Calibrating Use Case Points Using Bayesian Analysis

Kan Qi
University of Southern California
United States
kqi@usc.edu

Anandi Hira
University of Southern California
United States
a.hira@usc.edu

Elaine Venson
University of Southern California
United States
venson@usc.edu

Barry W. Boehm
University of Southern California
United States
boehm@usc.edu

ABSTRACT

Background: Use Case Points (UCPs) have been widely used to estimate software size for object-oriented projects. Yet, many research papers criticize the UCPs methodology for not being verified and validated with data, leading to inaccurate size estimates.

Aims: This paper explores the use of Bayesian analysis to calibrate the use case complexity weights of the UCPs method to improve size and effort estimation accuracy.

Method: Bayesian analysis is applied to integrate prior information (in this study, we use the weights defined by the UCPs method and the weights suggested by other research papers) with parameter values suggested by data. To validate the effectivenss of this approach, we evaluate the out-of-sample effort estimation accuracy using 10-fold cross validation over three datasets (D1, D2, and D3) of 105 use case driven projects in total.

Results: The Bayesian method outperforms the A-Priori, Original UCPs, and Regression methods by 13.4%, 15.9%, and 15.9% respectively for PRED(.25), and by 19.0%, 19.1%, and 20.0% respectively for MMRE, for D1, according to the cross-validation out-of-sample testing results. The paper also presents the accuracy evaluation results for datasets D2 and D3.

Conclusions: The results show that the Bayesian estimates of the use case complexity weights consistently provide better estimation accuracy, compared to the weights proposed by the original UCPs method, the weights calibrated by mulitple linear regression, and the weights suggested by experts.

CCS CONCEPTS

• **Software and its engineering → Software development process management**;

KEYWORDS

effort estimation, software sizing models, software size metrics, use case analysis, Use Case Points, functional size measurement, statistical model calibration, local calibration, Bayesian analysis, meta-analysis, use case driven process, project management

ACM Reference Format:
Kan Qi, Anandi Hira, Elaine Venson, and Barry W. Boehm. 2018. Calibrating Use Case Points Using Bayesian Analysis. In *ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '18), October 11–12, 2018, Oulu, Finland*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3239235.3239236

1 INTRODUCTION

Effort estimation has been regarded as a crucial driver for various software managerial decisions. For example, accurate estimates of project effort at the early stages help project managers effectively allocate project resources, make project plans, and react to the risks of being over schedule or budget[1][2][3]. To ensure the utility of effort estimation in decision making, it is necessary for an effort estimation model to provide estimates at early stages of a project, during which time, very little information about the project or system is known for software size measurement. Most of the effort estimation models rely on a size metric as the major effort predictor[4]. To measure software size early in a project, Karner developed Use Case Points (UCPs) which provides a functional size metric to estimate the effort for object-oriented projects that utilize the use case technique of gathering and understanding requirements [5]. Due to its early applicability during the software development lifecycle, UCPs has gained wide acceptance [6].

Table 1 presents the steps and rules to calculate UCPs. In summary, use cases are first classified into three levels of complexity, based on the number of internal transactions within each use case. Each complexity level is assigned a weight to represent its effect on software size. For example, if a use case contains 1-3 transactions, it is determined as a simple use case and has a weight of 5; whereas a use case with 5 transactions is determined as an average use case with a weight of 10. Actors are similarly classified and weighted. The sums of the weighted use cases and weighted actors are called Unadjusted Use Case Weight (UUCW) and Unadjusted Actor Weight (UAW) respectively, and the sum of UUCW and UAW is called Unajdusted Use Case Points (UUCP). Then, environmental (EF) and technical (TCF) complexity factors are evaluated to represent the influences from those aspects on project effort. UCPs is calculated by multiplying UUCP, EF, and TCF. Since UUCW contributes most to the software size measurements (more than 90% in our experimental datasets), we focus on calibrating the use cases complexity weights using the available datasets to avoid the

ESEM '18, October 11–12, 2018, Oulu, Finland
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5823-1/18/10…$15.00
https://doi.org/10.1145/3239235.3239236

Table 1: The UCPs calculation process

Step	Rules	Results
1	Classify the use cases (C) into 3 levels of complexity (LOC), based on number of transactions (NT) in each use case: $LOC_c = \begin{cases} Simple, & NT_c <= 3 \\ Average, & NT_c <= 7 \\ Complex, & NT_c > 7 \end{cases}$	LOC_c
2	Sum the number of weighted use cases as unadjusted use case weight ($UUCW$): $UUCW = \sum_{c \in C} W_c$ Where: $W_c = \begin{cases} 5, & LOC_c = Simple \\ 10, & LOC_c = Average \\ 15, & LOC_c = Complex \end{cases}$	$UUCW$
3	Classify the actors (A) into 3 levels of complexity and assign a weight for each actor based on its level of complexity. Sum the number of weighted actors as unadjusted actor weight (UAW) $UAW = \sum_{a \in A} W_a$ Where: $W_a = \begin{cases} 1, & LOC_a = Simple \\ 2, & LOC_a = Average \\ 3, & LOC_a = Complex \end{cases}$	UAW
4	Evaluate the 13 technical factors and calculate TCF based on the sum of their impact ($TFactor$): $TCF = 0.6 + (0.01 * TFactor)$	TCF
5	Evaluate the 8 environmental factors and calculate EF based on the sum of their impact ($EFactor$): $EF = 1.4 + (-0.03 * EFactor)$	EF
6	Calculate use case points (UCP) : $UCP = (UUCW + UAW) * TCF * EF$ $= UUCP * TCF * EF$	UCP

problem of overfitting. However, we maintain that the proposed approach is also applicable to calibrating actor complexity weights, used to calculate UAW.

While practitioners and research papers have reported the effectiveness of UCPs [7–12], the UCPs method has also been criticized for the complexity weights not being validated with data [6][13] - Karner defined the complexity weights based on his domain knowledge gained from Objectory Systems [5]. Also, as the software development environment has changed greatly since the development of the UCPs method, its weighting scheme may not be applicable for modern use case driven projects.

To overcome this well-known issue, we use Bayesian analysis to update the experts' estimates of use case complexity weights using empirically calibrated results with the goal of improving effort estimation accuracy with UCPs.

Different from the maximum likelihood method of estimating parameters, the Bayesian approach minimizes the posterior expected loss by combining the prior information with the sample information. Specifically, the posterior probability ($P(\theta|X)$) of a hypothesis is derived by updating the prior probability ($P(\theta)$) as more evidence becomes available. The evidence is the new data that is not used to define prior probability. The posterior probability can be computed according to Bayes' theorem by Eq. (1).

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)} \tag{1}$$

Using the derived posterior probability distribution, we are able to calculate the posterior mean as the estimate of a parameter or variable using Eq. (2).

$$\tilde{\theta} = E(\theta) = \int_{\theta} \theta P(\theta|X)\, d\theta \tag{2}$$

For the normally distributed $P(X|\theta)$ and $P(\theta)$, the posterior mean can be analytically computed as the weighted average of the prior mean and the sample mean using Eq. (3)[14]. The weights in the weighted average are the precision of the two sources of information (defined as the inverse of the variance).

$$\tilde{\theta} = (\alpha + \beta)^{-1} * (\alpha * \theta^{*} + \beta * \theta^{**}) \tag{3}$$

In Eq. (3), $\tilde{\theta}$ is the posterior mean, or in other words, the Bayesian estimate of the parameter θ. θ^{*} and θ^{**} are means of the prior and sample information, and α and β are the inverses of their variances, called precisions. The variance of the posterior distribution can be correspondingly computed as Eq.(4).

$$Var(\tilde{\theta}) = (\alpha + \beta)^{-1} \tag{4}$$

The prior information usually consists of expert judgment, which has not been validated by data, while the sample information can be derived from statistical analysis of data, for example, by maximizing the likelihood function ($P(X|\theta)$) that is defined on data. In this paper, the use case complexity weights proposed in previously published papers are used as the prior information, and the weights determined by running multiple linear regression (MLR) analysis on the datasets of 105 historical projects are used as the sample information.

To validate the effectiveness of our approach, we used k-fold cross validation to evaluate the out-of-sample effort estimation accuracy of the approach based on 105 historical projects. In addition to the proof of the improvement in effort estimation accuracy, we also make a few interesting observations about the interconnection between sample size, homegeneity of datasets, and selection of software size estimators, based on our empirical study, which we think can be used as effective guidelines to select appropriate size estimation methods in the typical software size calibration situations.

The rest of the paper is structured into 6 sections. Section 2 introduces previous work completed in modifying the UCPs method for better effort estimation. Section 3 details our approach in calibrating the use case complexity weights with Bayesian analysis. The datasets used for and the results from model calibration and validation are presented in Section 4. Lastly, we discuss the threats to validity of the results in Section 5 and make the conclusions of the empirical analysis in Section 6.

2 RELATED WORK

When Karner first proposed the UCPs method in 1993, he explained how the complexity weights were set by stating: "the weights in this article are a first approximation by people at Objective Systems."[15]

At that point, Karner also pointed out that more data was needed to adjust the model, weights and parameters. Since then, the method has been highly used, yet some limitations have also been reported. For example, Nassif et al. argues that the lack of granularity when classifying the complexity of uses cases negatively affects estimation accuracy [13].

To tackle criticism that the originally defined complexity levels and the weights assigned to the levels might not reflect the actual situations, new approaches were proposed to improve this aspect of the UCPs estimation method. We distinguished three main groups of approaches: the first group focused adding extra complexity levels, the second group focused discretizing the existing complexity levels, and the third group focused empirically calibrating the use case and the actor complexity weights.

Adding extra complexity levels. Mudasir Manzoor Kirmani and Abdul Wahid proposed the Re-UCP method as a revision of the UCP and e-UCP (extended Use Case Points method [16]). Re-UCP adds one extra rating level - "critical"- for both the use case and actor weighting schemes [17]. They conducted an experiment with 51 students, who were trained and divided in groups to estimate the effort of 14 projects. They observed that the effort estimated using Re-UCP method was closer to the actual effort in comparison with estimated effort using UCP & e-UCP methods. Nassif added three more use case complexity levels to the UCPs' original weighting scheme, extending their complexity weights to 20, 25, and 30 points. Including other improvements to the UCPs method, he evaluated the proposed model with 65 industrial data points and achieved promising results [18].

Discretizing existing compelxity levels. Using fuzzy logic, Wang et al.[19] and Nassif[18] suggested discretizing the levels of complexity into more granular options and assigning corresponding weights to differentiate their effects on software size. Wang et al. applied fuzzy set theory to smooth over the abrupt classification of use cases, extending it from three to five categories. The authors demonstrated the effectiveness of their method through a case study [19]. Nassif et al. proposed an enhancement to the model using fuzzy logic and neural networks. The authors used fuzzy logic to discretize the use case complexity levels into ten categories according to the number of transactions in a use case, maintaining the maximum number of transactions as 10 and the complexity weight applied to the largest use case as 15, as originally defined by Karner. The evaluation of this approach was conducted on 20 different projects and the results showed that the UCPs-based software estimation can be improved by up to 22% in some projects [13].

Empirically calibrating complexity weights. In another paper, Nassif proposed to empirically calibrate the weights assigned to the different use case complexity levels using neural networks [20]. However, specific experiment results or details of the approach were not found. Other than this research, we were unable to find other research papers that calibrated the use case complexity weights.

Although these studies showed good results by extending the UCPs method, none of them presented specific results or methods to empirically calibrate the complexity weights. Our study presents a valuable contribution to research in the UCPs method, by demonstrating how the use case complexity weights can be empirically calibrated. We demonstrate that it is possible to improve the accuracy of UCPs-based software effort estimation if we update the weights proposed by domain experts and the weights calibrated by data using Bayesian analysis.

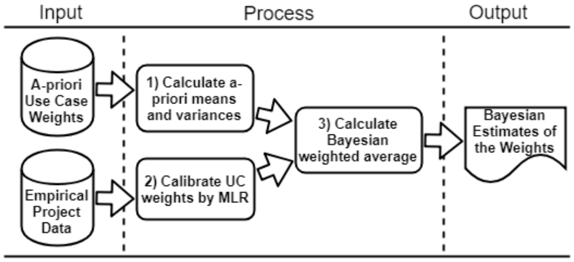
Input
Process
Output
A-priori Use Case Weights
Empirical Project Data
1) Calculate a-priori means and variances
2) Calibrate UC weights by MLR
3) Calculate Bayesian weighted average
Bayesian Estimates of the Weights

Figure 1: The Bayesian approach of UCPs weight calibration.

Bayesian Analysis, on the other hand, has been used in building the COCOMO®II effort estimation model to combine domain experience and empirical study results [1][21]. COCOMO®II combines effects of the cost drivers estimated by experts on project efforts and the effects calibrated by data to solve the unintuitive results from the calibration - the calibration returned negative values for some of the parameters, which are regarded as counter-intuitive to the experts. An empirical study verified that the parameter values resulting from Bayesian Analysis led to superior results compared to a model only based on expert judgment or data analysis [21].

In this paper, we followed the framework of Bayesian analysis and proposed the methods of synethizing experts' proposals of use case complexity weights, calibrating the weights from empirical data, and combinting the two pieces of information to achieve better effort estimation accuracy using UCPs.

3 THE BAYESIAN APPROACH TO CALIBRATE USE CASE WEIGHTS

3.1 The calibration process

As depicted in Figure 1, our approach of combining the prior and the sample information of use case complexity weights using Bayesian analysis generally goes through the following 3 steps:

(1) Calculate the means and variances of the complexity weights proposed by the experts as the prior information, which are denoted by the vectors $w_{a-pri} = \{w_1, w_2, w_3\}$ and $\delta^2_{a-pri} = \{\delta^2_1, \delta^2_2, \delta^2_3\}$. The details of our approach to deriving the prior information of the use case complexity weights are presented in Section 3.2.

(2) Calibrate the weights for simple use cases (UC_{simple}), average use cases ($UC_{average}$), and complex use cases ($UC_{complex}$) by running multiple linear regression (MLR) on an empirical dataset. The calibrated weights and their variances, denoted by vectors $w_{reg} = \{w^*_1, w^*_2, w^*_3\}$ and $\delta^2_{reg} = \{\delta^{*2}_1, \delta^{*2}_2, \delta^{*2}_3\}$, are used as the sample information input to the Bayesian analysis process. This is further explained in Section 3.3.

(3) Calculate the Bayesian estimates of the use case weights and their variances by perfoming a weighted average of the a priori means and the empirically calibrated weights for the use case complexity levels. The Bayesian estimates and variances are denoted by $w_{bayes} = \{\hat{w}_1, \hat{w}_2, \hat{w}_3\}$ and

Table 2: The weighting schemes from previous researches

Wght. Schm.	Study	Year	Metric
1	Karner[5]	1993	UCP
2	Wang et al.[24]	2009	EUCP
3	Kirmani and Wahid[17]	2009	Re-UCP
4	Nassif [18]	2012	Soft-UCP
5	Minkiewicz[25]	2015	UCP Sizing
6	Nassif et al.[13]	2016	Enhanced UCP

$\delta^2_{bayes} = \{\hat{\delta_1}^2, \hat{\delta_2}^2, \hat{\delta_3}^2\}$. The weights used in the averaging process are based on the variances of the two sources of information. The method to combine the prior information and sample information using Bayesian analysis is further explained in Section 3.4.

3.2 The prior information

In order to understand the differences in the use case complexity weights applied in practice, we did a systematic review of the papers related to use case calibration published between 2007 and 2017 in the 4 major research paper databases as suggested in [22]: Scopus, IEEE Xplore, ACM DL, and Science Direct. The aim of this literature review was to search for the previous studies that proposed different use case complexity weights. We also performed backwards snowballing, by examining the studies from the references of the selected papers.

Six papers were identified, which proposed new weighting schemes by distinct researchers and practitioners for the use case complexity levels (including the weighting scheme used in the original UCPs' definition). The sources of the weighting schemes are presented in Table 2. The weighting schemes are plotted in Figure 2. The weights proposed by these authors are based on their domain knowledge or analyses on datasets for the purpose of improving the estimation accuracy of the original Use Case Points (UCPs). For instance, Kirmani and Wahid introduced an extra level of complexity to cover the use cases with number of transactions being larger than 15 [23]. Wang et al. used fuzzy logic to calculate weights and determine how complexity levels should be set, leading to different weights being applied to different ranges of transactions from the original UCPs method [24]. We calculated the mean and the variance of the weights proposed by these experts for each of the three use case complexity levels, and use them as the prior information in the Bayesian analysis.

In Figure 2, the two vertical dashed lines separate the number of transactions into three ranges representing different use case complexity levels defined in the original UCPs method, and the weights for the different numbers of transactions in each range represent the experts' opinions on the effect a use case complexity level has on software size.

Based on the weighting schemes, we propose to calculate the prior information as follows:

(1) For each use case complexity level (l), an expert (i) may have different ratings ($r_t(i)$) about the effects that use cases, with different numbers of transactions ($t \in l$), have on software size.

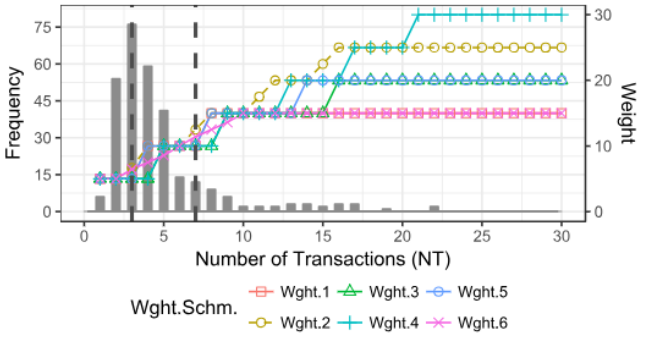

Figure 2: The UCPs complexity weight distribution with respect to NT.

(2) The mean value ($r_l(i)$) of the ratings provided by an expert for a use case complexity level (l) represents the expert's estimate of the weight ($w_l(i)$) that should be assigned to l.

(3) $w_l(i)$ is calculated as the weighted average of $r_t(i)$ where $t \in l$. The weights are decided by the probability ($Pr(t)$) for a use case having t transactions. $w_l(i)$ is formally defined by Eq (5).

$$w_l(i) = r_l(i) = \sum_{t \in l} r_t(i) * Pr(t) \tag{5}$$

(4) The probability ($Pr(t)$) is approximated by the relative frequency of use cases having t transactions with respect to the total number of use cases of the complexity level (l) where $t \in l$. The frequency distribution (f) of use cases with respect to the number of transations (NT) is plotted in Figure 2, which is based on a sample of 34 historial use case driven projects. The details of this dataset (D1) are introduced in Section 4.2.

$$Pr(t) = \frac{f(t)}{\sum_{k \in l} f(k)} \tag{6}$$

(5) After we calculate experts' estimates of the weights ($w_l(i)$) applied to each use case complexity level, the mean value (w_l) and variance (δ_l^2) are calculated over $w_l(i)$, where $l \in L$ and L is the complexity levels defined in the original UCPs method. w_l amd δ_l^2 are calculated by Eq. (7) and Eq. (8), where N is the total number of experts who provide estimates.

$$w_l = \frac{1}{N} * \sum_{i}^{N} w_l(i) \tag{7}$$

$$\delta_l^2 = \frac{1}{N} * \sum_{i}^{N} (w_l(i) - w_l)^2 \tag{8}$$

Following the process, we calculated the weights w_{a-pri} and their variances δ^2_{a-pri} for the three use case complexity levels based on the 6 weighting schemes plotted in Figure 2. The results are displayed in Table 3, which are used as the prior information for the Bayesian analysis. To test the normality of the prior distributions, we calculated the skewness (λ) and kurtosis (κ) for each complexity level (presented in Table 4) to evaluate the symmetry and tailedness,

Table 3: Different proposals of the weights for the UC complexity levels

Weight. Schm.	UC$_{simple}$	UC$_{average}$	UC$_{complex}$
1	5	10	15
2	5.93	10.24	18.95
3	5	7.66	15
4	5	7.66	16.84
5	5	10	16.84
6	5.81	8.49	14.19
w_{a-pri}	5.29	9.00	16.14
δ^2_{a-pri}	0.20	1.48	3.06
λ	0.52	-0.15	0.73
κ	2.08	1.20	1.56

and we observed that the distributions generally fit the bell curve. For this research, we assume the prior is normally distributed.

Our calculation of prior information relies on the distribution of use cases over different numbers of transactions(NT), which forms informative priors. However, to simplify the calculation, uninformative priors can be achieved by assuming that the frequency distribution of the use cases over NT is uniform, such that the expert's estimate of the effect a complexity level has on the software size can be calculated as the average of the ratings for the numbers of transactions within that use case complexity level. In this case, no empirical distribution of the use cases with respect to NT is needed. Therefore, the calculation can be simplified as Eq. (9), where $|l|$ represents the length of the NT range a complexity level l covers.

$$w_l(i) = \frac{1}{|l|} * \sum_{t \in l} r_t(i) \tag{9}$$

3.3 The sample information

Three steps were proposed to calculate the sample information from an empirical dataset. The sample information includes the weights ($w_{reg} = \{w_1^*, w_2^*, w_3^*\}$) and their variances ($\delta^2_{reg} = \{\delta_1^{*2}, \delta_2^{*2}, \delta_3^{*2}\}$) for the three use case complexity levels. The steps are as follows:

(1) Follow the normal UCPs counting process to calculate the UCPs for each project. Specifically, we evaluate the numbers of simple use cases (UC_{sample}), average use cases ($UC_{average}$), and complex use cases ($UC_{complex}$) to calculate the Unadjusted Use Case Weight ($UUCW$); the numbers of simple actors (Act_{simple}), average actors ($Act_{Average}$), and complex actors ($Act_{Complex}$) to calculate the Unadjusted Actor Weight (UAW); rated the environmental factors (EF) and technical complexity factors (TCF). Using all these numbers, we calculate the UCPs for each project by Eq. (10). An example of the counting results is presented in [26], which is also one of the three datasets (D1) used to evaluate the performance of the Bayesian approach in Section 4.3.

$$UCP = (UUCW + UAW) * TCF * EF \tag{10}$$

In this step, we also record the number of transactions (NT) of each use case to generate the frequency distribution (Figure 2) of the use cases with respect to NT, which is used in the prior information calculation process (introduced in Section 3.2).

(2) Calculate the normalized project effort ($Effort_{norm}$) for each data point using Eq. (11). $Effort_{norm}$ is the expected effort under the normal conditions of EF and TCF.

$$Effort_{norm} = \frac{Effort_{real}}{EF * TCF} \tag{11}$$

(3) Apply linear regression of actual effort on UCPs to calibrate the productivity factor α using Eq. (12). The empirical productivity factor α represents the number of person-hours required to develop one unit of UCPs.

$$Effort_{real} = \alpha * UCP \tag{12}$$

$Effort_{norm}$ and α are then used to calculate $UUCW_{emp}$ by Eq. (13), which represents the empirically measured system size in terms of Unadjusted Use Case Weight (UUCW).

$$UUCW_{emp} = \frac{Effort_{norm}}{\alpha} - UAW \tag{13}$$

(4) Perform multiple linear regression of $UUCW_{emp}$ on UC_{simple}, $UC_{average}$, and $UC_{complexity}$ according to Eq. (14) to calibrate parameters $w_{reg} = \{w_1^*, w_2^*, w_3^*\}$. The parameters represent the contributions of the use case complexity levels to system size.

$$UUCW_{emp} = w_1^* * UC_{simple} + w_2^* * UC_{average} + w_3^* * UC_{complex} \tag{14}$$

The variances of the parameters ($\delta^2_{reg} = \{\delta_1^{*2}, \delta_2^{*2}, \delta_3^{*2}\}$) can be estimated using Eq. (15), where X is the design matrix and δ^2 represents the variance of the error term.

$$\delta^{*2} = \Delta^2 * (X^T X)^{-1} \tag{15}$$

Δ^2 can be estimated by mean squared error (s^2) with Eq. (16) [27], where e is the residuals for the sample data, n is the number of observations, and p is the number of parameters.

$$s^2 = \frac{e^T e}{n - p - 1} \tag{16}$$

In our expriments, we applied the above procedure to three datasets (D1, D2, and D3) to derive the sample information ($w_{reg} = \{w_1^*, w_2^*, w_3^*\}$ and $\delta^2_{reg} = \{\delta_1^{*2}, \delta_2^{*2}, \delta_3^{*2}\}$) from each dataset. The sample information was then used to update the prior information to calibrate the use case complexity weights using Bayesian analysis. The results from the normality tests on the sample information are reported based on the skewness (λ) and kurtosis (κ) of the residuals of the multiple linear regression analyses.

3.4 The Bayesian approach of combining prior and sample information

We updated the prior information using the sample information by taking the weighted averages of the estimates w_{a-pri} from the experts and the empirically calibrated weights w_{reg}. The weights used in the averaging process are based on the precisions of the estimates, which are calculated as the inverses of the variances of the estimates: δ^2_{a-pri} and δ^2_{reg}. The Bayesian averaged estimates of the weights $w_{bayes} = \{\hat{w}_1, \hat{w}_2, \hat{w}_3\}$ for the different use case complexity levels and their variances $\delta^2_{bayes} = \{\hat{\delta_1}^2, \hat{\delta_2}^2, \hat{\delta_3}^2\}$ are calculated using Eq. (17) and Eq. (18), where $H_{a-pri} = \frac{1}{\delta^2_{a-pri}}$ and $H_{reg} = \frac{1}{\delta^2_{reg}}$.

$$w_{bayes} = (H_{a-pri} + H_{reg})^{-1} * (H_{a-pri} * w_{a-pri} + H_{reg} * w_{reg}) \tag{17}$$

$$\delta^2_{bayes} = (H_{a-pri} + H_{reg})^{-1} \tag{18}$$

This process was applied to D1, D2, and D3 to derive three sets of w_{bayes} and δ^2_{bayes} to evaluate the performance of the Bayesian estimates of use case complexity weights.

4 EMPIRICAL STUDY

4.1 Research Questions

To systematically evaluate our approach of calibrating use case complexity weights using Bayesian analysis, we set three research questions:

(1) **RQ1**: What influences does the Bayesian approach have on the use case complexity weights, in comparison with the weights suggested by experts and the weights calibrated by data?
(2) **RQ2**: Can the Bayesian approach improve effort estimation accuracy in comparison with other typical software size estimators? If so, how much?
(3) **RQ3**: How do the Bayesian approach and other size estimators perform in the model calibration situations having different sample sizes and homogeneou/heterogeneous datasets.

4.2 Datasets

The first dataset is composed of 34 data points collected from master-level computer science student projects at USC’s systems and software engineering center during 2014-2016, which lasted for 4-8 months. A wide range of software products were developed: web applications, mobile applications, mobile games, information systems, and scientific tools, which yielded source code from 1-10 KSLOC. 5-8 people collaborated on the projects by taking specific roles, including project manager, designer, architect, quality focal point, developer, and tester. All the projects followed formal software development methods, including use case driven, design-driven, risk-driven, plan-based, and agile methodologies. The requirements were given by real-world clients from start-ups, non-profits, education institutes, government agencies, etc., and the clients were closely involved in the engineering activities throughout the entire lifecycle. The products were tested and evaluated before their acceptance. Project effort was recorded through Jira tickets and weekly effort reports. The counting results for the factors used in calibrating use case complexity weights are presented in [26]. We call this dataset D1.

The second dataset (D2) is a Use Case Points benchmark dataset published by Radek Silhavy in the PROMISE Repository[28]. This dataset consists of 71 data points collected from three software houses and has been used by the authors in the research of selecting regression models for size estimation based on UCPs[29].

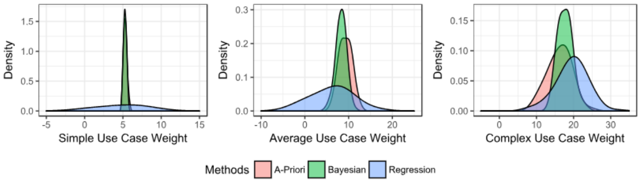

Figure 3: An example of updating UC complexity weights using Bayesian analysis.

Table 4: The sample information and the Bayesian estimates of the use case complexity weights

		D1		D2		D3	
Complexity	Estr.	w	δ^2	w	δ^2	w	δ^2
Simple	reg.	4.84	3.33	-0.71	52.13	-4.55	10.11
	bayes.	5.26	0.19	5.27	0.20	5.10	0.20
Average	reg.	6.41	4.89	9.46	8.36	12.49	6.08
	bayes.	8.41	1.13	9.08	1.26	9.69	1.19
Complex	reg.	19.15	4.71	19.56	9.42	20.22	7.06
	bayes.	17.32	1.85	16.97	2.31	17.37	2.13
		λ	κ	λ	κ	λ	κ
		1.98	7.69	-1.48	5.41	0.70	3.78

The projects are from different business sectors of software development, including manufacturing, banking, and communication. The software products were developed in 3rd generation programming languages, including java, C#, C++, etc., and were categorized into business applications, real-time applications, mathematically-intensive applications, etc. Different methodologies, for example, waterfall, personal software process, rapid application development, etc. were used in the development of the software products.

Since the projects from D1 and D2 were developed in noticeably different development environments according to their descriptions, we experimented our approach on these two datasets separately in order to evaluate the performance of our approach in the specific software development environments. A combination of D1 and D2, on the other hand, provides a more general software development context - a mixed environment - to test the robustness of the Bayesian approach of calibrating use case weights. We call this composite dataset D3.

4.3 Evaluation Methods and Results

The evaluation of the Bayesian approach of calibrating use case complexity weights was separated into two parts: calibrating the use case complexity weights, and assessing effort estimation accuracy. Specifically, we calibrated the weights based on the three datasets to understand how much each level of use cae complexity contributes to software size. The analysis of the calibration results answer **RQ1**. After that, we applied 10-fold cross validation on the empirical datasets to estimate the out-of-sample effort estimation accuracy of the Bayesian size estimator, and compared the accuracy measurements with other size estimators to analyze effectivness of the Bayesian approach (**RQ2**). We also discuss the interconnection

Table 5: The accuracy estimates of the Bayesian (B.), A-Priori (A.), Original (O.), and Regression (R.) estimators by 10-fold cross validation

D.	Estr.	M.	Std.	P.(.15)	Std.	P.(.25)	Std.	P.(.50)	Std.
D1	B.	0.714	0.196	0.442	0.106	0.492	0.101	0.758	0.075
	A.	0.882	0.167	0.308	0.100	0.358	0.095	0.475	0.086
	O.	0.883	0.164	0.308	0.100	0.333	0.095	0.558	0.068
	R.	0.892	0.177	0.217	0.085	0.333	0.113	0.475	0.086
D2	B.	0.209	0.021	0.418	0.071	0.618	0.057	0.957	0.022
	A.	0.275	0.022	0.296	0.045	0.488	0.078	0.888	0.047
	O.	0.276	0.022	0.268	0.054	0.502	0.081	0.888	0.047
	R.	0.285	0.02	0.307	0.067	0.521	0.056	0.873	0.045
D3	B.	1.203	0.218	0.235	0.057	0.407	0.067	0.666	0.054
	A.	1.819	0.731	0.187	0.029	0.273	0.040	0.617	0.038
	O.	1.795	0.714	0.197	0.024	0.284	0.039	0.627	0.039
	R.	1.209	0.293	0.265	0.040	0.369	0.049	0.646	0.057

between sample size, homegeneity of a data set, and applicability of the software size estimators based on our observations of the accuracy evaluation results (**RQ3**).

*4.3.1 Calibration of Use Case Weights (**RQ1**).* We applied the calibration processs introduced in Section 3.3 on the three datasets to calibrate the weights w_{reg} and their variances δ^2_{reg} for the use case complexity levels, which are used as the sample information. After that, we updated the prior information w_{a-pri} and δ^2_{a-pri} (calculated in Section 3.2 and presented in Table 3) with the sample estimates. The sample information and the results from Bayesian analysis for the three datasets are presented in Table 4. A graphical example (based on D1) of how the Bayesian approach combines the two pieces of information is presented in Figure 3 .

*Analysis of the calibrated weights (**RQ1**).* Based on the calibration results presented in Table 4, we summarize the properties of the Bayesian estimates as follows to answer **RQ1**:

(1) the Bayesian estimates of the weights for average use cases tend to be smaller than the estimate the original UCPs method, for example, by 15.9% for D1, 9.2% for D2, and 3.1% for D3, while the Bayesian estimates of the weights for complex use cases are generally larger than the estimate by the original UCPs method, for example, by 15.4% for D1, 13.1% for D2, and 15.8% for D3. This phenomenon implies that the influences from the different use case complexity levels toward project effort tend to be non-linearly increasing (for example, 1:1.6:3.3, based on D1), instead of the linear relationship - simple being 5, average being 10, and complex being 15 (1:2:3) - proposed by the original UCPs method. This phenomenon is similarly obeserved by COCOMO®II and formalized as the rule of "diseconomy of scale" in terms of SLOC [1].

(2) The Bayesian averaging approach corrects the counter-intuitive results from the sample estimates. For instance, the empirically calibrated weights for simple use cases are negative for both D2 and D3. Theoretically, this may be because of the small variances of the numbers of simple use cases in the empirical datasets [1][21], and this is demonstrated by the large variances of the sample estimates of simple use case weights (as shown in Table 4). For instance, the ratio between δ^2_{reg} : δ^2_{a-pri} is 261:1 for D2, and 51:1 for D3. The Bayesian approach corrects these counter-intuitive estimates for D2 and D3. Another potential solution to this problem is to adjust the classification rules of complexity levels to allow more use cases to be determined as simple. However, this is beyond the scope of this research, but an interesting direction for our future study.

(3) The variances of the Bayesian estimates of weights are smaller than both the experts' estimates and the sample estimates, which means the Bayesian estimates are more stable if the calibration process were applied to different datasets.

*4.3.2 Evaluation of Effort Estimation Accuracy (**RQ2** and **RQ3**).* *Accuracy Measures.* To evaluate the effectiveness of the Bayesian method in determining use case complexity weights, we evaluate the effort estimation accuracy in terms of MMRE and PRED, which are the commonly used accuracy measures in software engineering [1] [30]. Both MMRE and PRED rely on the quantity called magnitude of relative error (MRE), which is defined by Eq. (19).

$$MRE_i = \frac{|y_i - \hat{y_i}|}{y_i} \tag{19}$$

MMRE measures the sample mean of MRE, while PRED(x) measures the percentage of MRE within x. MMRE and PRED(x) can be calculated using Eq. (20) and Eq. (21) respectively. Low values of MMRE and high values of PRED(x) are desirable. These statistics give cost estimation practitioners the ability to state how often estimates can be expected to be within an acceptable margin of error.

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \tag{20}$$

$$PRED(x) = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1, & \text{if } MRE_i \le x \\ 0, & \text{otherwise}\end{cases} \tag{21}$$

Since MMRE and PRED(.25) are the most frequently used accuracy criteria [30], we emphasize these two accuracy metrics in the evaluation of the performance of the software size estimators. As pointed out in [30], there is no standard value of x for PRED(x) to be used for accuracy evaluation, and also we observed in our experiments that a model may perform better than another model in terms of PRED(.25) while performing worse than the same model for PRED(.30)). We propose to evaluate PRED(.01) to PRED(0.50) to comprehensively monitor the performance of the sizing estimators. Therefore, in addition to MMRE and PRED(.25), we also use the average of the differences in the values from PRED(0.01) to PRED(0.50) (Eq. 22) to certify which model performs better.

$$AVG\_PRED\_IMP = \frac{1}{50}\sum_{x=0.01}^{0.50}(PRED_1(x) - PRED_2(x)) \tag{22}$$

Out-of-sample Accuracy Assessment. Four size estimators were compared in terms of the out-of-sample effort prediction accuracy using 10-fold cross validation. The size estimators were the Bayesian approach of estimating the weights, the method of collecting prior

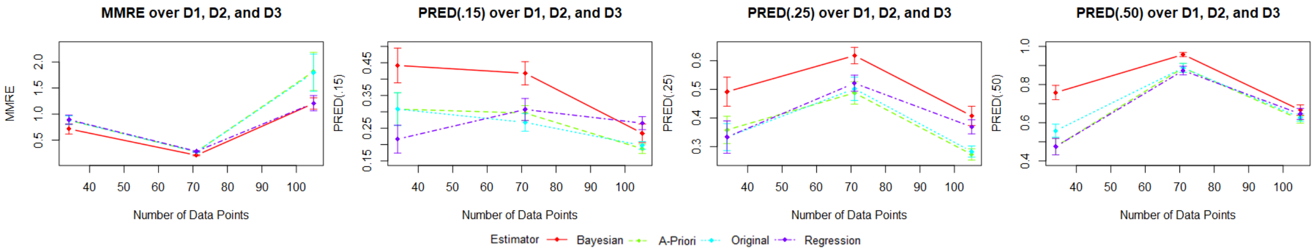

Figure 4: Accuracy evaluation of the size estimators over the datasets (D1, D2, and D3).

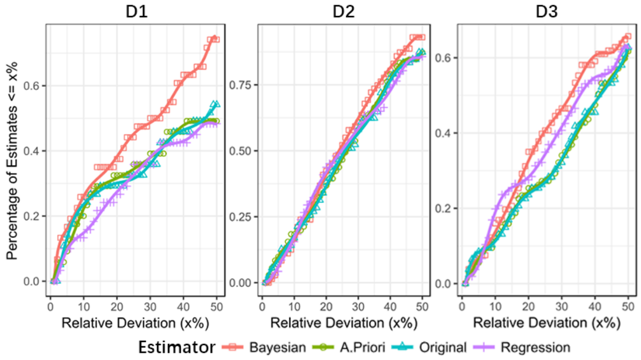

Figure 5: The evaluation of PRED(0.01) - PRED(0.50) for the four size estimators over the three datasets

Table 6: Relative improvements measured by the averages of the differences over PRED(0.01) - PRED(0.50)

Dataset	Avg.P.Imp	Bayes.	A-Pri.	Orig.	Reg.
D1	Bayesian	0.000	0.099	0.103	0.131
	A-Priori	–	0.000	0.004	0.032
	Original	–	–	0.000	0.028
	Regression	–	–	–	0.000
D2	Bayesian	0.000	0.030	0.034	0.028
	A-Priori	–	0.000	0.004	-0.002
	Original	–	–	0.000	-0.006
	Regression	–	–	–	0.000
D3	Bayesian	0.000	0.077	0.076	0.026
	A-Priori	–	0.000	-0.002	-0.052
	Original	–	–	0.000	-0.050
	Regression	–	–	–	0.000

information by synthesizing experts' estimates (introduced in Section 3.2), the original UCPs method, and the method empirically calibrate sample information using linear regression, which are called Bayesian, A-Priori, Original, Regression estimators respectively in the following sections for simplicity. They can be categorized into two kinds: expert-based estimators, which include the Bayesian, A-Priori, Original estimators, and data-driven estimators, which include the Bayesian and Regression estimators - the Bayesian approach belongs to both the categories for it combines the prior and sample information. The produced weights are then plugged in the same general form described by Eq. 23, and then used as the software size metrics for project effort estimation.

$$UUCW = w_1 * UC_{simple} + w_2 * UC_{average} + w_3 * UC_{complex} \tag{23}$$

Cross validation is a technique to test a prediction model on an independent dataset to better assess the performance of the model on new observations[20]. Specifically, each of three datasets was separated into 10 folds, and 10 runs of training and testing were applied to evaluate the effort estimation accuracy with the chosen metrics: MMRE, PRED(.15), PRED(.25), and PRED(.50). The averages of the values of MMRE, PRED(.15), PRED(.25), and PRED(.50) across the 10 runs were used as the final estimation accuracy indicators. The standard deviations of accuracy measurements were calculated and used to evaluate the statistical significance of the estimation accuracy improvements.

In each iteration of model training and testing, 90% of the total dataset (31 data points for D1, 64 data points for D2, and 95 data points for D3) were used as the training set to calibrate the weights (specifically, w_{reg} and w_{bayes}) using the Regression and Bayesian size estimators. w_{a-pri} and w_{org} (calibrated by the A-Priori and Original size estimators) remain the same for the 10 runs of training. The remaining 10% of the data points (3 data points for D1, 7 data points for D2, and 10 data points for D3) are used as the testing set to evaluate the estimation accuracy. $effort_{estimate}$ is calculated using Eq. (24), which is then compared with actual project effort ($effort_{actual}$) to calculate the chosen accuracy measures.

$$\begin{aligned} Effort_{estimate} = \alpha * (w_1 * UC_{simple} + w_2 * UC_{average} + \\ w_3 * UC_{complex} + UAW) * TCF * EF \end{aligned} \tag{24}$$

The testing results - the accuracy measurements and their standard deviations - are presented in Table 5 and Figure 4. The standard deviations are plotted as the virtical lines through the accuracy measurements in Figure 4. A more comprehensive evaluation of the prediction accuracy is presented in Figure 5, which plots PRED(.01) - PRED(0.50) for the four software size estimators over the three datasets. Also the results from assessing the averaged PRED improvements (AVG_PRED_IMP) are presented in Table 6.

*4.3.3 Comparison of the size estimators (**RQ2**).* In the comparison between the four software size estimators, the effort estimation accuracy of the Bayesian method consistently outperforms other size estimators for all the three datasets. For instance, as presented in Table 5, the Bayesian method outperforms the A-Priori, Original, and Regression methods by 13.4%, 15.9%, and 15.9% respectively for PRED(.25), and by 19.0%, 19.1%, and 20.0% respectively for MMRE, with respect to D1. We can also observe the improvements of effort estimation accuracy made by the Bayesian method with respect to D2 and D3. For instance, the Bayesian method outperforms the Original method by 24.3% for D2 and 33.0% for D3 in terms of MMRE, and by 11.6% for D2 and by 12.4% for PRED(.25) for D3. The improvements in terms of MMRE and PRED(.25) can also be found when comparing the Bayesian method with the A-Priori and Regression methods over D2 and D3.

Among the comparisons between the Bayesian estimator and other estimators over the three datasets, we found that the improvements brought by Bayesian analysis, in terms of PRED(2.5) for D1, MMRE for D2, and PRED(.25) for D3, are statistically significant according to the two-sample Student's t tests at significant levels asjuted by Benjamini-Hochberg procedure false discovery rate of 25% **??**. However, due to the small datasets that are available for the significance tests, further testing for the significance of the improvements is needed.

The plot from PRED(.01) to PRED(.50) in Figure 5 also certifies the significant improvements over other size estimators. Based on these observations, we conclude that the weights decided by Bayesian analysis performs better than the weights decided using only expert judgment or purely data driven method across the software engineering situations that the datasets represent.

*4.3.4 Interconnection between sample size, homogeneity of a dataset, and performance of size estimators (**RQ3**).* In addition to the proof about effectivenss of the Bayesian analysis in measuring software size, we also made a few observations about how sample size and homogeneity of a dataset affect effort estimation accuracy based on the accuracy evaluation results. We believe those observations provide the effective guidelines for selecting appropriate size estimation methods in the typical size modeling situations.

Stratification improves effort estimation accuracy. In the mixed environment (D3), all the four size estimators decrease in effort estimation accuracy significantly. For instance, as presented in Table 5, by comparing D1 and D2 with D3 respectively, the Bayesian method decreases by 68.5% and 475.6% in terms of MMRE, and 8.5% and 21.1% in terms of PRED(.25). This deterioration is especially noticeable from Figure 4 when we compare the accuracy measurements between D2 and D3, since D2 and D3 have close numbers of data points (71 and 105 respectively), but significant decreases in the estimation accuracy measurements. Similar deteriorations can be found in the performance of other size estimators in Table 5. This phenomenon strongly suggests that, in order to achieve better effort estimation accuracy, stratifying a dataset based on their inherent properties, for example, by teams, organizations, periods of time, or development environments is preferred, when developing and calibrating size metrics for effort estimation.

For small and homogeneous datasets, expert-based size metrics outperform the size metric decided by linear regression. As shown in Table 5 and Figure 4, the A-Priori and Original methods slightly outperfom the Regression method by 1.2% and 1.1% respectively for MMRE for D1. This trend can also be found for other accuracy metrics: PRED(.15), PRED(.25), and PRED(.50) from Table 5 and Figure 4. The accuracy improvements measured by AVG_PRED_IMP shown in Table 6, also confirms this point in considering that the A-Priori and Original methods outperform the Regression method by 4.0% and 3.9% respectively for D1. This phenomenon suggests that the Regression method tends to overfit small training datasets such that it provides worse estimation accuracy on new data points, compared to the expert-based methods.

However, as more data points are added to D2 for model calibration and testing, the regression method starts to perform better. For instance, as shown in Figure 4, the Regression size estimator perfoms slightly better than both the A-Priori and the Original size estimators by 3.3% and 1.9% respectively for PRED(.25). We can also observe that the advantages of the expert-based size estimators over the Regression estimator narrow down in Figure 5, and the Regression estimator provides the same level of accuracy as other size estimators in terms of MMRE, PRED(.15), and PRED(.50) in Table 5.

Therefore, to summarize, we suggest to use expert-based methods for small datasets. Since, as an example of small datasets, D1 has 34:3 sample size to parameter number ratio, the datasets that have sample size to parameter number ratio less than 11:1 are regarded as small datasets.

For heterogeneous datasets, data driven methods are preferred. Extending our finding in the last section about increasing performance of the regression method for the larger dataset D2, we also found regression method is more applicable to the heterogenous environment in the comparison between D2 and D3. For instance, as shown in Table 5, the regression method outperforms both the A-Priori and Original methods significantly for D3, for example, by 7.8% and 6.8% for PRED(.25), and by 33.5% and 32.6% for MMRE. In comparison to the slight improvements brought by the A-Priori and Original methods over D2, the increasing margins of improvement can also be observed for PRED(.15) and PRED(.50) from Table 5. The improvements measured by AVG_PRED_IMP - the Regression method is 2.8% and 2.6% better than the A-Priori method and the Original method - also demonstrate this point. Figure 5 and Figure 4 summarize this trend of improvements.

The advantage of data-driven methods over expert-based methods for D3 can be explained as, for heterogeneous environments, the effects a use case complexity level has on effort vary. For example, more strict environments may require the implemented functions (for realizing use cases) be more tested, while less strict environments may require less testing effort. Data-driven methods are sensitive to the conflicts due to its ability in finding a mean value to cover the different situations. However, the expert-based estimates of weights may be biased in the conflicted situations.

5 THREATS TO VALIDITY

In this section, we discuss the threats to the vadility of our proposed method and the experimental results, and also the possible ways to mitigate the threats.

Internal Validity. As mentioned in the model calibration process (Section 3.2 and Section 3.3), both the processes of gathering the prior information and the sample information rely on the properties of the empirical datasets. Specifically, the prior information relies on the use case distribution with respect to NT, while the sample information is calculated by applying multiple linear regression on the numbers of use cases of different complexity levels. Therefore, certain degree of variation may exist in the weight calibration results as well as the estimation accuracy measurements presented in our empirical study if different datasets were used. Local calibration is encouraged when an empirical dataset is available for a specific software development environment. Also the analytical steps relies on the assumptions of normal distributiosn of the prior and sample information, any considerable deviation from the assumptions should consider a full treatment of Bayesian analysis. Besides, as mentioned in the paper, due to the sample datasets of the accuracy measurements using 10 fold cross validation, further testing for significance of the improvements is needed.

External Validity. Some aspects of this research may also limit generalizability of the results. As mentioned in Section 4.2, the projects of D1 are considered small to medium projects, as the sizes range from 1-10 KSLOC and were done with 5-8 team members for D1. Also we don't know exactly how big the projects are in D2, in terms of the personnel and source lines of code, even though D2 covers a wide range of product types and business sectors. Therefore, the results presented in this paper may not be directly applicable to larger projects (>= 10KSLOC). More data points from large projects are desirable to further test the performance of the method.

6 CONCLUSIONS

In this paper, we used the Bayesian approach of combining prior information (complexity weights suggested previously by experts) and the sample information (complexity weights calibrated from data) to better calibrate use case complexity weights. To derive the prior information, we did a systematic review of previously published papers to summarize different proposals of the effects that use case complexity levels should have on software size and proposed the method to synthesize the different proposals. We introduced the method to derive the sample information from the empirical datasets and also the Bayesian approach of updating the prior information using the derived sample information. To validate the effectiveness of the Bayesian approach in adjusting use case complexity weights, we evaluated the effort estimation accuracy of the Bayesian approach with an empirical study of 105 projects and compared it with the A-Priori, Original UCPs, and Regression approaches (**RQ1** and **RQ2**). The results have shown that, in addition to the benefits of correcting counter-intuitive calibration results and increasing the stability of the estimates, the improvements made by Bayesian approach in effort estimation accuracy, compared to other size estimators, are statistically significant. Based on the analyses of the evaluation results, we futher provided suggestions to effectively select software size calibration methods in the typical software size calibration situations (**RQ3**).

Future directions include collecting more data points that are representative for wider ranges of software development sitations, especially, for large systems and engineering teams, and updating the calibrated results for more general use. As more data points are available, we would like to further evaluate the significance of the superiority of the Bayesian approach, and also extend the study to calibrating actor complexity weights. To alleviate the assumptions of normality of prior and sample information by the analytical method, Markov chain Monte Carlo (MCMC) can be an effective alternative to simulate the posterior distribution by simpling from prior distribution, so as to provide full access to posterior information. To further testing the significance of the improvements in terms of estimation accuracy, collecting more data points and using Bootstrapping to simulate the distribution of accuracy measurements should be csondered. As suggested in the paper, the underlying structure of classifying use cases also needs to be reconsidered or adjusted to be better adapted to modern use case driven projects, which woud be another interesting extension of this research.

REFERENCES

[1] B. W. Boehm, R. Madachy, and B. Steece, *Software cost estimation with Cocomo II.* Upper Saddle River, NJ: Prentice Hall, 2000.

[2] B. W. Boehm, *Software engineering economics.* Englewood Cliffs, N.J: Prentice-Hall, 1981.

[3] K. Qi and B. W. Boehm, "A light-weight incremental effort estimation model for use case driven projects," in *Software Technology Conference (STC), 2017 IEEE 28th Annual.* IEEE, 2017, pp. 1–8.

[4] B. Boehm, C. Abts, and S. Chulani, "Software development cost estimation approachesâĂŤa survey," *Annals of software engineering*, vol. 10, no. 1-4, pp. 177–205, 2000.

[5] G. Karner, "Metrics for objectory," *Sweden: University of Linköping*, 1993.

[6] M. W. Kamal and M. A. Ahmed, "A proposed framework for use case based effort estimation using fuzzy logic: building upon the outcomes of a systematic literature review," *International Journal of New Computer Architectures and their Applications (IJNCAA)*, vol. 1, no. 4, pp. 953–976, 2011.

[7] 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 *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th.* IEEE, 2008, pp. 5–B.

[8] M. Usman, E. Mendes, F. Weidt, and R. Britto, "Effort estimation in agile software development: a systematic literature review," in *Proceedings of the 10th International Conference on Predictive Models in Software Engineering.* ACM, 2014, pp. 82–91.

[9] G. Schneider and J. P. Winters, *Applying use cases: a practical guide.* Pearson Education, 2001.

[10] R. Agarwal, S. Banerjee, and B. Gosh, "Estimating internet based projects: A case study," in *Proceedings of the Quality Week 2001 Conference, San Francisco*, vol. 29, 2001.

[11] B. Anda, E. Angelvik, and K. Ribu, "Improving estimation practices by applying use case models," in *PROFES.* Springer, 2002, pp. 383–397.

[12] B. Anda, H. Dreiem, D. Sjøberg, and M. Jørgensen, "Estimating software development effort based on use casesâĂŤexperiences from industry," *«UML» 2001âĂŤThe Unified Modeling Language. Modeling Languages, Concepts, and Tools*, pp. 487–502, 2001.

[13] A. B. Nassif, L. F. Capretz, and D. Ho, "Enhancing use case points estimation method using soft computing techniques," *arXiv preprint arXiv:1612.01078*, 2016.

[14] A. O'Hagan and J. J. Forster, *Kendall's advanced theory of statistics, volume 2B: Bayesian inference.* Arnold, 2004, vol. 2.

[15] G. Karner, "Resource estimation for objectory projects," *Objective Systems SF AB*, vol. 17, 1993.

[16] K. Periyasamy and A. Ghode, "Cost estimation using extended use case point (e-ucp) model," in *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on.* IEEE, 2009, pp. 1–5.

[17] M. M. Kirmani and A. Wahid, "Revised use case point (re-ucp) model for software effort estimation," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 3, pp. 65–71, 2015.

[18] A. Bou Nassif, "Software size and effort estimation from use case diagrams using regression and soft computing models," 2012.

[19] F. Wang, X. Yang, X. Zhu, and L. Chen, "Extended use case points method for software cost estimation," 2009, pp. 1–5.

[20] A. B. Nassif, L. F. Capretz, and D. Ho, "Calibrating use case points," in *Companion Proceedings of the 36th International Conference on Software Engineering.* ACM, 2014, pp. 612–613.

[21] S. Chulani, B. Boehm, and B. Steece, "Bayesian analysis of empirical software engineering cost models," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 573–583, 1999.

[22] M. de Freitas Junior, M. Fantinato, and V. Sun, "Improvements to the function point analysis method: A systematic literature review," *IEEE Transactions on*

Engineering Management, vol. 62, no. 4, pp. 495–506, 2015.

[23] K. Periyasamy and A. Ghode, "Cost estimation using extended use case point (e-ucp) model," 2009, pp. 1–5.

[24] F. Wang, X. Yang, X. Zhu, and L. Chen, "Extended use case points method for software cost estimation," in *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on.* IEEE, 2009, pp. 1–5.

[25] A. Minkiewicz, "Use case sizing," 2015.

[26] K. Qi, A. Hira, E. Venson, and B. Boehm, "Use Case Points Complexity Weights Assessment Dataset (D1)," https://figshare.com/s/21d5ea9c5a142133e4dc, 2018, [Online; 19-May-2018; private-shared].

[27] T. Hastie, R. Tibshirani, and J. H. J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction.* New York: Springer, 2001.

[28] R. Silhavy, P. Silhavy, and Z. Prokopova, "Use case points benchmark dataset," Mar. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.344959

[29] ——, "Analysis and selection of a regression model for the use case points method using a stepwise approach," *Journal of Systems and Software*, vol. 125, pp. 1–14, 2017.

[30] D. Nguyen and T. M. WVU, "Studies of confidence in software cost estimation research based on the criterions mmre and pred," 2009.