Improved RPPI Strategy for Software Reliability Analysis Basing on Multi-objective Optimizing Method

Bingchao Lei

College of Mathematics and Computer Science, Guangxi University for Nationalities Guangxi 530006, China

Abstract

The RPPI Strategy is improved by involving in the testing costs as an evaluation factor. In software testing stage, the component software with both high RPPI value and low testing costs is given advantage to be tested. The new proposed method avoids the defects in common RPPI strategy, which is often impracticable because of too much extra testing costs. Since both the objective of higher RPPI value and the objective of low costs are simultaneously required, the problem is modeled as a multi-objective optimization problem and resolved by the optimization algorithm such as the goal programming method and the Pareto algorithm. The method proposed is practicable in software reliability analysis.

Keywords: RPPI Strategy, Software Reliability, Optimization Model, Reliability Prediction Prioritization Strategy

1. Introduction

According to several recent studies, software failures greatly reduce system reliability and availability, which contribute to 20-30% of system failures. Software bugs can crash the system, making the service unavailable, corrupting information, generating wrong results, and leading to unavailability. Besides software defects, administrative errors are another major cause for system failures. Recent studies have also shown that about 37% of failures in Internet services are caused by administrator mistakes. To improve software reliability and reduce software failures as well as administrative errors, many reliability assurance techniques have been used at different stages of the software life-cycle, including software testing, software patching, and online validation of administrative reconfigurations. As for software reliability, two factors must be noticed: firstly, proper mathematic model is needed to evaluate the software reliability; secondly, proper optimizing model is also needed to optimize the software reliability. Many researches have been devoted to these two tasks and the according research results are fruitful.

However, the traditional mathematic models for software reliability evaluation and optimizing models for software reliability optimization treat the software reliability as a deterministic value. In fact, the software reliability is often stochastic since the insufficiency of data for reliability evaluation. Insufficiency of data brings deviation to the reliability evaluation result and such deviation can accumulate during the integration of the software from component level to the system level. Clearly, it's of importance to decrease the evolution deviation of the reliability of the component software to avoid the deviation of the whole software system.

The paper proposes both single objective and multi-objective optimizing model for the optimization of the evaluation of the components software reliability. The basing on the optimization model proposed, a reliability prediction prioritization strategy is also proposed to evaluate the reliability of the software system at high efficiency and low costs.

2. Optimization Model for Components Software Reliability

The optimization models for the components software reliability can be classified into two types according to the number of the objective must be achieved, when only one objective must be achieved, in most case to decrease the deviation of the software reliability evaluated, the type of optimization model is called single objective optimization. However, when multi-objectives, such as the low costs, high efficiency and so on, are expected to be achieved, the optimization is called multi-objective optimization. The two types of optimization model meet different needs of the component software reliability evaluation and are discussed as the follows:

2.1 Single purpose optimization model

Sometimes, the architecture of the software is pre-selected and the selection strategy of the component is also predefined, the optimization objective is only to decrease the deviation of the evaluation value of the component software reliability. Since the components are not replaceable or redundantly configurable, the objective is confined to decrease the evaluation deviation by allocate extra testing resource for the component software needing to provide more accurate evaluation result. Unfortunately, the testing resource is often limited with a maximal amount, assuming that the limitation testing cost is C and the unit testing cost of the k th component software is ck, the evaluation deviation of the i th component is D(Ri), the single objective optimization model can be given as the follows:

$$Min \quad D(R_i) \quad i = 1, 2, \dots, s$$

$$S.t \quad \sum_{k=1}^{s} c_k n_k \le C$$

Where n_k is the number of the testing unit allocated to the *k* th component software. Then, the single objective optimization model gives the best solution to the evaluation of the component software reliability, which can be solved by many optimization algorithms such as linear programming and dynamic programming to achieve the best solution.

2.2 Multi-objective optimization model

The single objective optimization model emphasizes on improving a certain single objective of the reliability of the component software with the limitations such as the testing resource, the testing efficiency and etc. However, during the integration period, the number of the components increases dramatically and the software system becomes more and more complex, many objectives must be considered simultaneously, for example, the lowest testing costs ,the shortest testing time and highest evaluation precision must be achieved at the same time. Thus, the optimization problem must be modeled by multi-objective optimization model. As for the software reliability analysis, the most common form of the optimization problem is to maximize the software reliability as well as to minimize the deviation of the evaluation and the testing costs. So, in the multi-objective optimization model, the maximum of the component software reliability and the minimum of the deviation of the evaluation must be given in the objective function simultaneously and the optimization model can be given as the follows:

$$\max R_i, \quad \min D(R_i)$$

s.t: $\sum_{k=1}^{s} c_k n_k \le C$

Where R_i is the evaluation value of the *i* th component software, D(Ri) is the evaluation deviation of R_i . C_k and nk are the unit testing costs of the *k* th testing cost and the number of the testing units, respectively; *C* is the limitation of the total cost allocated to the I the component software.

The solution procedure of the multi-objective optimization is more complex than that of the single objective optimization problem. Goal programming method and Pareto algorithm are two most efficient methods to solve such optimization problem. The goal programming method transforms the multi-objective optimization problem to a series of single objective optimization problems and then given the solution by solving each single objective optimization problems and the testing engineer are acquainted to the scope the objective function, the goal programming method will surely give the best solution. However, when the knowledge of the component software and the whole software system is not so sufficient, the Pareto algorithm may be the preference to solve the optimization problem, which gives all the practicable solutions to the model then the analyst and testing engineer of the software can choose a best answer from all the solutions.

3. Reliability Prediction Prioritization Strategy

It's noticeable that the reliability evaluation values of different component software haven't the same influence to the whole software system: Some reliability value of the component software is negligible to the whole software system while the other component software is of significant importance to the software system. Such potential ability to improve the systemic reliability is often called the sensitivity of the reliability. The common method to decrease the evaluation deviation of the reliability of the software is giving extra testing. However, component software has different sensitivity and influence to the systemic reliability and the component software must be sorted in the order of its sensitivity to the whole software system. The sorted series of the component software with bigger sensitivity has more influence to the software system, and the reliability of such component must be improved to improve the whole system's reliability. The optimizing strategy to basing on the RPPI yields the RPP strategy model, which is defined as the follows:

$$\alpha_i = \frac{\delta_i^2 \cdot \sigma_i^2}{\sum_{j=1}^n \delta_j^2 \cdot \sigma_j^2}$$

Where σ_i is the deviation of the evolution value of the *i* th component software, δ_j is the sensitivity of the j th component software and has the form:

$$\delta_j^2 = \left[\frac{\partial R(t;r)}{\partial r_i} \bigg|_{r=r_i} \right]^2,$$

It's obvious that $\sum_{i=1}^{n} \alpha_i = 1$, according to the RPPI, the sensitivity of the component software are compared and sorted, and then different privilege groups are arranged. Assuming that the RPPI values are sorted in increasing order and a critical value θ is given, the component software with RPPI value less than θ will be arranged in group I, and on the contrary the component software with RPPI value bigger than θ will be arranged in group II. Different privileges are given to the two groups and the component software in the two group are tested differently: The component software in group II will be given more tests since it's sensitivity is bigger and has more influence to the reliability of the whole software system, however, the initially evaluated sensitivity value of the components in group I will not be updated since it has not so much influence to the whole system's reliability.

4. Improved Reliability Prediction Prioritization Strategy

The traditional RPPI strategy has the defects that it can only find out the component software that has higher sensitivity to the whole software system, which needs to decrease the evaluation deviation. However, it can not point out the difficulty to decrease the deviation and costs to decrease the deviation. If the testing cost is too exaggerate, even the component software with high enough RPPI will not become the preference of the testing engineers because both high RPPI and low testing cost are expected by the software developers.

Since high testing cost will cause the testing work is impracticable, the RPPI strategy must be improved to involve in the testing costs. By considering the testing cost when computing the RPPI value, the testing work will be practicable without the limitation of testing costs. To differentiate both the RPPI value and the testing cost of the component software, the RPPI value and the testing cost are given proper weight to make the difference. The improved reliability evaluation strategy is called RPPI-c strategy and the component series sorted by the RPPI-c value are called RPPI-c series. The model of the problem can be given as the follows:

$$\max w_{i1}\alpha_i, \quad \min w_{i2}(c_i \cdot n_i) \quad i = 1, 2, ..., n$$

s.t
$$\sum_{i=1}^n c_i n_i \le C$$

Where α_i is the RPPI value of the *i* th component software; c_i is the extra testing cost of the *i* th component software; n_i is the number of the extra testing unit allocated to the *i* th component software. W_{li} is the weight of the RPPI value of the *i* th component software. W2*i* is the weight of the testing cost of the *i* th component software. W2*i* is the weight of the testing cost of the *i* th component software. W2*i* is the weight of the testing cost of the *i* th component software. W2*i* is the weight of the component software is sorted and yields the RPPI series; and then, the component software with maximum value of RPPI is selected.

The solution procedure of the RPP-c model consists of 4 steps:

Step1: Compute the RPPI value α_i of the component software $r_i(t)$ are computed and arranged into group 1

or group 2 according to the critical value θ given. Then, the component software needs extra test is selected. **Step2:** each component software selected is evaluated to determine testing unit ni.

Step3: the RPPI-C values of the selected component software for testing are computed.

Step4: the testing resource and the testing time are allocated to the component software according to the RPPI-C value.

5. Conclusions

The RPPI Strategy is improved by involving in the testing costs as an evaluation factor. In software testing stage, the component software with both high RPPI value and low testing costs is given advantage to be tested. The

new proposed method avoids the defects in common RPPI strategy, which is often impracticable because of too much extra testing costs. Since both the objective of higher RPPI value and the objective of low costs are simultaneously required, the problem is modeled as a multi-objective optimization problem and resolved by the optimization algorithm such as the goal programming method and the Pareto algorithm. The method proposed is practicable in software reliability analysis.

References

D.H. Deng,Z.Y. Gong, Z.L. Guo and S. Phillip. (2008). Semantic programming of Web-enabled database applications, Proceedings of 1st IEEE International Workshop on Semantic Computing and Applications, IEEE Press, 2008, 51-60.

Priyadarshini, Pallavi,Qin, Fengqiong and Lim, Ee-Peng. (2004). Parameter driven synthetic web database generation, *Journal of Systems and Software*, Vol.69, Jan. 2004, pp.29-42.

Zhao Shuxin, Chang Elizabeth and Dillon Tharam. (2008). Knowledge extraction from web-based application source code: An approach to database reverse engineering for ontology development, 2008 IEEE International Conference on Information Reuse and Integration, IEEE Press, 2008, pp. 153-159.