# Calibrating Use Case Points Using Bayesian Analysis


## ABSTRACT

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. 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. Bayesian analysis integrates 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. We empirically validated the effectiveness of this calibration approach in effort estimation with 105 use case driven projects. 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 empirically calibrated weights, and the weights suggested by experts.



## KEYWORDS

effort estimation, software sizing models, software size metrics, use case analysis, use case points, function points, statistical model calibration, local calibration, Bayesian analysis, use case driven process, project management

**ACM Reference Format:**

. 2018. Calibrating Use Case Points Using Bayesian Analysis. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn


## 1 INTRODUCTION

Effort estimation has been regarded as a crucial driver for various software managerial decisions. For example, accurate estimation of project effort helps avoid the risks of being over schedule or budget, and effectively allocate project resources and make project plans[1][2]. 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, however, during which time, very little information about the project or system is known. Use Case Points (UCPs) provides a functional size metric based on use cases, which can be measured early in a project. Due to its early applicability during the software development lifecycle, UCPs has gained wide acceptance [3]. While practitioners and research papers have reported the effectiveness of UCPs [4–9], the UCPs method has also been criticized because the complexity weights are not validated with data [10] [3] - Karner set the complexity weights based on his domain knowledge gained from Objectory Systems [11]. 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, the authors of this paper use Bayesian analysis to update the experts' estimates of use case complexity weights with empirically calibrated results with the goal of improving effort estimaiton accuracy with UCPs. To validate the effectiveness of our approach, we empirically evaluated the out-of-sample effort estimation accuracy on 105 historical projects. The results showed that the Bayesian estimates of the weights consistently outperform the weights proposed by original UCPs method, the weights proposed by the domain experts, and the purely empirically calibrated weights. In addition to the proof of the improvement in effort estimation accuracy, we also make a few interesting observations based on our empirical study, which we think can be used as effective guidelines to select appropriate size estimation methods for 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 instroduces the background of this research in terms of the underlying theories and methods. Section 4 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 5. Lastly, we discuss the threats to validity of the results in Section 6 and make the conclusions of the empirical analysis in Section 7.

## 2 RELATED WORK

When Karner first proposed UCPs method in 1993, Karner explained how the complexity weights were set by stating: "the weights in this article are a first approximation by people at Objective Systems."[12] At that point, Karner also pointed out that more data were 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 the estimation accuracy [10].

To tackle the 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 is based on the addition of complexity levels, the second group is based on the discretization of the complexity levels, and the third group is based on the calibration of the use case and the actor complexity weights.

*Adding 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 point method [13]). Re-UCP adds one extra rating level - "critical"- for both the use case and actor weighting schemes [14]. They conducted an experiment with 51




*Conference'17, July 2017, Washington, DC, USA*





ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

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 [15].

*Discretizing compelxity levels.* Using fuzzy logic, Wang et al.[16] and Nassif[15] suggested discretizing the levels of complexity into more granular options and assigning corresponding weights to differentiate their effect 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 [16]. 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 [10].

*Emprically calibrating complexity weights.* In another paper, Nassif proposed to empirically calibrate the weights assigned to the different use case complexity levels using neural networks [17]. However, specific experiment results or details of the approach were not found. Other than this research, no related research about emprically calibrating use case complexity weights were found to the best effort of the authors.

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 this research in the UCPs method, by demonstrating how the use case complexity weights can be empirically calibrated. We demonstrated that it is possible to improve the accuracy of UCPs-based software effort estimation if we update the weights proposed by domain experts by the weights calibrated by data using Bayesian analysis.

Bayesian Analysis, on the other hand, has been used in building the COCOMO®II effort estimation models to combine domain experience and empirical study results [1][18]. 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 is however 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 [18]. 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 emprical data, and combinting the two pieces of information - prior and sample information - to achieve better effort estimation accuracy using UCPs.

**Table 1: The counting process of UCPs**

| 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 ($UCP$) : $UCP = (UUCW + UAW) * TCF * EF$ | $UCP$ |

## 3 BACKGROUND

### 3.1 Use Case Points (UCPs)

Karner developed Use Case Points (UCPs) to estimate the effort for objected-oriented projects using the use case technique of gathering and understanding requirements [11]. TABLE 1 summarizes the steps and rules to calculate UCPs. Use cases are 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 calculate the Unadjusted Use Case Weight (UUCW). The different weights represent their different effects on the 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. Unadjusted Use Case Weight (UAW) is similarly defined. Since UUCW contributes most of 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 problem of overfitting. However, we maintain that the proposed approach is also applicable to calibrating actor complexity weights for calculating UAW.

### 3.2 Bayesian Analysis

Different from the maximum likelihood method of estimating parameters, the Bayesian approach minimizes the posterior expected loss, which maintains many theoratical and practical advantages[19]. For instance, it provides a solid theoratical framework to combine prior information with data. The posterior probability ($P(\theta|X)$) of a hypothesis is derived by updating the prior probability ($P(\theta)$) as more evidence becomes available, which is defined as a likelihood function ($P(X|\theta)$) derived from a statistical model of the observed data. 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).

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

In the context of linear regression, the posterior mean can be computed as the weighted average of the a priori mean and the sample mean using Eq. (3)[20]. 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 $\theta$. $\theta^{*}$ and $\theta^{**}$ are means of the prior and sample information, and $\alpha$ and $\beta$ are the inverse of their variances called precision. 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 drived from statistical analysis of data. In this paper, the use case complexity weights proposed in prevously published papers are used as the prior information. The sample information consists of the use case weights determined by running multiple linear regression (MLR) on the dataset of 105 historical projects.

## 4 THE BAYESIAN APPROACH TO CALIBRATE USE CASE WEIGHTS

### 4.1 The calibration process

As depicted in Figure 1, our approach of combining the prior information and the sample information of use case complexity weights using Bayesian analysis generally goes through the following 4 steps:

(1) Calculate the means and variances based on the complexity weights proposed by the experts, as the prior information, 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 4.2.

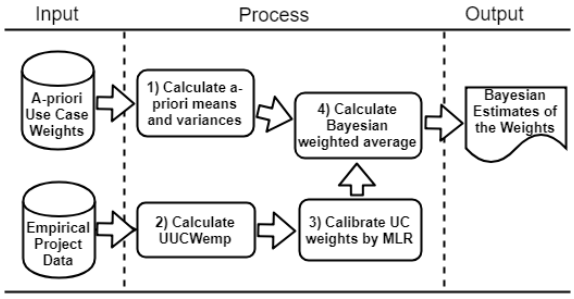


**Figure 1: Bayesian Averaged Weight Calibration**

(2) Calculate $UUCW_{emp}$ by having the project effort $Effort_{real}$ (person-hours) divided by $EF$, $TCF$, and $\alpha$ and minus $UAW$ based on an empriical dataset. $UUCW_{emp}$ is the measurements of software size in terms of $UUCW$, and $\alpha$ is the emprical productivity factor calibrated by running a linear regression of $Effort_{real}$ against UCPs. The details of calculating $UUCW_{emp}$ are in Section 4.3.

(3) 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 the 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 4.3.

(4) 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 $\delta^2_{bayes} = \{\hat{\delta_1}^2, \hat{\delta_2}^2, \hat{\delta_3}^2\}$. The weights used in the averaging process are based on the variances of the two sources of information. Combining the prior information and sample information is further explained in Section 4.4.

### 4.2 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 doing 2007 - 2017 at the 4 major research paper databases as suggested by [21]: Scopus, IEEE Xplore, ACM DL, and Science Direct, to search for the previous researches that proposed different use case complexity weights. Also references of the selected papers are examined.

6 papers were identified, which proposed new weighting schemes by different 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 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

**Table 2: The weighting schemes from previous researches**

| Wght. Schm. | Study | Year | Metric |
|---|---|---|---|
| 1 | Karner[11] | 1993 | UCP |
| 2 | Wang et al.[23] | 2009 | EUCP |
| 3 | Kirmani and Wahid[22] | 2009 | Re-UCP |
| 4 | Nassif [15] | 2012 | Soft-UCP |
| 5 | Minkiewicz[24] | 2015 | UCP Sizing |
| 6 | Nassif et al.[10] | 2016 | Enhanced UCP |

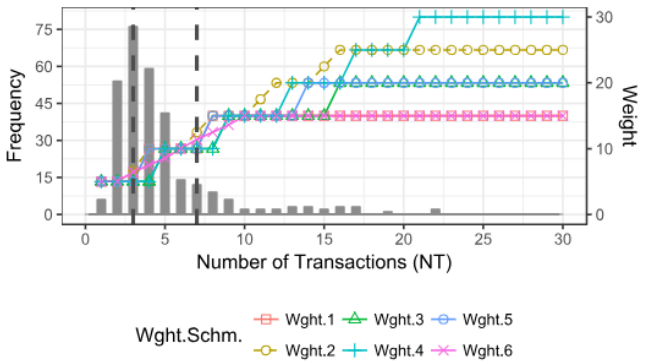


**Figure 2: Use Case Point Complexity Weight Distribution**

an extra level of complexity to cover the use cases with number of transactions being larger than 15 [22]. 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 [23]. The weighting schemes are plotted in Figure 2.

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 observations, we summarize the calculation process of 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.
(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).

**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 |
| $\delta^2_{a-pri}$ | 0.20 | 1.48 | 3.06 |

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

(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.

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

(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 ($\delta^2_l$) are calculated over $w_l(i)$, where $l \in L$ and $L$ is the complexity levels proposed by the original UCPs. $w_l$ amd $\delta^2_l$ are calculated by Eq. (7) and Eq. (8), where N is the total number of experts who provide estimates.

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

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

Following the process, we calculate the weights $w_{a-pri} = \{w_1, w_2, w_3\}$ and their variances $\delta^2_{a-pri} = \{\delta^2_1, \delta^2_2, \delta^2_3\}$ for the three use case complexity levels based on the 6 weighting schemes (Figure 2) identified from the previous studies. The results are displayed in TABLE 5, which are used as the prior information for the Bayesian analysis.

To simplify the calculation, one can also assume that the probability distribution of the use cases with respect to the number of internal transactions ($NT$) 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.

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

**Table 4: The Empirical Dataset (D1)**

| Proj. No. | Simple | Average | Complex | UAW | TCF | EF | PH | Proj. No. | S | A | C | UAW | TCF | EF | PH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 8 | 8 | 3 | 1 | 0.9 | 1.03 | 146 | **18** | 33 | 8 | 1 | 4 | 1.14 | 1.15 | 759 |
| **2** | 11 | 1 | 2 | 7 | 0.9 | 1.025 | 724 | **19** | 11 | 4 | 3 | 5 | 0.925 | 0.965 | 302.5 |
| **3** | 11 | 3 | 9 | 3 | 0.8 | 1.23 | 1125 | **20** | 9 | 2 | 5 | 16 | 0.935 | 0.935 | 1965 |
| **4** | 9 | 1 | 6 | 4 | 0.835 | 0.995 | 482 | **21** | 16 | 11 | 9 | 4 | 1.135 | 1.25 | 1392 |
| **5** | 8 | 2 | 17 | 2 | 0.925 | 1.04 | 657.5 | **22** | 10 | 13 | 3 | 2 | 0.89 | 1.025 | 804.5 |
| **6** | 12 | 6 | 3 | 2 | 0.925 | 0.995 | 263 | **23** | 6 | 15 | 13 | 3 | 0.94 | 0.875 | 1592 |
| **7** | 9 | 5 | 8 | 3 | 0.88 | 1.04 | 571 | **24** | 8 | 6 | 7 | 16 | 0.94 | 0.89 | 3113 |
| **8** | 12 | 33 | 2 | 1 | 0.96 | 1.03 | 547 | **25** | 3 | 4 | 15 | 1 | 1.175 | 1.31 | 737 |
| **9** | 15 | 6 | 3 | 1 | 0.915 | 1.025 | 268.5 | **26** | 2 | 4 | 12 | 3 | 1.25 | 1.025 | 1510.08 |
| **10** | 6 | 2 | 5 | 4 | 0.855 | 1.025 | 746 | **27** | 2 | 12 | 2 | 1 | 1.25 | 1.025 | 581.87 |
| **11** | 13 | 4 | 3 | 1 | 0.855 | 1.025 | 140.5 | **28** | 2 | 3 | 12 | 5 | 1.25 | 1.025 | 1560.53 |
| **12** | 6 | 4 | 3 | 4 | 0.795 | 0.92 | 1281 | **29** | 2 | 18 | 5 | 6 | 1.25 | 1.01375 | 3484 |
| **13** | 6 | 4 | 21 | 1 | 0.925 | 0.95 | 1482.5 | **30** | 5 | 4 | 22 | 3 | 1 | 1.0175 | 1561.38 |
| **14** | 6 | 12 | 5 | 6 | 1.04 | 1.12 | 1142 | **31** | 2 | 3 | 3 | 1 | 1.85 | 1.0025 | 784.4 |
| **15** | 16 | 16 | 6 | 2 | 0.805 | 1.025 | 617 | **32** | 2 | 12 | 14 | 28 | 1.25 | 1.030625 | 8094.34 |
| **16** | 9 | 5 | 2 | 13 | 0.885 | 0.89 | 1347 | **33** | 5 | 11 | 7 | 8 | 1.39 | 1.013 | 2810.95 |
| **17** | 13 | 2 | 9 | 27 | 1.12 | 1.325 | 3680 | **34** | 16 | 1 | 0 | 16 | 1.25 | 1.016 | 2130 |

## 4.3 The sample information

Three steps are required 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 Table 4, which is also one of the three datasets (D1) used to evaluate the performance of the Bayesian approach in Section 5.3. The details of this dataset are provided in Section 5.1.

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

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 calculation process.

(2) Apply linear regression of actual effort on UCPs to calibrate the productivity factor $\alpha$ using Eq. (11). The empirical productivity factor $\alpha$ represents the number of person-hours required to develop one unit of UCPs.

$$Effort_{real} = \alpha * UCP \tag{11}$$

$\alpha$ is then used to calculate the normalized effort ($Effort_{norm}$) for each data point using Eq. (12). $Effort_{norm}$ is the expected effort under the normal conditions of EF and TCF.

$$Effort_{norm} = \frac{Effort_{real}}{EF * TCF} \tag{12}$$

Using $Effort_{norm}$, $UUCW_{emp}$ is calculated by Eq. (13), which represents the empirically measured system size in terms of Unadjusted Use Case Weight (UUCW).

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

(3) Perform multiple linear regression of $UUCW_{emp}$ on $UC_{simple}$, $UC_{average}$, and $UC_{complexity}$ according to Eq. (14) to calibrate parameters $w_{reg} = \{w_1^*, w_2^*, w_3^*\}$, which 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 $s^2$ represents the sample estimate of the variance of the error term.

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

$s^2$ can be calculated by Eq. (16), where $e$ is the residuals for the sample data, $n$ is the number of observations, and $p$ is the number of parameters.

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

In our expriments, we apply the method to three datasets (D1, D2, and D3) in order to evaluate the performance of the Bayesian

approach of calibrating use case complexity weights. TABLE 5 provides the results of assessing the sample information from the three datasets, which includes the calibrated weights and their variances for the three levels of use case complexity.

## 4.4 The Bayesian approach of combining prior and sample information

We update the prior information with the sample information by taking the weighted average of the weights $w_{a-pri}$ from expert judgment and the empirically calibrated weights $w_{reg}$. The weights used in the averaging process is based on the precision of the estimates, which is calculated as the inverses of the variances of the estimates: $\delta^2_{a-pri}$ and $\delta^2_{reg}$. 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 $H_{reg} = \frac{1}{\delta^2_{reg}}$.

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

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

This process is also applied to D1, D2, and D3 to derive three sets of $w_{bayes}$ and $\delta^2_{bayes}$ to evaluate the performance of Bayesian estimates of use case complexity weights. The results are provided in Table 5.

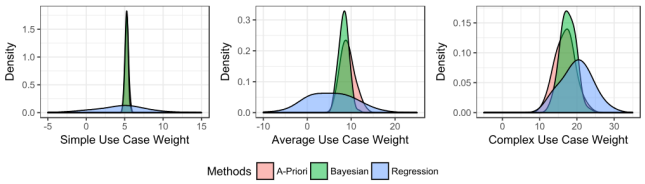


**Figure 3: Use Case Weights by Bayesian Analysis**

**Table 5: The sample information and the Bayesian estimates of the use case complexity weights**

| Dataset | Param. | Simple | Average | Complex |
|---|---|---|---|---|
| D1 | $w_{reg}$ | 4.84 | 6.41 | 19.15 |
| | $\delta^2_{reg}$ | 3.33 | 4.89 | 4.71 |
| | $w_{bayes}$ | 5.26 | 8.41 | 17.32 |
| | $\delta^2_{bayes}$ | 0.19 | 1.13 | 1.85 |
| D2 | $w_{reg}$ | -0.71 | 9.46 | 19.56 |
| | $\delta^2_{reg}$ | 52.13 | 8.36 | 9.42 |
| | $w_{bayes}$ | 5.27 | 9.08 | 16.97 |
| | $\delta^2_{bayes}$ | 0.20 | 1.26 | 2.31 |
| D3 | $w_{reg}$ | -4.55 | 12.49 | 20.22 |
| | $\delta^2_{reg}$ | 10.11 | 6.08 | 7.06 |
| | $w_{bayes}$ | 5.10 | 9.69 | 17.37 |
| | $\delta^2_{bayes}$ | 0.20 | 1.19 | 2.13 |

# 5 EMPIRICAL STUDY

## 5.1 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) RQ1: Does the Bayesian Approach improve effort estimation accuracy in comparison with other typical software size estimators? If so, how much?
(3) RQ2: How does the Bayesian Approach perform in the typical model calibration situations in considering sample sizes and homogenity of the datasets.

## 5.2 Datasets

We experimented our approach on three different datasets to evaluate the effectiveness of our approach in different software engineering settings.

The first dataset is composed of 35 data points collected from master-level computer science student projects during 2014-2016, which lasted for 4-8 months. The projects developed a wide range of software prodcuts: web applications, mobile applications, mobile games, information systems, scientific tools, etc., and 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, tester, etc. All the projects followed the formal methods of software development, including use case driven, design-driven, risk-driven, plan-based, and agile approaches. The requirements were given by the real-word clients from start-ups, non-profits, education institutes, government agencies, etc., and they 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 Table 4. We call this dataset as D1.

The second dataset is use case points benchmark dataset published by Radek Silhavy at the Promise Repository[25]. This dataset consists of 71 data points collected from three software houses and has been used in the research of selecting regression models for size estimation based on UCPs by the authors[26]. The projects are from different business sectors of software development, including manufacturing, banking, communication, etc. The software products were developed in 3rd generation programming languages, including java, C#, C++, etc., and were categorized into business application, real-time application, mathematically-intensive application, etc. Dfferent methodologies, for example, waterfall, personal software process, etc. were used in the development of the software products. We call this dataset as D2.

D3 is the combination of D1 and D2. With this dataset, we are able to observe the performance of the Bayesian estimator and evaluate its applicability in a mixed environment .

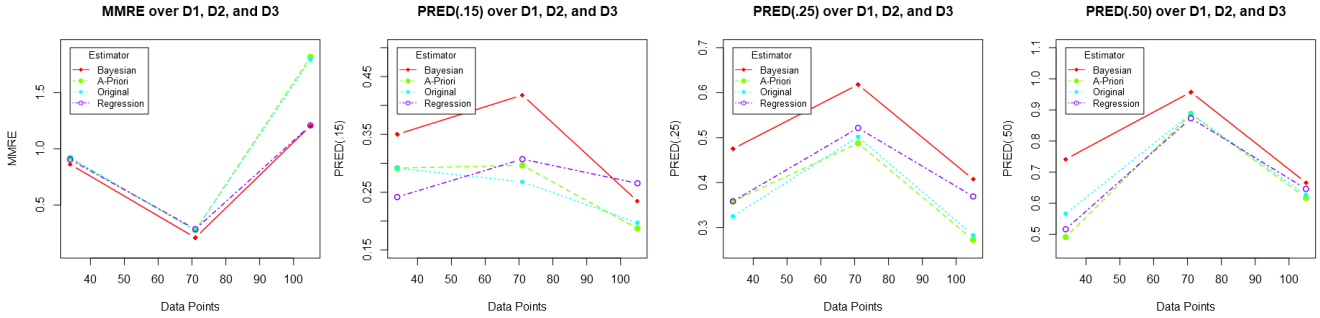


**Figure 4: Accuracy Evaluation over Datasets**

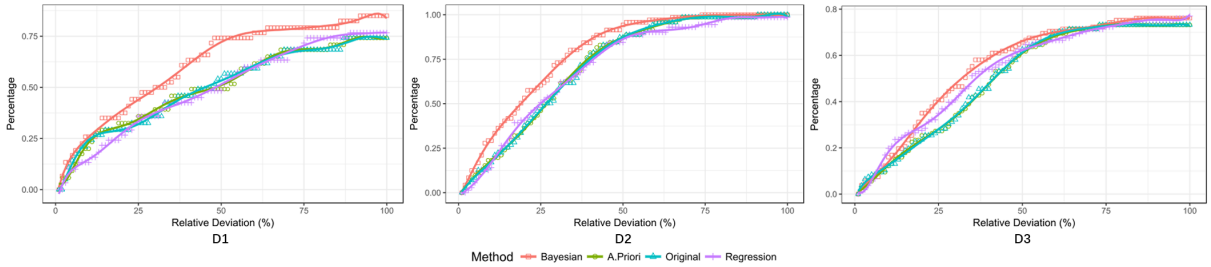


**Figure 5: 5-fold Cross Validation Estimation Error Rates**

**Table 6: Average MMRE, PRED(.15), PRED(.25), PRED(.50) for 5-fold cross validation**

| Dataset | Estimator | M. | P.(.15) | P.(.25) | P.(.50) |
|---|---|---|---|---|---|
| D1 | Bayesian | 0.862 | 0.350 | 0.475 | 0.742 |
| | A-Priori | 0.920 | 0.292 | 0.358 | 0.492 |
| | Original | 0.920 | 0.292 | 0.325 | 0.567 |
| | Regression | 0.905 | 0.242 | 0.358 | 0.517 |
| D2 | Bayesian | 0.209 | 0.418 | 0.618 | 0.957 |
| | A-Priori | 0.275 | 0.296 | 0.488 | 0.888 |
| | Original | 0.276 | 0.268 | 0.502 | 0.888 |
| | Regression | 0.285 | 0.307 | 0.521 | 0.873 |
| D3 | Bayesian | 1.203 | 0.235 | 0.407 | 0.666 |
| | A-Priori | 1.819 | 0.187 | 0.273 | 0.617 |
| | Original | 1.795 | 0.197 | 0.283 | 0.627 |
| | Regression | 1.209 | 0.265 | 0.369 | 0.646 |

## 5.3 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 compelxity weights and assessing effort estimation accuracy. Specifically, we calibrated the weights based on the three datasets to understand how each level of use cae complexity contributes to software size and compare the Bayesian estimates of the weights with the weights estimated by experts and the datasets. Our analysis of the results answer RQ1. To evaluate the estimation accuracy of the Bayesian approach (RQ2) and the interconnection between sample sizes, homegeneity of the data sets, and applicability of the software size estimators (RQ3), we applied 5-fold cross validation on the empirical datasets to estimate the out-of-sample effort estimation accuracy of the typical size estimators, and compared the results to make the conclusions.

*5.3.1 Calibration of Use Case Weights (RQ1).* We applied the calibration processs introduced in Section 4.3 on the three datasets to calibrate the weights $w_{reg}$ and their variances $\delta^2_{reg}$ for the use case complexity levels as the sample information. After that, we updated the prior information $w_{a-pri}$ and $\delta^2_{a-pri}$ (Table 3) calculated in Section 4.2 using the sample estimates. The sample information and the results from Bayesian analysis for the three datasets are presented in Table 5. A graphical representation of the model calibration results (based on D1) is presented in Figure 3.

*Analysis of the calibrated weiths (RQ1).* Based on the calibration results, we made the following observations to summarize the properties of the Bayesian estimates of use case complexity weights and answer RQ1 about how the Bayesian approach influences the calibration results:

(1) the Bayesian estimates of the weights for average use cases tend to be smaller than the estimates by the original UCPs (by 15.8% for D1, 8.7% for D2, and 1.8% for D3), while the Bayesian estimates of the weights for complex use cases are generally larger than the estimates by the original UCPs by 16.7%. This phenomenon implies that the influence from the complex use cases toward project effort tends to be non-linearly increasing (1:1.8:3.5), instead of the linear relationship (1:2:3)- simple being 5, average being 10, and complex being 15 - proposed by the original UCPs. 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. Theoratically, this may be because of the small variances of the numbers of simple use cases in the empirical dataset [1][18], which is also testified by the large estimated variances of the weights for simple use cases as shown in Table 5. For instance, the ratio between $\delta^2_{reg} : \delta^2_{a-pri}$ is 257:1 for D2 and 50:1 for D3. The Bayesian approach corrects this counter-intuitive estimates for D2 and D3. Another potential solution to this problem may be adjusting 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 expert estimates and the sample estimates, which means the Bayesian estimates of the weights are more stable if the calibration process uses the data points from different development environments.

*5.3.2 Evaluation of Effort Estimation Accuracy (RQ2 and RQ3).* *Accuracy Measures.* To evaluate the effectiveness of our proposed 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] [27]. Both MMRE and PRED rely on the quantity called magnitude of relative error (MRE) that 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 most frequently used accuracy criteria [27], in the secition of discussing the evaluation results (Section 5.4), we emphasize these two accuracy metrics to evaluate the performance of the software size estimators. Also, since as pointed out in [27], 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 perform worse than the same model for PRED(.30)), we propose to evaluate PRED(.01) to PRED(0.99) to comprehensively monitor the performance of the sizing models. In addition to MMRE and PRED(.25), we also use the average of the differences over PRED(0.01) to PRED(0.99) (Eq. 22) in our comparison to certify which model performs better.

**Table 7: Improvements by the average over PRED(0.01) - PRED(0.99)**

| Dataset | P.Avg.Imp | Bayes. | A-Pri. | Orig. | Reg. |
|---|---|---|---|---|---|
| D1 | Bayesian | 0.000 | 0.065 | 0.067 | 0.076 |
| | A-Priori | -0.065 | 0.000 | 0.001 | 0.011 |
| | Original | -0.067 | -0.001 | 0.000 | 0.009 |
| | Regression | -0.076 | -0.011 | -0.009 | 0.000 |
| D2 | Bayesian | 0.000 | 0.113 | 0.114 | 0.120 |
| | A-Priori | -0.113 | 0.000 | 0.00142 | 0.007 |
| | Original | -0.114 | -0.001 | 0.000 | 0.006 |
| | Regression | -0.120 | 0.007 | -0.006 | 0.000 |
| D3 | Bayesian | 0.000 | 0.049 | 0.047 | 0.0213 |
| | A-Priori | -0.049 | 0.000 | -0.002 | -0.028 |
| | Original | -0.047 | 0.002 | 0.000 | -0.026 |
| | Regression | -0.0213 | 0.0280 | 0.026 | 0.000 |

$$PRED\_AVG\_IMP = \frac{1}{99}\sum_{x=0.01}^{0.99} (PRED_1(x) - PRED_2(x)) \tag{22}$$

*Out-of-sample Accuracy Assessment.* We applied 5-fold cross validation to estimate the out-of-sample effort predication accuracy of the typical software size estimators using the selected accuracy measures. Specifically, four size estimators are compared in the evaluation to determine their effectiveness and mmake the conclusion about the effectiveness of our proposed approach. The four size estimators are acalled Bayesian, A-Priori, Original, Regression size estimators in the following sections for simplicity, which produce the Bayesian estimates of the weights, the a-priori weights, the weights proposed by the original UCPs, the empirically calibrated weights by linear regression respectively, following the methods introduced in Section 4.2, 4.3, and 4.4. The produced weights are then plugged in the same general form described by Eq. 23 to be 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 predication model on an independent dataset to better assess the performance of the model on new observations[20]. Specifically, for each of three datasets, it is separated into 5 folds, and 5 runs of training and testing are applied to evaluate the effort estimation accuracy with the chosen metrics: MMRE, PRED(.15), PRED(.25), and PRED(.50). The average of the values of MMRE, PRED(.15), PRED(.25), and PRED(.50) across the five runs are used as the final estimation prediction accuracy indicators.

In each iteration of model training and testing, 80% of the total dataset (27 data points for D1, 56 data points for D2, and 89 data points for D3) are used as the training set to calibrate the weights (specifcally, $w_{reg}$ and $w_{bayes}$) using the Regression and Bayesian size estimators. $w_{a-pri}$ and $w_{org}$ remain the same for the five runs of training according to A-Priori and Original size estimators. The

remaining 20% of the data points (7 data points for D1, 15 data points for D2, and 23 data points for D3) are used as the testing set to evaluate 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.

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

The predication accuracy testing results are presented in TABLE. 6, and a more comprehensive evaluation of the predication accuracy is presented in Figure 5, which plots PRED(.01) - PRED(0.99) for the four software size estimators over the three datasets.

*5.3.3 Comparison of the size estimators (RQ2).* In the comparison between the four software sizing methods, the effort estimation accuracy of the Bayesian method consistently outperforms other sizing methods for all the three datasets. For instance, based on our dataset D1, the Bayesian method outperforms A-Priori, Original, and Regression methods by 32.9%, 28.1%, and 37.0% respectively for MMRE, and 10.8%, 18.8%, and 11.6% respectively for PRED(.25) according to Table 6. The plot from PRED(.01) to PRED(.99) in Figure 5 also certifies this significant improvement. We can also observe the improvement brought by the Bayesian method on effort estimation accuracy for D2 and D3 when comparing it with the size estimators from Table 6. For example, the Bayesian method outperforms the Original method by 13.2% for D2 and 2.2% for D3 in terms of MMRE, and also by 5.6% and 0.3% for PRED(.25) for D2 and D3 respectively. The improvments in terms of PRED(.25) and MMRE can also be found when comparing the Bayesian method to the A-Priori and Regression methods over D2 and D3. Based on those observations, we conclude that the weights decided by Bayesian analysis, which combines expert-based estimates and data 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.

*5.3.4 Interconnection between sample size, homogenity of 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 the sample size and homogenity of the dataset affect effort estimation accuracy based on our empirial study. We believe those observations provide the effective guidelines for selecting appropriate size estimation methods for the typical size modeling situations.

*Stratification improves effort estimation accuracy* In the mixed environment (D3), all the four size estimator decrease in effort estimation accuracy significantly. For instance, as presented in Table 6, the Bayesian method decreases by 43.0% and 71.0% in terms of MMRE and 9.7% and 19.3% in terms of PRED(.25) when comparing D1 and D2 with D3 respectively. Similar deteriorations in effort estimation accuracy can be found for other size estimators from Table 6. 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 104 respectively), but sinificant decreases in accuracy measurements. This phenomenon strongly suggests that, in order to achieve better effort estimation accuracy, stratifying the 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 homogeneious datasets, expert-based size metrics outperform the size metric decided by purely data driven method* Expert-based size metrics based on Bayesian, A-Priori, and Original UCP methods, perform better than the purely data driven (by linear regression) size metric. For instance, Table 6 shows the significant improvements of Byesian, A-Priori, and Original methods on MMRE (by 37.0%, 6.1%, and 12.3% respectively) in the comparison with the regression based size metric. This trend can also be found for the accuracy metrics: PRED(.15), PRED(.25), and PRED(.50) according to Figure 4. The accuracy improvement measured by taking the average over the differences from PRED(.01) - PRED (0.99) for D1, shown in Table7, also confirms this point - Bayesian, A-Priori, and Original outperforms Regression by 8.3%, 1.6%, and 1.9% respectively. This phenomenon indicates the regression model tend to overfit the training dataset such that it provides worse estimation accuracy on new data points when compared to the expert-based metrics.

However, as shown in the accuracy assessment based on D2 in Figure 4, as more data points added for model calibration and testing, the regression method starts to perform better. For instance, the Regression size estimator outperforms both the A-Priori and the Original size estimator by 4.4% and 5.9% respectively. Also, we can observe the same trend from Figure 5 and Table. 6, for which the advantages of the expert-based size estimator over the regression size estimator narrow down and the regression method provides the same level of accuracy in terms of the chosen accuracy metrics.

Therefore, to summarize, we suggest, for small dataset - sample size to parameter number ratio is less than 12:1 (D1 as an example of small dataset, which has 35:3 sample size to prameter number ratio) - expert-based methods, including A-Priori, Original, and Bayesian methods, are preferred. Otherwise, the data driven methods - the regression and Bayesian methods (the Bayesian method is also driven by data) are preferred.

*For heterogeneous datasets, data driven methods are preferred.* Extending our finding in last section about increasing performance of the regression method for large dataset when comparing the regression method to the expert-based methods between D1 and D2, we also found regression method is more applicable to the heterogenous environment by comparing the accuracy measurements between D2 and D3. For instance, as shown in Table 6, the regression method outperforms both A-Priori and Original methods by 9.8% and by 9.7% respectively for PRED(.25) for D3, while there is only small improvement of 4.4% and 5.9% over A-Priori and Oirginal method for D2. The increasing margin of improvement can also be observed for MMRE and PRED(.15) from Table 6. The improvement measured by the average over PRED(.01) and PRED(.99) also testifies this point by being 5.0% better than both the A-Priori and the Original methods. Figure 5 and Figure 4 also tesitfy this point in considring the curve of regression accuracy is closing to that of the Bayesian method for D3, and the regression method performs better for both PRED(.15) and PRED(.25) for D3.

The advantage of data-drivens method over expert-based methods for D3 can be explained as, for heterogeneous environments, the effects of different levels of use case complexity on effort vary. For example, more strict environment may require the implemented

functions (for realizing use cases) be more tested, while less strict environment may require less testing effort. data-driven methods are more 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.

## 6 THREATS TO VALIDITY

*Internal Validity.* As mentioned in the model calibration process (Section 4.2 and Section 4.3), both the processes of gathering the prior information and the sample information rely on the properties of the data points. 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, some degree of variation may exist in the results of the weight calibration process 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.

*External Validity.* Some aspects of this research may also limit generalizability of the results. As shown in Figure **??**, the studied projects are considered small to medium projects as the sizes range from 1-20 ksloc and are done with 5-8 team members for D1. Also we don't know exactly how big the projects are for 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 (>= 20KSLOC). More data points from that large projects are desirable to further test the performance of the method.

## 7 CONCLUSIONS

In this paper, we introduce a systematic approach of combining prior information (complexity weights suggested previously by experts) and the sample information (complexity weights calibrated from empirical 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. We introduced the methods to synthesize the different proposals of the weights as the prior information and also the Bayesian approach to updating the prior information using sample information - the use case complexity weights calibrated from the emprical datasets. We performed an empirical study on 105 projects and evaluate the performance of the Bayesian approach of adjusting use case complexity weights by comparing it with A-Priori, Original UCPs, and Regression approaches, in terms of effort estimaiton accuracy. The results have shown the effectiveness of the Bayesian approach in improving the effort estimation accuracy and stability of the estimates. The validation was applied to three datasets which are representative to the typical software development environments and software size calibration situations. Threrefore, in addition to the proof of the effectiveness of the Bayesian approach of weight calibration, we also provided suggestions to effectively select software size calibration methods based on our analyses of the evaluation results.

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 also would like to extend the study to calibrating actor complexity weights and further evaluate the method based on the results. Also as suggested in the paper, the underlying structure of classifying use cases also needs to be reconsidered or adjusted to be more adaptable to modern use case driven projects, which woud also be an interesting extension of this research.

## REFERENCES


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

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

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

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

[9] 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.

[10] 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.

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

[12] ——, "Resource estimation for objectory projects," *Objective Systems SF AB*, vol. 17, 1993.

[13] 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.

[14] 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.

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

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

[17] 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.

[18] 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.

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

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

[21] 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.

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

[23] 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.

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

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

[26] ——, "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.
[27] D. Nguyen and T. M. WVU, "Studies of confidence in software cost estimation research based on the criterions mmre and pred," 2009.
[28] S. Chulani, B. Boehm, and B. Steece, "Calibrating software cost models using bayesian analysis," *IEEE Transactions on Software Engineering*, vol. 573583, 1999.
[29] B. Clark, S. Devnani-Chulani, and B. Boehm, "Calibrating the cocomo ii post-architecture model," in *Software Engineering, 1998. Proceedings of the 1998 International Conference on.* IEEE, 1998, pp. 477–480.
[30] R. Silhavy, P. Silhavy, and Z. Prokopova, "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. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016412121630231X
[31] 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.
[32] D. Kashyap, D. Shukla, and A. Misra, "Refining the use case classification for use case point method for software effort estimation," in *Int. Conf. on Recent Trends in Information, Telecommunication and Computing, ITC*, 2014, pp. 183–191.
[33] E. R. Carroll, "Estimating software based on use case points," in *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications.* ACM, 2005, pp. 257–265.
[34] M. Saroha and S. Sahu, "Tools & methods for software effort estimation using use case points modelâĂŤa review," in *Computing, Communication & Automation (ICCCA), 2015 International Conference on.* IEEE, 2015, pp. 874–879.
[35] E. E. Leamer, *Specification searches: ad hoc inference with nonexperimental data*. New York: Wiley, 1978.
[36] W. G. Cochran, "Errors of measurement in statistics," *Technometrics*, vol. 10, no. 4, pp. 637–666, 1968.
[37] M. R. Braz and S. R. Vergilio, "Software effort estimation based on use cases," in *Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International*, vol. 1. IEEE, 2006, pp. 221–228.
[38] D. McFall, F. G. Wilkie, F. Mc Caffery, N. Lester, and R. Sterrit, "Software processes and process improvement in northern ireland," in *16th Int. Conference SOFTWARE & SYSTEMS ENGINEERING and their applications*, 2003.
[39] A. J. Albrecht and J. E. Gaffney, "Software function, source lines of code, and development effort prediction: a software science validation," *IEEE transactions on software engineering*, no. 6, pp. 639–648, 1983.
[40] C. F. Kemerer, "An empirical validation of software cost estimation models," *Communications of the ACM*, vol. 30, no. 5, pp. 416–429, 1987.
[41] D. V. Ferens, "Software size estimation techniques," in *Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National.* IEEE, 1988, pp. 701–705.
[42] T. Rollo, "Functional size measurement and cocomo–a synergistic approach," in *Proc. of Software Measurement European Forum*, vol. 2006, 2006, pp. 259–267.
[43] S. Sparks and K. Kaspczynski, "The art of sizing projects," *Sun World*, 1999.