Deep-modified neutral network time of implementation (ms)
10	4380	3727
20	4881	4996
30	5242	6070
40	5476	6822
50	5871	7861

analysis with different models of neural networks is shown in Table 4.

Conclusion

This article emphasizes the customer's impact on accuracy by software professionals. Often because of lack of well-defined expectations, rising new requirements, a lack of competent customers and adequate decision-making, the most prevalent contributions to customer misconceived decisions alter. SDEE is effectively achieved with the proposed deep-modified neutral network algorithm. The EE targeting SD is no longer a monotonous process following installation of the proposed approach. The approach to deep learning via the modified neutral network analogized the problem of the limitless potential of well. Furthermore, it is easy to utilize in a very short time for any project due to its newness, simply textual data and readily accessible measurements are necessary. Further study needs to notify methods for text processing to improve the outcomes. The study is also intended since the inclusion of current project knowledge in the prediction is a key aspect in this present approach.

Future scope

In future research, additional aspects besides the size of the software and team experience will also be investigated, such as team sizes, software complexity measurements and software development methods, which might have an influence on estimating the efforts of software. We are encouraged to continue this study by leveraging actual data from software projects on different dimensions to implement the suggested approach. Further study on this topic is to overcome the problems of instability, a more general prediction model that is not heavily impacted by the size and kind of data collection and better to improve the optimization method itself. In addition, a modified hybrid strategy that includes the better qualities of several prediction plans would be useful. We will employ the sub-set selection technology in future to concentrate on significant features which impact effort and expand the work to work for any dataset. Other techniques, such as Bagging and Ensemble, can be used to estimate the work of projects in future. Results against each would be examined to identify improved performance with a

Table 4 Comparative analyses among different models of neural network to evaluate software effort estimation

Technique used	Description	Advantage
Radial basis function neural network	RBFN provides strategies for estimating effort. To discover the structure of clothed neurons, he employs *k*-means cluster algorithms. RBFN consists of three layers: cost drivers, cache neurons and output layer. RBFN consists of three layers. COCOMO and TUKUTUKU are the main datasets utilized	This article analyzes the radial base width using various formulae and RBFN gives a greater precision in estimating effort while testing different projects
Artificial neural network	The ANN design comprises input neurons linked to a weighted value, which compute and create the total of the inputs. The output is compared to the threshold value for a positive or negative outcome	In contrast to other model models (e.g., Halstead, Walston–Felix or Bailey–Basili), ANN provides the least MMRE and it is found to be more accurate than others
Functional link artificial neural network	FLAN is also a neural network approach that employs a single-layer feeding system. This method includes input neurons (cost driver) and generates the resulting output (work) by the expansion of cost drivers. The development effort is determined by the linear weighted sum of the input neurons outputs	FLAN concentrates on functioning and minimizes complexity as well. It also employs the learning principle to improve its approximation faster than other neural grid technologies
Artificial neural network and support vector machine	In the calculation of effort estimation, the ANN (artificial network) and SVM mechanism (support vector machine) mechanism may be coupled. ANN employs the idea of the learning and training error reverse propagation mechanism Weights will be set during the forward phase, and weights will be adjusted based on the estimated inaccuracy during the reverse phase. SVM is a supervised learning technique based on regression analysis and classification. SVM is used mostly for enhancing precision	It concentrates on learning ways to calculate the mistake. It results better since weights may be changed by the training data
Use of neural network function points, case-based rationalization and regression analyses	For the calculation of system size (case-based justification, regression analysis and neural network). The criteria for assessment are based on MRE and MMRE	Case-based reasoning that employs the expert assessment mechanism and neural network results better than regression analyses
Model of connection utilizing neural back propagation network techniques	Comparative research has been carried out on genuine projects and commercial initiatives. The term connecting model is employed because inputs are distorted and lead to a complicated relationship between inputs and outputs	The effort on huge commercial data sets was determined to be above 75% within 25% of the real projects
Artificial neural network and regression model	Neural networks may be used to assess numerous components of the development environment instead of utilizing the function points separately to determine the effort of development. This approach was therefore called a network-based estimate model for software	Performs better than the traditional technique with simply COCOMO or function point (FP)

view to precise estimations and prevision. The effort predicted by the expert evaluation technique must be considered for the future work to optimize the final estimate. Optimization's initial value is the expected effort by an expert opinion.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

Altuntas TU, Alptekin SE (2017) Software development effort estimation by using neural networks—a case study. Int J Comput 2:115–122

Al Yahya M, Ahmad RB, Lee S (2010) Impact of CMMI based software process maturity on COCOMO II's effort estimation. Int Arab J Inf Technol 7(2):129–137

Araujo RA, Oliveira ALI, Meira S (2017) A class of hybrid multilayer perceptrons for software development effort estimation problems. Expert Syst Appl 90:1–12

Aroba J, Cuadrado-Gallego JJ, Sicilia MÁ, Ramos I, García-Barriocanal E (2008) Segmented software cost estimation models based on fuzzy clustering. J Syst Softw 81(11):1944–1950

Azath H, Wahidabanu RSD (2012) Efficient effort estimation system viz. function points and quality assurance coverage. IET Softw 6(4):335–341

Azath H, Mohanapriya M, Rajalakshmi S (2020) Software effort estimation using modified fuzzy *C* means clustering and hybrid ABC–MCS optimization in neural network. J Intell Syst 29(1):251–263

Azzeh M (2017) Dataset quality assessment: an extension for analogy based effort estimation. Int J Comput Sci Eng Survey. https://doi.org/10.5121/ijcses.2013.4101

Azzeh M, Nassif AB (2015) Analogy-based effort estimation: a new method to discover set of analogies from dataset characteristics. IET Softw 9:39–50

Azzeh M, Neagu D, Cowling PI (2011) Analogy-based software effort estimation using fuzzy numbers. Syst Softw 84(2):270–284

Bardsiri VK, Jawawi DNA, Hashim SZM, Khatibi E (2012) Increasing the accuracy of software development effort estimation using projects clustering. IET Softw 6:461–473

Cuadrado-Gallego JJ, Sicilia MÁ, Garre M, Rodríguez D (2006) An empirical study of process-related attributes in segmented software cost-estimation relationships. J Syst Softw 79(3):353–361

Dan Z (2013) Improving the accuracy in software effort estimation using artificial neural network model based on particle swarm optimization. In: Proceedings of IEEE international conference on service operations and logistics, and informatics (SOLI), 28–30 July 2013, Dongguan, China

Dejaeger K, Verbeke W, Martens D, Baesens B (2012) Data mining techniques for software effort estimation: a comparative study. IEEE Trans Softw Eng 38:375–397

El Bajta M (2015) Analogy-based software development effort estimation in global software development. In: Proceedings of IEEE 10th international conference on global software engineering workshops, 13–13 July 2015, Ciudad Real, Spain

Hamdan K, El Khatib H, Moses J, Smith P (2006) Software cost ontology system for assisting estimation of software project effort for use with case-based reasoning. In: 2006 innovations in information technology, 19–21 Nov 2006, Dubai, UAE

Kad S, Chopra V (2012) Software development effort estimation using soft computing. Int J Mach Learn Comput 2(5):548

Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. J Appl Math Comput 214:108–132

Kocaguneli E, Menzies T, Keung J, Cok D, Madachy R (2013) Active learning and effort estimation: finding the essential content of software effort estimation data. IEEE Trans Softw Eng 39:1040–1053

Kodmelwar MK (2019) Software estimation using deep learning. Int J Innov Technol Explor Eng 8(6S):565–568

Kodmelwar M, Joshi S, Khanna V (2018) A deep learning modified neural network used for efficient effort estimation. J Comput Theor Nanosci 15:3492–3500. https://doi.org/10.1166/jctn.2018.7651

Lee W-T, Lee J, Hsu K-H, Kuo JY (2012) Applying software effort estimation model based on work breakdown structure. In: Proceedings of 6th international conference on genetic and evolutionary computing, 25–28 Aug 2012, Kitakyushu, Japan

Malathi S, Sridhar S (2012) Estimation of effort in software cost analysis for heterogenous dataset using fuzzy analogy. Int J Comput Sci Inf Secur 10.

Merlo-Schett N, Glinz M, Mukhija A (2002) Seminar on software cost estimation WS 2002/2003. Department of Computer Science, p 3–19

Morrow P, George Wilkie F, McChesney IR (2014) Function point analysis using NESMA: simplifying the sizing without simplifying the size. Softw Q J 22(4):611–660

Murillo-Morera J, Quesada-López C, Castro-Herrera C, Jenkins M (2017) A genetic algorithm based framework for software effort prediction. J Softw Eng Res Dev 5(1):4–10

Oliveira ALI, Braga PL, Lima RMF, Cornelio ML (2010) GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. Inf Softw Technol 52:1155–1166

Popovic J, Bojic D, Korolija N (2015) Analysis of task effort estimation accuracy based on use case point size. IET Softw 9(6):166–173

Pospieszny P, Czarnacka-Chrobot B, Kobyliski A (2017) An effective approach for software project effort and duration estimation with machine learning algorithms. J Syst Softw 137:184–196

Qi K, Boehm BW (2017) A light-weight incremental effort estimation model for use case driven projects. In: Software technology conference (STC), 2017 IEEE 28th annual, IEEE, 2017, p 1–8

Satapathy SM, Acharya BP, Rath SK (2016) Early-stage software effort estimation using random forest technique based on use case points. IET Softw 10(1):10–17

Seo YS, Bae DH, Jeffery R (2013) AREION: software effort estimation based on multiple regressions with adaptive recursive data partitioning. Inf Softw Technol 55(10):1710–1725

Silhavy R, Silhavy P, Prokopova Z (2017) Evaluating subset selection methods for use case points estimation. Inf Softw Technol 97:1–9

Urbanek T, Prokopová Z, Silhavy R, Sehnálek S (2014) Using analytical programming and UCP method for effort estimation. In: Silhavy R, Senkerik R, Oplatkova Z, Silhavy P, Prokopova Z (eds) Modern trends and techniques in computer science. Springer, Cham, pp 571–581

Yadav CS, Singh R (2014) Prediction model for object oriented software development effort estimation using one hidden layer feed forward neural network with genetic algorithm. Adv Softw Eng 2014 (**Article ID 284531**)