IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 7, 2020, accepted May 11, 2020, date of publication May 29, 2020, date of current version August 6, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.2998581

Efficiency Improvement of Function Point-Based Software Size Estimation With Deep Learning Model

KUI ZHANG[1], XU WANG[2], (Member, IEEE), JIAN REN[1], AND CHAO LIU[1], (Member, IEEE)
[1]State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing 100191, China
[2]China Ship Research and Development Academy, Beijing, China

Corresponding author: Jian Ren (renjian@buaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602021, in part by the State Key Laboratory of Software Development Environment under Grant SKLSDE-2019ZX-10, and in part by the Beijing Municipal Science and Technology Commission under Grant 20191111845496.

ABSTRACT Software cost estimation is crucial to software management, which has received considerable attention from both industry and academia. Software size is an important metric that forms the cornerstone of software cost estimation. The function point has been proven to be a useful software size unit for size estimation and has been successfully implemented in many countries. However, in current practice, the rule of function point size method is complicated and performed manually. Consequently, it is costly in both time and resources spent to apply these methods, especially in the scenario of large-scale software development in the industry. In this paper, a deep learning-based named entity recognition (NER) model was designed in place of manual function point recognition. In particular, a BiLSTM-CRF model was trained on previously labeled requirements in the industry to classify the function point type of new requirements in the same domain. The proposed method was verified on 29 real projects provided by our industry partner. A comparative experiment was designed for the quantitative evaluation of efficiency improvement of the proposed NER model aided function point estimation. The result suggests that, for the NER model, the precision and F1 of the BiLSTM-CRF-based function point analysis on test samples achieved 94.5% and 80.3%, respectively. Moreover, the improvement in the efficiency of the software size estimation process achieved an average of 38.6%, which is a significant enhancement for the function point-based software size estimation.

INDEX TERMS Software size estimation, BiLSTM-CRF, function point, NER.

I. INTRODUCTION

Software cost estimation (SCE) has been recognized as one of the most important tasks in software project management. During software project management, the manager needs to balance development efficiency, software quality, and the cost of software development to achieve successful software development. The underestimation of software development costs causes overbudget and schedule delays. In contrast, the overestimation causes a waste of resources and further loss of investment opportunities for the organization. Therefore, the manager must estimate the budget, schedule, and resource expenses accurately during software development under a given software quality and schedule to obtain an executable plan.

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

Many models have been proposed to estimate software cost, where an accurate estimation of SCE needs to take many factors into consideration in both software and organization related aspects. For example, the size of software [1], the complexity of software [2], the reliability requirement of software [3], the maturity of software development organization [4], and the effect of misleading information provided by the software developer [5]. Expert judgment is one of the most traditional techniques since the very beginning of the estimation problem raised [6]. However, it relies too much on expert experience, which can be subjective and lacks standardization. Therefore, model-based estimation attempts to bridge this gap by constructing an estimation model from the historical metric data of software. The model-based estimation can be further classified into parametric and nonparametric models. COCOMO is a famous parametric model that was first proposed in 1981 by Boehm *et al.* [7]. After that, many variant

extensions have been proposed, such as COCOMOII [8] and COSYSMO [9]. Similarly, SLIM, PUTNAM [10], PRICE-S, SEER, and ESTIMACS can also be classified into this kind of model. To obtain a more precise and portable estimation model, a nonparametric model such as machine learning techniques is applied to the estimation model by training on historical project data, such as a neural network (NN) and its extension [11]–[14]. The accuracy of the model, as mentioned above, has been proven to reach an acceptable level with the careful instruction of the model construction guideline, whereas little focus has been on the efficiency of the estimation process. A significant barrier to obtaining an efficiency estimation is the accurate metric for the software within an acceptable limit time.

Software size is an essential metric of software, which is also the primary input of the estimation model and lies on the critical path of the SCE process. The time consumed on this procedure has a critical impact on the total estimation time consumed. Many software size units have been proposed to estimate the software scale under different scenarios, such as the source lines of code (SLOC) and function point (FP) methods for the size estimation of information systems [15], story points developed for agile software development [16], and use case points (UCPs) for model-based development [17], [18]. In particular, function point metrics are one of the most accurate and useful metrics and have been proven successful in many areas. Compared with the SLOC, the FPA method can be applied in the different phases of software development life cycle(SDLC). It can be helpful to reach a common view on the software effort spent between the supplier and vendor and thus improve the management ability of SDLC. As illustrated in Fig. 1, the primary process of the FPA

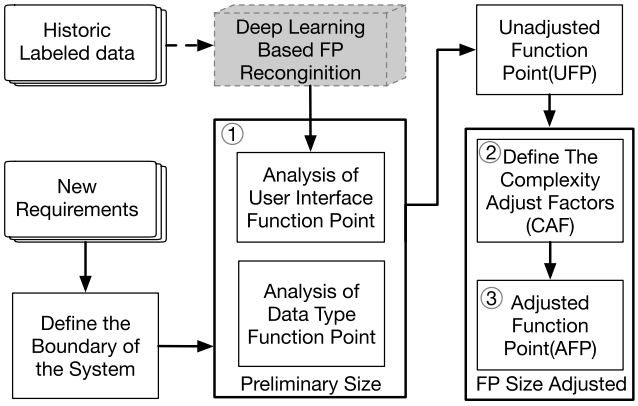

FIGURE 1. Deep learning-based FPA process.

metric can be divided into three steps: the first step is the user interface function point and its complexity analysis to obtain the unadjusted function point(UFP) size; In the second step, the consultant needs to evaluate the complexity adjustment factors(CAF)(e.g., distributed communication, performance requirement), which can execute parallel with the first step when working through the requirement document; The last step is the combination of the results of the previous two steps to obtain the adjusted function point (AFP), which is the final software size.

The methods were further developed into many function point-based variation size methods and standardized into many ISO standards, such as IFPUG [19] MKII [20], NESMA [21], and COSMIC [22].

However, the methods mentioned above are primarily manually executed, which is too slow and costly to apply to large-scale software projects (e.g., 10,000 function points in size). Consequently, there are two common difficult challenges in current industry practice. The first is the steep learning curve; the rule for extracting the FP from the requirement is complicated, and the learner needs to thoroughly understand the rules and their variants according to the scenarios defined in the guideline book in which nearly a thousand rules are defined. Furthermore, for certification, the learner needs to practice over 50,000 function points to completely master the rule before applying them into practice in the industry. The second is the effort used to extract the FP unit (e.g., EI, EO, EQ, ILF, EIF) from requirement or design. Take IFPUG FP analysis as an example. A certified function point consultant can proceed at speeds between 400 and 600 function points per day, according to the study of Jones [23]. In large-scale software, the number of pages in requirement documents can quickly reach over a thousand, filled with terminology and logic. It is difficult for a consultant to understand all the terminology and give a correct classification according to the rules in a short time (e.g., 10,000 function points would require 20 days and cost $30,000), which is unacceptable in practice.

Considering the FPA method is costly both in terms of time and money, the main motivation of this research is to reduce the time consumed on this procedure to bridge the gap between the FPA method and industry practice. This research study presents a deep learning-based FPA model for mitigating the challenge mentioned above, as shown in Fig. 1. A deep learning model for function point recognition replaced the traditional analysis of the user interface function point. Specifically, a BiLSTM (bidirection long short-term memory)-based deep neural network (DNN) combined with conditional random field (CRF) layer called BiLSTM-CRF model was applied for the NER model training. The model was then used for FP recognition; in that case, the consultant only needs to check the result given by the model rather than thoroughly working through the requirement. An industry case was conducted to validate the efficiency improvement of the proposed method. An experiment with three senior certified function point consultants as participants was designed for the quantitative evaluation of efficiency improvement. The result shows that the accuracy of the model can be comparable with the certified function point consultant, and efficiencies can be significantly improved with the help of a deep learning-based FPA model.

The contribution of this research study is twofold: the first contribution is the presentation of a deep learning-based NER

model for FPA. The second contribution is the improvement evaluation of efficiency by comparing the time consumption of the FPA process with and without the help of the previous NER model. To the best of our knowledge, this is the first study on the improvement of the FPA method by constructing a learning model to consolidate the experience of a certified function point consultant. Moreover, we make a quantitative evaluation of efficiency improvement in an industry scenario.

The remainder of the paper is organized as follows. Section II describes the related work, while Section III will define the problem. The research methodology used in the research study is explained in Section IV. Section V explains the experiments and our interpretation, and a discussion on the research analysis is also be discussed in section VI. Finally, Section VII summarizes our conclusions and highlights future directions.

II. RELATED WORK

A. SOFTWARE SIZE METRIC

Generally, software size is the number of units that can be used to decompose the software. Many works look into the artifact of the software to determine the indicator of the software size. The first widely used metric is an estimation by SLOC (source lines of code) [15], and then Halstead was applied to the SLOC size method to improve the precision. SLOC focuses on the metric of the code, but it cannot be simply applied in the early phase of software development. Moreover, the number of lines of code depends mainly on the programming language, which can vary if the same software is developed in different languages [24].

To address the software size problem in different phases and with different developed languages, many other software size methods have also been proposed. Function point is a size unit that can be applied in the early phases of the software life cycle. Story point is a size unit for the task within an agile project that can represent the quantity of work, and the velocity differs from team to team [25]. The case point was first introduced by Karner [26] to estimate the software size in the early phases and has been applied widely in model-based development, such as UML-based software development [27], [28]. CC (full object-oriented metric) was proposed to estimate object-oriented system software size [29]. OO-HFP (object-oriented hypermedia function points) [30] is an extension of OO-H [31] for model-driven web application size estimation that complies with the FPA method. The TEF (testing-effort function) is a software development & test effort curve with excellent predictive capability [3].

B. FUNCTION POINT

Function point analysis (FPA) was first formally proposed by Albrecht [15], in which the FP was defined in five elements, ILF, EIF, EI, EQ, EO, and it proved to be a better size metric compared with SLOC.

1) FUNCTION RECOGNITION

According to the Albrecht study [15], the FPA process decomposes the system into these five functional components.

1) External Input Type (EI): Count each unique user data or user control input type that enters the external boundary of the application being measured and add or change data in a logical internal file type.
2) External Output Type (EO): Count each unique user data or control output type that leaves the external boundary of the application being measured.
3) Internal Logical File Type (ILF): Count each major logical group of user data or control information in the application as a logical internal file type.
4) External Interface File Type: (EIF): Files passed or shared between applications should be counted as external interface file types within each application.
5) External InQuiry Type: (EQ): Count each unique input/output combination, where an input causes and generates an immediate output, as an external inquiry type.

2) FUNCTION POINT COUNTING

After the preliminary function recognition, the function needs to be analyzed for the further function point size under the complexity consideration, as shown in Table 1.

TABLE 1. Function complexity (FC) count.

Description	Complexity		
	simple	Average	Complex
External Input Type(EI)	3	4	6
External Output Type (EO)	4	5	7
Internal Logical File Type: (ILF)	7	10	15
External Interface File Type: (EIF)	5	7	10
External IQuiry Type: (EQ)	3	4	6

The complexity is defined according to the number of data elements that need to deal with and also operations on the data, the detail definition can be seen in [15], [32], [33]. To obtain the unadjusted function point (UFP) number of each user interface function, the consultant needs to understand the software requirements and also evaluate the complexity of the function according to the rules defined by given FPA method.

3) ADJUST FUNCTION POINT SIZE

The UFP needs to be adjusted under specific software to obtain the final software size. In the IFPUG, the software characteristics that may have an effect on the AFP are defined in Table 2.

The weight of each characteristic can be the number range [0, 5] according to its influence on the software effort, which relies on the judgment of the consultant. Consequently, the adjusted function point size (AFP) can be reached by multiplying the UFP by the adjustment factors as defined in the following equation:

$$AFP = UFP * CAF \tag{1}$$

TABLE 2. Characteristic definition.

ID	characteristics	ID	characteristics
C1	Data Communications	C8	Online Update
C2	Distributed Functions	C9	Complex Processing
C3	Performance	C10	Reusability
C4	Heavily Used Configuration	C11	Installation Ease
C5	Transaction Rate	C12	Operational Ease
C6	Online Data Entry	C13	Multiple Sites
C7	End User Efficiency	C14	Facilitate Change

where CAF is the complexity adjustment factor defined as:

$$CAF = 0.65 + 0.01 * \sum F_i, \quad (1 \leq i \leq 14) \qquad (2)$$

F_i is the weight of the characteristics.

C. IMPROVEMENT FOR THE FUNCTION POINT METHOD

The improvement for functional size measurement (FSM) methods can be classified into two categories: the first is the evolution of the software size metric methods into a new method. In contrast, the second is the efficiency improvement of the existing software size method with other techniques.

1) FSM METHOD EVOLUTION

For the first category, after the FP proposed by Albrecht and Gaffney [15] in 1979, several variants have been proposed to make the FPA estimation more accurate and suitable for different scenarios. The FPA metric was first standardized by the International Organization for Standardization (ISO) as IFPUG [32]. NESMA FP fixes the disparity that the IFPUG FP method cannot be used for early estimation of the information system and extends the application scenario to software maintenance projects. NESMA FPA varies little from IFPUG, which has been established as a measurement unit and an ISO standard [33]. The MK II function point method was published by Charles Symons [20]. The essential improvement of this method is that its unit improved IFPUG by considering the internal complexities of data handling, which is the key feature of the business system. The Common Software Measurement International Consortium (COSMIC) was established in 1998. The COSMIC function points, which represent the 2nd generation FSM method, were proposed to find a measurement unit capable of being successfully applied to the highest possible number of software types. COSMIC function points solve the problem that the NESMA FP cannot be directly used in the embedded system and are considered to be the second generation of the FP method [34]. After that, a COSMIC FFP (full function point) method was proposed for the improvement of COSMIC FP by the application scenario [35].

2) EFFICIENCY IMPROVEMENT

The mentioned evolution of FSM methods has made considerable contributions to the widespread use of software size methods. The cost and time consumption of these methods are still essential obstacles for their wide acceptance by industry. To address this problem, a simplified FP named the simple function point (SiFP), which requires much less time and effort than IFPUG FPA and is fully compatible with IFPUG, was proposed in 2011 [36]. Capers Jones in 2013 [37] proposed a method based on pattern matching of high-speed function point counting. It can size different applications in less than two minutes and can predict application growth during development and for five years after release. To make the evaluation process more rapid, a CRF named entity recognition (NER) model named ESSE was applied to extract features from the requirement and construct the regression model to predict the software size [38] An ontology model base COSMIC FP was also proposed to eliminate effort and subjectivity coming from manual measurement [39]. With the development of DNN, a prediction model based on a combination of deep learning architectures was also applied for story point-based estimation [40].

D. NAMED ENTITY RECOGNITION

Name entity recognition (NER) is a subtask of natural language processing (NLP). With the development of NLP, the NER model was also developed from the rule-based approach or statistical machine learning approach to today'DNN models. Most research regards the NER task as a sequence labeling task. The model for modeling this task includes maximum entropy (ME) [41], support vector machine (SVM) [42], hidden Markov model (HMM) [43] and conditional random field (CRF) algorithms [44].

With the rapid development of deep learning neural networks in NLP tasks outperform previous statistical algorithms. The recurrent neural network (RNN) is an outstanding NN model for sequence labeling that learns long-distance dependencies better than CRF, which utilizes features found in a specific context window [45]. As an extension of RNN, long short-term memory (LSTM) and its updated bidirectional recurrent neural network (BRNN) [46], [47] have proven to be efficient in modeling sequential text. BRNN can consider an effectively infinite amount of context on both sides of a word and eliminate the problem of limited context that applies to any feedforward model. Convolutional neural networks (CNN) have also been investigated for extracting character-level features for use in NER and POS tagging [48]. However, the LSTM-CRF obtains state-of-the-art performance in the NER task in four languages (English, Dutch, German, and Spanish) [49]

Chinese named entity recognition (CNER) is more complicated in Chinese (e.g., the lack of word boundaries, the complex composition forms, the uncertain length). There are mainly three kinds of named entities (NE), namely, locations, persons, and organizations in CNER research. The abovementioned models also achieve similarly good results in Chinese [50], [51].

III. PROBLEM DEFINITION

Generally, the procedure of SCE can be divided into three parts: estimating software size, establishing the software effort estimation model and finally constructing a software

cost model. FPAs have been established as ISO standards and are widely used in industries that integrate the three parts mentioned above into a whole. In the estimation model of software development in the early phase, five types of FP (EI, EO, EQ, ILF, EIF) are used. Therefore, software cost estimation task is transferred into an accurate estimation of the size of the function point. However, in most research studies, software size data are obtained from an existing dataset, such as the COCOMO dataset, NASA dataset, or ISBSG dataset. To the best of our knowledge, little work has discussed obtaining software size accurately and efficiently. Moreover, in practice, the software size needs to be estimated toward a given artifact (e.g., requirement or design document) according to a given standard FPA procedure (e.g., early estimation in NESMA, COSMIC FFP). With the development of the software design (detail design), more information is needed for the more accurate estimation (e.g., data element type, record element type), and the estimation accuracy needs to reach 85% of the actual result to get an valid cost estimation. A static provide by Jones and Bonsignour [52] can be seen in Table.3

TABLE 3. The FP and requirement(REQ) quality.

FP Size	FP Per Page	Page In Total	Integrity of REQ
10	1.35	14	97%
100	1.15	115	95%
1000	0.75	750	80%
10000	0.60	6000	60%
100000	0.15	15000	20%
Average	0.5	5005	39%

As shown, with the increase in the size of the software, the length of requirement increases while the integrity decreases, which means that more effort will be used to recognize the function point. Moreover, during the software life cycle, the estimation activity should be repeated two to three times. If two many person-days of measurement effort is required, even if it is only a small fraction of the total cost, many managers will not accept this apparently ''nonproductive'' cost in the project budget, thus, efficiency is the key to practice. Therefore, our research questions can be stated as follows while the fundamental research hypotheses is that all the requirement document will be translated into natural language for the further analysis:

A. RESEARCH QUESTIONS

RQ1 (Sanity Check): Is the proposed method suitable for estimating the function point? This sanity check requires us to compare efficiency improvement with the manual FPA as a baseline benchmark. In current practice, a manual analysis procedure implies function point recognition, removing the redundancy, and then evaluating the factors that have an effect on the software effort. The output of the NER-based learning model is a probabilistic classification that cannot be directly used as the final result. It is evident that the model cannot simply substitute the manual procedure in practice, which needs to be inspected manually. Therefore, the NER model needs to be combined into the traditional procedure in which the function point recognition is replaced with NER recognition and manual inspection. Therefore, in this scenario, whether the efficiency can be improved by this new revised method compared with pure manually FPA is the question that needs to be answered in this research study.

RQ2 (Accuracy of the Model): How does the accuracy of the proposed model compared with manual FPA? In industry, the ISO standard provides a method for identifying the function point type in the given context. Nevertheless, the requirement is specified in natural language, which can be ambiguous between different domains and depend on the understanding of the consultant. However, many function point analysis works have been performed, and the rule for identifying these function points is the same in the determined FPA model, so whether the rules can be extracted from these labeled documents and to what extent the recognition accuracy can reach are also questions that need to be answered.

RQ3 (Efficiency Improvement with the Model): How much efficiency improvement can be achieved by the proposed model? As described in RQ1, compared with the traditional method, the output of the model needs to be inspected by the consultant. Therefore, the improvement of the efficiency needs to be evaluated against the same input under different scenarios (with and without the help of the NER model). Moreover, the efficiency needs to be evaluated against two indexes, which are the correction and time consumed. The number of correctly recognized functional points and the time consumption of the experiment need to be recorded for further quantitative analysis. Therefore, the quantitative evaluation of the improvement in efficiency is the third research question that needs to be answered.

IV. METHODOLOGY

The objective of this research study is to answer the research questions raised in the previous section. The research involves constructing a deep learning-based NER model to replace the manual function point analysis (FPA) and evaluating the improvement on the efficiency with the help of the NER model. A BiLSTM-CRF model was applied for the NER task in the FPA process, and a case from our industry partner was designed to validate the efficiency improvement of the proposed model. The process for the deep learning-based FPA can be seen in Fig. 2.

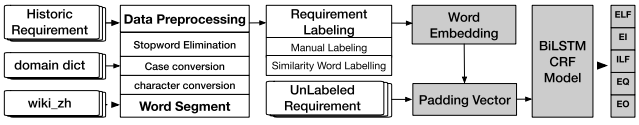

FIGURE 2. Deep learning-based NER model.

As shown, from the left side, the already analyzed requirement documents, domain dictionary, and wiki in Chinese were used as the input of the data preprocessing process. Specifically, the historical requirement documents were used as the raw input for data preprocessing and then segmented

by word segment. The wiki content was added for the training of the word2vec model, while the domain dictionary was used to improve the accuracy of the word segment. After that, the requirement needed to be labeled for further training. In this study, in addition to manual labeling, an edit distance-based similarity word labeling method was introduced to label the existing requirement automatically. Sequentially, the content of requirement and its label was converted into word vector and one-hot vector, which was used for the BiLSTM-CRF model training. As a result of BiLSTM-CRF model training, the unlabeled requirement was automatically classified into five given function types for further estimating software size. The detailed description of each part of the methodology is illustrated in the following paragraph.

A. DATA PREPROCESSING

In most supervised NLP tasks, the first and most important work is data cleaning and labeling. In this research study, data preprocessing is also a preset task before model training. A stop-word list was used for the preliminary data cleaning, and the high-frequency words that appeared in the requirement were extracted into a domain dictionary, which was used to improve the accuracy of the word segment in the next step. After transforming traditional Chinese to simplified Chinese, removing non-UTF8 chars and unifying different styles of punctuation, open-source Chinese word segment software was then applied to separate the requirements into words.

After the word segment, the words need to be combined for further labeling. However, in practice, the function point item, which represents the feature of the function point, is extracted from the original requirement text, and the content of a similar function point item is slightly different from simple word combination. In this study, an edit distance algorithm [53] was applied to bridge this gap by labeling the similar function point item automatically. The word edit distance algorithm recognizes two similar words by defining a value of edit distance, which counts the number of operations by converting one word to the other. Two main parameters of the edit distance algorithm are word length and edit distance, which is defined in the specific case.

The BIO label was chosen for further single word labeling, and the sequence label is illustrated in Fig.3:

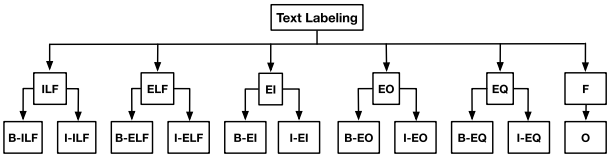

FIGURE 3. BIO sequence label.

As illustrated, the function point item can be classified into six types, which are EI, EO, EQ, ILF, EIF, and F, in which F means that it is not the function point item. For the function point item labeling, the function point type can be further decomposed into B (Begin), I (Interval), and O (Out of scope) labels. A simple example of BIO labeling according to the NESMA FPA rules can be seen in Table 4:

TABLE 4. An example with BIO sequence label.

Words:	can	refer	to	information	manage	system
Labels:	O	O	O	B-EIF	I-EIF	I-EIF

As shown, the function point item in the requirement is ⟨*information manage system*⟩, which can be identified as an EIF so that the word can be labeled into B-EIF and I-EIF sequentially, and the unrelated word is labeled as O.

B. WORD EMBEDDING

The function point item needs to be converted into a low-dimensional, continuous, and real-valued vector because the input of the neural network is vectors. However, traditional one-hot coding will give a vector with a high dimension and discard the relationship between similar words. In this study, a skip-gram-based word2vec model was used for word embedding training. A GenSim package that contains a Python version implementation of word2vec was applied.

The skip-gram structure uses the center word as input while adjusting the neural network weight by backpropagation of the word in a given window size. This structure is more suitable for the text corpus that consists of more domain words. The training process is described in Section V.

C. BiLSTM-CRF BASED FP SIZING

Different from traditional NLP techniques, DNN becomes a better solution for NER [49], which relieves the heavy labor work of feature engineering. In this research study, a BiLSTM-CRF model was applied for function point recognition, which combined RNN-based BiLSTM (bidirectional long short-term memory) neural network and CRF (conditional random field) model to process different parts of the prediction model. The structure and a simple example of the model can be seen in Fig. 4.

As shown, in the bottom part, the input of the model is the word vectors obtained from the word embedding. The BiLSTM is trained and used to predict the label type of every single word. The output of the BiLSTM is the score of the input word regarding each label type (e.g., B-EI), which is the input of the CRF model. The CRF model is trained on the input of the BiLSTM for further named entity prediction.

Here, we briefly introduce the BiLSTM and CRF models.

1) BiLSTM

LSTM is a variant of RNN (recurrent neural network) architecture designed to fix the vanishing gradient problem with long short-term memory units as hidden units [54]. BiLSTM is a deep neural sequence model that considers an effectively infinite amount of context on both sides of a word. A forward long short-term memory (LSTM) layer and a backward LSTM layer are incorporated to eliminate the problem of

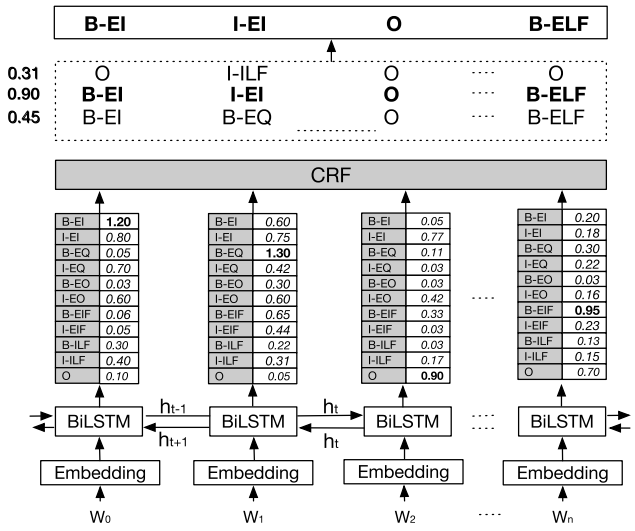

FIGURE 4. BiLSTM-CRF-based FP recognition.

limited context that applies to any feedforward model [55]. As shown in Fig. 4, one LSTM processes the sequence from left to right, and the other processes the sequence from right to left. At each time step t, a hidden forward layer with hidden unit function $\overrightarrow{h}$ is computed based on the previous hidden state $\overrightarrow{h_{t-1}}$. The input at the current step x_t and a hidden backward layer with hidden unit function $\overleftarrow{h}$ is computed based on the future hidden state $\overleftarrow{h_{t+1}}$. After target extraction by BiLSTM, all given sentences are classified into BIO labeled EI, EO, EQ, EIF, ILF, according to the number of targets extracted from them. As illustrated in Fig. 4, the BiLSTM is designed for label prediction, which uses two LSTMs to learn each token of the sequence based on both the past and the future context of the token. The forward and backward context representations, generated by $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$, respectively, are concatenated into a long vector. The combined outputs are the predictions of the given target label of each word.

2) CRF TAGGING MODEL

Despite the success of the BiLSTM model'in problems such as POS tagging, its independent classification decisions are limiting when there are strong dependencies across output labels. As shown in the example, if simply combining the highest score of the label, the prediction result of successive labels can be B-EI, B-EQ, which is not correct. Therefore, the BiLSTM can only predict the relation between the word and its label but cannot predict the relation between labels. In that case, the CRF model is designed to bridge this gap. Conditional random fields (CRFs) are a type of discriminative undirected probabilistic graphical model that represents a single log-linear distribution over structured outputs as a function of a particular observation input sequence. A transition matrix models the relation between labels.

Given observations variables X

$$X = (x_1, x_2, x_3, \ldots, x_n) \tag{3}$$

Consider P to be the matrix of scores output by the BiLSTM network. P is of size $n * k$, where k is the number of distinct tags, and $P_{i,j}$ corresponds to the score of the j^{th} tag of the i^{th}word in a sentence. The random variables Y, whose values require the model to predict,

$$Y = (y_1, y_2, y_3, \ldots, y_n) \tag{4}$$

$\sigma(X, y)$ is the score function defined as follows:

$$\sigma(X, y) = \sum_{i=0}^{n} A_{y_i, yi+1} + \sum_{i=1}^{n} P_{i,y_i} \tag{5}$$

where A is a matrix of transition scores and $A_{y_i,yi+1}$ represents the score of a transition from tag y_i to y_{i+1}. n is the length of a sentence, P is the matrix of scores output by the BiLSTM network, and P_{i,y_i} is the score of the $y_i^t h$ tag of the ith word in a sentence. A softmax over all possible tag sequences gives a probability for the sequence y as defined in Equation (6).

$$p(y|X) = \frac{e^{\sigma}(X, y)}{\sum_{\widetilde{y} \in Y_X} e^{\sigma(X,\widetilde{y})}} \tag{6}$$

Maximize the log-probability of the correct tag sequence during training:

$$\begin{aligned} log(p(y|X)) &= \sigma(X, y) - log(\sum_{\widetilde{y} \in Y_X} e^{\sigma(X,\widetilde{y})}) \\ &= \sigma(X, y) - logadd_{\widetilde{y} \in Y_X}(\sigma(X, \widetilde{y})) \end{aligned} \tag{7}$$

where Y_X represents all possible tag sequences for a sentence X. The output of the sequence that obtains the maximum score is given by:

$$y^* = argmax_{\widetilde{y} \in Y} \sigma(X, \widetilde{y}) \tag{8}$$

V. INDUSTRY CASE STUDY

The purpose of the case study is to validate the proposed methodology and evaluate the efficiency improvement quantitatively and to answer the research question raised in Section III. The NER model proposed in Section IV was applied for FP counting in place of manual recognition. Three groups of experiments with three senior consultants as participants were designed and carried out to validate the efficiency improvement of FP recognition with the aid of the proposed model.

A. DESCRIPTION OF INDUSTRY CASE

The case is from our industry partner, State Grid, which is the largest electricity operator in China. In the company, many software programs need to be designed, developed, updated, or maintained each year to support the efficient operation of the business and organization. Therefore, a total of an average of three billion CNY budgets for software need to be divided into different projects according to the requirements from the various business departments. However, the proposed requirement from these departments needs to be evaluated and then divided according to the estimation

TABLE 5. Requirement(REQ) sample from industry*.

ID	Requirement	Status	Requirement Description (DESC).	Function DESC.	Label
1	Account management of multirents	New	The operation includes account create and delete	Account **create**	EI
2	Account management of multirents	New	The operation include account create and delete	Account **delete**	EI
3	Monitor& manage of multirent resource	New	The renter can judge the further investment by monitoring the resource, which includes the monitor on the utilization ratio of CPU, hard disk, memory, network	Monitor the utilization of CPU	EO

* the content is translated from Chinese

result in a short time, which is only three months. Therefore, it is an urgent and critical task for both the budget sector and software supplier to settle down mature. The function point-based NESMA standard has been used for consistent and accurate evaluation progress and has been established as a national standard (NS) in China. However, even with enough experienced consultants, it is still difficult for the company to complete the work adequately in such a short time. Moreover, the accuracy of the result can vary between consultants without enough discussion, so how to make this procedure more efficient and more consolidated is a crucially important issue in the industry.

In this case study, the deep learning model proposed in Section IV was introduced to address this problem. Twenty-nine real project requirement documents were provided by our partner in the format of a spreadsheet for model training. However, the raw data of requirements is confidential so that it can not be accessed by the public. As shown in Table.5, each requirement was decomposed into several columns for further analysis. The second column is the original requirement proposed by the business sector, while the third column labels the requirement with *New* or *Add* by comparing the requirement with the existing system. The fourth column is the extension of the requirement, which is provided by the requirement analyst and provides a technical explanation of the requirement. The next two columns are the function point item and its type given by the consultant, in which the function point item is the feature of the function point type extract from the requirement description. As seen in the table, the description of the first and second requirements is the same while the function description is not; in this case, this single requirement needs to be decomposed into different functions and further into different function points.

B. DATA PREPROCESSING

As required by the proposed BiLSTM-CRF model, the input of the training sample needed to be clean and transferred into a low-dimensional, real-value vector. The data cleaning process included stop-word elimination which include 1396 english and Chinese stop words, case transition, and traditional to simple Chinese conversion. Then, the requirement was segmented into words, and the redundant word was removed. As a result, a whole 81,966 characters of requirements in which 3,457 unique words with wiki Chinese were added for the word2vec training to improve the accuracy of the word embedding model training and reduce the unknown word number. The requirement and its label were separately embedded and transferred into one-hot, which will then be padded for further training.

The total sample was divided into training and testing sets for training the BiLSTM-CRF model. After the sample set was generated, which consisted of 7,785 labeled requirement items, cross-validation was used to obtain the training set and validation set, which obtained 5,000 training samples and 2,785 test samples.

C. EXPERIMENTAL DESIGN

1) MODEL CONSTRUCTION AND TRAINING

For construction of the model, the first step is transfer the requirement and its label into vector, which have been described in previous. After this three layers were defined as described in Section IV, which represented the embedding layer, BiLSTM layer, and CRF layer. The model structure in this study can be seen in Table 6.

TABLE 6. Model structure.

Layer type	Output shape	ParamNumber
Embedding	(none,none,200)	228,600
Bidirection	(none,none,200)	240,800
CRF	(none,none,11)	2,354

As shown, the total number of parameters that were trained was 471,754. The training set was first padded into a $50 * 300$ array, while the label needed to be coded into a one-hot vector, which was combined according to each training sample. The input of the embedding layer was the data after the padding operation, while the output dimension in this study was defined as 200. Thus, the input number of neural networks in the BiLSTM was the same as the output dimension of the embedding layer, in which the backpropagation and forward propagation were 100. As described in Section IV, 11 labels were defined for recognition so that the output dimension of the CRF should be the same as the number of labels. In the learning process, some super parameters need to be defined. The maximum length of a single sentence was used for the definition of the recursive time step value, which is set to 624 in this case. Moreover, sentences in which the length is less than the step value were masked by 0. The epoch was set to 100, while the batch size was 16.

As the traditional evaluation of effectiveness, the quality of the model was evaluated in three indexes: precision, recall and

F1 value. These indexes were defined as follows:

$$Precision = \frac{Correct_{num}}{Predict_{num}} * 100\% \quad (9)$$

$$Recall = \frac{Correct_{num}}{Labeled_{num}} * 100\% \quad (10)$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} * 100\% \quad (11)$$

A PC with an 8-core CPU (Intel i7 9700), 16 GB DDR 1667 Memory, 1 TB HDD, an Nvidia RTX2070s GPU was used to execute the experiments. The software environment of the experiment was an Ubuntu 16.04 LTS system, and Python 3.6, Keras with TensorFlow-GPU-1.15 backend.

2) EFFICIENCY IMPROVEMENT EVALUATE EXPERIMENT

A designed experiment conducted a quantitative evaluation of efficiency. Three groups of FP recognition tests were carried out. Each group was set to compare the time consumption under three different scenarios, of which the difference was with and without the aid of the NER model.

TABLE 7. Experiment design.

GroupID	Quiz Type		
	No sugg	Half Sugg.	Full Sugg.
Group1	REQ.1	REQ.3	REQ.2
Group2	REQ.2	REQ.1	REQ.3
Group3	REQ.3	REQ.2	REQ.1

As illustrated in Table 7, these three scenarios are traditional manual FP recognition with no suggestion (No Sugg), FP recognition with only FP item suggestion (Half Sugg), and FP recognition with FP item and corresponding FP type suggestion (Full Sugg). Moreover, the difference between these groups is the information provided in the requirement, as shown in Table 8.

TABLE 8. Interpretation for the Feature.

requirement sample	No sugg	Half Sugg.	Full Sugg.
Insert record into the table	{}	{**Insert**}	{ **EI: Insert**}

In the no suggestion group, only the original requirement was provided, while in the half suggestion, the FP entity was provided; furthermore, in the full suggestion group, the FP entity and its type were provided. Moreover, each requirement was evaluated under different scenarios while keeping them independent of the test within the group. Each group consisted of three different types of quiz and included three different requirement documents (e.g., REQ1, REQ2, REQ3)

The quiz was designed for the experiment, which contained the requirement from different software within the same domain. The content of the quiz is described in Table 9.

As shown, each quiz contained 100 requirement items. The number of characters varying from groups is listed in Table 9. The distribution of FP types in these three requirement documents is presented in Fig. 5.

TABLE 9. Description of the designed quiz.

	REQ.1	REQ.2	REQ.3
Num of Req Item	100	100	100
Num of Character	3692	2033	3270

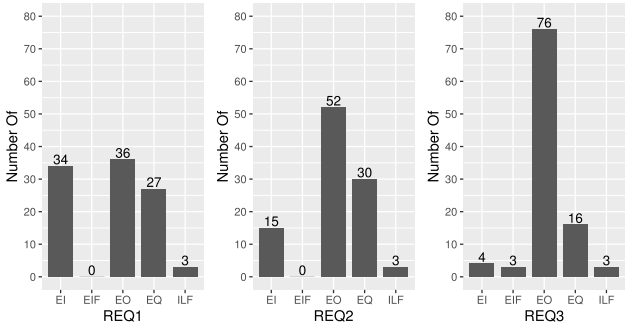

FIGURE 5. Function point distribution within quiz.

As shown in Fig. 5, EI, EO, and EQ were far more than ILF and EIF types in all three requirements. Moreover, the prediction given by the NER model gave one prediction for one given requirement in a single request. Therefore, the instruction for the quiz was predefined as below to obtain a correct evaluation:

1) Finish the quiz as fast as possible.
2) One FP for each requirement item and if the multitype of FP can be conducted in the requirement, EI, EQ, EO for the priority.

Finally, the experiments were conducted with three senior software cost estimation consultants separately, and the time consumption was recorded in the accuracy of seconds.

TABLE 10. Prediction result sample*.

ID	Orignal Labeled Data			**Prediction Result**	
	Requirement	FP Item	Type	FP item	Label
1	Support checking for the user signed information	Checking	EO	Checking	EO

* The content is translated from Chinese

VI. RESULT ANALYSIS AND ANSWER TO RQs

The sample of the prediction result can be seen in Table 10. As shown, each column included two parts. The first part showed the originally labeled data, while the second part showed the prediction result given by the NER model. The originally labeled data included the description of the requirement, the function point item, and the label given by the consultant. As seen in Table 10, the prediction result gave the predicted function point item and its type. The prediction result of all the samples used for the testing can be seen in Table 11.

As shown in Table 11, the number of labeled and predicted sample sets was the same at 2,785, while the accurately predicted label was 2,313. It can be seen that the predicted number was equal to the labeled number, which means that

TABLE 11. Statistical result of prediction.

FP type	Number of Samples		
	Correct	Predict	Labeled
EI	530	566	617
EO	1475	1550	1756
EQ	70	72	103
ILF	224	245	293
EIF	14	352	16
Total	2313	2785	2785

the model predicted each sample, but not every prediction was correct. As shown, the prediction sample number of EIF was much less than the other function point types, and the labeled EIF was only 16, while the prediction result was 352, which was not correct. The result of the evaluation index defined in Equation (9), Equation (10), and Equation (11) can be seen in Table 12.

TABLE 12. Result analysis.

Index	EI	EO	EQ	ILF	EIF
Precision	0.94	0.95	0.97	0.91	0.04
Recall	0.86	0.84	0.68	0.76	0.88
F1	**0.90**	**0.89**	**0.80**	**0.83**	**0.08**

In addition to the EIF type, the precision of other types of function point recognition achieved a sound performance compared with the manual work by the consultant. The precision of the other four types of FP reached over 94.25%, while the F1 value achieved over 80%. Furthermore, the reason for the overfitting of EIF prediction is that the number of labeled EIFs was too small for training.

As illustrated, the improvement in the efficiency of learning-based FPA is remarkable, reaching an average of 38.6%. Moreover, the time consumption in REQ2 and REQ3 is less than that in REQ1, and less improvement can be observed between the half suggestion and the full suggestion version. The efficiency of the proposed method evaluates the time consumption in a given requirement under different scenarios. The improvement of the efficiency can be defined as Equation (12):

$$\phi_{improve} = \frac{Time_{NoSugg} - Time_{FullSugg}}{Time_{NoSugg}} * 100\% \qquad (12)$$

The time consumption is recorded in seconds. The result is illustrated in Fig. 6:

As shown, there is an apparent trend of decreasing time consumption in the no suggestion, half suggestion and full suggestion groups in REQ1, REQ2 and REQ3. Moreover, the time consumption of REQ1 is higher than that of REQ2 and REQ3 in every scenario.

Therefore, several conclusions can be reached from these experiments. First, the FP type distribution of the requirement may have an impact on the time spent. For that, in REQ2, the EO type takes over 52% while the number increases to 76% in REQ3, which can save much time for switching between the different FPA rules. The second is that the most

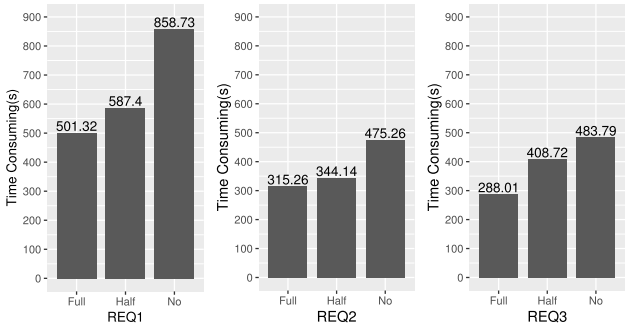

FIGURE 6. Efficiency improvement.

time-consuming work during the experiment is the correct identification of the feature in the FP entity. If the feature was given, then the FP type classification was a relatively simple operation compared with finding the FP entity in the requirement document.

Considering the final accuracy of the FPA process with the aid of the NER model, the description of the function is derived from the requirement so that the difference of FP type recognition can be argued under a different interpretation of the assumption context by the consultant. Moreover, the classification of FP type is simple for the senior software consultant because they only need to determine which kind the FP type is and do not need to determine if there exists an FP. Under this design, the worst different result between the consultant is 10. In that case, each group can obtain over 90% accuracy under a different scenario. Consequently, with the help of the proposed methodology, the conclusion can be reached that the consultant can conduct the FPA more efficiently without the reduction in accuracy.

A. ANSWER TO RESEARCH QUESTIONS

Three research questions accompanied the objective of this study. After the experiment on the proposed learning model and implementation of research objectives, we are now ready to respond to the answers to research questions.

RQ1 (Sanity Check): Is the proposed method suitable for estimating the function point? Traditional FP recognition is conducted manually by the consultant. In this research study, an NER-based FP recognition model substitutes in this part, as illustrated in Fig. 1. The deep learning NER model learns from the historic labeled data and predicts the FP item and its type for each sentence of the requirement. The combination of the output of the NER model and the consultant inspection performed much more effectively with less time consumed according to the experiment in the case study. As seen in Table 13, the result shows that the newly proposed FPA process makes FP recognition more efficient. Therefore, the proposed method in this paper outperforms manual FP recognition, thus passing the sanity check required by RQ1.

RQ2 (Accuracy of the Model): How does the accuracy of the proposed model compare with manual FPA? According to our research study, a domain NER model was

TABLE 13. Efficiency improvement result analysis*.

	No Sugg.	Half Sugg.	Full Sugg.	Improvement
REQ1	14:18.73	09:47.40	08:21.32	41.6%
REQ2	07:55.26	05:44.14	05:15.26	33.7%
REQ3	08:03.79	06:48.72	04:48.01	40.5%
Average				38.6%

* Unit:minite

constructed for FP prediction. The accuracy here can be divided into two parts: the first is the accuracy of the model, while the second is the accuracy of the proposed FPA process with the aid of the NER model. The performance of the model is shown in Table. 11 and Table 12, which achieved approximately 94.25% precision, and the F1 value achieved over 80% in addition to the EIF type. The total accuracy was obtained from the result of the prediction result in Table 11, which was over 83%. Moreover, compared with manual labeling, the final accuracy of the FPA process with the aid of the NER model achieved at least over 90% accuracy.

RQ3 (Efficiency Improvement with the Model): How much efficiency improvement can be achieved by the proposed model? In this research study, the proposed FPA method replaces the manual FP recognition process with a deep learning-based NER model. Consequently, the reading requirement procedure, recognizing the FP item, and then classifying them into five types was tested under a comparative experiment. Through the validation of the industry case, the proposed NER model aided procedure achieved over 90% accuracy, as conducted in RQ2, and an average efficiency improvement reached 38.6%, as shown in Table 13, which can be a significant enhancement for the FP base function point software size estimation.

VII. CONCLUSION AND FURTHER WORK

Considerable work has been done on software effort estimation toward the improvement of accuracy and applicable scenarios [14], [56], [57]. However, most of the proposed methods construct the effort prediction model on the given metric of the software or even on the existing datasets, such as ISBSG and NASA. In the industry scenario, these metric datasets are generally not already available, especially the metric data for SCE. Software size is a fundamental metric for the SCE, which is costly and time consuming according to practice. It becomes a barrier to the widespread existing SCE model, which is the primary concern in this research study.

In this paper, a novel approach was presented to improve the efficiency of sentence-level function point analysis via a deep learning-based function point type classification model. The approach adopts BiLSTM-CRF to extract function point features in given sentences and classifies them into different types according to the FPA standard. A case from our industry partner was conducted to validate the proposed method and evaluate the efficiency improvement quantitatively. The empirical results show that our approach performs state-of-the-art performance on function point analysis and can significantly enhance the efficiency of FP-based software size estimation.

However, there are still several problems that remain, which can be a direction for further research. According to the experiment, the readability and interpretation speed are crucial for accuracy and efficiency, so what and how to make the output of the learning model more suitable for the FPA consultant is also an interesting research aspect. Moreover, in this research study, the provided sentence already contained the FP item, which was derived from the original requirement in advance. Therefore, this scenario can be extended into extracting the FP item directly from the original requirement document and evaluating the efficiency improvement, which will significantly decrease the cost of the FPA. In addition to the extension of the learning model to the original requirement, there are many SCE-related metrics defined in the SCE model, such as the recognition of the data element type (DET) and the adjustment factors, which are also important metrics of the software. Therefore, how to integrate these elements for a full automatic estimation is also a direction for future research. Finally, the data used for training is domain-specific and the accuracy of the recognition can be not as much higher when a completely new requirement from a new domain, therefore, how to combined different learning models from different domains can be a future research direction.

ACKNOWLEDGMENT

The authors would like to thank Prof. Chao Liu, who unfortunately passed away just before this article was submitted for publication. Prof. Liu served in an essential role in the research described herein and he is greatly missed by all of us.

REFERENCES

[1] T. E. Hastings and A. S. M. Sajeev, "A vector-based approach to software size measurement and effort estimation," *IEEE Trans. Softw. Eng.*, vol. 27, no. 4, pp. 337–350, Apr. 2001.
[2] S. Grimstad and M. Jørgensen, "A framework for the analysis of software cost estimation accuracy," in *Proc. ACM/IEEE Int. Symp. Empirical Softw. Eng.*, 2006, pp. 58–65.
[3] C.-Y. Huang and M. R. Lyu, "Optimal release time for software systems considering cost, testing-effort, and test efficiency," *IEEE Trans. Rel.*, vol. 54, no. 4, pp. 583–591, Dec. 2005.
[4] M. A. Yahya, R. Ahmad, and S. P. Lee, "Effects of software process maturity on COCOMO II's effort estimation from CMMI perspective," in *Proc. IEEE Int. Conf. Res., Innov. Vis. Future Comput. Commun. Technol.*, Jul. 2008, pp. 255–262.
[5] M. Jørgensen and G. Stein, "Avoiding irrelevant and misleading information when estimating development effort," *IEEE Softw.*, vol. 25, no. 3, pp. 78–83, May/Jun. 2008.
[6] R. T. Hughes, "Expert judgement as an estimating method," *Inf. Softw. Technol.*, vol. 38, no. 2, pp. 67–75, Jan. 1996.
[7] B. W. Boehm, *Software Engineering Economics*, vol. 197. Upper Saddle River, NJ, USA: Prentice-Hall, 1981,
[8] B. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. J. Reifer, and B. Steece, *Cost Estimation With Cocomo II*. Upper Saddle River, NJ, USA: Prentice-Hall, 2000.
[9] R. Valerdi, B. W. Boehm, and D. J. Reifer, "COSYSMO: A constructive systems engineering cost model coming of age," in *Proc. INCOSE Int. Symp.*, vol. 13, no. 1. Hoboken, NJ, USA: Wiley, 2003, pp. 70–82.
[10] L. H. Putnam, "A general empirical solution to the macro software sizing and estimating problem," *IEEE Trans. Softw. Eng.*, vol. SE-4, no. 4, pp. 345–361, Jul. 1978.
[11] Y. Singh, P. K. Bhatia, and O. Sangwan, "ANN model for predicting software function point metric," *ACM SIGSOFT Softw. Eng. Notes*, vol. 34, no. 1, pp. 1–4, Jan. 2009.

[12] I. F. de Barcelos Tronto, J. D. S. da Silva, and N. Sant'Anna, "An investigation of artificial neural networks based prediction systems in software project management," *J. Syst. Softw.*, vol. 81, no. 3, pp. 356–367, Mar. 2008.

[13] K. V. Kumar, V. Ravi, M. Carr, and N. R. Kiran, "Software development cost estimation using wavelet neural networks," *J. Syst. Softw.*, vol. 81, no. 11, pp. 1853–1867, Nov. 2008.

[14] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," *J. Syst. Softw.*, vol. 137, pp. 184–196, Mar. 2018, doi: 10.1016/j.jss.2017.11.066.

[15] A. Albrecht, "Software function, source lines of code, and development effort prediction," *IEEE Trans. Softw. Eng.*, vol. SE-9, no. 6, pp. 83–92, Nov. 1979.

[16] M. Salmanoglu, T. Hacaloglu, and O. Demirors, "Effort estimation for agile software development: Comparative case studies using COSMIC functional size measurement and story points," in *Proc. ACM Int. Conf. Proc.*, 2017, pp. 41–49, doi: 10.1145/3143434.3143450.

[17] P. Mohagheghi, B. Anda, and R. Conradi, "Effort estimation of use cases for incremental large-scale software development," *Softw. Process Improvement*, pp. 309–326, May 2006.

[18] V. Saxena and M. Shrivastava, "Performance of function point analysis through UML modeling," *ACM SIGSOFT Softw. Eng. Notes*, vol. 34, no. 2, pp. 1–4, Feb. 2009.

[19] *Function Point CPM IFPUG. Release. 4.1.*, IFPUG, Westerville, OH, USA, 1999.

[20] C. R. Symons, *Software Sizing and Estimating: MK II FPA* (Function Point Analysis). Hoboken, NJ, USA: Wiley, 1991.

[21] F. Niessink and H. Van Vliet, "Predicting maintenance effort with function points," in *Proc. Int. Conf. Softw. Maintenance*, 1997, pp. 32–39.

[22] R. Meli, A. Abran, V. T. Ho, and S. Oligny, "On the applicability of COSMIC-FFP for measuring software throughout its life cycle," in *Proc. 11th Eur. Softw. Control Metrics Conf.*, 2000, pp. 18–20.

[23] C. Jones, "A new business model for function point metrics," Capers Jones Associates LLC, Tech. Rep., 2008.

[24] C. Jones, *Software Assessments, Benchmarks, and Best Practices*. Boston, MA, USA: Addison-Wesley, 2000.

[25] S. Kang, O. Choi, and J. Baik, "Model-based dynamic cost estimation and tracking method for agile software development," in *Proc. IEEE/ACIS 9th Int. Conf. Comput. Inf. Sci.*, Aug. 2010, pp. 743–748.

[26] G. Karner, "Metrics for objectory. No. LiTH-IDA-Ex-9344: 21," Ph.D. dissertation, Univ. Linköping, Linköping, Sweden, Dec. 1993.

[27] M. Heričko and A. Živkovič, "The size and effort estimates in iterative development," *Inf. Softw. Technol.*, vol. 50, nos. 7–8, pp. 772–781, Jun. 2008.

[28] C. M. B. da Silva, D. S. Loubach, and A. M. da Cunha, "Applying the use case points effort estimation technique to avionics systems," in *Proc. IEEE/AIAA 27th Digit. Avionics Syst. Conf.*, Oct. 2008, p. 5.

[29] F. Fioravanti, P. Nesi, and F. Stortoni, "Metrics for controlling effort during adaptive maintenance of object oriented systems," in *Proc. IEEE Int. Conf. Softw. Maintenance (ICSM) Softw. Maintenance Bus. Change*, Aug. 1999, pp. 483–492.

[30] S. Abrahão, J. Gómez, and E. Insfran, "Validating a size measure for effort estimation in model-driven Web development," *Inf. Sci.*, vol. 180, no. 20, pp. 3932–3954, Oct. 2010.

[31] J. Gomez, C. Cachero, and O. Pastor, "Conceptual modeling of device-independent Web applications," *IEEE MultimediaMag.*, vol. 8, no. 2, pp. 26–39, Apr. 2001.

[32] IFPUG Counting Practices Committee *et al.*, "Function point counting practices manual," Tech. Rep. 4.2.1, 2005.

[33] *Counting Guidelines for the Application of Function Point Analysis*, Definitions NESMA, Dordrecht, The Netherlands, 1997.

[34] A. Abran, J.-M. Desharnais, and F. Aziz, "3.5 measurement convertibility–from function points to COSMIC FFP," in *Proc. Cosmic Function Points, Theory Adv. Practices*, 2016, p. 214.

[35] A. Abran, "FFP reléase 2.0: An implementation of COSMIC functional size measurement concepts," in *Proc. FESMA*, Amsterdam, The Netherlands, Oct. 1999.

[36] R. Meli, "Simple function point: A new functional size measurement method fully compliant with IFPUG 4. X," in *Proc. Softw. Meas. Eur. Forum*, Jun. 2011.

[37] C. Jones, "Function points as a universal software metric," *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 4, pp. 1–27, Jul. 2013.

[38] C. Zhang, S. Tong, W. Mo, Y. Zhou, Y. Xia, and B. Shen, "ESSE: An early software size estimation method based on auto-extracted requirements features," in *Proc. 8th Asia–Pacific Symp. Internetware*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 112–115, doi: 10.1145/2993717.2993733.

[39] S. Bagriyanik and A. Karahoca, "Automated COSMIC function point measurement using a requirements engineering ontology," *Inf. Softw. Technol.*, vol. 72, pp. 189–203, Apr. 2016.

[40] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," *IEEE Trans. Softw. Eng.*, vol. 45, no. 7, pp. 637–656, Jul. 2019.

[41] A. Borthwick and R. Grishman, "A maximum entropy approach to named entity recognition," Ph.D. dissertation, New York Univ., New York, NY, USA, 1999.

[42] H. Isozaki and H. Kazawa, "Efficient support vector classifiers for named entity recognition," in *Proc. 19th Int. Conf. Comput. Linguistics*, vol. 1. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 1–7.

[43] G. Zhou and J. Su, "Named entity recognition using an HMM-based chunk tagger," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2001, pp. 473–480.

[44] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and Web-enhanced lexicons," in *Proc. 7th Conf. Natural Lang. Learn. (HLT-NAACL)*, vol. 4. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 188–191.

[45] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010, pp. 1045–1048.

[46] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[47] J. Hammerton, "Named entity recognition with long short-term memory," in *Proc. 7th Conf. Natural Lang. Learn. (HLT-NAACL)*, vol. 4. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 172–175.

[48] M. Labeau, K. Löser, and A. Allauzen, "Non-lexical neural architecture for fine-grained POS tagging," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 232–237.

[49] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," 2016, *arXiv:1603.01360*. [Online]. Available: http://arxiv.org/abs/1603.01360

[50] W. Chen, Y. Zhang, and H. Isahara, "Chinese named entity recognition with conditional random fields," in *Proc. 5th SIGHAN Workshop Chin. Lang. Process.*, 2006, pp. 118–121.

[51] G. Fu and K.-K. Luke, "Chinese named entity recognition using lexicalized HMMs," *ACM SIGKDD Explor. Newslett.*, vol. 7, no. 1, pp. 19–25, Jun. 2005.

[52] C. Jones and O. Bonsignour, *The Economics of Software Quality*. Reading, MA, USA: Addison-Wesley, 2011.

[53] E. S. Ristad and P. N. Yianilos, "Learning string-edit distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 5, pp. 522–532, May 1998.

[54] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[55] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[56] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Trans. Softw. Eng.*, vol. 23, no. 11, pp. 736–743, Nov. 1997.

[57] S. P. Singh, V. P. Singh, and A. K. Mehta, "Differential evolution using homeostasis adaption based mutation operator and its application for software cost estimation," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 33, no. 16, pp. 740–752, Jun. 2018, doi: 10.1016/j.jksuci.2018.05.009.

KUI ZHANG received the master's degree in information management and information systems from the Beijing Institute of Technology, Beijing, China, in 2010. He is currently pursuing the Ph.D. degree with the Software Engineering Institute (SEI), Beihang University. His research interests include model-driven engineering, model-based real-time analysis, airworthiness certification, model-based safety analysis, and general model-based software engineering.

XU WANG (Member, IEEE) received the bachelor's degree in computer technology from Beihang University. He is currently an Assistant Engineer with the China Ship Research and Development Academy. His research interests include search-based software engineering, natural language processing, and machine learning.

JIAN REN received the dual M.Sc. degree from the Queen Mary University of London and King's College London and the Ph.D. degree in computer science from University College London. He is currently an Assistant Professor with the School of Computer Science and Engineering, Beihang University, Beijing. His research interests include search-based software engineering, software project planning and management, requirements engineering, and evolutionary computation.

CHAO LIU (Member, IEEE) received the M.S. degree in computer software and theory and the Ph.D. degree from Beihang University. He was a Professor of software engineering with Beihang University. His research interests include software quality engineering, software testing, model-driven software development, and software process improvement. During the preceding decade, he primarily focused on the modeling and verification of safety-critical software and systems, including safety requirement modeling and analysis, evidence-based software safety analysis and evaluation, software safety and reliability analysis based on the software development process, and model-driven software testing.