Chinese Journal of Electronics
Vol.26, No.2, Mar. 2017

Decision-Level Defect Prediction Based on Double Focuses*

HU Changhong[1], XUE Xucheng[1], HUANG Liang[1,2], LYU Hengyi[1,2], WANG Heqi[1], LI Xiangzhi[1], LIU Hailong[1], SUN Ming[1] and SUN Wu[1,2]

(1. *Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130022, China*)

(2. *University of Chinese Academy of Sciences, Beijing 100084, China*)

Abstract — This research mainly expounds upon the decision-level software defect prediction theory. The defect characteristics is the first research focus. For the first research focus, a characteristic comparison set is built out of the existing defect characteristics according to the dissimilarity of defect characteristics and the defect characteristics are organized outside the characteristic comparison set into some defect characteristic clusters to reduce the scale of the characteristic data. The defects is the second research focus. For the second research focus, the vector weights are assigned to the defect characteristics contained in the defects according to the minimum critical characteristic set. Moreover, the multi-agent algorithm integration technology is used to predict defects according to the repulsive relationship between similar defect clusters.

Key words — Decision, Prediction, Defect, Focus.

I. Background

Software defect prediction remains a new research area in software engineering. In the birth stage of computers and electronics, computer hardware had many defects, and nobody cared about software defects and their impacts on the quality of software systems. Therefore, software defects did not arouse widespread attention and concern. Recently, the reliability and quality of computer hardware have been continuously improved while computer software defects are increasingly prominent. In the face of an obvious software crisis, more importance is attached to the detection and repair of software defects[1−3]. Because the quantity and negative effect of software defects are underestimated, software defects have become one of the main reasons for the failure of large engineering projects, leading to huge economic losses in the 21st century. Thus, ensuring the quality of software systems enjoys top priority in software development and is an urgent issue that needs to be timely identified to predict software defects for software development. For a major engineering project, it is essential to measure, predict, and evaluate software defects. Especially for a major research and development (R&D) project in the fields of aviation, spaceflight, and national defense, software defects will severely affect the R&D progress and even cause failure of the project, thus leading to huge financial, manpower, and material losses. Defect prediction is the only way to accurately locate the defect distribution in a large project, shorten the downtime, reduce the overall project risk, and ensure the overall project quality. Therefore, software defect prediction has become a requisite means for the smooth implementation of a large engineering project. In summary, software defect prediction is of great significance in ensuring the quality of a large engineering project.

With the continuous advancement in information science, a great variety of software (especially large-scale embedded software) has been developed, which tends to be more complex, larger, and diversified. In this situation, software defects are on the increase. Each software defect is generated under different conditions and environments, and hence differs in its specific characteristics. A software defect may have an enormous negative effect upon software quality. Usually, such massive and complex defect data and defect characteristic data are referred to as big data. For research on software defect prediction, determining how to statistically classify, model, and formularize the defects according to the defect characteristics and thus effectively predicting the possible defects contained in software remains a critical issue. If the weight of each defect characteristic in the defect characteristics

*Manuscript Received Apr. 11, 2016; Accepted Sept. 19, 2016. This work is supported by the National Natural Science Foundation of Jilin Province in China (No.20150520059JH)

 DOI:10.1049/cje.2017.01.005

is exhaustively used to predict the defects, the complexity of computation will be increased exponentially (as illustrated in Fig.1), thus greatly affecting the prediction efficiency. Too many factors (defect characteristics and defects) need to be considered during the prediction process, thus affecting the accuracy of prediction. As set forth in Ref.12, Tim Menzies of University of North Carolina discussed the prediction effect of regional data and global data and observed the following phenomenon: 1) regional defect prediction was usually more effective and efficient than global defect prediction and 2) in most cases, the prediction effect for regional data was better than that for big data except that not all projects had sufficient regional data. Therefore, a new hot spot of research has become determining how to reduce the computational scale of the defects and defect characteristics during the defect prediction process and how to more clearly and simply distribute the defects.

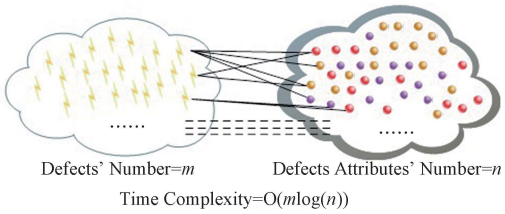

Fig. 1. Complexity

II. Related Studies

Today, the globally prevailing technologies of software defect prediction can be roughly classified into two types: static software defect prediction and dynamic software defect prediction.

Static software defect prediction includes metric-based defect prediction, defect prediction based on software defect distribution, and model-based defect prediction.

1. Metric-based defect prediction

The metric-based defect prediction technologies include 1) software defect prediction based on software scaleseven typical models and standards: defects per thousand source code lines (also called CMMI standard), Akiyama model, predicate model, Halstead model, Lipow model, Gaffney model, and Compton and Withrow model and 2) software defect prediction based on software complexity, which is represented by the McCabe Cyclomatic Complexity Metric proposed by McCabe & Associates in the 1970s.

2. Defect prediction based on software defect distribution

Defect prediction based on software defect distribution is a typical analytical and reasoning technique, including Principal component analysis (PCA), Linear discriminant analysis (LDA), Boolean discriminant function (BDF), Clustering analysis (CA), and regression analysis (typical methods include Artificial neural network (ANN), Multiple regression analysis (MR), and Case-based reasoning (CBR)).

3. Model-based defect prediction

Model-based defect prediction includes the Constructive quality model (COQUALMO), Defect removal efficiency (DRE) model, and Bayesian network model.

Model-based defect prediction technologies are also dynamic software defect prediction technologies, which are also called multi-objective software defect prediction technologies. Typical model-based defect prediction technologies include the Rayleigh model, exponential distribution model, and S-curve distribution model.

Between 2012 and 2014, several scholars successively posited various methods, theories, and models for software defect prediction. The following section introduces typical examples, including static and dynamic software defect prediction technologies:

1) In 2013, Professor Tim Menzies (From the faculty of Computing Science at West Virginia University, he is currently working at the University of North Carolina, and his paper on defect prediction ranks in the top five globally by citation count) published a paper on the effect of defect prediction results on global and local defect samples based on machine learning. In the paper, he comprehensively analyzed the effects of various prediction methods and algorithms on the defect prediction for global and local defect samples. He ignored the defect classification for global defect samples, made full use of existing local defect samples, and utilized, under certain constraints, the defect samples closely correlated to the data to be predicted, finally determining that constrained use of local samples could sometimes have a better effect than the use of global samples.

2) In August 2013, Tim Menzies together with researchers at Tsinghua University studied the equilibrium issue in defect prediction for the inter-institutional private defect samples and global defect samples and evaluated the effect of the CLIFF+MORPH algorithm in defect prediction, finding that the CLIFF+MORPH algorithm is significantly more effective than other algorithms in equilibrium defect prediction[4,5].

3) In 2014, Robert M. Bell, Thomas J. Ostrand, and Elaine J. Weyuker from AT&T Laboratory noted that the effect of local defect data on prediction results was very limited[6].

4) In 2014, Xiamen University and the University of Science and Technology of China jointly noted that the efficiency of defect prediction could be significantly improved if defect prediction was designed to incorporate various requirement information.

5) In 2012, Lourdes Pelayo and Scott Dick from the University of Alberta (Canada) noted that the stratified

selection technology could significantly improve the equilibrium of global and local defect data in defect prediction and further increase the accuracy of defect prediction[7].

6) In 2013, Gerardo Canfora, Andrea De Lucia, and Massimiliano Di Penta of Italy jointly proposed a multi-agent and cross-project defect prediction algorithm. By integrating different curve prediction algorithms, the algorithm could integrate and calculate the defect prediction samples of multiple projects simultaneously, thus improving the efficiency of defect prediction.

7) In 2012, Jayalath Ekanayake and Jonas Tappolet of Switzerland studied the effect of time shift upon defect prediction and verified the effect for the first time.

8) In 2012, Zhongbin Sun, Qinbao Song, and Xiaoyan Zhu (from Xi'an Jiaotong University) proposed a code-based machine-learning algorithm that used the global defects contained in codes as samples. The machine learning algorithm could significantly improve the efficiency of defect prediction[8].

Among the methods, algorithms, and models described above, 1)–5), 7), and 8) are static software defect prediction technologies,[9,10] and 6) is a dynamic software defect prediction technology. In addition, 1)–3), 5), and 8) are software defect prediction technologies based on machine learning[11−15], 4) is a multiple regression analysis technology, 6) is a multi-objective defect prediction technology, and 7) is a defect prediction technology based on time shift[16−19]. These methods represent not only the latest achievements made in software defect prediction in recent years but also the global research trend.

III. Problem Description

Today, academic studies on software defect prediction mainly focus on cross-project defect data, multi-agent integration of historical defect data, multi-agent stratification, the effects of different defect data on prediction results, and integration of prediction algorithms. Most scholars focus their studies on algorithms for software defect prediction and have published over tens of thousands of technical papers accordingly. They have a far higher enthusiasm for algorithms than for software defects themselves. Similarly, they are accustomed to applying their algorithms to the existing defect data and comparing the experimental results with those attained using other algorithms. Such defect prediction methodology is called algorithm-level defect prediction.

Meanwhile, a minority of scholars and engineers focus instead on defect distribution, selecting optimal algorithms according to the analysis of defect data distribution and characteristics and applying the optimal algorithms to the data used for defect prediction. Such defect prediction methodology is called panoramic defect prediction. However, the ultimate goal of defect prediction is not to compare the advantages/disadvantages of different algorithms or to select an algorithm considered by scholars to be the most suitable for defect prediction but to make a decision on defect prediction by scholars or engineers. Comparison is the easiest way to make a decision. In Fig.2, sets of stereographs with three types of characteristics are presented, and to be compared, graphic a can only possess one of the three types of graphic characteristics. By comparison, it is apparent that graphic a does not belong to Set A and surely belongs to Set B. Therefore, it can be concluded that if Set A is determined, the remaining graphics surely belong to Set B. While the graphics with three types of characteristics are classified, a set of graphics with similar characteristics can be obtained only by determining a set uncorrelated to the graphics to be classified.

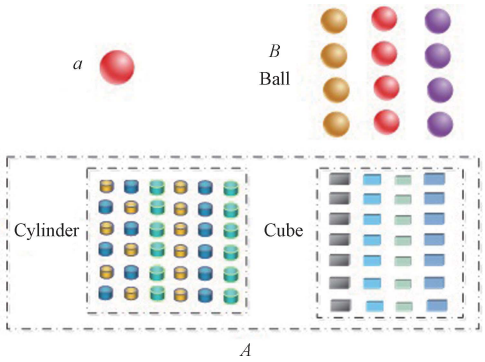

Fig. 2. Contrast

The defects is processed according to their distribution and characteristics, and a comparison set is determined according to the dissimilarity between defects. According to the comparison set, defects are clustered to reduce the scale of defects, and according to the repulsive relationship between different data, the scale of data is further reduced. In this manner, the defect prediction will acquire higher efficiency and lower complexity. The results of defect prediction will provide a basis for judging software quality and repairing software defects. This defect prediction methodology is called decision-level defect prediction. As illustrated in Fig.3, there only exist two types of stereographic characteristics and three graphics (a, b, and c). a and c are repulsive to each other, and b and c are also repulsive to each other. Then, a and b are surely attractive to each other and thus are clustered. During the clustering process, the relationship between a and b can be determined only by determining the c that is repulsive to both a and b. During the process of defect prediction, the remaining data are the data similar to the data used for defect prediction only by finding the data repulsive to the data used for defect prediction. Algorithm-level defect prediction and panoramic defect prediction involve too many subjective factors, ignore its own characteristics

of defects, and resort to certain algorithms that are extremely complex, inefficient, and difficult to understand. Therefore, these technologies cannot ensure the objectivity of the defect prediction results and severely affect the judgment of the software quality and repair of software defects.

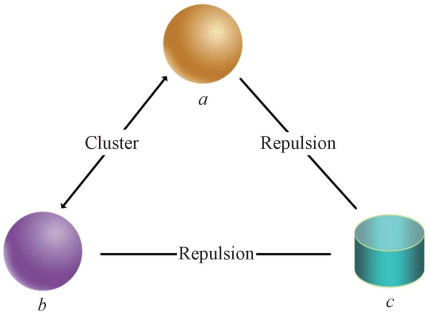

Fig. 3. Repulsion

To overcome the above deficiencies, this paper puts forth decision-level defect prediction, which focuses on the defects and defect characteristics. Using three technical means (including defect simulation prediction, laboratory software testing, and laboratory defect gathering), this paper predicts the bounded defects and uncovers the effect upon defect prediction made by the minimum critical characteristic set of two focuses of the bounded software. Bounded defect prediction refers to the defect prediction for the local and global defect data that is similar to the local defect data under certain constraints. The defects and defect characteristics for defect prediction are similar two walking persons involved in a mathematical meeting problem. If the two persons walk in the same direction, it will be difficult for them to meet (the process is time-consuming). Even if they can meet each other, their total walking distance will be greater than the relative distance between them, the meeting point will surely be beyond the shortest distance between them, and it is difficult to accurately predict the meeting point. If their speed difference is less than zero, they will never meet each other. If they walk in directions opposite to each other, significant time will be saved, their total walking distance will be equal to the relative distance between them, and the meeting point can be accurately positioned within the shortest distance between them. For details, refer to Fig.4. If defect prediction is studied from the perspective of double focuses, defect prediction will acquire higher efficiency and accuracy, and the negative effects of human factors will be reduced. However, it is difficult to determine the minimum critical characteristic set of the defect characteristics and use the minimum critical characteristic set for defect prediction. To attain this goal, this paper expounds on the following key issues: the extraction and acquisition of defect characteristics, dissimilarity between defect characteristics, clustering of defect characteristics, modeling for the characteristic comparison set, modeling for the minimum critical characteristic set, modeling for the defect comparison set, algorithm and modeling for reducing the scale of the defects via the minimum critical characteristic set, and methodology and modeling for defect prediction. This study will provide a theoretical and model basis for improving the efficiency and accuracy of defect prediction, developing the defect prediction software, and assuring the software quality in aerospace projects and help to reduce the risks of failure arising from software defects in spaceflight, aviation, and other major projects.

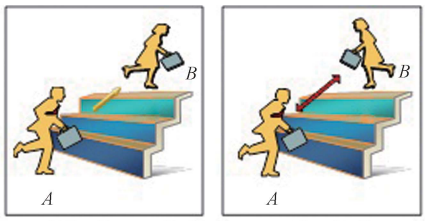

Fig.4(*a*) Fig.4(*b*)

If A' velocity <B' velocity then
In Fig.4(*a*) A never meet B;
In Fig.4(*b*) time cost is distance/(A' velocity + B' velocity):
elsif A' velocity > B' velocity then
In Fig.4(*a*) time cost is distance/(A' velocity -B' velocity):
In Fig.4(*b*) time coost is distance/(A' velocity + B' velocity):
elsif A' velocity - B' velocity then
In Fig.4(*a*) A never meet B:
In Fig.4(*b*) time cost is distance/(A' velocity + B' velocity):
End if;

Fig. 4. Time cost

IV. Defect and Characteristic Comparison Models

The Negative_K_means method will be introduced to the defect comparison model and characteristic comparison model. First, data are preprocessed via the CANOPY algorithm to determine the category value K contained in Negative_K_means. As demonstrated in Fig.5, a start point p is then randomly selected from the data set ALL (the data can be defect characteristics or defects), and the data node a_1 that is the farthest from p is selected for use as the first cluster center to generate the a_1-centered set A_1. The second cluster center a_2 is the farthest from the central point between p and a_1, and the a_2-centered cluster A_2 is generated. There exists no intersection between A_2 and A_1. The third cluster center a_3 is the farthest from the central point between p and a_2, and the a_3-centered cluster A_3 is generated. There exists no intersection between A_3 and A_2, A_1. Likewise, the kth cluster center a_k is the farthest from the central point between p and a_{k-1}, and the a_k-centered cluster a_k is generated. There exists no intersection between a_k and a_{K-1} to $A_1 \cap a_K \cup A_{K-1} \cup \ldots A_1$ is a comparison data set, and A is a similar data set.

V. Multi-agent Integration Mathematical Model

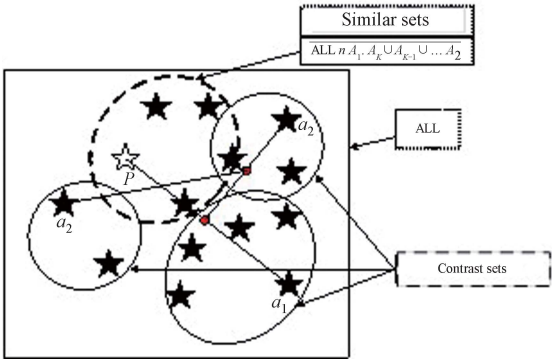

Fig. 5. Comparison model

The attraction and repulsion strategies determine the quantity of minimum characteristic sets and create the prediction Eq.(1). w is the weight undetermined coefficient, n is the dimensionality of the minimum characteristic set, x is the characteristic state, and the order u is an undetermined coefficient. Eq.(2) is a SVM prediction equation, where x is the data point of a defect. Eq.(3) is a Bayesian forecasting equation, where H is the initial probability, $P(H)$ is a prior probability, $P(D)$ is the prior probability of data D, $P(D|H)$ is the probability of data D based on the assumption of H, and $P(H|D)$ is a posterior probability. Eq.(4) is a neural network prediction equation, where p is the input defect value, O_i is an output value, and $\boldsymbol{W}$ is an undetermined coefficient.

$$V(b) = w_0 + w_1 x_1 + w_2 x_2^2 + w_3 x_3^3 + \cdots + w_n x_n^u \quad (1)$$

$$\text{SVM agent: } f(x) = \boldsymbol{W}^{\mathrm{T}} x + b \quad (2)$$

Bayesian agent:

$$p(H|D) = P(D|H)^* P(H)/P(D) \quad (3)$$

Neural network agent:

$$O_i = F_L(\cdots(F_2(F_1(P_i W^{(1)})W^{(2)})\cdots)W^{(L)}) \quad (4)$$

First, the quantity of variables x in the prediction equation is determined according to the quantity of defect characteristics in the minimum characteristic set, and Eq.(2) is created with the unknown quantities w and u. Then, multi-dimensional binary classification of the defect learning data is conducted via the SVM (agent) Eq.(2), and initial values are assigned to w and u. Subsequently, the Bayesian agent (Eq.(3)) is used to calculate the posterior probability of each characteristic according to the calculated real classification, and the w and u values are adjusted. Finally, the neural network agent (Eq.(4)) is used to conduct iterations and result comparison for Eq.(1) and real calculation, adjust the w and u values accurately according to the differences between the real and calculated values, and ultimately determine Eq.(1).

VI. Experimental Results

Within a period of 36 months, the software defects involved in an astronautic camera project were tracked and predicted. Feedback data revealed that the software and devices could provide quality assurance for the software of the astronautic remote sensing cameras, thus laying a firm foundation for success of the astronautic project. Figs.6–9 present the medium-term prediction within the first 18 months. Fig.6 presents the cluster analysis for the 1521 defects recorded within the first 18 months, and a total of 97 similar defect categories are generated. Fig.7 presents the repulsion analysis based on the center characteristics of each defect cluster shown in Fig.8, and there are a total of 26 defect clusters. Fig.7 presents the distribution of the results; it can be observed that the highly frequent defects mainly appear at four coordinate points including $(-1.1,-1)$, $(1,1)$, $(0,0.8)$, and $(-3,1.3)$. Eq.(5) is a prediction function that is obtained via calculation. Fig.9 presents the medium-term defect prediction results, and it can be observed that the software defects in the astronautic remote sensor project increase within the first 18 months and are likely to increase continuously in the subsequent months. The overall prediction result can be attained by comparing the above prediction results with the actual quantity of defects observed by the project team. In the 18^{th} month, a difference of +60 defects (the error rate is approximately +4%) exists between the prediction and actual results. According to the prediction of defect distribution, it is estimated that 70% of the unfound defects appear at the four coordinate points including $(-1.1,-1)$, $(1,1)$, $(0,0.8)$, and $(-3,1.3)$. In the 19^{th} month, the project team conducted an overall test for the work performed within the first 18 months and observed a total of 21 defects. Overall, 52% of the total defects appeared at the four coordinate points $(-1.1,-1)$, $(1,1)$, $(0,0.8)$, and $(-3,1.3)$. The accuracy rate of the prediction results was as high as 97%. Figs.10–13 present the prediction results within the entire 36 months, and Eq.(6) is a defect prediction function. In the analysis data for defect clustering, a total of 3754 defects are observed, and a total of 153 defect clusters are generated, as shown in Fig.10. In the analysis data for defect repulsion, a total of 27 defect clusters are generated. The main defect characteristics are distributed at the coordinate points including $(-1.1,-1)$, $(-2.8,-1.3)$, $(-1.8,-2)$, $(-0.8,-0.3)$, $(-0.2,1.3)$, $(0,0.7)$, $(0,-0.5)$, $(0.3,0)$, $(0.8,1)$, $(1.5,0)$, and $(1.6,-1.2)$. In the 36^{th} month, the project team compared the prediction results with the observed defects and observed a difference of +44 defects between them. The difference value accounts for 1% of the total defects, which far less than the error rate of 3% available in medium-term prediction. The required error-free rate for the project should be higher than 97%. Therefore, the software quality meets the quality requirements for the astronautic re-

mote sensor project.

$$V(b) = -34.79 + 69.47x_1 + 42.73x_2^2 + 17.399x_3^3 - 3.03x_4^4 + 0.26x_5^5 - 0.01x_6^6 \quad (5)$$

$$V(b) = -33.39 + 69.47x_1 + 42.43x_2^2 + 16.299x_3^3 - 2.03x_4^4 + 0.16x_5^5 - 0.03x_6^6 \quad (6)$$

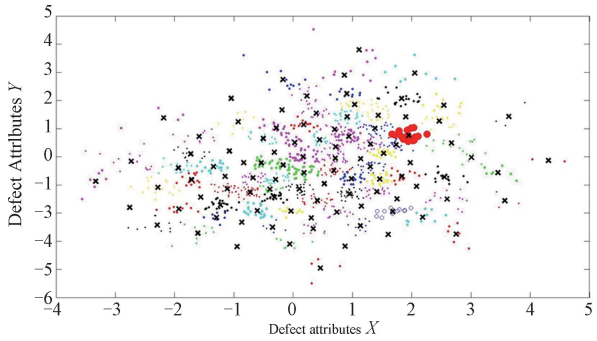

Fig. 6. Medium-term prediction of defect cluster

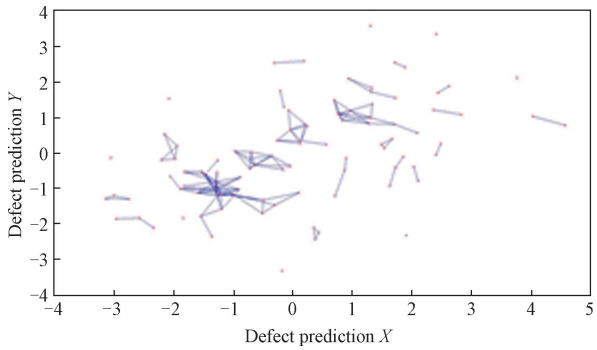

Fig. 7. Medium-term prediction of defect repulsion

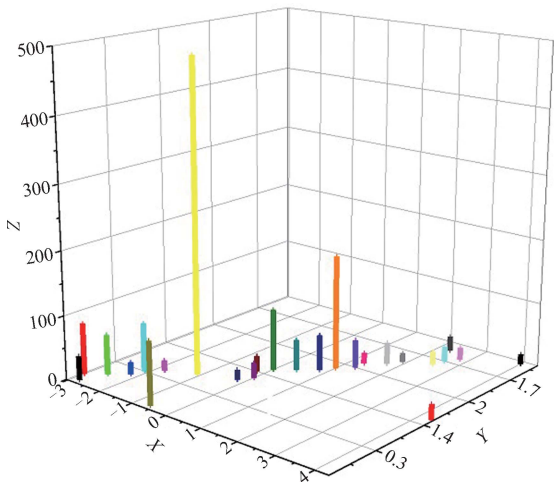

Fig. 8. Medium-term prediction of defect distribution

VII. Conclusion

This paper mainly expounds on the decision-level software defect prediction by reducing the scale of two focuses (including the defect characteristics and defects). This paper provides an in-depth and systematic analysis of various key issues, including how to create the defect characteristic comparison set and defect comparison set, a repulsion theory for defect characteristics and defects, and a methodology and model for defect prediction. Thus, a theory on the minimum critical characteristic set for defects is proposed. To predict the software system defects accurately, this paper resorts to the multi-agent defect prediction methodology (note: the agents include the SVM, Bayesian forecasting, and ANN). The experimental results demonstrate that this multi-agent prediction methodology is very effective in predicting the quality of astronautic project software.

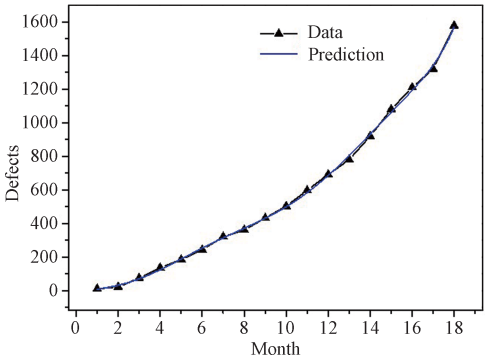

Fig. 9. Medium-term defect prediction

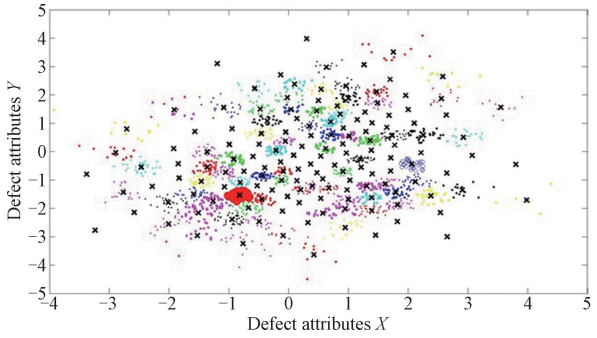

Fig. 10. Late-stage prediction of defect clusters

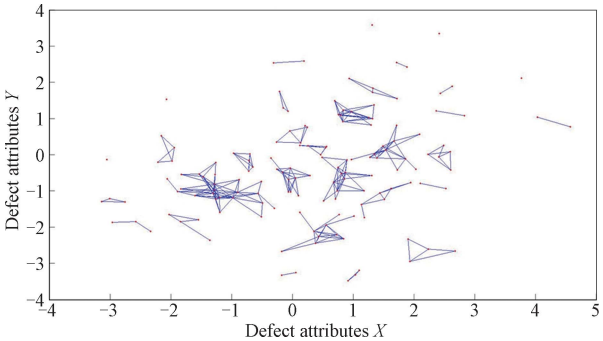

Fig. 11. Late-stage prediction of defect repulsion

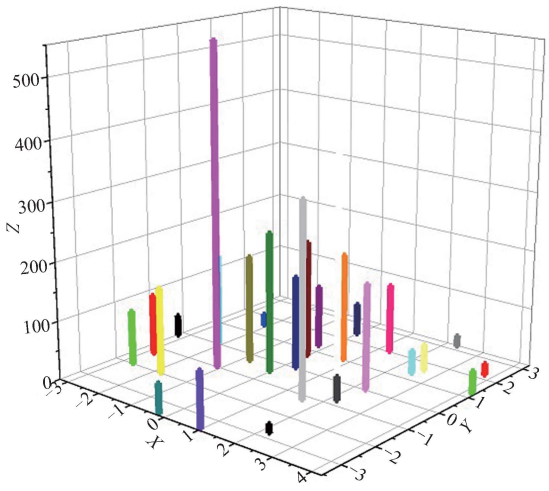

Fig. 12. Late-stage prediction of defect distribution

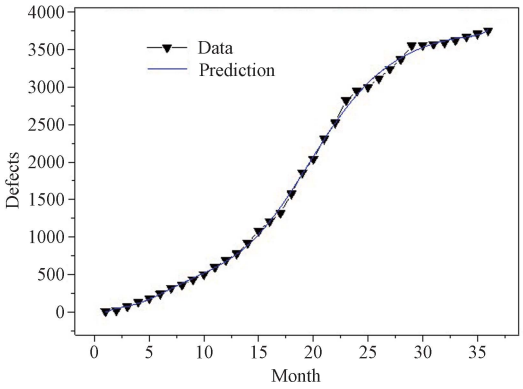

Fig. 13. Late-stage defect prediction

References

[1] M. Altman and M.P. McDonald, "Choosing reliable statistical software", *Political Science and Politics*, Vol.34, No.4, pp.681–687, 2001.

[2] G. Rudolph, "Convergence analysis of canonical genetic algorithms", *IEEE Trans on Neural Networks*, Vol.5, No.1, pp.1021–1027, 1994.

[3] B. Yang, D.Y. Liu, J.M. Liu, *et al.*, "Complex network clustering algorithms", *Jounal of Software*, Vol.20, No.1, pp.54–66, 2009.

[4] Tim Menzies and Andrew Butcher, "Local versus global lessons for defect prediction and effort estimation", *IEEE Transactions on Software Engineering*, Vol.39, No.6, pp.822–834, 2013.

[5] T.M. Khoshgoftaar, A. Herzberg and N. Seliya, "Resource oriented selection of rule-based classification models: An empirical case study", *Software Quality Control*, Vol.14, No.4, pp.309–338, 2006.

[6] M. Bell Robert, J. Ostrand Thomas and J. Weyuker Elaine, "The limited impact of individual developer data on software defect prediction", *Empir. Software. Eng*, Vol.20, No.1, pp.201–214, 2013.

[7] Lourdes Pelayo and Scott Dick, "Evaluating stratification alternatives to improve software defect prediction", *IEEE Transactions on Reliability*, Vol.61, No.2, pp.20–34, 2012.

[8] Z.B. Sun, Q.B. Song and X.Y. Zhu, "Using coding-based ensemble learning to improve software defect prediction", *IEEE Transactions on Systems*, Vol.42, No.6, pp.34–50, 2012.

[9] V. Basili, L. Briand and L. Walcelio, "A validation of object oriented design metrics as quality indicators", *IEEE Trans on Software Engineering*, Vol.22, No.10, pp.751–761, 1996.

[10] Taghi M. Khoshgoftaar and NaeemSeliya, "Fault prediction modeling for software quality estimation: Comparing commonly used techniques", *Empirical Software Engineering*, Vol.3, No.3, pp.255–283, 2008.

[11] A.K. Jain and Richard C. Dubes, "Algorithms for clustering data", *Technometrics*, Vol.32, No.2, pp.227–229, 1990.

[12] T.M. Khoshgoftaar and D.L. Lanning, "A neural network approach for early detection of program modules having high risk in the maintenance phase", *Journal of Systems and Software*, Vol.29, No.1, pp.85–91, 1995.

[13] J.C. Munson and T.M. Khoshgoftaar, "Regression modeling of software quality: Empirical investigation", *Information and Software Technology*, Vol.32, No.2, pp.106–114, 1990.

[14] R. Chillarege, I. Bhandari, K. Jarik, *et al.*, "Orthogonal defect classification–A concept for in-process measurements". *IEEE Trans on Software Engineering*, Vol.18, No.11, pp.943–956, 1992.

[15] N.E. Fenton, N. Martain, M. William, H. Peter, R. Lukrad, *et al.*, "Predicting software defects in varying development lifecycles using Bayesian nets", *Information and Software Technology*, Vol.49, No.1, pp.32–43, 2007.

[16] S. Yamada, M. Ohba and S. Osaki, "S-Shaped reliability growth modeling for software error detection", *IEEE Trans on Reliability*, Vol.32, No.5, pp.475–478, 1983.

[17] Yamada Shigeru, Ohba Mitsuru and Osaki Shunji, "S-Shaped software reliability growth models and their applications", *IEEE Transactions on Reliability*, Vol.33, No.4, pp.289–292, 1984.

[18] Tim Menzies and Andrew Butcher, "Local versus global lessons for defect prediction and effort estimation", *IEEE Transactions on Software Engineering*, Vol.39, No.6, pp.822–834, 2013.

[19] Fayola Peters, Tim Menzies, Liang Gong, *et al.*, "Balancing privacy and utility in cross-company defect prediction", *IEEE Transactions on Software Engineering*, Vol.39, No.8, pp.200–221, 2013.

HU Changhong was born in Jilin Province, China, in 1982. He is an Assistant Professor at Changchun Institute of Optics, Fine Mechanics, and Physics. His research interest are in the fields of distributed systems, software engineering, network management technology, electronics, and cloud computing, with a specific focus on distributed computing and reliability in system architecture (Email: changhonghu@rocketmail.com).

XUE Xucheng was born in Hebei Province, China, in 1980. He is an Associate Professor and Master Degree Director at the Chinese Academy of Sciences. His research interests include remote sensing cameras, data mining, and electronics. (Email: Xuexucheng@ciomp.ac.cn)

Huang Liang was born in Guangxi Province, China, in 1986. He is an Assistant Professor and a Ph.D. candidate at Changchun Institute of Optics, Fine Mechanics, and Physics. His research interest focuses on digital electronics for high-resolution space cameras (Email: 401353081@qq.com).