International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.8 (2015), pp.1-12
http://dx.doi.org/10.14257/ijmue.2015.10.8.01

Residual Defect Prediction using Multiple Technologies

WanJiang Han[1], LiXin Jiang[2], TianBo Lu[1] and XiaoYan Zhang[1]

[1]*School of Software Engineering, Beijing University of Posts and Telecommunication, Beijing 100876, China*
[2]*Department of Emergency Response, China Earthquake Networks Center, Beijing 100036, China*
Email: hanwanjiang@bupt.edu.cn

Abstract

Finding defects in a software system is not easy. Effective detection of software defects is an important activity of software development process. In this paper, we propose an approach to predict residual defects, which applies machine learning algorithms (classifiers) and defect distribution model. This approach includes two steps. Firstly, use machine learning Algorithms and Association Rules to get defect classification table, then confirm the defect distribution trend referring to several distribution models. Experiment results on a GUI project show that the approach can effectively improve the accuracy of defect prediction and be used for test planning and implementation.

Keywords: *residual defect prediction, defect distribution model, software defect classification, defect trend, classifiers*

1. Introduction

With the rapid development of information technology, the application of computer software is more and more widely. High efficient and security software system is highly dependent on the software reliability, and the software defects have become the underlying causes which lead to the systems error, failure, collapse or even the disaster. In order to raise the effectiveness and efficiency of testing, software defect prediction has been used to identify defect-prone modules in an upcoming version of a software system and help to allow the effort on those modules [1].

Software defect prediction is a very important research topic in software engineering. It is based on the defect records in historical data to predict the defects in the future. It helps software project planning and process management.

The growing demand for higher operational efficiency and safety in industrial processes has resulted in a huge interest in defect detection techniques. Engineering researchers and practitioners remain concerned with accurate defect prediction when building systems. Defect prediction has both safety and economic benefits in technical systems by preventing future failures and further improves process maintenance schedules. Lack of adequate tools to evaluate and estimate the cost for a software system failure is one of the main challenges in systems engineering.

For a software development project, it is highly desirable to reduce software defects. A variety of information products emerged and the requirements of product quality are also increasing. By analyzing and studying defects, building defect model can help to find solutions for defects quickly and efficiently, which provides a good exploration to improve the quality of product and will help to forecast the reliability of the software and improve software reliability [2].

Software defects include detection defects and residual defects. Detection defects are the defects which were detected before software release. Residual defects are the defects

ISSN: 1975-0080 IJMUE

which were detected after software release. The purpose of residual defects' prediction is to keep the software defects number under the acceptable level in testing times [3-5].

During testing process, there will be a large accumulation of defect data, it is necessary to merge the similar defect to the same type of defect for unify solution easily. And defect model can predict residual defects. Defects can be effective managed and predicted by defect classification and prediction model [6].

Software defect prediction models are known as software reliability growth models. A large number of prediction models have been proposed and investigated over the last four decades [7].

This paper discusses the residual defect prediction based on more techniques using a large number of historical defect from testing. Firstly, this paper describes how to use classifiers to classify software defects, then gives the distribution of residual defects. Finally, the residual defects can be predicted. It provides a favorable recommendation for product development and design.

2. Related Classifier Models

From a holistic point of view, software defect prediction studies can be categorized as statistical and machine learning (ML) approaches. And the use of machine learning approaches to fault prediction modeling is more popular. Most defect prediction methods are based on product metrics [1].

Among these analysis studies, linear regression and logistic regression are the ones most used in defect prediction. Linear regression is often used to predict the number of defects a program unit contains, while logistic regression is usually used to predict the likelihood that a program unit is defect-prone [7].

In recent years, the use of machine learning algorithms (classifiers) has proven to be of great value in solving a variety of problems in software engineering including software defect prediction [8-9].

The classifier is a hypothesis about the true classification function that is learned from, or fitted to, training data. The classifier is then tested on test data. In supervised learning, especially for multivariate data, a classification function y = f (x) predicts one (or more) output attribute(s) or dependent variable(s) given the values of the input attributes [10].

This section will discuss three classifiers: Association rules, Decision tree and k-Nearest Neighbour [10-12].

2.1. Associations Rules

Association Rule, Proposed by Agrawal *et al* in 1993, is an important data mining model studied extensively by the database and data mining community. Assume all data are categorical. It is the task of discovering association rules that occur frequently in a given transaction data set [13].

An association rule is a pattern that states when X occurs, Y occurs with certain probability.

Its task is to find certain relationships among a set of data (item set) in the database.

Association rules mining is interested in finding frequent rules that define relations between unrelated frequent items in databases, and it has two main measurements: support and confidence values. Confidence values are measurements of rule's strength, while support value corresponds to statistical significance.

An association rule states that an item or group of items implies the presence of another item with some probability. Association rules reflect the relationship among data or is a study whether the generation of a data can speculate the generation of another data. Data association reflects the relationship in Database. The association

degree can be expressed through support and confidence. Thus, finding the relationship among data in the transaction database is the mining purposes of association rule.

Apriori algorithm has been shown as a classical association rule in mining algorithms that have been cited so far, which is used to find frequent item sets in a database and to generate Association Rules from the frequent item sets. Algorithm is how to regenerate all the frequent itemsets. It uses an iterative approach layer by layer, applying K-Itemset to search for (K+1)-Itemset [14].

2.2. Decision Tree

Decision tree (DT) induction is one of the simplest and yet most successful forms of supervised learning algorithm. It has been extensively pursued and studied in many areas such as statistics and ML [15] for the purposes of classification and prediction.

DTs are non-parametric (no assumptions about the data are made) and a useful means of representing the logic embodied in software routines. A decision tree takes as input a case or example described by a set of attribute values, and outputs a Boolean or multi-valued decision.

A classification tree, as opposed to a regression tree means that the response variable is qualitative rather than quantitative. In the classification case, when the response variable takes value in a set of previously defined classes the node is assigned to the class which represents the highest proportion of observations. Whereas, in the regression case, the value assigned to cases in a given terminal node is the mean of the response variable values associated with the observations belonging to a given node. Note that in both cases, this assignment is probabilistic, in the sense that a measure of error is associated with it. Clustering trees just group instances in leaves.

2.3. K-Nearest Neighbour

One of the most venerable algorithms in machine learning is the nearest neighbour (NN). Nearest-neighbour methods are sometimes referred to as memory-based reasoning or instance-based learning (IBL) or case-based learning (CBL) techniques and have been used for classification tasks. They essentially work by assigning to an unclassified sample point the classification of the nearest of a set of previously classified points.

A further non-parametric procedure of this form is the k-nearest neighbour (k-NN) approach. To classify an unknown pattern, the k-NN approach looks at a collection of the k nearest points and uses a “voting” mechanism to select between them, instead of looking at the single nearest point and classifying according to that with ties broken at random. If there are ties for the k^{th} nearest observations, all candidates are included in the vote [16].

3. Related Defect Distribution Model

Software defect prediction technology includes lots of models. Here, we will discuss several models used in this paper.

3.1. Rayleigh Distribution Model

Rayleigh model is a common reliability model. It can forecast the defect distribution during software life cycle. This kind of model is based on the WeiBull statistical distribution. WeiBull distributed reliability analysis is widely used in different field. The

probability density function of WeiBull distributed tag end is gradually converge to 0, but never equal to 0. The experts of Trachtenberg and IBM have researched on the software project faults and they found that the distribution accords with Rayleigh distribution model. The probability distribution density function of Rayleigh model is as in (1).

$$f(t) = 2K(t / c^2)e^{-(t/c)^2} \quad (1),$$

the cumulative distribution function is as in (2)

$$F(t) = K(1 - e^{-(t/c)^2}) \quad (2),$$

where K is the total faults. T stnds for time, c is an constant and $c = \sqrt{2}t_m$, where t_m is the time when $f(t)$ reached maximum, and $F(t_m) / K$ approximately equal to 0.4. Therefore, we can estimate the total number of defects at a certain time as well as the specific Rayleigh distribution parameter. Thus we can simplify the counting process. It is easy to control the quality of the enterprise performance goal by using Rayleigh model. It is necessary to take measures to correct it when the process appeared abnormal [15-16].

3.2. Exponential Distributed Model

Exponential model accords with defect distribution for the testing phase, especially the acceptance testing phase. The basic principle is that the defect occurs at this stage or the failure mode is a good indication of the reliability of the product. Exponential model, also known as reliability growth model, is divided into fault/failure count model and the failure interval model. The defect probability distribution's density function of the exponential model is as in (3).

$$f(t) = K(\lambda e^{-\lambda t}) \quad (3)$$

And the defects cumulative distribution function is as in (4).

$$F(t) = K(1 - e^{-\lambda t}) \quad (4)$$

Where t is time, K is the total number of defects, λ is the defect detection rate or failure occur rates. Exponential model is the most simple and most important model in software reliability research, and also is a basis of many other reliability growth model. Its basic assumptions are:

- Each defect leading to failure found in test phase is equal;
- The defect repair time is negligible;
- Each defect can be perfectly repaired and the repair will do not introduce new defects, and so on.

Defects exponential probability density function and cumulative distribution function are shown in Figure 1 and Figure 2 [17].

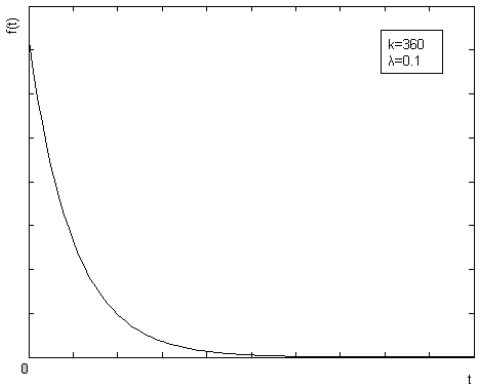

Figure 1. The Defect Exponential Distribution Probability Density Function

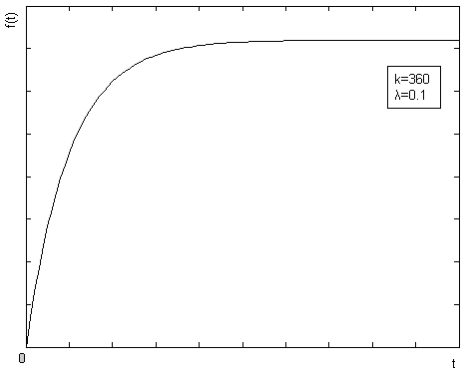

Figure 2. The Defects Exponential Cumulative Distribution Function

3.3. S-curve Distributed Model

Yamada and other people proposed a test period not only includes detecting the defects but also includes isolating defects. When there comes a defect, we need to find the reason why defects are disabled. Therefore, there exists a time delay for finding a defect until we report it. The accumulated time delayed defects accord with S curve distribution and this is called delayed S-curve model which is also a reliable increment model. S-curve model meet the requirement of nonhomogeneous Poisson process. Its CDF is as in (5).

$$F(t) = K(1 - (1 + \lambda t)e^{-\lambda t}) \tag{5}$$

where t represents time; k represents total fault number; λ represents the possibility of detecting a fault. The PDF is as in (6).

$$f(t) = K\lambda^2 t e^{-\lambda t} \tag{6}$$

In 1984, Oghba [20] proposed another S-curve distributed model called transformed S model. This model considers the detected faults are interdependent and the more defects are detected the more faults will be detected later. The accumulated fault distributed function (CDF) is as in (7).

$$\mathrm{F(t)} = \mathrm{K}(1 - \mathrm{e}^{-\lambda t}) / (1 + \phi \mathrm{e}^{-\lambda t}) \tag{7}$$

where t represents time; k represents total fault number; λ represents the possibility of detecting a defect. Its PDF is as in (8) [17].

$$\mathrm{f(t)} = \mathrm{K}\lambda \mathrm{e}^{-\lambda t} \mathrm{K}(1 + \phi) / (1 + \phi e^{-\lambda t})^2 \tag{8}$$

3.4. The Lognormal Distribution Model

Lognormal model which can predict the defect distribution in the entire life cycle of software is a most commonly used reliability model. Lognormal model is based on the Weibull statistical distribution, and Weibull distribution is widely used statistical distribution family in different areas of a reliability analysis, which has a large number of statistical data to support [17-19].

Set software defects in the program block, B1, B2, Bm are m blocks of the program; b1, b2, bn are n branches of the program. The probability of each branch being executed obeys normal distribution, the probability of each block being executed can be expressed as in (9).

$$\mathrm{P(B)} = \prod_{j=1}^{n} \mathrm{P}(\mathrm{b}_{ij}) \tag{9}$$

Since $P(b_{ij})$ obeyed normal distribution, according to the central limit theorem, $P(B_i)$ obeys the lognormal distribution. The lognormal distribution of defects exposure rate λ can be expressed as in (10).

$$P(\lambda) = \frac{N}{\lambda\delta\sqrt{2\pi}} e^{\frac{-(\ln\lambda-\mu)^2}{2\delta^2}} \tag{10}$$

Where N is the initial number of defects; λ as a variable; μ is the average of the logarithm of λ; δ^2 is the variance of the logarithm of λ. Defects lognormal probability density

Function and cumulative distribution functions are shown in Figure 3 and Figure 4.

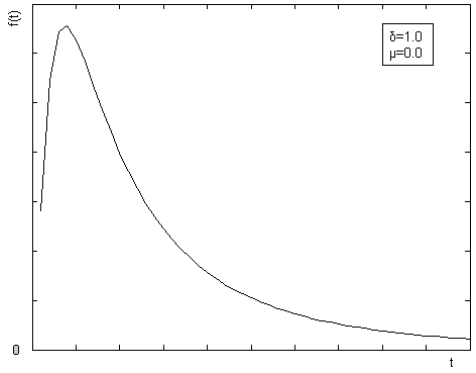

Figure 3. Defects Probability Density Function

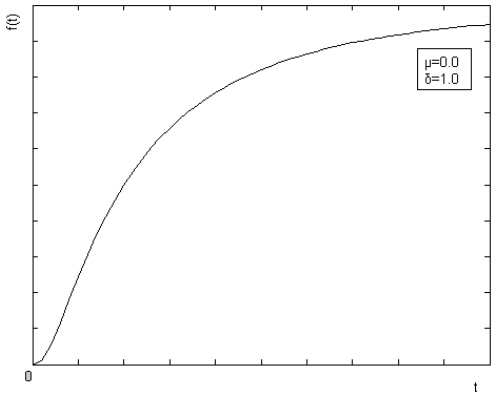

Figure 4. The Number of Defects Cumulative Distribution Function

3.5. Bayesian Belief Networks

Bayesian Belief Networks (BBNs) is a graphical network. The graph is made up of nodes and arcs where the nodes represent uncertain variables and the arcs represent the causal relationships between variables.

There are many advantages of using BBNs. The benefits of using BBN include:

- specification of complex relationships using conditional probability statements;
- use of "what - if? Analysis and forecasting of effects of process changes;
- explicit modeling of "ignorance" and uncertainty in estimates, forecasting with missing data.

There are three stages to set up software residual defects prediction model using BBNs : (1) Setting variables set and field, (2) Constructing Network topology, (3) ascertainment local probability distribution. Conditional probability expresses the relation of nodes and their parents' nodes. Its transcendental probability is conditional probability which has no parents node. These three steps are implemented interactively, not serially [17].

4. Defect Classification using Classifiers

Through a large number of test data, defect distribution can be summed up. We applied association rules of data mining, and studied the relative defects, which can analyze efficiently and prevent product defects, then improve product quality.

An association rules-based method has the strength of not requiring prior knowledge on the system. The association rules approach compares with existing data mining methods but has the slight advantage of detecting some anomalies which (otherwise) could have been overlooked by conventional approaches.

In order to show the steps of residual defect prediction, this paper uses a typical GUI defect data to analyze the classification and defects distribution. The data is shown in Table 1.

Table 1. GUI Defect Classification

Term	Defect classification
1	Interface display problem
2	Interface indication problem
3	Wrong character problem
4	Punctuation format problem
5	Inconformity semantic expression
6	Unreadable codes
7	Messy versions
8	Inconformity pictures
9	Inappropriate learning problem
10	Sequencing problem
11	Boundary problem

From Table 1, we use the association rules-based technology to get a defect association relationship, shown as Table 2.

Relative analysis informs that a happening of one defect may incur another or many other defects occur. For example, the problem of ‘interface displaying’ is associated with ‘interface indication problems’,’ consistency problems’ and ‘boundary problems’.

After the threshold values of minimum support and minimum confidence support are given, frequent item sets and association rule will be generated by applying Apriori arithmetic. We focus on the strong association rule which is the rule satisfying the minimum support and minimum confidence [20-22].

Table 2. GUI Defect Association

Defect types	Defect associations
Interface display problem	Interface indication problem, consistency problem, boundary problem
Interface indication problem	Interface display problem, consistency problem, boundary problem, messy codes

Defect types	Defect associations
Messy Versions	Interface display problem
Graphical inconsistency	Interface indication problem, consistency problem
Learnability problem	Interface indication problem, consistency problem, boundary problem
Boundary problem	Interface display problem, Interface indication problem, Consistency problem
Consistency problem	Interface display problem, Interface indication problem, Boundary problem, Graphical inconsistency

By this relationship, we can predict more defects.

5. Defect Prediction using Distribution Model

After getting the defect relationship, we can study the defects distribution, then we can predict the residual defects. In order to obtain accurate distribution of software defects, this paper uses many typical project defect data to analyze the distribution of defects. Table 3 shows the average data of several projects.

Table 3. Defect Data

Date	Defects
1	6
2	7
3	9
4	11
5	13
6	15
7	16
8	19
9	22
10	25
11	28
12	31
13	35
14	40
15	46
16	51

We draw the defect trend picture using data from Table 3, as shown in Figure 5. We can infer from the trend of the data fitting curve that the distribution of software defects is more consistent with the lognormal distribution. So we can select lognormal distribution model to predict residual defects.

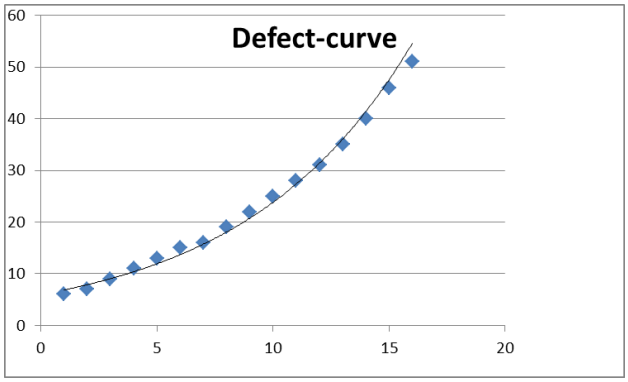

Figure 5. Defect Trend Fitting

In our study, the least square method is used to make the regression analysis about the parameters of the model.

In the last, we have achieved the expression of the model, the analysis results are presented in (8).

$$F(t) = 123(1 - e^{-0.0325t}) \tag{8}$$

Then, using this model to predict the project defects during the test phase. Figure 6 shows the difference between estimated data and actual data. Obviously, the two curves are basically identical.

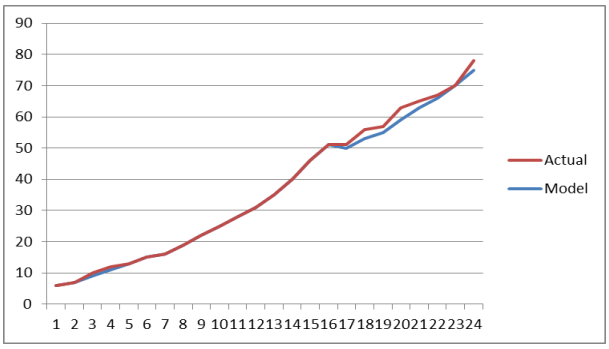

Figure 6. The Comparison between the Predicted and Actual Data

We can evaluate the accuracy of estimation by MRE (Magnitude of Relative Error). Namely,

$$MRE_i = \frac{Actual_i - Estimated_i}{Actual_i}$$

The MMRE (Mean of the Magnitude of Relative Error) is also a useful evaluation tool, Its Equation is as in (9).

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \tag{9}$$

For Figure. 6, the MMRE is 8.6%. The calculated MMRE of similar projects is close to 8.9%.

6. Conclusions

This paper analyzed the related technologies about classifiers and distribution model. From the representative collected software defects data of GUI projects, the

paper used several classifier algorithms to get defect classification table, then applied mathematical methods to show that the distribution of this kind of software project defects is consistent with the lognormal distribution better. If we can find out which distribution the software defects obeyed in accordance with the defects classification, we can use the fault injection method to simulate software fault, and study the accelerated test method under certain defects distribution, which can effectively improve the software test coverage, reduce test time, reduce cost of test.

Further study will continuously work on specializing projects and keep on researching other prediction model as well.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61170273).

References

[1] Xiao D. M., Rui H. C. and Li Z., "Software Defect Prediction Based on Competitive Organization CoEvolutionary Algorithm," Journal of Convergence Information Technology (JCIT), vol. 7 no. 5, March, **(2012)**.
[2] Gaffney J. E., "Estimating the number of faults in code," IEEE Transactions on Software engineering, vol. 10 no. 6, **(1984)**, pp. 459- 464.
[3] Fenton N. and Neil M., "A Critique of Software Defect Prediction Models," IEEE Transaction on Software Engineering, vol. 25 no. 5, **(1999)**, pp. 675- 689.
[4] May J., "Fault prediction for software," Oxford: Oxford University Press, **(1995)**.
[5] C. Wohlin and P. Runeson, "Defect Content Estimations from Review Data, In Proceedings of the Twentieth International Conference on Software Engineering," IEEE Computer Society Press, **(1998)**, pp.400–409.
[6] Fenton N., Krause P. and Neilm M., "A probabilistic Model for Software Defect Prediction."
[7] Kazu O., "Software Defect Prediction Based on Stability Test Data," IEEE, **(2011)**.
[8] L. Breiman, "Bagging predictors," Machine Learning, vol. 26 no. 2, **(1996)**, pp. 123-140.
[9] L. Breiman, J. Friedman, R. Olshen and C. Stone, (1984). "Classification and Regression Trees," Wadsworth, **(1984)**.
[10] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting and variants," Machine Learning, vol. 36 no. 1/2, **(1999)**, pp. 105-139.
[11] T. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," Machine Learning, vol. 40 no. 2, **(2000)**, pp. 139-158.
[12] Bhekisipho T., "Software Faults Prediction Using Multiple Classifiers," *www.cnki.net.*
[13] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," Data Mining and Knowledge Discovery. Kluwer Academic Publishers, **(1998)**.
[14] Wanjiang H.,Tianbo L. and Sun Y., "Research on the Problem Model of GUI based on Knowledge Discovery in Database," Proceedings of the 2013 International Conference on Software Engineering and Computer Science, **(2013)**, pp. 5-9.
[15] J.R. Quinlan. C.4.5: Programs for machine learning. Los Altos, California: Morgan Kauffman Publishers, INC., 1993.
[16] P. Domingos, "On the optimality of the simple Bayesian classifier under zero-one loss," Machine Learning, vol. 29, **(1997)**, pp. 103-130.
[17] Ouyang X., Shuai C. Y., "The research of lognormal probability distribution of software defects," International Conference on Circuit and Signal Processing, **(2010)**.
[18] Mullen R., "The lognormal distribution of software failure rates application to software reliability growth modeling," Proc of the 9th international symposium on software reliability engineering Washington DC: IEEE computer society, **(1998)**, pp. 134.
[19] Mullen R., "The lognormal distribution of software failure rates: origin and evidence," Proc of the 9th international symposium on software reliability engineering Washington DC: IEEE computer society, **(1998)**, pp. 124.
[20] Wanjiang H., "Research of the Defect Model Based on Similarity and Association Rule," unpublished.
[21] Wanjiang H., "Research on Software Fault Distribution for Web Application," unpublished.
[22] Li C. and Fan M., (2004) "Generating association rules based on threaded frequent pattern tree. Comput Eng Appl, (in Chinese), vol. 4, **(2004)**, pp. 188-192.

Authors

Wan-Jiang Han, was born in Hei Long Jiang province, China, 1967. She received her Bachelor Degree in Computer Science from Hei Long Jiang University in 1989 and her Master Degree in Automation from Harbin Institute of Technology in 1992.

She is an assistant professor in School Of Software Engineering, Beijing University of Posts and Telecommunication, China. Her technical interests include software project management and software process improvement.

Li-Xin Jiang was born in Hei Long Jiang province, China, 1966. He received his Bachelor Degree and Master Degree in physical geography from Beijing University in 1989 and 1992.

He is a professor in the department of Emergency Response of China Earthquake Networks Center. His technical interests include software cost estimation and Emergency Response software development.

Tian-Bo Lu was born in Guizhou Province, China, 1977. He received his Master Degree in computer science from Wuhan University in 2003 and his PH.D Degree in computer science from the Institute of Computing Technology of the Chinese Academy of Sciences in 2006.

He is an Associate professor in School of Software Engineering, Beijing University of Posts and Telecommunications, China. His technical interests include information and network security, trusted software and P2P computing.

Xiao-Yan Zhang was born in Shandong Province, China, 1973. She received her Master Degree in Computer Application in 1997 and her PH.D Degree in Communication and information system from Beijing University of Posts and Telecommunication, China, in 2011.She is an Associate professor in School of Software Engineering, Beijing University of Posts and Telecommunications, Beijing, China. Her technical interests include software cost estimation and software process improvement.

