Quality Management of Software Testing from the Perspective of Process Measurement

Guiquan Shen¹, Xiaojiang Chen², Zhenyue Long¹, Jijun Zeng¹, Daohuan Jiang¹
¹Guangdong Power Grid Corporation Information Center, Guangzhou 510600, Guangdong Province, China
²Guangdong Power Grid Corporation Jiangmen Power Supply Bureau, Jiangmen 529099, Guangdong Province, China

Abstract
This paper has studied the compliance of software testing to user requirements from the perspective of process measurement. The goal is to achieve scientific management of software testing process and ensure a high success rate of software testing. The paper has analyzed the use of software metrics to quantify the software testing process, and constructed process metrics and analyzed the effectiveness of process metrics data. At the same time, the paper has discussed the test efficiency, the software flaw as the necessity of the software testing workload and the achievement index, and analyzed the testing process by using the workload and the result model. Through the construction of process metrics by quantifying software-related attributes, this paper has analyzed the measurement results by using the workload and result models after implementing the metrics, and evaluated the projects and takes corresponding measures. In the case of existing resources, we can achieve effective management for the implementation of quality management, monitoring software testing process and software testing process.

Keywords: Process measurement; Software testing; Quality management

1. Introduction

With the rapid development of information technology, software products have been applied to all areas of society, and the quality of software products naturally become the focus of common concern. Just as the production of any product is inseparable from the quality inspection of products, software testing is an indispensable step in the development of software products. It occupies a very important position in the software life cycle, and is of great significance to software reliability assurance. Software testing work directly determines the level of software product quality. Software testing is to find software errors and the implementation of a program process, and the purpose is to find and correct as much as possible the error of the tested software to improve the reliability of the software. In the process of finding software errors, the management of the process is still important. A successful software testing project cannot be separated from the scientific organization and monitoring of software testing process. The management of software testing process has become an important guarantee for software testing success. However, the effective management of the quality of the software testing process cannot be separated from the measurement of the software testing process. In this paper, some measures are used to quantify the software testing process, and the results of the testing program are used to evaluate the progress of the testing project, so that the software testing process can be effectively managed, and the software testing efficiency can be improved as much as possible.

2. The theoretical model of software testing quality management from the perspective of process measurement

2.1 Problem analysis of software testing quality management
Based on the measurement results of the software testing process, this paper uses the workload and result model (EOM) to achieve quality management in software testing process. Optimization software testing methods can usually be divided into two categories, including random software testing and deterministic software testing, which is further divided into unconstrained optimization software testing and constrained optimization software testing. Software testing methods of the overall optimal quality is none of the essence of constraint optimization. In addition, the optimization software testing method can be divided into discrete optimization and continuous optimization. The “optimal testing mode based on the continuous software testing method” in this paper belongs to the unconstrained optimal software testing in deterministic models. Many constraints can be translated into unconstrained problems to study, and many discrete cases can also be converted into a continuous situation to study, and that is, the results of the continuous case can be applied to discrete situations. Compared with the local optimal software testing method, the testing method of global optimal software is far from being mature and perfect in theory research and practical quality management method due to its inherent characteristics, which is a worldwide theoretical and engineering problem. For unconstrained optimal software test questions, the following relation is satisfied in equation (1):
\[ \min F(X), X \in D \{ X \mid F(X) \} \subset R^n \] (1)

At the same time, many scholars at home and abroad have done a lot of good work, and put forward many effective software test quality management methods. However, most of these software test quality management methods are testing the optimality of local software management, and people are often interested and much needed to obtain the overall management optimality of the software test method in its domain, which is better than the local test. Therefore, the best quality is much more difficult. The current research on this issue is generally divided into two types of certainty and non-certainty. The former has Branin descent trajectory method, Levy tunneling method, Asaithambi and Shen interval method, and Ge Renpu filling software testing methods. Such methods can search for local minima based on a certain deterministic strategy, and then try to skip the local optimum obtained while seeking the best of other tests. The latter has Anderssen's uniform distribution search method, Hartman's multi-start method, Solis and Wets combined gradient information search method, and HarioFgel method, and the use of interval analysis can do this work.

2.2 The best quality model of software testing

The content studied in this paper is to extract the model expressed by the expression of software testing method from the practical application, and test the overall optimal software quality by the method of valley-peak to achieve the purpose of optimal software testing. Achieved by the peak method to test the best overall quality of continuous software testing method named FindMin-VM, all VPM refers to the valley method, and FindMin-VPM refers to the software tested in this paper. This paper relates to a unified software testing methods, binary software testing methods and \( n(\geq 2) \) Yuan software testing methods are described in the concept of a unified software testing methods.

Definition of the software test method \( f(x) \) is as in the interval \( (a, b) \), and \( x_0 \) is a point within \( (a, b) \). If there is a disintering test neighborhood at point \( x_0 \), \( f(x) < f(x_0) \) holds for any point \( x \) in the de-test neighborhood, and \( f(x_0) \) is the software test method. And point \( x_0 \) is called the optimal point. If there is a disintering test neighborhood at point \( x_0 \), for any point \( x \) in the de-centered test neighborhood, there is \( f(x) > f(x_0) \) holds, and it is said that \( f(x_0) \) is a test optimal method of software test \( f(x) \), and point \( x_0 \) is called the test best point, which satisfies the following relation:

\[ \max_{x \in D} f(x) = \min_{x \in D} (-f(x)) = f(x_0), x \in [a, b] \] (2)

The optimal management of software testing methods is collectively referred to as the best quality of software testing methods. The concept of management optimality for software test methods is partial. If \( f(x_0) \) is an optimal value for the software test method \( f(x) \), then \( f(x_0) \) is an optimal test for \( f(x) \) for only a local range around \( x_0 \). The \( f(x_0) \) is not necessarily the best test method for the entire domain, and is calculated as the following equation:

\[ f(x) \geq f(x_0), \forall x \in D \] (3)

The problem of finding the optimal software quality can be transformed into the problem of the optimal test. During the discussion, we only discuss the problem of the optimal test. When it comes to the local solution of the unconstrained optimal software test problem, the global solution refers to the problem of the local test the best point and the overall test the best point.

2.3 Function module design of software test

In order to use the simulation system to better achieve VPM software of test quality management methods, the testing of various types of continuous software are required to call the best overall quality and reasonable design of the test method, according to its testing procedures and software test quality management. The flowchart of each part of the method divides the overall structure of the software testing quality into multiple functional modules of the tree structure, which will be described in detail below.

Software test method module **Main**: This needs to call the software test user interface module **UI**, software test drawing module **Plot**, test module calculation module **Calculate**, as shown in Figure 1.

![Figure 1: Software testing method module](image)

Software testing user interface **UI** module: This includes a unified software testing method of user interface...
function UI1D, binary software testing method user interface function UI2D, and \( n \) element software testing method interface UInD, as shown in Figure 2.

![Figure 2 User interface UI module of software test](image)

Plot software testing plots: This includes a one-unitary software testing method of Plot1D drawing function, binary software testing method of Plot2D drawing functions, which is shown in Figure 3.

![Figure 3 software test plot module](image)

Software test calculation module: This includes one-unitary software test method of overall optimal quality function of Ex1D, binary software test method of overall optimal quality function of Ex2D, binary software test method of local best quality function of AEEx2D, \( n \) element software test method of overall best quality functionality of ExnD, and \( n \)-element software test method of local best quality function of AEExnD, which is shown in Figure 4.

![Figure 4 Calculate module of software test](image)

### 2.4 Step and process design of software testing method

1. One-unitary software testing method step
   - Let \( f(x) \) define the domain from \( a \) to \( b \), and define two lists of \( list_1 \) and \( list_2 \), which can store the optimality of all the software test methods.
   - The first step: software testing methods \( f(a) \) and \( f(b) \), are deposited into \( list_1 \) and \( list_2 \).
   - The second step: Suppose \( a_1 = a \), and set the appropriate step \( h > 0 \).
   - The third step: If \( a_1 + h > b \), turn to the seventh step, otherwise turn to the fourth step.
   - Fourth step: For the software test of \( f(a_1 + h) \), if \( f(a_1 + h) < f(a_1) \), turn to the fifth step, otherwise turn to the sixth step.
   - The fifth step: Use software testing golden section method, which is starting from \( a_1 \), and is testing the optimality of adjacent software of \( f(x_{Min}) \), and if \( x_{Min} > b \), go to the seventh step, otherwise store them in \( list_1 \), and therefore \( a_1 = x_{Min} \), and go to the sixth step.
   - The sixth step: Let the software test method \( f(x) = -f(x) \), and test the corresponding software optimality according to the test points of \( (a_1, f(a_1), f(x_{Max})) \). If \( x_{Max} > b \), go to step 7. Otherwise, store in \( list_2 \), and let \( a_1 = x_{Max} \), \( f(x) = -f(x) \).
   - The seventh step: Compare all \( f(x_{Min}) \) values in \( list_1 \) to find the optimal test method, which is the optimal test for the software test method. Compare all \( f(x_{Max}) \) values in \( list_2 \) to find the optimal test method, which is the optimality of the software test method.

2. Binary software testing method step
   - Let both arguments of \( f(x_1, x_2) \) be from \( a \) to \( b \). Define list of \( list_1 \) and \( list_2 \), which can store test management
optimality.

First step: Let $ax_1 = a$, and test the local management optimality of the continuous software test method $f(ax_1, x_2)$.

The second step: Determine the optimality of local test management, and judge whether there is a downward trend in the quality of the test point, and if there is, turn to the third step; and if not, then come to the optimal test, and turn to the fifth step.

The third step: Discard the point where the test quality is going up, and test the optimality of binary local test management corresponding to all points with a decreasing trend, and find the nearest point of $ax_1$ in the local test management optimality.

The fourth step: Determine whether the point of cross-border, if there is not cross-border, turn to the fifth step, or turn to the eleventh step.

The fifth step: Save the point list of (0), so that $ax_1$ is equal to the value of the point of $x_1$.

The sixth step: Test the global test management optimality of the univocal software test method of $f(ax_1 + h, x_2)$, which is making it $TMin_1$.

The seventh step: Determine whether $TMin_1$ is smaller than list (0), if it is smaller than 8, turn to Step 8; otherwise, turn to Step 9.

The eighth step: Test the optimality of binary local software management corresponding to $TMin_1$, and turn to Step 4.

The ninth step: Let $ax_1 = ax_1 + h$, to determine whether $ax_1$ cross-border, if the cross-border turn to the twelfth step, otherwise turn to Step 7.

The tenth step: Test the global software management optimality of the univocal software test method of $f(ax_1 + h, x_2)$, which is making it $TMin_2$.

The eleventh step: Determine if $TMin_2$ is less than $TMin_1$, if it is less than $TMin_1$, $TMin_1 = TMin_2$, go to Step 8; otherwise $TMin_1 = TMin_2$ and go to Step 9.

The twelfth step: Compare list storage software management, and draw the global software management optimality.

(3) The best method step for continuous software testing

Let both arguments of $f(x_1, x_2)$ be from $a$ to $b$. Define the list of test lists, which is used to store the optimal software management.

The first step: Let $ax_1 = a$, and test the optimality of all the local software for the univocal software test method of $f(ax_1, x_2)$.

The second step: Determine if there is any downward trend in the optimality of all local software management, and if there is, turn to the third step; otherwise, turn to Step 6.

The third step: Discard the point of upward trend, and test the binary local software management optimality corresponding to all points with downward trend, and find the nearest points in the local software management optimality to $ax_1$.

The fourth step: Determine whether the point of cross-border, if not cross-border, turn to the fifth step, or turn to the eighth step.

The fifth step: Save the point list (0), so that $ax_1$ is equal to the value of the point of $x_1$.

The sixth step: Let $ax_1 = ax_1 + h$, and determine whether $ax_1$ is cross-border, if it is cross-border turn to the eighth step, otherwise turn to the seventh step.

The seventh step: Test all the local software of the software test method of $f(ax_1, x_2)$ to manage the optimality and turn to the second step.

The eighth step: Test the optimality of binary local software management corresponding to $TMin_1$, and turn to Step 4.

(4) Flow chart of software test quality management method
The Calculate module in the software test of the various modules need to call methods include comparison of the software testing for the best (Compare method), and find the quality of the downward trend point (DownTrend) method, and find the binary software test method Local (Powell method), and all local optimal quality of the software testing method (AEx1D method, AEx1D method in turn calls MI to determine the local optimal quality interval, and EGD to find the local optimal gold partitioning method). The software test quality management method of flow chart is shown in Figure 5.

3. Experimental results of software testing quality management from the perspective of process measurement

3.1 Experimental result of one unitary software test method

A continuous software testing method for the overall best quality program is as follows:
Expression is selected as the software test method expression. The starting point is selected as the left boundary of the domain of (0.1), and the end of the domain is selected as the right boundary of the domain of (14.56), and the selection of test precision is 0.01, and the range of test results is selected as (11.0855, -11.0407). The software testing method is shown in Figure 6.

3.2 Experimental results of binary software test method

Binary continuous software testing method to find the overall best quality program is as follows:
Experimental results of binary software test method are shown in Figure 7. The left boundary of the software test domain is (-3), and the right boundary of the test domain of x is (3), and the left boundary of the test domain is (-3), and the right boundary of the test domain is (3), and the test accuracy is 0.1, and the test results are (0.22826, -1.6255, -6.5511) with all local best quality of the binary software test method.

4. Conclusion

In this paper, the introduction of software process metrics into test quality management provides reference information for checking software code quality and project progress. In the measurement of the software testing process, it is important to measure the validity rather than the number of measurements. Although metrics are easy to set up, poor metrics are not only useless but in fact can lead to the opposite effect and increase project overhead. Therefore, in the process of setting up the measurement, we should pay attention to that each measure should follow the basic principle of test theory, and reflect the experience value. The concept, the definition of implementation, the unit of measure, the validity and the reliability issues need to be carefully considered. In this paper, the determination of the weight of defects may bring some subjective factors, which requires that we must have a good understanding of the measurement and data in order to make the right decision. Of course, the data analysis part of software testing quality management needs to be further tapped. We can also ask more questions about the quality of software testing and find out their answers. This is also the focus of our work in the future.

References


Wang Z, Feng X, Yang H. Research on design process control of ground testing software for the small satellite, Control and Decision Conference. IEEE, 2013:4085-4090.