Application of Multi-Model Combinations to Evaluate the Program Size Variation of Open Source Software

Shih-Min Huang
Fab. 12 Equipment Automation
Taiwan Semiconductor Manufacturing Company Limited
Hsinchu, Taiwan
n0140105@hotmail.com

Chin-Yu Huang
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
cyhuang@cs.nthu.edu.tw

***Abstract*—As one of the most important internal attributes of software systems, the estimation of software size is crucial to project management and schedule creation/tracking. Program size can be typically described by the length, functionality, or complexity of the software, but in practice most people customarily use the lines of code (LOC) as a major measure of program size. In actuality, LOC is still widely used, and can be easily measured upon project completion. In this study, we used the methods of multi-model combinations to predict and analyze the size distribution and size-change rate of Open-Source Software (OSS). Experiments based on real OSS data will be performed, and the evaluation results have shown that the proposed method had a better prediction capability of size distribution and size-change rate of OSS. Our goal is not to add one more model to the already-existing large number of models, but to emphasize a new approach for the evaluation of program size variation and reveal different issues of software sizing.**

Keywords—Size estimation, Code length, Combinational Model, Laplace distribution, Normal distribution.

I. INTRODUCTION

During the software development life cycle (SDLC), size is one of the metrics related to the properties and specifications of the software. For the purpose of software measurement, acquiring objective and quantitative results can be important for applications regarding schedule and budget planning, cost estimation, quality assurance testing, etc [1]. Some research has shown that the statistical distribution of source-code file size is a lognormal distribution [2]. But Concas et al. [3] reported that the statistical distribution of source code file sizes follows a double Pareto distribution. For years to come, we may see numerous research that concentrates on file size rather than on the software size-change rate. As a matter of fact, if the focus is directed to the software size-change rate, this metric may give particular insight into software metrics.

Presently, the commercial off-the-shelf (COTS) systems can be roughly divided into two kinds: Closed-Source Software (CSS, e.g., Microsoft Windows, Adobe Photoshop, etc.) and Open-Source Software (OSS, such as Ubuntu Linux, Samba, Apache HTTP Server, etc.) The source code of CSS is typically not available, but OSS generally allows users to access the source code. Many successful OSS projects like Linux, Apache and others are growing steadily. But the main challenge facing CSS and/or OSS companies at present is how to develop the software and complete all the required tests on time. In the past, Amit et al. [4] reported that the number of OSS projects is growing at an exponential rate. But it can be found that the software size-change has a similar tendency with the interest rate change in Economics. Samuel et al. [5][6] once used the asymmetric Laplace distribution (ALD) model to analyze the interest rate and currency exchange rate in finance.

In this study, we have proposed a method of multi-model combinations to analyze the program size variation in OSS [7]. We utilized two weighted combination methods, equal and dynamic, for reliability estimation of real OSS data. Three probability distribution models, namely the ALD model, the normal distribution (ND) model, and the Laplace distribution (LD) model, were selected as the candidate models for model combinations. Actually, these models represent different features: the ND and LD models are usually used in the research fields of nature and science for real-valued random variables, and the ALD model was used in observing the asymmetrical phenomena such as the exchange rate and interest rate change.

Specifically, we first studied the performance comparisons between the single model and the proposed model combinations regarding the distribution of the software size-changing rate. Additionally, we further adopted a modified dynamic weight decision approach for the proposed model combination. In addition, we compared the single model to the equally-weighted combined model using different weight decision approaches. The proposed weight decision approach may be considered as another way to determine the optimal single software size-change rate distribution model. We collected the number of LOC edited per month for OSS systems, and transformed the datasets into software size-change rate per month for OSS systems.

The rest of the paper is organized as follows. Section 2 gives a brief review of the existing literature, which mentions the related works about the software size-change rate in OSS systems, and introduces the concepts of model combination. Section 3 illustrates the dynamic weight decision approach and the proposed modified Bayesian inference dynamic weight decision approach (MBIWDA). Section 4 shows the experimental results of the OSS data collected from Ohloh.net using the proposed approach. We discussed different weight decision approaches in weighted combinational models and compared the prediction performance between the various combined models. Finally, section 5 summarizes the experimental results and discusses potential research directions in the future.

II. Related Works

2.1 Software Size Estimation

Software quality control (SQC) and software quality assurance (SQA) typically rely heavily on fault removal and forecasting. If developers and/or managers could identify the most error-prone components that are difficult to maintain early on, they would be able to optimize testing-resource allocation, enhance defect removal efficiency, and increase fault detection effectiveness, etc. Thus the customer-detected faults after delivery could be greatly reduced. Software project data could be systematically collected and analyzed during testing and operational phases, and they are assumed to provide additional information for the developer's reference.

OSS has become an essential and primary way to develop software. In the software market, some software products reuse OSS components (e.g., web servers, internet browsers, client email, etc.). We have to take advantage of one probability distribution model to observe its current state and estimate the future development of OSS if we want to understand how OSS changes and grows in the future.

The distribution of file size has been discussed intensely in recent years. Some research reported that the size is lognormally distributed. Additionally, several researchers have also shown that in some cases power law distribution offers a better explanation [8]. For instance, the Pareto distribution model and related distribution models are widely used in social, scientific, and many other types of observable phenomena. Therefore, the "80-20 rule" may be true, which argues that 80% of the effects results from 20% of the causes. We can attribute 80% of the faults to 20% of the modules in software engineering [9][10][11].

However, it is noted that previous research may not have come to an explicit conclusion about the distribution of file sizes. Another important issue is the growth rate of the software size. In the past, some research reported that the number of OSS projects was growing at an exponential rate. Nevertheless, less research focused on the single OSS size-change rate fitted by which probability distribution model.

Let us consider a motivating example to explain this part. Here we assumed that the software size-change rate can be presented by the natural logarithm of the line of code (LOC) ratio for two consecutive months defined as follows:

$$Y_m = \log(i_m / i_{m-1}) \tag{1}$$

where i_m is the LOC of the month m, and the resulting values of the loragithmic Y_m is the software size-change rate in month m. The Mozilla Firebug LOC whole version size-change rate data per month from May 2008 to July 2012 were collected and presented in Fig. 1(a). As seen from Fig. 1(a), we found that there are no negative values in this data set. Fig. 1(b) also depicts the Mozilla Firefox whole version LOC size-change rate data per month from April 2002 to July 2012. As shown in Fig. 1(b), some negative values existed in the collected data. From Figs. 1(a) and 1(b), it can be seen that the traditional log-normal distribution model or the Pareto distribution model may not be appropriate for handling software size-change rate data which could have both positive and negative values in some cases.

But we can find that from Figs. 1(a) and 1(b), the software size-change rate demonstrates a similar trend to the interest rate change in economics. Figs. 2(a)-2(c) further plots the number of size-change rate versus size-change rate for Apache HTTP Server, Eclipse Platform, and Ubuntu. It is obvious that these distributions are quite skewed or fat-tailed and peaky. In the past, the asymmetric Laplace distribution (ALD) model was commonly used to analyze stock yield, interest rate, and currency exchange rate, among other factors in economics. Samuel et al. [5][6] once used the ALD model to analyze the interest rate and currency exchange rate in finance. Klein [12] studied the yield rates of 30-year Treasury bonds from 1977 to 1993, finding that the empirical distribution was too "peaky" and "fat-tailed". Kozubowski and Podgórski [13] proposed an ALD model for analyzing interest rates, proving that this relatively simple model is able to capture the peakedness, fat-tailedness, skewness, and high kurtosis observed in the data. The *Probability Density Function* (PDF) of the ALD model is given as follows:

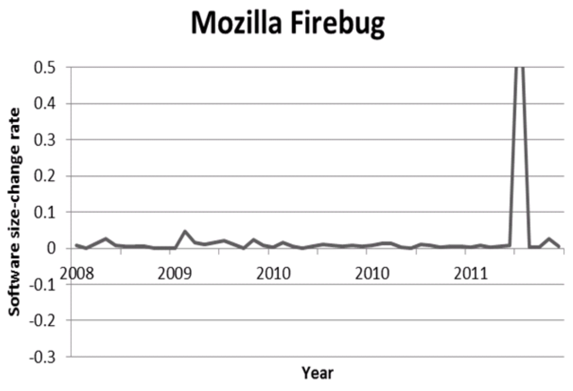
$$f_{ALD}(x) = \frac{1}{\sigma}\frac{\kappa}{1+\kappa^2}\begin{cases}\exp(-\dfrac{\kappa}{\sigma}x), \text{ for } x \ge 0\\ \exp(\dfrac{1}{\sigma\kappa}x), \text{ for } x < 0\end{cases} \tag{2}$$

where κ is a skewness parameter and σ is a scale parameter. In addition, the Laplace distribution (LD) and the normal distribution (ND) were also widely used in reliability engineering, economics and in the finance fields [14][15], and their PDFs are defined by

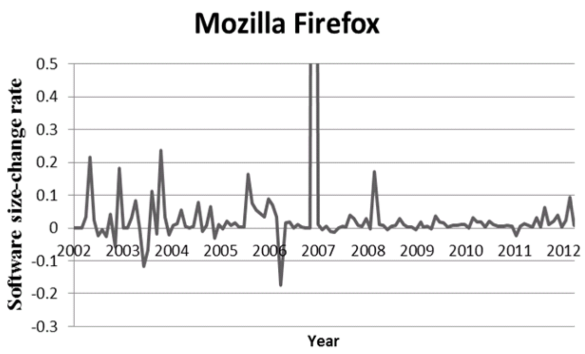
$$f_{ND}(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{3}$$

$$f_{LD}(x \mid \mu, b) = \frac{1}{2b}\exp(-\frac{|x-\mu|}{b}), \tag{4}$$

where μ is the mean, σ is the standard deviation, and b is the scale parameter. But the practical problem is that sometimes the selected models disagree in their desired predictions, while no single model can be trusted to provide consistently accurate results across various applications [16]. In the following, the ALD model, the LD model, and the ND model will be selected as the candidate models and we will use the methods of multi-model combinations to predict and analyze the size distribution and size-change rate of OSS.

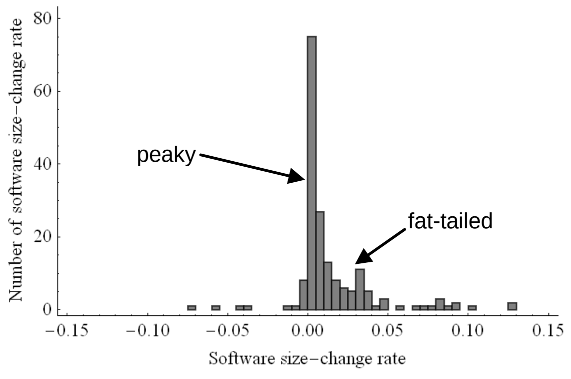


(a) Mozilla Firebug.

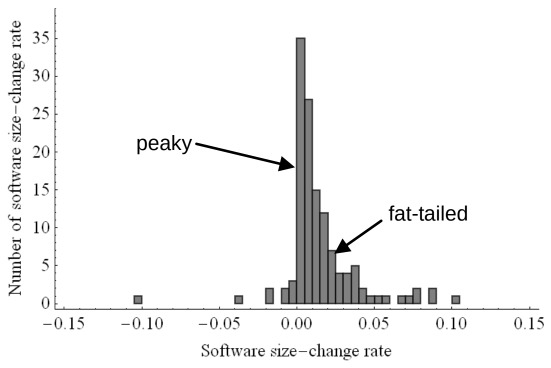


(b) Mozilla Firefox.

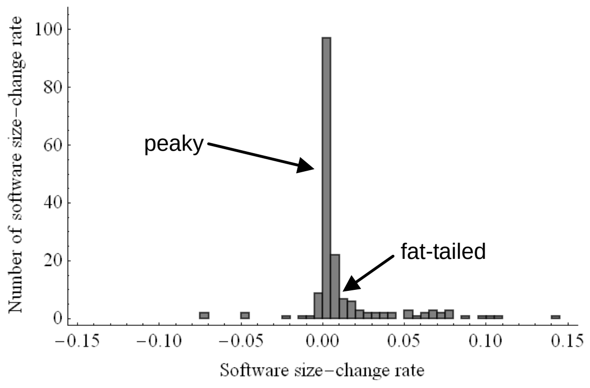
Fig. 1. OSS program size-change rate.



(a) Apache HTTP Server.



(b) Eclipse Platform.



(c) Ubuntu.

Fig. 2. Number of OSS size-change rate vs. size-change rate.

2.2 *Weighted Combinational Models*

Theoretically, different models represent a range of underlying assumptions that are only applicable to certain validated data or testing conditions. This could pose a challenge for managers in selecting the most suitable model and in assessing the quality measures of software. Rather than selecting the best model, some researchers have also suggested that using more than one model can reduce the risk of only trusting a single model [16], [17]. In principle, applying combinational models could pose an annoying problem in terms of determining proper weights for each model. Some research has suggested that the equally-weighted linear combination methods are suitable for generating a new combined model [18]. It is a simple, low-risk, and high-performance method to solve the above problem. For example, Lyu and Nikora [16][17] used the arithmetic average of each candidate model's prediction as a linearly equally-weighted (LEW) combination prediction. Later, Su and Huang [19] adopted the neural-network-based approach with dynamic weight decision methods for creating the combinational models. Additionally, Hsu and Huang [7] also reported that the relationship between models is the nonlinear geometric weight (NLGW).

Another relationship between models is the nonlinear harmonic weight (NLHW). In some circumstances, the LEW, NLGW, and NLHW decision approach may not provide more accurate results. Consequently, Hsu and Huang [7] proposed the Genetic Algorithm (GA) to determine the weight assignment. Moreover, some studies also proposed using machine learning techniques, and implemented the adaboosting algorithm to combine the model. These researches used the adaboosting-based self-combination model (ASCM) and adaboosting-based multi-combination model to improve the reliability estimation of software systems [20]. Other research also suggested that the Bayesian inference dynamic weight decision approach (BIWDA) be used to determine the dynamic weight of the candidate model [21].

It is noted that the BIWDA originated from Bayes' theorem, which depicts the relation between two conditional probabilities that are the reverse of each other [22]. Here we will adopt the Bayes' theorem to calculate the weight for prediction system *r* at prediction stage *j* [21]:

$$w_j^{\,r} = PL_{1,j-1}^r / (\sum_{k=1}^{m} PL_{1,j-1}^k), \qquad (2)$$

where $PL_{1,j-1}^r$ is the prequential likelihood from time 1 to *j*-1 of the prediction system *r*, and w_j^r has to comply with the rules of Eq. (3):

$$\sum_{r=1}^{m} w_j^r = 1 \qquad (3)$$

We can use both Eq. (2) and Eq. (3) to predict $T_{j,}$ which is the inter-failure time at stage *j* described as follows.

$$P(T_j < t \,|\, t_{j-1,\ldots,}t_1) = \sum_{r=1}^{m} w_j^r \hat{F}_j(t) \qquad (4)$$

But it also has to be noted that above combined model could have great performance with one specified dataset but exhibit unstable performance for unknown datasets. Therefore, it is hard for us to determine the appropriate weight decision approach for each model. In the following, a modified BIWDA model is proposed.

III. Weighted Combinational Models

3.1 Equal Weight Combinations

First, for the ND, LD, and ALD models, we selected the equal weight decision approach to form a combinational model. The equal weight combination model is the easiest combinatorial method. For each candidate model, the weight is constant and equal. Furthermore, Lyu and Nikora [16][17] suggested that the equally-weighted linear combinatorial method can be used to reduce the risks of depending on a specified model. That is,

$$\hat{E}(t) = \sum_{i=1}^{n} w_i \times E_i(t) \text{ and } w_i = \frac{1}{n}, \qquad (5)$$

where *n* is the number of models, $E_i(t)$ is the estimated result (i.e., cumulative number of number of faults) of the *i*th model by time *t*, and w_i is the fixed weight of the *i*th model. Additionally, Hsu and Huang [7] asserted that the relationship between models is nonlinear, and NLGW can be described as follows:

$$\hat{E}(t) = \sqrt[n]{\prod_{i=1}^{n} E_i(t)^{w_i}} \text{ and } \sum_{i=1}^{n} w_i = n \qquad (6)$$

The definition of NLGW is based on the nth root of the geometric product of estimated results with exponential weights. Here we will assign a static equal weight for NLGW, known as the nonlinear geometric equal weight (NLGEW), where each candidate model has the same and static weight in the combinatorial model to correspond to the software size-change rate. Finally, the harmonic mean is also suitable for calculating the average rates and is defined as the reciprocal of the arithmetic mean or the reciprocal of a set of positive real numbers [23]. Another relationship that can be found between models is the nonlinear harmonic equal weight (NLHEW), which can be expressed by

$$\hat{E}(t) = \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} \frac{w_i}{E_i(t)}} \text{ and } \sum_{i=1}^{n} w_i = n. \qquad (7)$$

3.2 Modified BIWDA Linear Combination Model

In practice, there are some drawbacks in the traditional weight decision approaches and these drawbacks influence the estimated results if the accurate weight decision approach is not adopted. For instance, the equally-weighted approach may exhibit better performance in specified applications of software reliability engineering [17]. Moreover, the GA for the weight assignment can avoid the above problem if the equally-weighted decision approach is used, but the drawback is that it cannot be converged in the limited time. Hence, the following section describes the approach we proposed that appropriately reflects the relative importance of the candidate model.

Additionally, we could not directly select the best probability distribution model to satisfy all OSS datasets. The problem was addressed by seeking the combination model that mostly corresponded to the true distribution of the software size-change rate. As mentioned in the preceding section, some widely-used probability distribution models in economics, such as the ND model, the LD model, and the ALD model will be considered as the candidate models for our proposed weighted decision approach.

Here we have assumed that the weights with MBIWDA at time t_j are determined by the past predictive accuracy (ie.t_1,..., t_{j-1}). Moreover, we have considered that each weight of the candidate model was a learning process of earlier predictions. Therefore, the weight of the candidate model was influenced by the previous prediction results. In short, we applied and modified the theory of BIWDA for calculating the weight of each candidate model in this section. In the MBIWDA, the weight for candidate model *i*, at time *t* is similar to Eq. (2):

$$w_t^i = P(M_i \,|\, t_1,\ldots,t_{t-1}) = \frac{P(t_1,\ldots,t_{t-1} \,|\, M_i)}{\sum_{k=1}^{m} P(t_1,\ldots,t_{t-1} \,|\, M_k)} = \frac{p(t_1,\ldots,t_{t-1} \,|\, M_i)}{\sum_{k=1}^{m} p(t_1,\ldots,t_{t-1} \,|\, M_k)} = \frac{\prod_{l=1}^{j-1} f^{\,i}(x_l)}{\sum_{k=1}^{m} \prod_{l=1}^{j-1} f^{\,k}(x_l)} \qquad (8)$$

where M_i is a candidate system numbered i. That is,

$$w_t^i = \frac{PL_{1,t-1}^i}{\sum_{k=1}^{n} PL_{1,t-1}^k}, \tag{9}$$

where $PL_{1,t-1}^i$ is the prequential likelihood regarded as the criterion for judging the predictive performance of each candidate model. Also, this criterion can be viewed as a global comparison of goodness of prediction for one candidate model with another from time 1 to t-1 of the candidate model i, and $PL_{1,t-1}^i$ is expressed as:

3.3 $$PL_{1,t-1}^i = \prod_{j=1}^{t-1} f^i(x_j), \tag{10}$$

where $f^i(x_j)$ is the probability density function of the candidate model i, and x_j is the software size-change rate at time j.

We utilized both Eq. (9) and (10) to find the closest degree of each candidate model to identify the true distribution at specified time t. Therefore, we obtained the weight of the specified model by time t. Furthermore, we could accumulate the weight of the specified model for time t and calculate the average accumulative weight of the specified model expressed as follows.

$$w_i = \frac{1}{t}\sum_{j=1}^{t} w_t^i \text{ and } \sum_{i=1}^{n} w_i = 1 \tag{11}$$

where w_i is the average weight given to the ith model. Finally, we assumed that the average weight w_i can represent the overall and stationary weight of the model i in the combination model. Additionally, we substituted Eq. (11) into Eq. (5) and calculated the estimated results of the dynamic weight combination model, as defined in Eq. (12).

$$\hat{E}(x) = \sum_{i=1}^{n} w_i \times E_i(x) \text{ and } \sum_{i=1}^{n} w_i = 1 \tag{12}$$

where n is the number of models, $E_i(x)$ is the estimated result of the ith model at stage x, and w_i is the modified dynamic weight of the ith model.

IV. Experiment and Analysis

4.1 *Selected Data Sets*

In this study, the Apache HTTP Server data collected from Ohloh.net are used for the comparison of model performance [24]. The experiments are performed by the following steps:

Step 1. First, we recorded the direct measures of software from Ohloh.net on a monthly basis. In this step, we excluded comments and blankness in codes, and thus fetched the accurate LOC.

Step 2. Using the normalized proportion of the monthly cumulative OSS software size-change rate, the software size-change rate arranged in an increasing order.

Step 3. The parameters of these candidate models were estimated by applying the MLE and LSE.

Step 4. MBIWDA was applied to a linear combination of the two models (of ND, LD, and ALD).

4.2 *Evaluation Criteria*

For the purpose of comparing the performance of the different combined models, several evaluation criteria were selected and listed as follows.

(a) The *Mean Squared Error* (MSE) is typically defined as[25],[26] ,[27] ,[28], [29]:

$$MSE = \frac{1}{n-\theta}\sum_{i=1}^{n}(X_i - O_i)^2, \tag{13}$$

where θ is the number of the estimated model's parameters. The mean squared error (MSE) can be regarded as the average of the squares of errors. The MSE value should be as small as possible.

(b) The *Coefficient of Determination* (R^2) is defined as [30]

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(X_i - O_i)^2}{\sum_{i=1}^{n}(O_i - \overline{O})^2}, \text{ where } \overline{O} = \frac{\sum_{i=1}^{n} O_i}{n} \tag{14}$$

The R^2 measures the degree of fitting data by the model. Eq. (14) shows that the R^2 is between 0 and 1. It is noted that the higher R^2 value is considered with a better fitness of the model.

(c) The *Akaike Information Criterion* (AIC) is defined as [31]:

$$AIC = 2k - 2\log[L], \tag{15}$$

where k is the number of parameters in the model and L is the maximized value of the likelihood function for the model. AIC evaluates the performance of MLE. A smaller AIC is better.

(d) The *Kolmogorov-Smirnov Test* (K-S test) [32], [33], [34] The K-S test is a nonparametric test for measuring the equality of a random sample with a reference probability distribution. The K-S test calculates the distance between the CDF of the actual data and that of the theoretical probability distribution model. The K-S statistic for a given $F(x)$ is

$$D_n = \sup_x |F(x_i) - F_n(x_i)|, \tag{16}$$

where sup_x is the supremum of the set of distances. If the empirical distribution function is accommodated by the theoretical distribution function, D_n may approach 0. Moreover, we applied the null hypothesis to test the relationship between the empirical distribution function and theoretical distribution function. The null and the alternative hypothesis are H_0: *The data follow a specified distribution.* H_1: *The data do not follow the specified distribution.* The hypothesis regarding the distributional form was rejected at a significance level α=0.05 if the test static, *D* calculated by Eq. (16) is greater than the critical value (CV) shown in Eq. (17):

$$CV_{(1-\alpha,n)} = 1.358/\sqrt{n},\ \alpha = 0.05,\ n > 35. \qquad (17)$$

In this experiment, we will use arrows (↑,↓) to indicate whether the specified model accommodated the data. In addition, models of better performance should exhibit a smaller K-S statistic.

4.3 Experiments and Discussions

In order to validate the performance of our proposed method, we selected equally-weighted combinatorial model as the basis of comparison. To compare the models, we assigned each model's rank for each criterion and summed up these ranks. The superior models showed lower summed values whereas the worse model had higher summed values. Generally, there are two widely-used estimation methods for model parameters: the Least Square Estimation (LSE) and the Maximum Likelihood Estimation (MLE). The method of MLE estimates parameters by solving a set of simultaneous equations and is better in deriving confidence intervals. However, the equation sets are typically complex and usually have to be solved numerically. On the other hand, the method of LSE minimizes the sum of squares of the deviations between what we actually observe and what we expect. Here we employ the methods of MLE and LSE to estimate the parameters of selected models. Table 1 shows the estimated values of all selected models when the methods of MLE and LSE are used.

When using LSE, the MBIWDA's weight assignment for the ND and LD were $\boldsymbol{W_{ND}=0.003}$, $\boldsymbol{W_{LD}=0.997}$, for the ND and ALD were $\boldsymbol{W_{ND}=0.002}$, $\boldsymbol{W_{ALD}=0.998}$, and for the LD and ALD were $\boldsymbol{W_{LD}=0.100}$, $\boldsymbol{W_{ALD}=0.900}$, respectively. The estimated results shown in Table 2 suggest that the MBIWDA can achieve excellent performance for reflecting the actual software size-change rate. As we can see in Table 2, the MBIWDA can get the relatively well rank expressed in parentheses in most criteria. That is, the combined models with MBIWDA demonstrated higher performance compared to the combined model with other weight decision approaches. Here we also devoted our attention to the comparisons of model performance when the method of MLE is used. Similarly, we separately calculated the weight using the MBIWDA. The ND and LD were $\boldsymbol{W_{ND}=0.839}$, $\boldsymbol{W_{LD}=0.161}$, for the ND and ALD were $\boldsymbol{W_{ND}=0.106}$, $\boldsymbol{W_{ALD}=0.894}$, and for the LD and ALD were $\boldsymbol{W_{LD}=0.020}$, $\boldsymbol{W_{ALD}=0.980}$, respectively. As shown in Table 3, the combined model with MBIWDA exhibited better performance in almost every criterion as did the LSE method in this dataset. Overall, the combined model with MBIWDA produced relatively high fit values.

Table 1. **Estimated values of model parameter**

ND (LSE)	μ=0.0163, σ=0.0119
LD (LSE)	μ=0.0172, b=0.011
ALD (LSE)	σ=0.0061, κ=0.3504
ND (MLE)	μ=0.0223, σ=0.0651
LD (MLE)	μ=0.0055, b=0.1532
ALD (MLE)	σ=0.0140, κ=0.482

Table 2. **Comparison results (LSE)**

ND+LD				
Criteria	MSE	R^2	K-S Test	Rank
LEW	0.001(3)	0.997(2)	0.082↓(4)	3
NLGEW	0.001(1)	0.997(3)	0.082↓(2)	2
NLHEW	0.001(3)	0.997(4)	0.082↓(2)	4
MBIWDA	0.001(2)	0.998(1)	0.008↓(1)	1
ND+ALD				
Criteria	MSE	R^2	K-S Test	Rank
LEW	0.0004(4)	0.998(4)	0.069↓(4)	4
NLGEW	0.0004(2)	0.998(3)	0.069↓(3)	2
NLHEW	0.0004(3)	0.998(2)	0.069↓(2)	3
MBIWDA	0.0003(1)	0.999(1)	0.059↓(1)	1
LD+ALD				
Criteria	MSE	R^2	K-S Test	Rank
LEW	0.0004(3)	0.998(2)	0.068↓(4)	2
NLGEW	0.0003(2)	0.998(3)	0.068↓(3)	3
NLHEW	0.0004(3)	0.998(4)	0.068↓(2)	3
MBIWDA	0.0003(1)	0.999(1)	0.061↓(1)	1

Table 3. Comparison results (MLE)

ND+LD					
Criteria	MSE	R^2	AIC	K-S Test	Rank
LEW	0.016(4)	0.924(4)	1089(2)	0.345↑(4)	4
NLGEW	0.013(3)	0.934(3)	26268(3)	0.341↑(3)	3
NLHEW	0.012(2)	0.938(2)	51718(4)	0.337↑(2)	2
MBIWDA	0.010(1)	0.953(1)	1034(1)	0.305↑(1)	1
ND+ALD					
Criteria	MSE	R^2	AIC	K-S Test	Rank
LEW	0.003(4)	0.985(4)	2462(1)	0.198↓(4)	4
NLGEW	0.003(2)	0.987(3)	26948(3)	0.189↓(2)	2
NLHEW	0.003(3)	0.987(2)	51435(4)	0.193↓(3)	3
MBIWDA	0.001(1)	0.995(1)	2730(2)	0.129↓(1)	1
LD+ALD					
Criteria	MSE	R^2	AIC	K-S Test	Rank
LEW	0.009(4)	0.959(4)	718(2)	0.256↑(4)	4
NLGEW	0.004(2)	0.978(3)	1704(4)	0.222↑(3)	3
NLHEW	0.004(2)	0.978(2)	-392(1)	0.191↓(2)	2
MBIWDA	0.001(1)	0.996(1)	792(3)	0.116↓(1)	1

V. CONCLUSIONS

To conclude, the present study is an application of multi-model combinations on the software size-change rate. Here we adopted different combinational models to describe the distribution of OSS size-change rates and our findings confirmed that OSS size-change rates can be fitted by the methods of multi-model combinations. We also provided a MBIWDA that determined the weight of the each candidate model. Furthermore, the proposed method was used to conduct experiments from real OSS project. Experimental results show that the proposed method could acquire better performance than other equally-weighted methods for fitting OSS size-change rate data. It has to be noted that our proposed combinations would not be restricted to a particular kind of candidate models or data sources. Additionally, it is also worth noting that the parameter estimations could become more complicated and tedious with the use of an increasing number of combined models. However, the additional calculations can be fully automated.

Acknowledgment

The work described in this paper was supported by the Ministry of Science and Technology, Taiwan, under Grants NSC 101-2221-E-007-034-MY2, NSC 101-2220-E-007-005, and MOST 103-2220-E-007-022.

References

[1] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 7th Edition, 2010.

[2] I. Herraiz, D. M. German and A. E. Hassan, "On the Distribution of Source Code File Sizes," *Proceedings of the 6th International Conference on Software and Data Technologies* (ICSOFT 2011) pp.18-21, Seville, Spain, July 2011.

[3] G. Concas, M. Marchesi, S. Pinna, and N. Serra, "Power-laws in a Large Oriented Software System," *IEEE Trans. on Software Engineering*, Vol. 33, Issue 10, pp. 687-708, Oct. 2007.

[4] D. Amit, and D. Riehle, "The Total Growth of Open Source," *Open Source Development, Communities and Quality*, pp. 197-209, Springer US, 2008.

[5] K. Samuel, T. J. Kozubowski, and K. Podgorski, "The Laplace Distribution and Generalizations: A Revisit with Applications to Communications," *Economics, Engineering, and Finance*, No. 183, Springer, 2001.

[6] S. Gennady, and M. S. Taqqu, "Stable Non-Gaussian Random Processes," *Econometric Theory*, Vol. 13, pp. 133-142, 1997.

[7] C. J. Hsu and C. Y. Huang, "Reliability Analysis Using Weighted Combinational Models for Web-based Software," *Proceedings of the 18th International World Wide Web Conference (WWW 2009)*, pp. 1131-1132, Madrid, Spain, April 2009.

[8] M. Mitzenmacher, "A Brief History of Generative Models for Power Law and Lognormal Distributions," *Internet Mathematics* Vol.1, No. 2, pp. 226-251, 2004.

[9] R. Cooper, and T. J. Weekes, *Data, Models, and Statistical Analysis*, Rowman & Littlefield, 1983.

[10] S. L. Pfleeger, F. Wu, and R. Lewis, *Software Cost Estimation and Sizing Methods: Issues and Guidelines*, Vol. 269. Rand Corporation, 2005.

[11] B.W. Boehm, *Software Engineering Economics*, Pearson Education, 1981.

[12] G. E. Klein, "The Sensitivity of Cash-flow Analysis to the Choice of Statistical Model for Interest Rate Changes," *Insurance: Mathematics and Economics*, Vol. 16, No. 2, pp.187-187, 1995.

[13] T. J. Kozubowski, and K. Podgorski, "A Class of Asymmetric Distributions," *Actuarial Research Clearing House*, Vol. 1, pp. 113-134, 1999.

[14] H. Okamura, T. Dohi, and S. Osaki, "Software Reliability Growth Model with Normal Distribution and its Parameter Estimation," *Proceedings of the IEEE International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE 2011)*, pp. 411-416, Xi'an, China, June 2011.

[15] G. Bottazzi, and A. Secchi, "Why are Distributions of Firm Growth Rates Tent-shaped?" *Economics Letters*, Vol. 80, Issue 3, pp. 415-420, Sep. 2003.

[16] M. R. Lyu, *Handbook of Software Reliability Engineering*, McGraw Hill, 1996.

[17] M. R. Lyu, and A. Nikora, "A Heuristic Approach for Software Reliability Prediction: the Equally-weighted Linear Combination Model," *Proceedings of IEEE International Symposium on Software Reliability Engineering (ISSRE 1991)*, pp. 172 -181, Austin, TX, USA, May 1991.

[18] M. R. Lyu, and A. Nikora, "Applying Reliability Models More Effectively (software)," *IEEE Software*, Vol. 9, Issue 4, pp. 43-52, July 1992.

[19] Y. S Su, and C. Y. Huang, "Neural-network-based Approaches for Software Reliability Estimation Using Dynamic Weighted Combinational Models," *Journal of Systems and Software*, Vol. 80, Issue 4, pp. 606-615, Apr. 2007.

[20] H. Li, M. Zeng and M Lu, "Adaboosting ‐ based Dynamic Weighted Combination of Software Reliability Growth Models," *Quality and Reliability Engineering International*,Vol. 28, Issue. 1, pp.67-84, 2012.

[21] M. Lu, S. Brocklehurst, and Bev Littlewood, "Combination of Predictions Obtained From Different Software Reliability Growth Models," *Predictably Dependable Computing Systems*, ESPRIT Basic Research Series, pp. 421-439, 1995.

[22] G. E. P. Box, and G. C. Tiao, *Bayesian Inference in Statistical Analysis*, Vol. 40. John Wiley & Sons, 2011.

[23] P. S. Bullen, *Handbook of Means and their Inequalities*, Springer, 2003.

[24] http //www.ohloh.net/, accessed 21 March 2013

[25] S. D. Conte, H. E. Dunsmore, V. Y. Shen, *Software Engineering Metrics and Models*, 1986

[26] P. Rook, *Software Reliability Handbook*, Elsevier Science Inc., New York, NY, USA, 1990.

[27] N. F. Schneidewind, "Finding the Optimal Parameters for a Software Reliability Model," *Innovations in Systems and Software Engineering*, Vol. 3, Issue 4, pp. 319-332, Dec 2007.

[28] K. Shibata, K. Rinsaka, and T. Dohi, "PISRAT: Proportional Intensity-based Software Reliability Assessment Tool," *Proceedings of the 13th IEEE Pacific Rim International Symposium on Dependable Computing* (PRDC 2007), pp. 43-52, Melbourne, Victoria, Australia, Dec. 2007.

[29] M. Xie, Q. P. Hu, Y. P. Wu, and S. H. Ng, "A Study of the Modeling and Analysis of Software Fault ‐ detection and Fault ‐ correction Processes," *Quality and Reliability Engineering International* Vol.23, Issue 4, pp. 459-470, Dec. 2007.

[30] G. Keller and B. Warrack, *Statistics for Management and Economics*, Duxbury, 1999.

[31] H. Akaike, "A New Look at the Statistical Model Identification," *IEEE Trans. on Automatic Control*, Vol.19, Issue 6, pp.716-723, Dec. 1974.

[32] A. L. Goel, "Software Reliability Modeling and Estimation Technique," *Technical Report, RADC-TR-82-263*, Rome Air Development Center, Oct. 1982.

[33] W. J. Conover, *Practical Nonparametric Statistics*, John Wiley and Sons,1980.

[34] G. Triantafyllos, and S. Vassiliadis, "Software Reliability Models for Computer Implementations— An Empirical Study," *Software: Practice and Experience* Vol.26, Issue 2, pp.135-164, Feb. 1996.

Shih-Min Huang *received the B.E. degree (2011) in Department of Information Management from National Central University, Taoyuan Taiwan and the M.E. degree (2013) in Institute of Information Systems and Applications from National Tsing Hua University, Hsinchu, Taiwan. He is currently an IT engineer in the Taiwan Semiconductor Manufacturing Company, Ltd. His research interests include software quality and software measurement.*

Chin-Yu Huang *(M'05) is a Professor in the Department of Computer Science at National Tsing Hua University (NTHU), Hsinchu, Taiwan. He received the M.S. (1994), and the Ph.D. (2000) in Electrical Engineering from National Taiwan University, Taipei, Taiwan. He was with the Bank of Taiwan from 1994 to 1999, and was a senior software engineer at Taiwan Semiconductor Manufacturing Company from 1999 to 2000. Before joining NTHU in 2003, he was a division chief of the Central Bank of China, Taipei. His research interests are software reliability engineering, software testing, software metrics, fault tree analysis, and system safety assessment. He has published over 100 papers in these areas. He received the Ta-You Wu Memorial Award from the National Science Council of Taiwan in 2008. He also received an Honorable Mention Best Paper Award in the 2010 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM 2010). In 2010, he was ranked at the 7th place of top scholars in Systems and Software Engineering worldwide between 2004 and 2008 by the Journal of Systems and Software based on his research on software reliability, software testing, and software metrics. He is a member of IEEE.*