Calibrating Use Case Points Using Bayesian Analysis

# 1. Abstract

Use Case Points has been widely used to estimate project effort for object-oriented projects. Yet, many research papers criticize the Use Case Points methodology for not being verified and validated by data or domain experts, leading to inaccurate effort estimates. To improve effort estimation accuracy with Use Case Points, this paper introduces an approach to calibrate the complexity weights of use cases using Bayesian Analysis. Bayesian Analysis integrates a priori information (weights defined by the Use Case Points method, weights suggested by other research papers) with parameter values suggested by data. The effectiveness of the calibration approach is being demonstrated in this paper with an empirical study on 40 use case driven projects. ~~through a comparison between estimation accuracy of different the weighting schema.~~

# 2. Introduction

Despite the amount of research that goes into cost and effort estimation, a survey from 2003 concluded that “out of 15 areas of software engineering and engineering management addressed, the topic of estimation was the single most problematic area” [cite]. Most effort models use source lines of code (SLOC) as the software size input because it is easy to calculate after projects are complete, has high correlation to effort, and seems objective [cite]. However, SLOC cannot be accurately estimated until the project is nearly complete [cite].

Developers, especially unfamiliar with the existing source code of the project, must spend a lot of time understanding and learning the source code, and which “parts of the (source code) are relevant to their current task” [cite]. The source code requiring changes with maintenance tasks may be distributed among the “system’s components and modules” making the learning process “both time-consuming and difficult” [cite]. Therefore, estimating the size of the modules and code that need to be changed, and the amount of changes that will be required in SLOC becomes practically impossible early in the maintenance task’s lifecycle.  
   
Albrecht developed Function Points (FPs) to size software projects early in the lifecycle, which describe software by the number and complexity of transactions, file types, and general system characteristics. The International Function Point Users Group (IFPUG) recognized the problem that FPs did not consider non-functional requirements (i.e., data operations and interface design), and constructed the Software Non-functional Assessment Process (SNAP) as a solution. IFPUG expects that “when used in conjunction with function points” using SNAP will increase the accuracy of effort estimation [cite]. Analyses performed with FPs and SNAP on Unified Code Count (UCC)’s dataset found that using FPs and SNAP in conjunction led to acceptable effort estimation models [cite].

Despite the efficiency of FPs and SNAP, they require detailed design or architecture specifications for accurate size and effort estimation [cite]. Use Case Points (UCPs) provides a functional size metric based on use cases, which several development teams use to gather and understand requirements [cite]. While many practitioners and research papers have reported the effectiveness of UCPs, a few research papers criticize the UCPs method because the complexity groups and weights are not validated with data [cite]. To counter the criticism, the authors of this paper use Bayesian Analysis to use the complexity weights defined in the UCPs method, complexity weights suggested by other research efforts, and data to define the complexity weights.

# 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 [cite]. Several practitioners and researchers have reported its effectiveness for use case driven projects [cite]. Table 1 summarizes the steps and rules to calculate UCPS. The empirical experiments Karner ran showed that a UCP required 20 person-hours to develop [cite], but Schneider and Winters found that the ratio between UCPs and person-hours depend on the environmental factors [cite]. Other datasets found that the UCP to effort ratio could range from 15 to 30 person-hours, which concludes that a team or organization should use historical data to determine the ratio that best describes their environment [cite].

|  |  |  |
| --- | --- | --- |
| Step | Rules | Results |
| 1. | Identify the number of transactions (NT) for each use cases and classify the use cases into 3 levels of complexity and assign the weight for each use case.  If (NT < 3), then simple use case and weighted as 5  If (NT < 5), then average use case and weighted as 10  If (NT > 8), then complex use case and weighted as 15  Sum the number of weighted use cases as unweighted use case weight | Unweighted use case weight (UUCW) |
| 2. | Identify the actors and classify the actors into 3 levels of complexity and assign the weight for each actor. | Unweighted Actor Weight (UAW) |
| 3. | Evaluate the 13 technical factors and their impact to calculate TCF | Technical Complexity Factor (TCF) |
| 4. | Evaluate the 8 environmental factors and their impact to calculate EF. | Environmental Factor (EF) |
| 5. | Calculate use case point (UCP) based on:  UCP = (UUCW+UAW)\*TCF\*EF | UCP |

However, a good portion of decisions in the counting process is determined based on the domain experience, for example, the weights assigned for different levels of complexity for use cases and actors, as explained in the original paper for use case – “the weight schema works best in our organization”. As the software engineering environment have changed greatly compared before, this weighting schema has fallen short in estimating modern use case driven projects. Therefore, an approach of updating the weights by integrating the empirical data is needed to make use case points better adapt to modern status of use case driven approach of software engineering. Specifically, the approach we take to solve the problem is by Bayesian analysis to combine the apriori information and empirical information.

## 3.2 Bayesian Analysis

Bayesian analysis has been used in project effort estimation as the framework to combine domain experience and empirical study results in COCOMO II. For example, COCOMO II combines the effects of reuse factor estimated by experts’ estimation and calibrated by data to solve the unintuitive calibrated results, and thus improve the estimation accuracy. Specifically, the Bayesian approach of estimation is to minimize the expected error of estimation by having weighted average over the possible estimates by different models. In our case, we take the weighted average over the estimate by experts and the estimate from data, and the estimates are weighted by their variances. The process is formalized with the equation (1).

Where is the Bayesian averaged estimate, b is the regression estimate, and is the expert estimate; is the design matrix of the predictors and s is the standard deviation of the residual for the sample data and is the precision matrix – the inverse of the variance of expert estimates. Also the variance of the Bayesian estimation is calculated by equation (2).

# 4. Related Works

To calibrate the weights of use cases of different level of complexity, an approach based on neural network has been proposed by Nassif, however, no specific results or details of the approach have been discussed. Other than that, there is very limited amount of papers to discuss the approach of calibrating use case weight. Also, Bayesian analysis has already been applied in the software project effort estimation, for example, by COCOMO II, in which, an example is provided to calibrate the weights for ruse factor to solve the uninterpretable calibrated weight.

Model Calibration

As depicted in Figure 1, the Bayesian calibration process goes through the follow steps:

1. The empirical project data is input to run multiple linear regression to calibrate the weights for simple use case (), average use cases (), complex use case (), simple actor (), average actor (), and complex actor () of equation (3), as denoted by vector. is calculated by having the real project effort divided by EF and TCF according to equation (2). W is the productivity factor calibrated by running linear regression between real effort and use case points with respect to the data set.

(3)

(4)

1. The apriori evaluations for use case weights are input to calculate the means and variances for , , , , , , as the vectors }.
2. Bayesian averaged weights are calculated based on the least squared calibrated weights and the weights calculated from apriori information. Specifically the weights are calculated by equation (3) and (4).

(3)

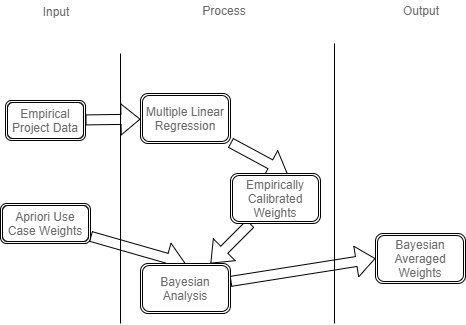


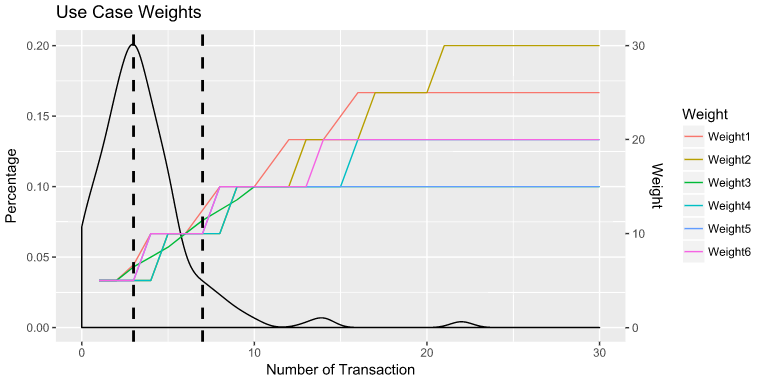
Figure 1. Bayesian Averaged Weight Calibration

Estimation of Apriori

The weights that are selected apriori knowledge:

|  |  |
| --- | --- |
| Number | Source |
| Weight1 | Extended Use Case Points Method for Software Cost Estimation by Wang Fan, Yang Xiaohu, Zhu Xiaochun, and Chen Lu |
| Weight2 | Software Size and Effort Estimation from Use Case Diagrams Using Regression and Soft Computing Models by Ali Bou Nassif (PhD Thesis) |
| Weight3 | Enhancing Use Case Points Estimation Method Using Soft Computing Techniques by Ali Bou Nassif, Luiz Fernando Capretz, and Danny Ho |
| Weight4 | Revised Use Case Point (Re-UCP) Model for Software Effort Estimation, Mudasir Manzoor Kirmani Research Scholar, School of CS & IT |
| Weight5 | UCP, Garner |
| Weight6 | Use Case Sizing Arlene Minkiewicz |

Different weighting schemes are collected from five different papers/sources. The weights with respect to the number of transactions are provided by Fig X. Here we calculate the estimated means and variances for each level of complexity by taking the average of the weights proposed by the experts and the variances of the weights for each level of complexity. As displayed in Figure X, for each level of complexity (defined in original UCP), the expert may use different weights for different number of transactions, so we take the weighted average of the weights over different number of transaction for that level of complexity as the estimated mean for the level of complexity. The weights are decided by the probability of the number of transactions with respect our dataset, to simulate the probable situation for the weight being used for the level of complexity if the weighting schema is used.



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Complexity Level | TranNum | Weight1 | Weight2 | Weight3 | Weight4 | Weight5 | Mean | Variance |
| Simple | 1-3 | 5.78 | 5.00 | 5.68 | 5.00 | 5.00 | 5.29 | 0.161 |
| Average | 4-7 | 10.19 | 7.93 | 8.64 | 7.93 | 10.00 | 8.94 | 1.199 |
| Complex | >7 | 16.21 | 14.14 | 13.80 | 13.28 | 15 | 14.48 | 1.320 |

Empirical Study

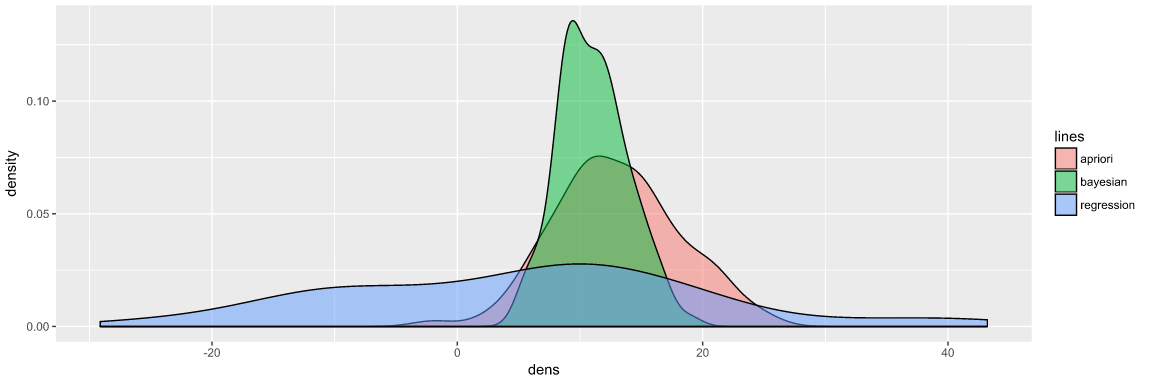
Data collection

Empirical study has been done on 40 projects to collect the data needed to calibrate the Bayesian averaged weights. The projects are educational projects, which produces mobile applications, web applications, and information systems. The use case counting process are applied to get the ratings for the number of simple use cases, averages use cases, complex use cases, simple actors, average actors, and complex actors, environmental factors, and technical factors. The effort data was collected by weekly effort report. The charts provided in Figure 2 (to be done) provides general descriptions about the ranges of the projects. On the other hand, apriori data for use case and actor weights are collected from published papers and interviews with the experts in use case point estimation domain. In our case, we collected ratings for the weights from X sources.

Model Calibration

The Bayesian averaged weights and variances are calculated using the weights and variances calibrated by multiple linear regression and the weights and variance by apriori information. The results are presented in Table 1. As we can observe, the variances for Bayesian calibrated estimates are smaller than both the apriori estimates and regression calibrated estimates. A graphical representation of the calibrations are presented in figure 2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Apriori | | Regression | | Bayesian | |
| Estimates | Mean | Variance | Mean | Variance | Mean | Variance |
| Simple Use Case | 5.292 | 0.16112 | 3.4849 | 1.3637 | 5.216 | 0.110 |
| Average Use Case | 8.938 | 1.20407 | 6.0577 | 11.964 | 9.032 | 0.647 |
| Complex Use Case | 14.486 | 1.32028 | 22.509 | 18.749 | 14.624 | 0.697 |

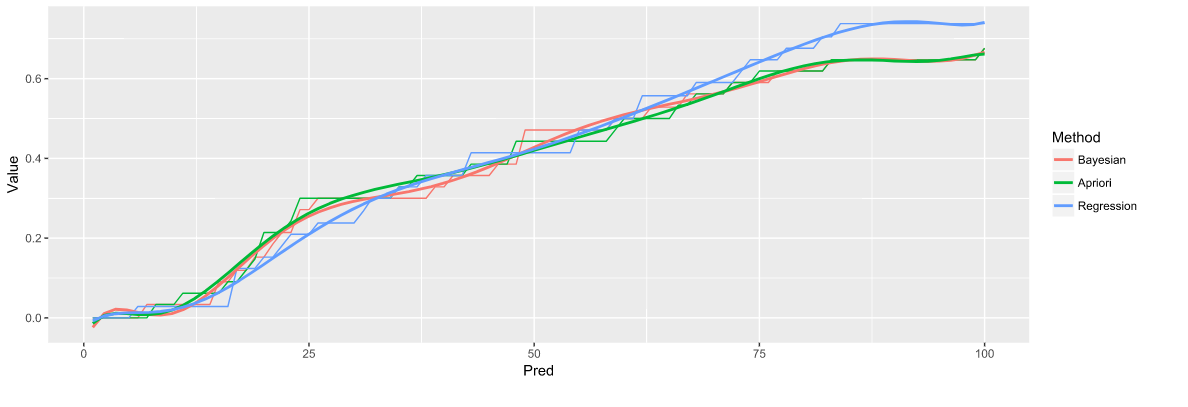


Model Evaluation

To further evaluate the calibrated models, we evaluate the estimation accuracy for the three estimators by 5 fold cross validation. Specifically, the 40 training data points are separated into 5 folds, and 5 runs of training and testing are applied to calculate the MMRE, PRED(.25), and PRED(.50), and the average of the five runs of MMRE, PRED (.25), and PRED(.50) are used as the final estimations for the estimation accuracy of the models. The results are presented in Table 3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Apriori | | | | Regression | | |  | Bayesian | | |  |
| Run | MMRE | PRED | PRED | PRED | MMRE | PRED | PRED |  | MMRE | PRED | PRED |  |
| 1 | 1.19175 | 0 | 0.285714 | 0.428571 | 1.144814 | 0 | 0.142857 | 0.285714 | 1.191725 | 0 | 0.285714 | 0.428571 |
| 2 | 1.47253 | 0.142857 | 0.428571 | 0.428571 | 1.836527 | 0 | 0.428571 | 0.428571 | 1.560733 | 0.142857 | 0.428571 | 0.428571 |
| 3 | 0.66624 | 0.166667 | 0.5 | 0.5 | 0.582729 | 0 | 0.333333 | 0.5 | 0.671106 | 0.166667 | 0.5 | 0.5 |
| 4 | 0.81688 | 0 | 0 | 0.428571 | 0.628557 | 0 | 0.142857 | 0.428571 | 0.757049 | 0 | 0 | 0.571429 |
| 5 | 0.815597 | 0.142857 | 0.285714 | 0.428571 | 0.767113 | 0.142857 | 0.142857 | 0.428571 | 0.839742 | 0.142857 | 0.285714 | 0.428571 |
| Average | 0.99259934 | 0.090476 | 0.3 | 0.442857 | 0.99194791 | 0.028571 | 0.238095 | 0.414286 | 1.00407104 | 0.090476 | 0.3 | 0.471429 |

[Just an example of the possible result] As we can observe, the Bayesian approach outperforms both the apriori and regression’s estimation accuracy. The improvement is about \*\* percentage better than apriori and \*\* percentage better than regression estimation.



Local Calibration

[Need to be done]

Threads to Validity

There are some aspects of the research limit the results been applied to other more general areas. For example, as shown in the project description figure, the projects that are collected are small sizes ranging from 5-20 sloc with 7-8 team members, and the results may not be applicable to other larger projects. Also the data set is relatively small, the ratio between data points and parameters are about 6:1. It is possible that the situation of being overfitting exists, which could affect the calibrated weights by regression, which may also influence the stability of the conclusions. Also in the fivefold cross validation, only 8 data points are in the test data points, still it is a relatively small testing data set to make conclusion about the estimation accuracy and the superiority of the Bayesian analysis based model.

Conclusions

In this paper, we introduced a framework of calibrating use case and actor weights by integrating the apriori information and empirical information, and the empirical study has shown the effectiveness of the approach in improving the estimation accuracy of use case points. However, due to the nature of the data points used in the empirical study, further application to other software development environment needs to collect data points from the specific environment and calibrate for the environment.

References

1. cocomoII
2. calibrating cocomoII using Bayesian analysis
3. Calibrating Use Case Points