International Journal of Security and Its Applications
Vol.9, No.9 (2015), pp.189-196
http://dx.doi.org/10.14257/ijsia.2015.9.9.17

Fault Analysis in Software with the Data Interaction of Classes

Yan Xiaobo[1] and Wang Yichen[2]

[1]*Science & Technology on Reliability & Environmental Engineering Laboratory, Beihang University, Beijing 100191, China*
[2]*Science & Technology on Reliability & Environmental Engineering Laboratory, Beihang University, Beijing 100191, China*
yxbbuaa@buaa.edu.cn[1], wangyichen@buaa.edu.cn[2]

Abstract

With the development of science, software becomes more and more complex, which makes it difficult to do fault propagation analysis. Software Network is a general tool in studying fault propagation, but because of the diversification of software's architecture, we can no longer oversimplify the data flow of nodes in the software network which is universal in conventional method to analyze fault propagation. This paper put forward a new structure Class Interactive Network based on the data interaction of classes in software to analyze software's fault propagation in the class hierarchy and simulate software classes' performance in fault propagation with quantitative analysis in Class Interactive Network.

Keywords: *Fault Analysis; Class; software; data interaction*

1. Introduction

Fault analysis is a key link in the whole software reliability research; and there are a lot of researches for the fault analysis at present, most of them are based on Software Network since it was proved that massive software system is a complex network [1-5]. For example, Mac Cormack proposed DSMs method for extracting software source files and dependencies and used The Change of Price to measure error propagation [6]. J. Liu studied on measuring the volatility of software structure by using Software network model [7]. Hassan and Holt presented a propagation model focused on software changes [8]. However, these researches didn't consider the connection of data in different modules of software.

Complex software system is under ongoing evolution as a whole [9], so we should consider the data interaction between different nodes in the Software Network. Some researches do consider the connection of different nodes or modules, such as Pan's software quality metrics to describe fault propagation [10], Li's fault propagation model based on signals and module level [11]. But these researches oversimplified the data flow of nodes in the software network, which is also not accurate.

In the process of fault propagation, conventional method can't tell the performance of a module or node in software accurately when it has a plurality of outputs, in which situation conventional method can't describe the changes of software structure introduced by the data interaction.

So this paper aims to develop a new method that represents Class-calling output network to study the fault propagation in software based on the data interaction in class hierarchy. For the purpose of studying the data connection of different classes in Fault propagation process, Interaction coefficient has been proposed at the first time, which is very valuable in describing the degree of data interaction between different classes. With this concept, some quantitative analysis has been studied and proved. At last, an experiment is designed to prove the correctness of the mode.

ISSN: 1738-9976 IJSIA

2. Class-calling Output Network

Definition 1 Class-calling output network CCON can be represented by directed graph P(M,E), in which M is the set of software's classes and E is a set of directed edges which represents the call relationship for the outputs of different classes and directed by the callee to caller.

Which should be noted is that we defined a class's output as a set of outputs of all this class's member functions. Generally speaking, class A have a output A.f(), in which f() is a output of class A's member function, if A.f() call, inheritance or reference class B's output B.h(), then there is call relationship between class A and class B, in which class A is the caller and class B is the callee and we can call class A as the caller-class and call class B as the callee-class.

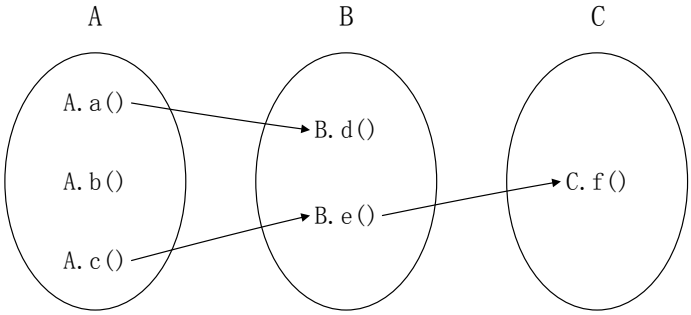


Figure 1. An Example of CCON

In this example, A, B, C are three classes of the software, A.a(),A.b(),A.c() are three outputs of class A's member functions. B.d(), B.e() are two outputs of class B's member functions. C.f() is a output of class C's member functions. The arrows indicate the call relationships between these outputs and directed by callee to caller.

3. Interaction Coefficient and Class Interactive Network

Definition 2 Interaction coefficient is a description of two adjacent classes' strength of data interaction in CCON.

Interaction coefficient values between 0-1. The bigger interaction coefficient the two adjacent classes have, the higher degree of data interaction they have.

Let U_i indicate the number of class i's all member functions' outputs in CCON; U_{ij} represent the number of class i's outputs that were called by class j; U_j is the number of outputs which belong to class j's member functions ; $U_{j\leftarrow i}$ represent the number of class j's outputs which called class i's outputs. Then we can define the interaction coefficient K_{ij} from class i to class j as:

$$K_{ij} = \frac{U_{ij}}{U_i} \times \frac{U_{j\leftarrow i}}{U_j} \quad (1)$$

For further study, we put forward Class Interactive Network which is transformed from Class-calling output network and easier to realize. Class Interactive Network applies Interaction coefficient to Class-calling output network. We can use directed graph N=(M,C,K) to describe Class Interactive Network. M is the set of software's classes and E is a set of directed edges which represents the call relationship of different classes and directed by the callee-class to caller-class in CCON.K is the set of interaction coefficient in CCON, which contains every edge's interaction coefficient from callee-class to caller-class. For example, if we change the example of CCON above into Class Interactive Network, the conversion is as follows:

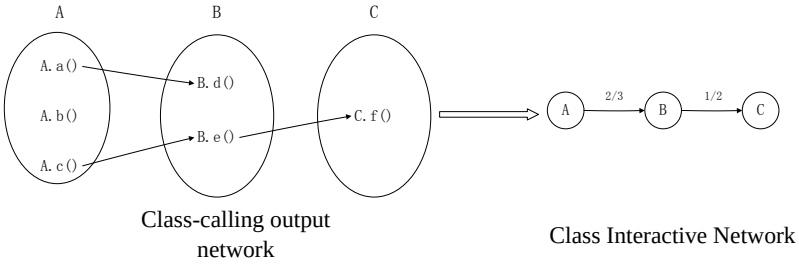


Figure 2. The Conversion Example of CCON to Class Interactive Network

According to the definition of Class Interactive Network, if we make the value of Interaction coefficient as 1 in Class Interactive Network, then the degree of two adjacent classes rely on each other's data will be the maximum, in which case caller-class will run totally wrong if callee-class has fault. In this situation, our Class Interactive Network will change into conventional method in which the data dependency between different classes is ignored.

4. Fault Classification and Interaction Failure Rate

As we know, faults have so many species that we cannot study every kind of their impact on software. Some faults can cause the software memory overflow, but most of them just make the software to produce output that does not meet the requirements of users [10]. To simplify the problem, we divided fault into two types in CCON: physical fault and interactive fault. Physical fault is the situation that the machine registers and other hardware devices have errors or there is data error in member function of class. Interactive fault represents the situation that a class calls the wrong data as input, resulting in the error of output where the class may have right outputs given correct inputs. What should be noted that we assume the physical fault is the cause of interactive fault. So the fault propagation in this paper can be denoted as 3 cases:

a. Callee-class has physical fault in which case the caller-class will get error input and have error output.
b. Callee-class has interactive fault in which case the caller-class will have interactive fault either.
c. Callee-class has no fault in which case the caller-class will have error output only if the caller-class has physical fault.

Obviously, we can define the fault propagation as the circulation of case a and case b.

According to the definition of Interaction coefficient, we can use Interaction coefficient to denote the probability of case a and case b between two adjacent classes, but for any two classes in a software, it will fail when there have different paths between class A and class B in the Class Interactive Network. So, combined with fault classification above, we put forward the concept of Interaction Failure Rate to describe two classes' data interaction strength in this situation.

Definition 3 Interaction Failure Rate indicates the possibility that the caller-class have interactive fault caused by callee-class' physical fault or interactive fault in the Class Interactive Network.

If class *i* is the callee-class, class *j* is the caller-class. *G* is the set of all paths directed by class *i* to class *j*. K_{ijp} is the Interaction coefficient between class *i* and class *j* through path *p* in the Class Interactive Network. R_p is the set of Interaction coefficient *K* on path *p*. Then we can define Interaction Failure Rate S_{ij} from class *i* to class *j* as :

$$S_{ij} = \sum_{p \in G} K_{ijp} \quad (2)$$

$$K_{ijp} = \prod_{K \in R_p} K \quad (3)$$

For example, in the example of Class Interactive Network above, there is a path p between class A and class C:$A \rightarrow B \rightarrow C$, $R_p=\{2/3,0.5\}$.

Then the Interaction Failure Rate between class A and class C is $S_{AB}=K_{ABp}=2/3\times 1/2$.

According to the definition of Interaction Failure Rate, if there is only one path between two classes i and j, then the Interaction Failure Rate S_{ij} is the same as Interaction coefficient K_{ij} from class i to class j, so Interaction coefficient is the exceptional of the Interaction Failure Rate. As an extension of Interaction coefficient, Interaction Failure Rate is a description of any two classes' data interaction strength in the Class Interactive Network. The bigger Interaction Failure Rate to two classes, the higher degree that the two classes rely on each other's data, in which situation that caller-class is more prone to have interactive fault when the callee-class have physical fault or interactive fault.

5. Interaction Factor

In the fault propagation process, it's difficult to make measurement methods to describe fault propagation process exactly. This is due to the software complexity of the structure itself, but we can build some quantitative indicators to estimate the impact of software's structure. To achieve it, we put forward Interaction coefficient and Interaction Failure Rate to describe mutual influence between the two classes, which can't describe the impact of a class to whole software. So we propose Interaction factor.

Interaction factor T_i indicates class i's ability to spread fault in the Class Interactive Network. Using directed graph $N=(M,C,K)$ to describe Class Interactive Network, S_{ij} represents the Interaction Failure Rate from class i to class j, then class i's Interaction factor T_i:

$$T_i = \sum_{j \in M, j \neq i} S_{ij} \quad (4)$$

According to the formula above, Interaction factor T_i is the sum of Interaction Failure Rate from class i to all of the rest classes. If class i has a great value of Interaction factor T_i, then other classes have great possibilities to get interactive fault when class i has interactive fault or physical fault. Because of this property of Interaction factor, software analyst can get theoretical value of software's classes by building the Class Interactive Network. Then analyst can simulate software classes' fault propagation process by counting Interaction factor of classes timely.

6. Fault Propagation based on the Data Interaction of Classes

As a conclusion, our quantitative analysis put forward a probability model of fault propagation in the class hierarchy. In this paper, the source of fault propagation is physical fault in a class which can be a bug, a defect of algorithm, a program error or even a failure of hardware, the physical fault may cause other interactive faults in different classes. What should be noted is that we don't consider the specific of faults, in this paper, the trend of fault propagation is what we focused on. The architecture or our model is shown as below:

a. Construct the CCON of software
b. Get the Class Interactive Network and Interaction coefficient
c. Calculate the Interaction Failure Rate and Interaction factor
d. Quantitative analysis for fault propagation

7. Tests

To test the validity of the theory, we did fault injection experiment to the SHA-1 software. The SHA-1 program is a classic algorithm used to encrypt the information

processing. The program is an iterative procedure which has 80 rounds of iteration. In order to avoid redundancy, the test only selected the first 20 rounds of iteration of SHA-1 as the experimental object.

Table 1. The Theoretical Value of Interactive Factor in Class Interactive Network

Table 1

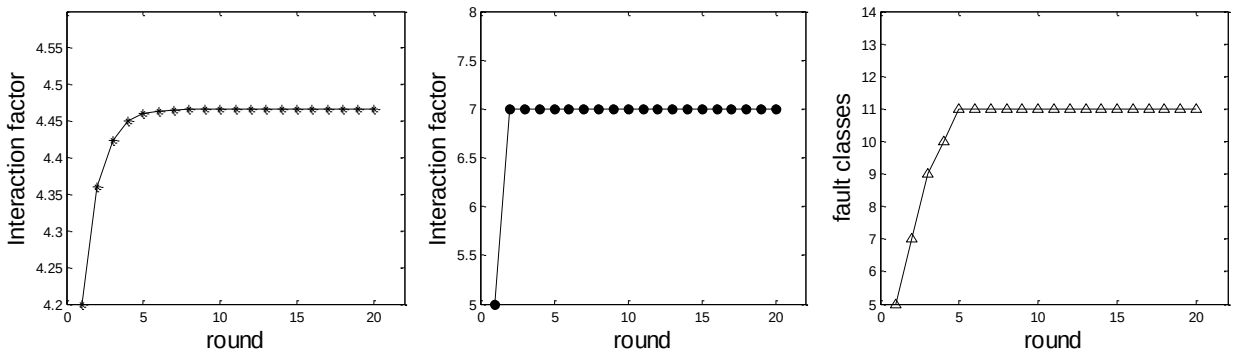
	the theoretical value of Interactive factor in Class Interactive Network						
classes	1 (round)	2	3	4	5	……	20
ft	4.2	4.3600	4.4240	4.4496	4.4598	……	4.4667
S_5	3.2	3.6000	3.7600	3.8240	3.8496	……	3.8667
XOR_1	3.2	3.4800	3.5920	3.6368	3.6547	……	3.6667
XOR_2	2.2	2.6400	2.8160	2.8864	2.9145	……	2.9333
S_30	0.2	1.2320	1.8099	2.1335	2.3148	……	2.5454
……							

Table 2. The Number of Fault Classes in the Fault Propagation Experiment

Table 2

	the number of fault classes						
classes	1 (round)	2	3	4	5	……	20
ft	5	7	9	10	11	……	11
S_5	4	7	9	10	11	……	11
XOR_1	4	6	9	10	11	……	11
XOR_2	3	5	9	10	11	……	11
S_30	1	3	7	10	11	……	11
……							

In the test, we made three groups of contrast experiments. At the first experiment, we built Class Interactive Network for SHA-1 program and calculated every class's Interaction factor in every round by the definition. In the second experiment, we did the first experiment again but made the value of Interaction coefficient as 1 in Class Interactive Network this time, in which situation that Class Interactive Network is the same as conventional method. The last experiment is a fault propagation experiment, we inserted physical fault into every class in program and got the statistical number of fault classes in the process of program operation in every round. According to the theory above, we could insert a data error into class's member functions to make class produce physical fault. To be brief, we selected class ft as an example to analyze the result, Other classes were very similar to class ft. The experimental results are shown in Figure 3:



(a)The model of Class Interactive Network (b) Conventional method (c) fault propagation experiment

Figure 3. The Experimental Results

As can be seen from the graph above, the Interaction factor of class ft has a steady increasing from round 1 to round 5 and then reaches an equilibrium state in the model of Class Interactive Network. In conventional method, the Interaction factor of class ft reaches the max value after the first round which is very abrupt. In the fault propagation experiment, the number of classes that get interactive fault caused by class ft's physical fault has a steady increasing from round 1 to round 5 and then reaches an equilibrium state after round 5, which is very similar to the variation of interaction factor in the model of Class Interactive Network.

In order to avoid the divergence caused by the results of experiment data in different units, we processed the test data with the following formula:

$$E_n = \frac{I_n - I_1}{I_{max} - I_1} \tag{5}$$

In the first and second experiment,$I_n$is the Interaction factor of class ft in the nth round. I_{max} is the max Interaction factor of class ft in the experiment. In the fault propagation experiment, I_n is the number of fault classes in nth round, I_{max} is the max number of fault classes in the fault propagation experiment. $\mathrm{E_n}$ is the percentage of data changing. It is necessary to minus I_1 to balance the divergence caused by the results of experiment data in different units. The results are as follows:

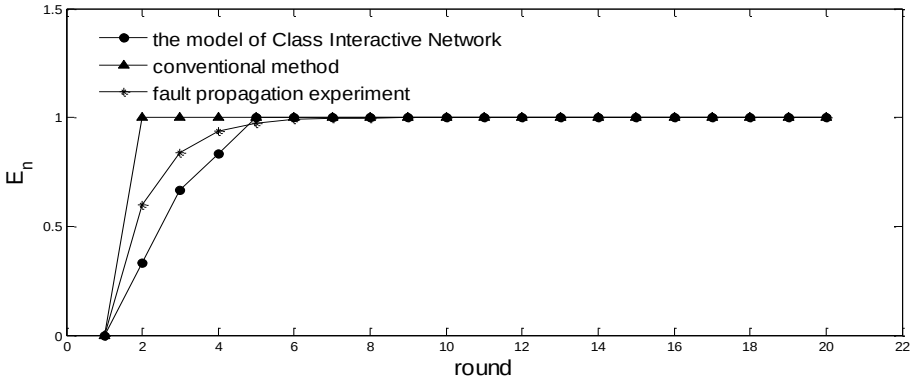


Figure 4. The Result of Data Changing

As we can see in the graph, the percentage of data changing $\mathrm{E_n}$ has a great change from round 1 to round 2 in conventional method, which is overhyped 100%. At the same time, the performance of model of Class Interactive Network is very similar to the actual situation of the fault propagation. So we can simulate one class's fault propagation with the model of Class Interactive Network.

Table 1 is the result of the theoretical value of Interaction factor in the model of Class Interactive Network. Table 2 is the result of the number of classes that get interactive fault caused by specific class's physical fault in fault propagation experiment. From the tables, we can see that class A's Interaction factor in every round is in direct proportion to the number of fault classes caused by class A's fault every round in fault propagation process. If two classes had similar Interaction factor like class S_5 and class XOR_1, the number of fault classes caused by them were nearly the same in every round. So it is feasible to simulate classes' process of fault propagation with Interaction factor.

8. Conclusion

Class-calling output network CCON is suitable in analyzing software's data structure in the class hierarchy and Interaction factor T_i does indicates class i's ability to spread fault in the Class Interactive Network. With the model of Class Interactive Network, we can simulate software's fault propagation on class hierarchies by analyzing Interaction

factor of classes. It also provides theoretical help in further improving the reliability of software and finding the vulnerabilities in a software.

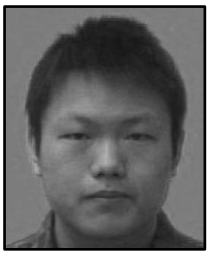
Acknowledgements

I am grateful to Mr. Wang, Yichen, and Associate professor, Science & Technology on Reliability & Environmental Engineering Laboratory, Beihang University for his invaluable assistance, counsel and acute criticism.

References

[1] C. R. Myers, "Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs [J]", Journal of Physical review, vol. 68, nos. 2, 4, (**2003**), p. 15.
[2] K. He, B. Li and Y. Ma, "Software network [M]", Beijing: Science Press, (**2008).**
[3] B. Li, Y. Ma and J. Liu, "The progress in the study of complex networks [J]", Advances in Mechanics, vol. 38, no. 06, (**2006**), pp. 805-813.
[4] D. Yan and G. Qi, "The scale-free feature and evolving model of large-scale software systems [J]", Chinese Journal of Physics, vol. 55, no. 08, (**2006**), pp. 3799-3806.
[5] Y. Ma and K. He, "Empirical Study on the Characteristics of Complex Networked Software [J]", Journal of Software, vol. 22, no. 3, (**2011**), pp. 381-407.
[6] A. Mac Cormack, J. Rusnak and C. Y. Baldwin, "Exploring the structure of complex software designs: An empirical study of open source and proprietary code [J]", Management Science, vol. 52, no. 7, (**2006**), pp. 1015-1030.
[7] J. Liu, J. Li and K. He, "Characterizing the Structural Quality of General Complex Software Network [J]", International Journal of Bifurcation and Chaos, vol. 18, no. 4, (**2008**), pp. 605-613.
[8] H. A. E. and R. C. Holt, "Predicting Change Propagation In Software System [C]", In Proceeding(s) of the 20th IEEE Conference International Conference on Software Maintenance, Chicago, IL,USA, (**2004**), pp. 284-293.
[9] H. Wang and W. Wu, "Development of complex software systems structure and adaptive evolution [J]", China Science Information Science, vol. 44, (**2014**), pp. 743-761.
[10] W. Pan and B. Li, "Quality metric of software based on software network error propagation [A]", Journal of Central South University, Natural Science Edition, vol. 43, no. 11, (**2012**)-11.
[11] A. Li, "Error propagation analysis in the software [J]", Computer research and development, vol. 44, no. 11, (**2007**), pp. 1962-1970.

Authors



Yan Xiaobo, he was born in 1991. He is a master at Beihang University. His research interests include software fault propagation, software testing, etc.

Wang Yichen, he was born in 1977. He is a Master supervisor and Senior Engineer at Beihang University. His research interests include software fault propagation, software testing, Analysis of embedded software fault etc.