http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 10 (11): 2214-2219, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.2214.2219

Dynamic Software Reliability Maintenance Based on Component Monitoring and Resource Allocation

L. Chen, W.U. Kaigui and H.E. Pan
College of Computer Science of Chongqing University, Chongqing, China

Abstract: Reliability is one of the major concerns for software engineers. Due to dynamic properties of the open environment, visited probability of each component might change after a certain amount of time. To maintain user's requirement of the software system reliability, this study proposes a dynamic reliability maintenance mechanism based on monitoring and resources allocation. An open source monitoring software, named Glassbox, is adopted to observe the change of number of visits to each component. Besides, the path-based method is used to analyze the reliability of software system from components reliability in the run time. Sensitivity of each component is calculated using this model and the difficulty to improve each component's reliability is analyzed. After that, considering the limited system resources, we propose a greedy algorithm for re-allocating resources reasonably according to component sensitivity and the difficulty to improve component reliability which can improve the system reliability. Finally, some simulation experiments are presented to evaluate our method. The results showed the effectiveness of the proposed mechanism in this study.

Key words: Reliability maintenance, component, monitoring, resource allocation

INTRODUCTION

With the rapid developments of information and network technology, the structure of software system is more and more complex. Software development almost gives up "start from scratch approach" and extensive component reuses are preferred. In order to evaluate the reliability of software system, the reliability of components, implementation features of components and interactions among components are integrated to establish the relationship model between components and software system on the basis of the architecture.

In recent decades, a number of reliability models have been developed to evaluate the reliability of a software system. They are mainly classified into three categories (Goseva-Popstojanova and Trivedi, 2001; Gokhale and Trivedi, 2002): State-based models (Pietrantuono *et al*., 2010), path-based model (Yacoub *et al*., 1999) and additive model. These models do well in the periods of software development and test phase on these conditions: 1, the reliability of individual components is known; 2, the transition probability among components can be informed; 3, the operating environment and user needs do not change. However, once the system is developed, in the operational phase, the system's reliability is difficult to be guaranteed. Possible reasons are as follows: 1, different people have different operation habits which resulted in large differences in the operational profile. 2, with the development of the Internet, more and more systems are in an open environment. Because of the uncertainty of an environment itself, software system had to change with the environment. These two factors will make system reliability a certain difference between software distribution and running. During the period of software operation, when the access number of a component increases and resources attached to this component are relatively few, performance of this component will be worse and reliability of the system will slip in synchronization; Conversely, the access number of a component reduces and resources attached to this component are relatively surplus, that is, resources are not fully utilized which also leads to reduce the reliability of the software system.

Monitoring is not only a traditional field of research but also a common engineering technology. The purpose is to monitor and control the system. Monitoring technology has been a widely used in various fields (Zulkernine and Seviora, 2005). To solve above problems, we attempt to make use of an open source monitoring software, named Glassbox which is a management agent with a browser-based interface that automatically pinpoints common problems in software system. We experience Enterprise Java applications problem diagnosis in the operational phase, especially monitor the transition probability among components.

Besides, we use software components to construct a software system, assess the reliability of the system by investigating the architecture and calculate the reliability

Corresponding Author: L. Chen, College of Computer Science of Chongqing University, Chongqing, China

of the system by path-based model. Sensitivity of each component is calculated using this model. When the system has been monitored for a certain amount of time, resources among various components should be adjusted dynamically to make them more reasonable on the condition that total resources in the system are limited and then the system is more reliable.

DYNAMIC RELIABILITY MAINTENANCE MECHANISM

Due to the dynamic properties of the system and variety of visited probability, we need to monitor the behaviors of each component, such as the frequency of visits to each component. According to the state monitored, resources could be re-allocated to improve the reliability of the system. Figure 1 describes the main framework of our mechanism.

Reliability evaluation of modular software system
Notations:

- R_i: reliability of component i
- R_1: initial state
- R_n: absorbing state
- F_i: the difficulty to improve reliability of component i
- P_{ij}: probability of X_{n+1} being in state j given X_n is in state i
- μ_i: expected value of the number of visits to component i
- R_s: reliability of the system S

Reliability assessment: To construct a component-based software system, it involves assembling components together and interactions among these logically independent components which can be implemented and tested individually. For this approach, we have the following assumptions:

- The failures of the components are independent. A software system can be viewed as composed of logically independent modules which can be implemented and executed individually. That is a component failure which will not affect other components
- The transfer of control among software components follows a Markov process. The exchanges of control among these components are characterized according to the rules of a Markov process

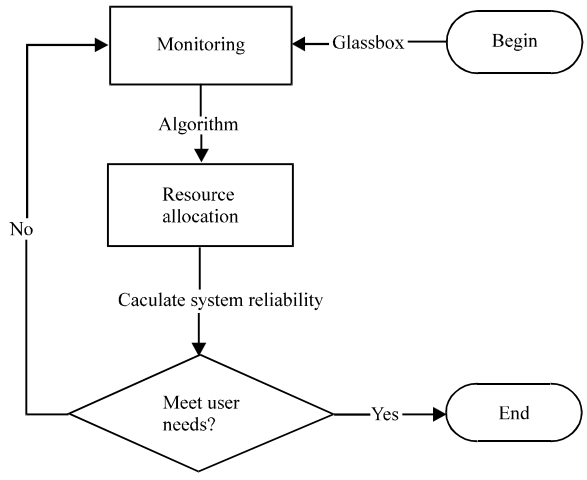

Fig. 1: Main framework of our mechanism

Under the assumption of independence among the successive executions of the components, the reliability of a system can be predicted by the path-based model (Gokhale and Trivedi, 2002). For example, if a system consists of n components with reliabilities denoted by R_1, R_2,....,R_n, respectively the reliabilities of an execution path, 1, 3, 2, 3, 2, 4, 3, 4, n.

The reliability of this path is:

$$R_L = R_1 \times R_2^2 \times R_3^3 \times R_4^2 \times R_n$$

R_1, R_n can be viewed as a virtual node which can solve the problem that a system has multiple initial/absorbing states.

The reliability of the system is (Lo *et al*., 2005):

$$R_s = R_1 \times R_n \times \prod_{i=2}^{n-1} (R_i^{\mu_i}) \quad (1)$$

MONITORING

The purpose of monitoring is to monitor and control the system. It is not only a traditional field of research but also a common engineering technology. Monitoring technology has been a widely used in various fields. We can use remote monitoring technology to observe and control the state of the object which not only simplifies the management process but also improves the management efficiency. Monitoring is essential in some place, for example: the critical distributed applications (the main business of telecommunications and banking). Monitoring software allows managers to master the system operating conditions in time and timely treatment of a number of exceptions to ensure the safe operation of the system.

Glassbox is a management agent with a browser-based interface that automatically pinpoints common

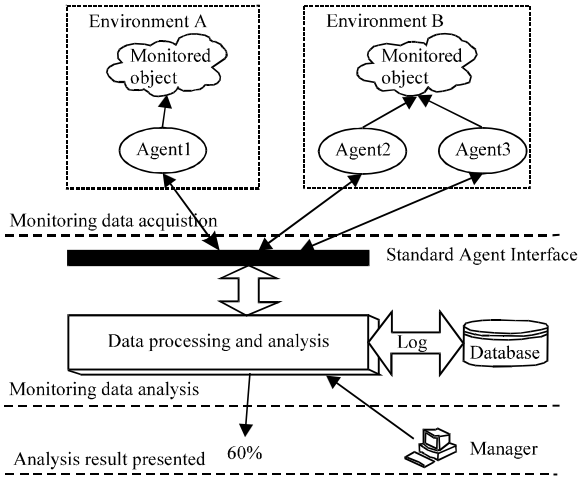

Fig. 2: Shows the hierarchical structure of the monitoring system

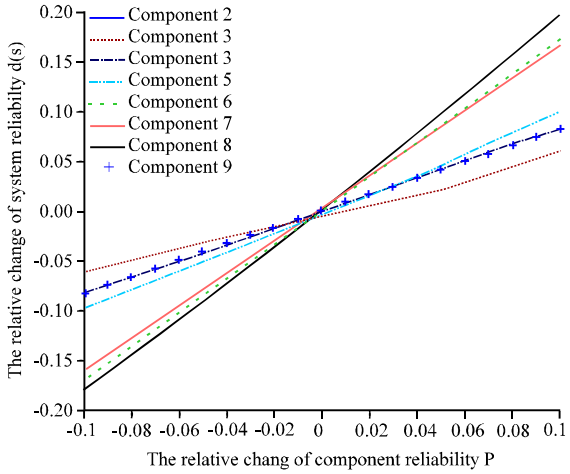

Fig. 3: The relationship between component and system

(AOP) and Java Management Extensions (JMX) technology to monitor the enterprise java, without forcing you to embed anything or change a single line of code. It provides a real time diagnosis of the system and cross-references it against both the service levels and our knowledge base of failures. Glassbox can carry out depth performance monitoring, for example: capture the total number of requests, time; database requests in all requests. In this study, we conduct experiments with the Glassbox 2.0 version and mainly calculate the access probability of individual components within a period of time (Fig. 2).

RESOURCE ALLOCATION

Sensitivity analysis: According to the (1), it would be helpful to know which component reliability most affects the reliability of the system, so that more accurate measurements can be made for the important one.

The following formula is:

$$\left|\frac{\delta R_S}{\delta R_i}\right| \geq \left|\frac{\delta R_S}{\delta R_j}\right| \tag{2}$$

We define S_p, R_i as the relative change of this system reliability, when is changed by 100 p%, so

$$S_{P,R_i} = \frac{\left|R_s(R_1,R_2,\ldots,R_i + p\times R_i,\ldots,R_n) - R_s\right|}{R_s} \tag{3}$$

$$\left|\frac{S_{P,R_i}}{P}\right| \geq \left|\frac{S_{P,R_j}}{P}\right| \tag{4}$$

On the base of (1), (3), (4), we can get that:

$$(1+p)^{\mu_i} \geq (1+p)^{\mu_j}, p > 0; (1+p)^{\mu_i} \leq (1+p)^{\mu_j}, p < 0 \tag{5}$$

According to (5), we have the result that the component which is most frequently accessed is the most sensitive one.

In Fig. 3, frequency of visits to component 8 is the highest, so component 8 is the most sensitive. However, frequency of visits to component 3 is the lowest, so component 3 is the least sensitive. We can reduce the resources which belong to the low sensitivity component to the high sensitivity one which can improve the system reliability.

Algorithm: Assuming that all components of the system share one kind of limited resources, such as redundant resources which can be treated as a resource pool, resources are deployed to the various components in the optimal way when the system are released. We can infer the following information from Fig. 4:

- It is an exponential growth relationship between resources cost and component reliability
- To achieve maximum reliability, the cost will be infinite in theory
- Higher reliability requirements, higher resource cost needs. The reliability of a component increases from 0.8 to 0.9 is easier than from 0.9 to 1.0

Because we do not focus on the research about the relationship between resources cost and component reliability in this study, we ignore the type of resources

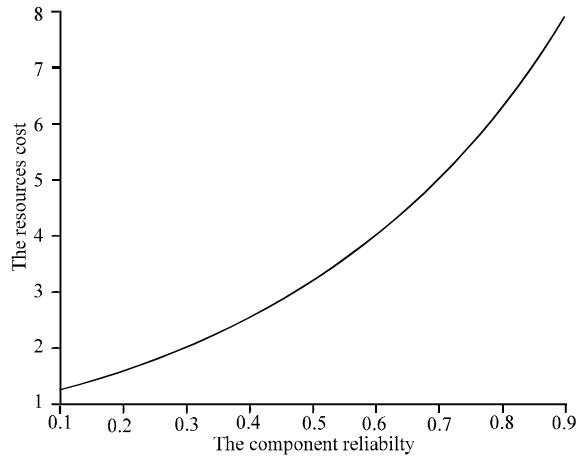

Fig. 4: The relationship between resources cost and component reliability

and simply view that F_i equal R_i and changes with R_i in synchronism. The values of F_i can range from 0-1, the bigger F_i is the component reliability is more difficult to enhance while added the same number of resources.

Because of the open environment is dynamic and unpredictable, how often to re-allocate resources becomes an important issue. If the cycle is too long, it may lead the system hysteretic and can't respond to user's needs in time; On the contrary, cycle is too short, component fails to make full use of resources. Frequent re-allocation would increase the system overhead and bring bumpiness to the system. Therefore, the granularity of resource allocation cycle is very important, usually, experts in this field have proposed methods to decide the allocation frequency (Tao *et al*., 2006).Greedy algorithm proposed in this paper is an improved method which solve problem by a step by step manner. According to the final goal which may be the objective function or not, every step must ensure that the solution is local optimal solution. Literature (Tian and Dai, 2007) has verified that greedy algorithm is a better method in the resource allocation.

Equation 5 represents the result that different components have different sensitivity because of the different transition probability. Figure 4 shows the bigger F_i is, the component reliability is more difficult to enhance on the unit resource. As total resources are limited, we can adjust the resources dynamically to improve the reliability of the system according to the following two aspects. On the one hand, less resources are allocated when low access to components and more resources are allocated when high access to components. On the other hand, when F_i is too high, we needn't to continue to improve the reliability of component I.

The process of design adjustment algorithm is as follows:

Step 1: Sort components from low to high by μ_i

Step 2: Compute system reliability using (1). If the current system reliability does not meet users' requirements, go to Step 3, else the end

Step 3: For (I = 0; i<n; i++) do remove unit resource from component i, calculate the new R_i, F_i as R'_i, F'_i . Then find the component i with the min (R_i-R'_i)

Step 4: For (j = n; j>0; j--) do add unit resource from component j, calculate the new R_j, F_j, as R'_j F'_j. Then find the component j with the max (R'_j-F'_j)

Step 5: Remove unit resource from component i to component j

Step 6: Repeat step 2

EXPERIMENTS AND RESULTS

The case study chosen for illustrative purpose is a J2EE application. The tools of Glassbox 2.0 and web server Tomcat 5.5 are adopted. These experiments run on a local pc with Intel i3 370 2.40GHz CPU, 2GB RAM, Windows 7.

The system consists of 10 components where component 1 is the initial state with the reliability 0.99 and component 10 the absorbing state with the reliability 0.99. Each component represents an application module, such as access to the database module, login module, registration module and so on. Figure 5 depicts the control-flow graph of the first example and the transition probabilities among the components are given as follow: $P_{1,2}$ = 0.6, $P_{1,3}$ = 0.2, $P_{1,4}$ = 0.2, $P_{2,4}$ = 0.3, $P_{2,5}$ = 0.4, $P_{2,7}$ = 0.3, $P_{3,6}$ = 0.5, $P_{3,9}$ = 0.5, $P_{4,3}$ = 0.4, $P_{4,5}$ = 0.4, $P_{4,8}$ = 1.0, $P_{5,7}$ = 1.0, $P_{6,4}$ = 0.3, $P_{6,7}$ = 0.4, $P_{6,8}$ = 0.3, $P_{7,8}$ = 0.5, $P_{7,10}$ = 0.5, $P_{8,9}$ = 0.3, $P_{8,10}$ = 0.7, $P_{9,5}$ = 0.6, $P_{9,6}$ = 0.4.

Resource allocation: When the system is released, the expected number of visits on each component before absorbing state from the initial state and the reliability of each component, the difficulty to improve reliability of component i can be derived as listed in Table 1.

Thus, the system reliability when the system released is 0.810 according to Table 1.

System running after a period of time: The Glassbox is monitoring the behaviors of each component, especially μ_i. After a period of time, due to the dynamic properties of the system and variety of visited probability, the expected number of visits on each component has changed and we obtain the new μ_i as the Table 2.

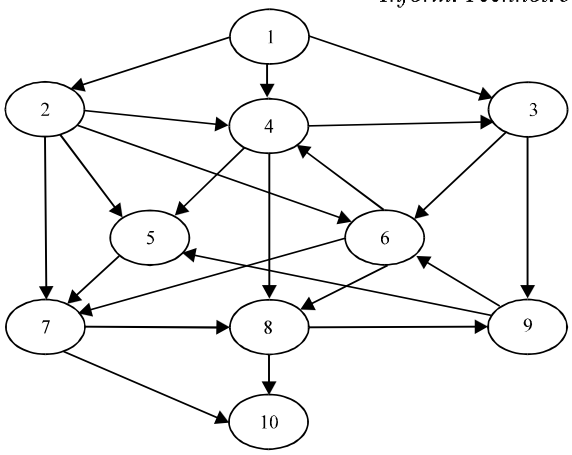

Fig. 5: Component based architecture

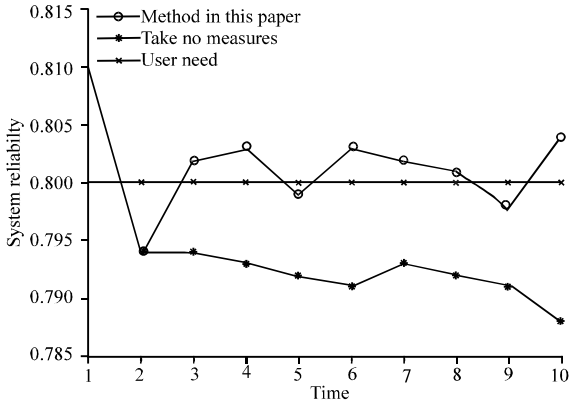

Fig. 6: The system reliability in the run time

Table 1: Component reliability and access frequency when the system is released

Component	2	3	4	5	6	7	8	9
R_i	0.97	0.97	0.97	0.99	0.97	0.98	0.99	0.98
F_i	0.97	0.97	0.97	0.99	0.97	0.98	0.99	0.98
μ_i	0.61	0.59	0.82	1.42	0.91	1.62	1.88	0.83

Table 2: Component reliability and access frequency after a period time

Component	2	3	4	5	6	7	8	9
R_i	0.97	0.97	0.97	0.99	0.97	0.98	0.99	0.98
F_i	0.97	0.97	0.97	0.99	0.97	0.98	0.99	0.98
μ_i	0.61	0.59	0.82	0.98	1.72	1.62	1.88	0.83

Table 3: Component reliability and access frequency after adjustment

Component	2	3	4	5	6	7	8	9
R_i	0.964	0.963	0.97	0.99	0.976	0.98	0.992	0.98
F_i	0.97	0.96	0.97	0.99	0.97	0.98	1.00	0.98
μ_i	0.61	0.59	0.82	0.98	1.72	1.62	1.88	0.83

Compared to Table 1, μ_5 in Table 2 drops to 0.980 and μ_6 increases to 1.720. According to (1), the new system reliability is estimated as 0.794.

If the user's requirement is not less than 0.800, we need to take measures to achieve this reliability of the system.

After re-allocate resources: As it is listed in Table 2, μ_8 is 1.880, then component 8 is the most sensitive, 8 is 0.590, so component 3 is the least sensitive. According to our algorithm, remove unit of resources from component 3, allocate it to the component 8. While $(R_3\text{-}R'_3)^{\mu 3} < (R_2\text{-}R'_2)^{\mu 2}$, we begin to remove resources from component 2 which is the second lowest sensitive component. On the contrary, when $(R'_8\text{-}R_8)^{\mu 8} < (R'_6\text{-}R_6)^{\mu 6}$, we begin to add resources to component 6 which is the second highest sensitive component. After adjustment, the reliability of each component listed in Table 3.

From Table 3 we can see that R_3 drops to 0.963, R_2drops to 0.964, R_8 increases to 0.992 and R_6 increases to 0.976 According to (1), the system reliability is recalculated and the satisfied result is 0.803. At this time, 0.803 exceeds the user's requirement 0.800, so we have completed the task successfully in one monitoring and resource allocation cycle.

Comparison: We have known that user's requirement is not less than 0.800. As Fig. 6 shows, when the system is released, the system reliability is 0.810, due to the dynamic expected number of visits on each component, the new reliability is 0.794. If no measures are taken to deal with in this situation, the reliability is still 0.794 and changes randomly in the future. According to our algorithm, unit of resources from lowest sensitive component is removed and allocated to the highest sensitive component, so the reliability is 0.803. As time goes on, resources can be re-allocated dynamically to meet user's needs. We can conclude that taking measures to monitor and re-allocate resources are necessary and our method can work well and effectively.

CONCLUSIONS AND FUTURE WORK

We presented an online reliability monitoring approach that takes advantage of static modeling and dynamic analysis to give continuous estimation of the system reliability. Due to the dynamic properties of the system and variety of visited probability, we use path-based method and analyze the reliability of software system on the level of components. Sensitivity analysis provides a way to analyze the impact of the different components. Glassbox 2.0 is used to monitor an enterprise java application. Furthermore, we propose a greedy algorithm for allocating resources to improve the reliability of the system in this study. Finally, some experiments are evaluated to validate and show the effectiveness of the

proposed method. We aim to do these in the future. First, we will modify the source code of Glassbox 2.0 to meet our requirements. Second, a better algorithm will be designed to re-allocate resources. Finally, the proposed mechanism will be improved in a reasonable way.

REFERENCES

Gokhale, S.S. and K.S. Trivedi, 2002. Reliability prediction and sensitivity analysis based on software architecture. Proceedings of the 13th International Symposium on Software Reliability Engineering, (SRE'02), IEEE Xplore, pp: 64-75.

Goseva-Popstojanova, K. and K.S. Trivedi, 2001. Architecture-based approach to reliability assessment of software system. Perform. Eval., 45: 179-204.

Lo, J., C. Huang, I. Chen, S.Y. Kuo and M.R. Lyu, 2005. Reliability assessment and sensitivity analysis of software reliability growth modeling based on software module structure. J. Syst. Software, 76: 3-13.

Pietrantuono, R., S. Russo and K.S. Trivedi, 2010. Software reliability and testing time allocation: An architecture-based approach. IEEE Trans. Software Eng., 36: 323-337.

Tao, J., Q.L. Wu and Q. Wu, 2006. Application research of network resource allocation algorithm based on non-cooperative bidding game. Acta Electron. Sin., 34: 241-246.

Tian, J. and Y.F. Dai, 2007. Study on durable peer-to-peer storage techniques. J. Software, 18: 1379-1399.

Yacoub, S.M., B. Cukic and H. Ammar, 1999. Scenario-based reliability analysis of component-based software. Proceedings of the 10th International Symposium on Software Reliability Engineering, Nov. 1-4, Boca Raton, FL., USA., pp: 22-31.

Zulkernine, M. and R. Seviora, 2005. Towards automatic monitoring of component-based software systems. J. Syst. Software, 74: 15-24.