International Journal of Hybrid Information Technology
Vol.9, No.2 (2016), pp. 275-284
http://dx.doi.org/10.14257/ijhit.2016.9.2.25

Research of Software Testing Model based on Correlation Defect

Jin Jiangang[1], Sun Shibao[2] and Bao Xiaoan[3]

[1]*School of Software, North China University of Water Resources and Electric Power, Zhengzhou, Henan, China, 450011*
[2]*Network Information Center, Henan University of Science and Technology Luoyang, Henan, China, 471023*
[3]*The College of Informatics and Electronics, Zhejiang Sci-Tech University, Hangzhou, Zhejiang, China, 310018*
E-mail:hn_jjg@126.com

Abstract

Analysis of existing research results of defect testing, software defects during testing should consider the relevance of the defect test software, the design of a Markov chain with the right to test the model, the design of a multi-objective algorithm weights given Markov chain with the right test strategy based on defects associated with the state transition strategy based on multi-objective test weight matrix. The experimental results show that this strategy compared with Controlled Markov Chain testing strategy that can significantly reduce test cases and improve the defect detection rate.

Keywords*: Defects Correlation, Controlled Markov Chain, Software Testing, Multi-target weights, Status Shift*

1. Introduction

In recent years, with the continuous extension of the range of software applications, the scale is growing, structure is increasingly complex. Computer applications with the software as a core have penetrated into all fields, including industry, agriculture, defense, education, economy and people's daily life and work, and are playing as an important role increasingly. The problems of software reliability [1] are becoming more and more prominent. Although there are many techniques and methods that have been proposed and adopted to improve software quality in designing software systems (such as software architecture assessment [2]), software testing is still a primary means to ensure its quality and reliability.

Currently, the process of software testing takes a lot of development costs in most software development companies. Researches show that the effort of software testing is often accounted for more than 40% of the total workload in the software development. For some software that requires high reliability and high security (such as aerospace, military, *etc*.), the testing cost is the equivalent of 80% of the entire software life cycle of the total cost of the project. In addition, the cost which is used to correct the software defects will grow with the postponed time to discover the defects. Even so, the study for software testing did not cause enough attention. Software testing is still a relatively small part to research in the software development process.

In order to effectively improve the efficiency of software testing, many researchers put forward new testing methods based on the automatic generation [3], the priority of testing cases [4] and defect localization [5] as well as optimizing the testing strategy et al, but most of these studies are based on the assumption that the defects are independent, that is, the relationship between defects is ignored. In the actual testing process, a large number of studies have shown that many software defects are not independent to each other. There is a certain relationship between

ISSN: 1738-9968 IJHIT

them, that is, the failure correlation [6]. It is noted [7] that the *Space* software contains a total of 36 defects, of which their detection rate is almost zero if there only exists the three defects of *D*1, *D*2 and *D*32. Only in the case of other related defects present can they be detected. Therefore, the correlative information between defects should be taken full use of, which can help to improve the efficiency of software testing.

2. Correlation Defect

All printed material, including text, illustrations, and charts, must be kept within the parameters of the 8 15/16-inch (53.75 picas) column length and 5 15/16-inch (36 picas) column width. Please do not write or print outside of the column parameters. Margins are 1 5/16 of an inch on the sides (8 picas), 7/8 of an inch on the top (5.5 picas), and 1 3/16 of an inch on the bottom (7 picas).

Although software companies strive to develop the products in the standard of high cohesion and low coupling, but for most programs, they still cannot do absolutely independent modules. The relationship between defects may appear in the control flow or in the data stream of the program, both of which may be regarded as inherent representation of defects and defects associated. Besides, the programmers' inherent styles when programming will lead to the defects with similarity [8]. Analyzed from the defect itself, it is because there exist the control flow and data flow which cause the presence of the relationship between them.

For correlation defects, Katerina and Trivedi [9] believe that in the actual software testing process, software failures are not independent. They propose a way of using updated Markov models for the reliability of failure associated software modeling. Some experts [10] proposed a binary Markov process model on the basis of a test of round associated. Based on the assumption of mutual interference before and after the round test, they regarded time as a basis for the relationship of correlation defects, which, to a certain extent, explained the reasons why correlation defects were generated. Bishop [11] and others carried out empirical studies on detected defects in the process of the PODS (the Project on Diverse Software) development. They interpreted software defects as a model of a context-dependent state machine, and thought the relationship between the software failures could be explained by the failure shielding effect. Xu GaoChao, *et al.*, [6] proposed P-NHPP (Phase-Nonhomogeneous Poisson Process) reliability model. They thought that if we prematurely rejected individual correlation defect will shield the detection capability of other related defects, which is one of the roots of software failure. This will affect the assessment results of the software reliability model. For the detection of correlation defect, Jing Tao and others [12] analyzed the failure correlation from the perspective of software defects. They demonstrated the relationship between defects by the examples of *Space* software, proving that defects associated didn't meet the exchange law and associative law, and applied correlation defects in the software testing process. They also put forward a test method of putting the detected defect back to eliminate correlation defects. Simulation results show that this method can effectively detect correlation defects existing in software, but the defects back method will have certain impact on test efficiency.

As is pointed out in some references [13], test cases covering the same or similar test requirements tend to detect the same or similar software defects. Therefore, taking full use of relevant information between defects will make the common defects easier to be found by testers. However, the test strategy of software optimization designed for the correlation defects has rarely been studied. Besides, these studies did not quantify the associated defects. So this paper will focus on the research of how to quantify the association of correlation defects. In this paper, through the study of associated defects, testers will try to quantify the association between defects and provide a formal description of the

relationship. Testers will apply related information to the test model of controlled Markov chain. Based on the thoughts of software cybernetics, during the testing process testers will have a formal description of the feedback mechanism in order to achieve the self-adaptive adjustment of the testing strategy. Therefore, making a rational use of defect-related information and combining it with the test model based on software cybernetics, and designing a set of optimizing test strategies of software defects provide a new idea to improve the reliability of software and optimize the test efficiency.

3. Controlled Markov Chain Model

Markov model is a statistical model, which is characterized by stochastic processes of discrete time. In this process, on condition that the current knowledge or information is given, it is irrelevant to regard the status of the past for predicting the future state.

Markov chain model is applied to software testing. The number of defects which is present in the system of the software to be tested is treated as the state of the model. Suppose there are *N* defects in the system of the software to be tested, it is desirable to detect all the defects of the software systems, that is, in ideal situation; the number of defects in the tested software systems is zero. The state with *N* defects is treated as the starting state of Markov chain model, and the state without defects is treated as the termination of the state. By the influence of the test cases, the state transition table of Markov chain model is determined. Once the state transition table is determined, the Markov chain will be subsequently determined.

Adaptive testing strategies which have existed are mostly based on the theory of Markov decisions, and their structures are controlled by the test models of Markov chain. On certain conditions [7-11] these models do too idealistic treatment, so there are some deficiencies, including: ① assumed that the number of defects contains in the tested software is certain. However, in the actual test, the number of defects is unknown. It is impossible in a test event to detect all the defects. It is necessary to optimize the test regularly, and then check again. The optimization of test cases - detecting operation again may be repeated many times. Although each time the number of defects detected is variable and limited, there exists a desired value. The desired value can be used to measure the effect of the test. So how to find a way to get the desired number of defects is the key to the study.② assuming that an equal probability of different software defects is detected. But that is not the case. Not only is the rate of the defect detection related to the selected decisions, but it is also associated with the properties of the defect itself. Therefore, a reasonable way is needed to be found to quantify and calculate the probabilities of different defects detected. ③It is assumed that if a defect is detected, it is immediately removed without introducing new defects. In the actual test, it is found that prematurely cutting detected defects may not only generate new defects, but may cause software failure, affecting the assessment results of the model of software reliability [1]. ④ assumed that the decision eliminate defects does not consume system resources (*i.e.*, cost).⑤ assumed that it is rebate equal to eliminate different defects. However, the possible pay of any decision has nothing to do with the software state. It is only related to the corresponding software failure itself. Finding a serious software failure is clearly more rewarding than finding a slight software failure. Therefore, it is necessary to analyze the fault features of software defects detected in the process of software testing [13].According to the severity, the defects are classified (such as fatal, serious, general, secondary, test suggestions or prompts, etc.) and the weight of all types of defects is reasonably distributed. The more serious the flaws are, the greater the weight is, and the corresponding rebate is greater.

These studies above are mostly controlled by simplified Markov chain model. In order to describe the change of software state formally, we do some specialized treatment for certain conditions, making the scope of the model limited. At the same time these models

are based on the assumption that the defects are independent, ignoring the impact of correlation defect on the testing process, so making the assessment of software reliability produce distortion.

4. Controlled Markov Model Associated with Defects in Software Testing Based

Author names and affiliations are to be centered beneath the title and printed in Times New Roman 12-point, non-boldface type. Multiple authors may be shown in a two or three-column format, with their affiliations below their respective names. Affiliations are centered below each author name, italicized, not bold. Include e-mail addresses if possible. Follow the author information by two blank lines before main text.

As is shown in Figure 1, in the traditional testing model of Controlled Markov chain (Controlled Markov Chain, abbreviation as CMC) test model [14], its state is defined follow as:

$S = \{st\} = \{M, M\text{-}1, \ldots, 1, 0\}$ $(t \geq 0)$

of which *st* refers to a remaining defect state of software for a certain time *t*. In Figure1, *at* refers to at time *t*, selecting an action of the test case. In this model, we only consider the relationship between the transfer rate of software state and the defect detection rate, but ignore the correlation between defects. To take advantage of the defect-related information to improve the test efficiency, expand the model of the detection conditions of associated defects, we introduce multi-objective weights based on the defects, so the improved test model of controlled Markov chain based on associated defects (Correlation Defects-based Controlled Markov Chain, referred CDCMC)is shown in Figure 2,of which *st* and *at* have the same meaning as before, and *wi* is right to take a particular action a_t value, that is, for a time the probability of selecting an action a_t taken by the test cases.

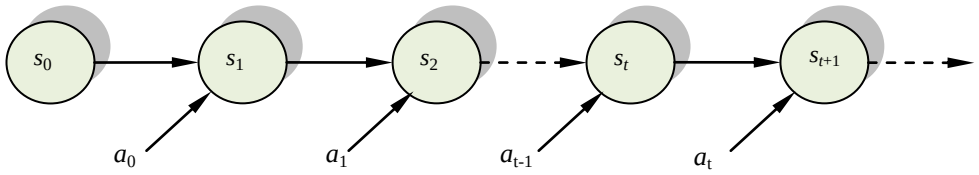

Figure 1. Traditional Controlled Markov Chain Model

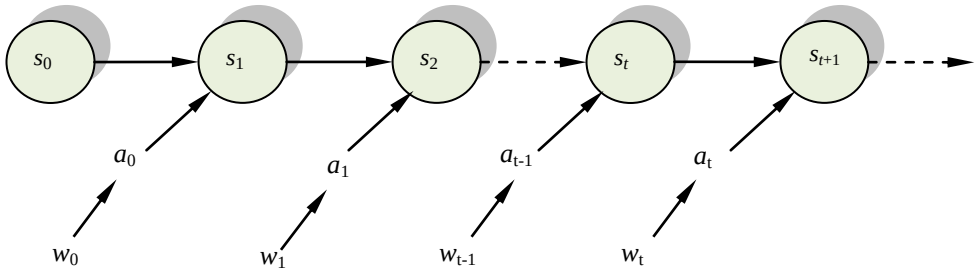

Figure 2. Improved Markov Chain Controlled Test Model

In order to construct multi-objective weights in the test model, and improve the transition conditions of software state, in this paper coupling degree function *coupler*(*x*, *y*, *type*) based on module weighted is designed to make formal descriptions of the coupling between modules, of which the parameters *x* and *y* indicate the module code, and *type* indicates coupling way. The function analyses the coupling between modules to deduce the relationship between defects that may exist in these modules, of which the return

value of the function is regarded as the coupling weight between modules. Considering the Pareto theorem of software testing [15], we found that 80% of the errors probably originated from 20% of the modules in the tests. Therefore, here we define when *x* equals *y*, the function returns a value of 0.5, that is, when defects are discovered in a certain module, the next time the probability to continue testing in the module is 50% so as to provide testing decisions with a standard of the software state transition. So the relationship between the modules can be represented as a triangular matrix ***R***:

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1i} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2i} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ r_{i1} & r_{i2} & \vdots & r_{ii} & \vdots & r_{in} \\ \vdots & & & & & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{ni} & \cdots & r_{nn} \end{bmatrix} \quad (1)$$

In this relationship, *n* represents the number of modules contained in the tested software, and r_{ij} represents the coupling weight between the *i-th* module and the *j-th* module. Obviously, according to the previous considerations, when *i* equals *j*, r_{ij} equals 0.5, *i.e.*, $r_{11} = r_{22} = ... = r_{nn} = 0.5$.

In order to construct multi-objective weights ***W***, a value function is constructed first.

$$weight(r_{ij}, \rho_i^j, \theta_i^j) \quad (2)$$

In this function, r_{ij} represents the coupling weight between the *i-th* module and the *j-th* module; ρ_i^j represents the expense that after detecting a defect in the *i-th* module, a defect in the *j-th* module is detected; θ_i^j represents a defectable rate that after detecting a defect in the *i-th* module, a defect in the *j-th* module is detected. The return value w_j of the function is the probability of detecting a defect in the *j-th* module to execute the corresponding test cases. According to the return value, we can adjust weight matrix ***W*** of more goals immediately so that the test strategy will be adjusted.

$$\boldsymbol{W} = [w_0,\ w_1,\ \cdots,\ w_j,\ \cdots,\ w_t]$$

To implement the strategy, the historical data need to be collected and they will be stored in the database. Using the test feedback to carry on the online parameter estimation and update multiple targets weight matrix W based on the coupling function of the weighted module, we can build a more accurate and reasonable model. As is shown in Figure 3.

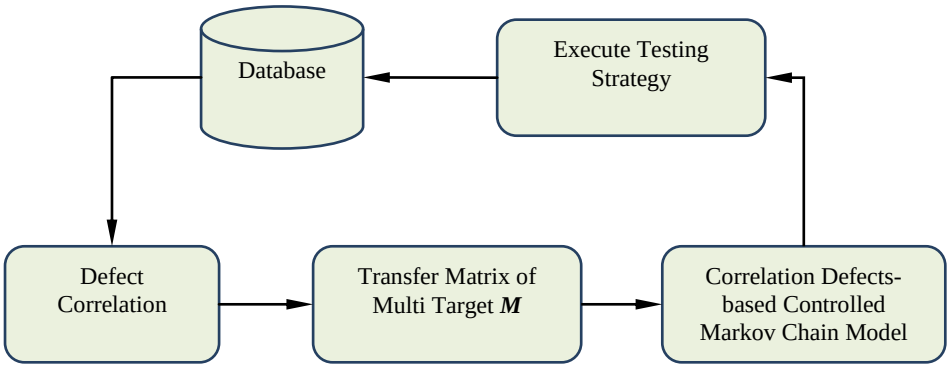

Figure 3. Dynamic Adjustment the Defects Correlation Relationship

5. Experimental Data

The following experiments were performed in the following environment.
Operating System: Windows 7.0 SP1, 64 bit.
Processor: Intel® Core ™ i5-2410M CPU @ 2.30GHz.
Memory: 8.00GB.

In Java integrated development environment Eclipse, the open-source project CDT using Java to implement is loaded and in the Eclipse environment, the unit testing procedures using JUNIT to develop are used. Using the software testing tools of Eclipse open-source plug EclEmma to achieve the test of CDT's coverage and the test data will be dynamically analyzed in the test. In the above test environment, different test cases can be loaded separately for testing according to the needs, and the results of several tests can be merged [16].

In the above environments, in order to verify the efficiency of software CDCMC testing model, in this paper we compare the model with CMC software testing strategy. The educational administration system of a school was carried out 50 tests, and the testing process was recorded in detail. It was found that the two methods detect the same number of defects in a particular test, all 39. The experimental data before and after the experiment of a total of 10 times are compared with the two test strategies, the results are shown in Figure 4. The number of test cases was used in the two strategies when the 39 defects were detected, as is shown in Figure 5. It can be found that in the CMC software testing strategy, with the increase of the number of detected defects, the number of test cases increase very fast, almost exponential growth. Compared with it, CDCMC software testing strategy tends to slower growth, whose detailed data are shown in Table 1. As can be seen from Table 1 at the beginning of the test there are no significant differences between the number of test cases which the two strategies need, but with the increase of the number of detected defects, the difference of the number of test cases which the two strategies need is growing. As can be seen from the ratio of the number of test cases and the number of defects detected, CDCMC software testing strategy is 6.19, while CMC software testing strategy is 13.38.Taking the 50 tests into consideration, the former is 6.21 while the latter is 13.51. Therefore, the defect detection rate of CDCMC software testing strategy is better than CMC software testing strategy.

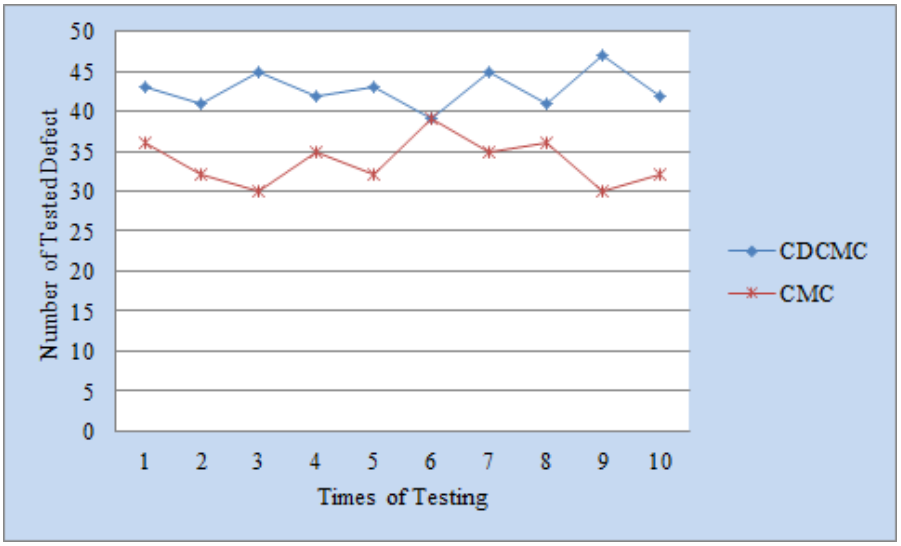

Figure 4. Comparison of Two Test Strategy

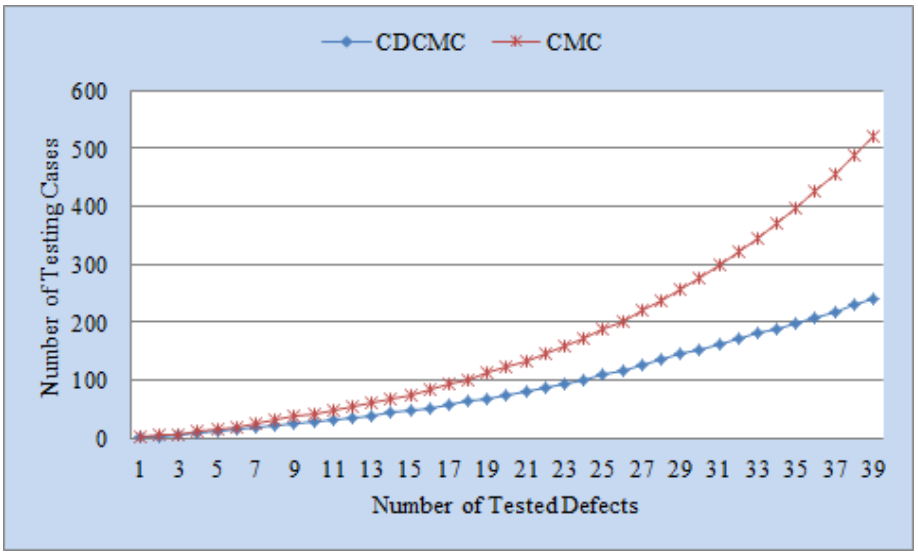

Figure 5. Relationship between the Number of Defects Detected, the Number of Test Cases used

Table 1. Compare CDCMC Test Strategy with Test Strategy CMC

Defects Number	CDCMC	CMC	Defects Number	CDCMC	CMC	Defects Number	CDCMC	CMC
1	2	3	14	44	68	27	127	220
2	4	6	15	48	76	28	136	237
3	7	7	16	53	84	29	145	256
4	9	11	17	58	93	30	154	276
5	12	15	18	63	102	31	163	298
6	15	20	19	69	112	32	172	321
7	18	27	20	75	122	33	181	345
8	21	32	21	81	133	34	190	370
9	25	37	22	88	145	35	199	397
10	28	42	23	95	158	36	209	425
11	32	48	24	102	172	37	219	456
12	36	54	25	110	187	38	230	487
13	40	61	26	118	203	39	241	522
Test Cases/Detected Defects Number					CDCMC	6.18	CMC	13.38

6. Conclusion and Outlook

In this paper the controlled Markov chain test model based on correlation defects is proposed on the basis of the controlled Markov chain test model, mainly to solve the problem of software testing in the situation of the relationship of software defects. The coupling between software modules is quantified by the formula (2) to calculate a multi-target transfer matrix to quantify the relationship of associated defects. Combining historical data to adjust multi-target transfer matrix online, we use this transition matrix as a transfer standard of the software testing strategies for the software state. By analyzing the results, we can see CDCMC testing strategy is more effective than CMC test strategy.

The shortcoming in this paper is that we only consider how to use the best detection rate to detect the defects, we cannot do further research on how to eliminate the defects, which will be the focus of our future research.

References

[1] C. Kai-Yuan, D. Zhao and L. Ke, "On Several Issues in Software Reliability Testing", Chinese Journal of Engineering Mathematics, vol. 25, no. 6, **(2008)**, pp. 967-978.
[2] Z. Li, G. Hui and W. Shou-Xin, "Software architecture evaluation", Journal of Software, vol. 19, no. 6, **(2008)**, pp. 1328−1339.
[3] H. Jun-Hao, H. Chun and Z. Zhu-Lin, "Automatic System Testing Test Case Generation Based on UML", Computer Systems & Applications, vol. 20, no. 2, **(2011)**, pp. 178-181.
[4] Q. Bo, N. Chang-Hai and X. Bao-Wen, "Test Case Prioritization Based on Test Suite Design Information", Chinese Journal of Computer, vol. 31, no. 3, **(2008)**, pp. 431-439.
[5] L. Wei, Z. Zheng, H. Peng, *et al.*, "Predicate Execution-Sequence Based Fault Localization Algorithm", Chinese Journal of Computer, vol. 36, no. 12, **(2013)**, pp. 2406-2419.
[6] X. Gao-Chao, L. Xin-Zhong, H. Liang, F. Xiao-Dong and D. Yu-shuang, "Software Reliability Assessment Models Incorporating Software Defect Correlation", Journal of Software, vol. 22, no. 3, **(2011)**, pp. 439-450.
[7] G. Rothermel, R. H. Untch and M. T. Harrold, "Prioritizing test cases for regression testing", IEEE Trans. Software Engineering, vol. 27, **(2001)**, pp. 929-948.
[8] Z. Jian, Z. Hong-Yu and L. David, "Where should the Bugs be fixed", In: Proc. of the Int'l Conf. on Software Engineering. Zurich: IEEE Computer Society, **(2012)**, pp. 14-24.
[9] G. P. Katerina and K. S. Trivedi, "Failure correlation in software reliability models", IEEE Trans. on Reliability, vol. 49, no. 1, **(2001)**, pp. 37-48.
[10] S. Chen and S. Mills, "A binary Markov process model for random testing", IEEE Trans. on Software Engineering, vol. 22, no. 3, **(1996)**, pp. 218-223.
[11] P. G. Bishop and F. D. Pullen, "PODS revisited-A study of software failure behavior", In: Proc. of the IEEE Int'l Symp. On Fault Tolerant Computing, **(1998)**, pp. 2-8.
[12] J. Tao, J. Chang-Hai, H. De -Bin, B. Cheng-Gang and C. Kai-Yuan, "An Approach for Detecting Correlated Software Defects", Journal of Software, vol. 18, no. 1, **(2005)**, pp. 17-28.
[13] S. Chang-Ai, "A Constraint-Based Approach to Identifying and Analyzing Failure-Causing Regions", Journal of Software, vol. 23, no. 7, **(2012)**, pp. 1688−1701.
[14] Z. De-Ping, N. Chang-Hai and X. Bao-Wen, "Cross-Entropy Method Based on Markov Decision Process for Optimal Software Testing", Journal of Software, vol. 19, no. 10, **(2008)**, pp. 2770-2779.
[15] B. Zhao, "Software Testing Technology Classic Course", Beijing: Science Press, **(2007)**, (in Chinese).
[16] A. Jin-Xia, W. Guo-Qing, L. Shu-Fang and Z. Ji-Hong, "Dynamic Evaluation Method Based Multi-Dimensional Test Coverage for Software Testing", Journal of Software, vol. 21, no. 9, **(2010)**, pp. 2135-2147.

Authors

Jin Jiangang, He was born Henan, China on 11/20/1972, Male, Gushi County, Henan Province, China. Lecturer, postgraduates. The main research directions: Software Engineering and Adaptive Software. National Natural Science Foundation of China (No.: 61202050). E-mail:hn_jjg@126.com. Tel: (+86)13523448299.

Shibao SUN, He was born in Henan, China on 07/10/1970. M.S. His research interests can be summarized as intelligent information processing, machine learning. Particularly, he is currently interested in rough sets theory and approaches, as well as their applications in information processing.

Bao Xiao'an, He born in 1973, M.S. He is a professor and mainly researches adaptive software, software testing and intelligent information processing.

