2012 4th International Conference on Computer Engineering and Technology (ICCET 2012)
IPCSIT vol.40 (2012) © (2012) IACSIT Press, Singapore

The Parallelization Index for Efficient Testing Resource Management

Seung-yeon You [1] and Sung-chun Kim [2]

[1] Quality Manager in SK Communications, South Korea

[2] Computer Science and Engineering Department Professor in Sogang University, South Korea

Abstract. Most tests are implemented either even without any specific planning or with a rough plan. It causes to leave efficient resource management for testing out of consideration. As you know, effectiveness and efficiency are important things in the software engineering area. In this paper, we suggest the concept of parallelization index for efficient software test which can exercise helpful influence on planning parallel test execution. This index is meaningful in that it can be utilized as the threshold value to express the resource efficiency of a software test in execution.

Keywords: Software Test, Parallel Test, Resource Management, Testing Plan

1. Introduction

A software test process progresses in the procedures of plan, design, test, result analysis and report, and test process improvement just the same as in a software development process. Among these respective procedures the step which has the closest relation with efficient resource management is the planning of a test. Nevertheless, it seems, in so many cases in reality, tests are implemented either even without any specific planning or with a rough plan in which simply the matters that a certain numbered teams will conduct the test for a certain numbered days under a certain environment are set. In contrast that normally a software development plan is laid up in detail to the level of the plan for specific modules, in which many developers can participate and implement a plan in parallel, in many cases a software test plan is made so roughly and ambiguously that in the implementation step of the test the resources become to be inefficiently managed and wasted. Resources in a software test are time, people, and instruments which are needed for the overall test processes including test planning, test implementing, test environment setting, error analysis, aftermath test assessment, reporting, and derived defects ascertaining and debugging. This study takes much note of testing time and human resources among these resources, and suggests a method for parallel test planning particularly focusing on the matter of human resources. The core concept of the method which we suggest in this study is to derive the parallelization index for a test. If a parallelization index is figured out according to software architecture, it can be used as the threshold value for the test, which can be much helpful for the planning and processing of a test. The suggested parallelization index is also meaningful in that it can provide a cost efficient solution for human resource management. Ahead, in Chapter 2, the concrete concept of the parallelization index is elucidated, and in Chapter 3, the method is explained in detail through actual examples and the conclusion is drawn.

2. What is The Testing Parallelization Index

The names of independent parts of a software are various according to the kind of development languages by which a software is developed and the kind of the software itself. Of course, though the components of a software may be separated into functions, interfaces, classes, components, modules, items, operations and menus, what is important among these parts is the unit which can be allocated to the individual members of human resources, thus the software parts for allocation in real tests may be classified

with convenience according to specific cases. In this study, the unit of software parts for allocation in a test is defined as a module for our convenience. The interrelationship between software modules can be grasped by being based on the software design structure, by which the processes which can be tested in parallel can be assorted for making an effective test planning. In many cases in reality, failure is caused in the management of resources in a test, as efficient processing is not prepared in the test into which many testing members or teams are input into and accordingly which is shown magnified externally. Sometimes unnecessary many human resources are input into a test, or sometimes, on the contrary, test quality is worsened because of the lack of human resources and time compared to the overly planned original scale. Naturally and deservedly, what is most important for test quality enhance is substantial planning. In this study, the parallelization index for software test is drawn out based on the interrelationship between software modules for making out a substantial test plan.

If a test plan for QC team was written down without the consideration for resource efficiency, the testing by this plan will be processed according to the sequence of modules in the software design structure. Though there are cases in which an individual member of the same test team conducts the test work by a different test scenario from fellows of the team or with a different test case one another, these cases can not be regarded as an actual systematic parallel test in view of the flow of a test. For example how many members/teams will be needed for conducting test in the case that the software design structure in the figure 1-a, below, is given?

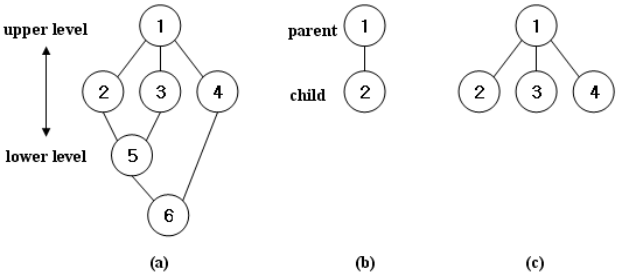

Fig. 1: (a) SW design structure, (b) parent/child node, (c) 3 child nodes

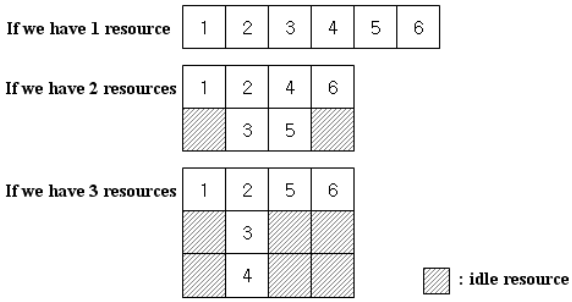

Fig.2 assigning modules into resources in each possible case

In this study, the discretion between an individual member and a team for a test is not employed. This is because what our consideration should be laid on is the unit for the execution of parallel test, and this unit may be either an individual member or a team. In our paper, the simple term resource is used for the unit of human resources in a test. As in Figure 1-a, if a software design structure is schematized with a graph, the grasp of the interrelationship between grows easier. In the graph, a software module to test is expressed as a node and the interrelationship of modules as an edge. As seen in Figure 1-b, For the modules which have the interrelationship of a parent node and a child node, it is more efficient to conduct testing in sequence than in parallel. It is because, in this case the test execution result of the parent node exercises influence on that of the child node, only when the test is processed in sequence, the test result of the parent node can be reflected straightly and immediately to the subsequent procedure, that is, the procedure of child node. This fact is very easily identified in a test scenario. In contrast, in case that a parent node has numbers of child nodes, the interrelationship between child nodes is very low, separated test progress according to the node connection will be available. In Figure 1-c, the parent node has 3 units of child nodes for which independent test progressing is available, and it is assumed that the leader of the test group made a plan to execute the test by allocating one module per one human resource for the entire 3 units of resources respectively. Though in this case the plan was made out by considering parallel testing for available modules, it is not desirable overall. Although the entire test execution time can be shortened, some specific periods in which some test resources do not have any allocated works may occur, and maintenance and management cost will be spent uselessly. This shows an example that a test plan by considering resource efficiency resulted in an inefficient solution reversely to the original intention.

Revealing the conclusion first, the parallelization index for software testing is the value which maximizes the average usage rate of a resource per a unit time, which can be used for making out a plan for the most efficient test. For example, if a testing parallelization index is 3, it means that to apply 3 test

resources, that is 3 testing team is most appropriate for the test. In order to draw out this parallelization index, a graph based on a software design structure should be created first and the longest path from the start node to the end node should be found in this graph, as referred above. The modules included in one same path is tested in sequential, not in parallel. Because of this, if it is assumed that the test time for one unit module is defined as a unit test time and the scale of this unit test time is similar for all modules, the total test time cannot be shortened less than the number of modules included in the longest path, no matter what way of parallel processing is applied. In another aspect, it is also because processing an integrated test or a system test with the interrelationship between modules being neglected may cause meaningless testing. Thus we allocate modules to respective resources so that the test might be conducted in parallel mode with the interrelationship between modules being considered. The basic allocation rules are like below.

(1) The allocation is conducted priorly for higher level in sequence in the graph.

(2)The allocation is conducted priorly for the nodes which are included in the longest path from the beginning node to the end node among the various paths.

(3-1) The test for the parent node (the upper node) of a node for allocation should have been completed in advance as it had been allocated to a resource.

(3-2) If the test for the parent node of a node for allocation has not been completed in advance, the node for allocation is to stay in wait state till the test for the parent node will have been completed.

It is natural that in parallel test, the testing time is shortened compared to the case that the test is conducted in sequence with only one resource. However, it is necessary to find out the case in which the average resource usage rate per a unit time becomes maximal and use the rate value because only to increase resource input to shorten test time is not efficient in the aspect of cost. And we figure out this resource usage value by calculation and define as the software testing parallelization index. If resources of more than the software parallelization index are input, resource waste will be caused, and if less is input, testing time will be increased uselessly. This parallelization index defined for software test is meaningful in that it can be used as the threshold value by which the efficiency of a parallel test can be weighed in this way.

3. Parallelization Index

We define that the testing time for a module is an unit time. We can get the resource threshold value, the number of resources, when the average resource utilization per unit time becomes maximal. It is the parallelization index.

(1) Draw the SW design structure using graph expression

(2) Assign the modules into available human resources

(3) Calculate the average resource utilization per unit time for all possible cases

(4) Select the maximum value among the results of (3)

(5) At that time, the number of resources is the parallel index

Let us calculate the parallelization index for the example graph, figure 1-a. We may have 1/2/3 resources or more. If we assign modules to test into resources, it becomes like figure 2. In the case of 2 resources, it takes 4 unit times to finish the test, and there are 2 idle resources at the first and the forth unit times. The resource utilizatin rate is 6/8 for total 4 unit times, and resource utilization rate, 6/8, is devided by 4, total time to test. It means the utilization rate for one resource per unit time. Like this way, we can get values for all possible resources. Finally, we select the maximal value. It becomes the resource threshold value to test.

Tab.1 simulation example

The number of Resource	Time to test	Utilization rate for one resource per unit time
1	6 unit times	0.17 (= 6 / (1 x 6^2))
2	4 unit times	0.19 (= 6 / (2 x 4^2))
3	4 unit times	0.13 (= 6 / (3 x 4^2))

$$\left(\frac{N_M}{N_R \times T_{tot}} \right) \times \frac{1}{T_{tot}} \quad (1)$$

We can simulate by using the equation above and the result is table 1. Given the software design structure, our simulator, which developed by C language calculate for all available resources. In the equation (1), NM is the number of modules, NR is the number of resources, and Ttot is total time to test in our equation. Table 1 shows the maximal utilization rate per unit time is about 19%. It means that 2 testing teams are sufficient to test. If we have resource team under 2, resource is insufficient to test, and it takes longer time. On the other hand, if we have resource team over 2, resource is overinvestment, and we have lots of idle resources per unit time. So, in this example, 2 is the testing parallelization index and it is the threshold value for efficient parallel test.

If we have a software design structure, we can figure out the parallelization index(threshold value) to test it by using our equation. But, we are not sure that our scheme is applied to various software design structure. Just like our example architecture, it is true iff the graph is started and ended by each one node.

4. Summaries

So far, in the sector of software parallel processing, most studies have focused on the method to efficiently allocate works to the resources in the environment that plural numbered resources exist. However, in the sector of software test, there have not been sufficient studies on parallel processing as well as on the theme that how many resources will be efficient for a specific test. In reality, tests are implemented either even without any specific planning or with a rough plan. If we have a good plan to test, we will achieve the efficient resource management. This study suggests the concept of parallelization index for software test which can exercise helpful influence on planning parallel test execution. Out index for software testing is the value which maximizes the average usage rate of a resource per unit time, which can be used for making out a plan for the most efficient test. This index is meaningful in that it can be utilized as the threshold value to express the efficiency of a software test in execution.

In this paper, we consider the software structure which has one start node and one end node. Is our method true for the software structure which has one start node and many end nodes? Now, we plan to advance our studies for the utilization of the parallelization index in different tests according to the variety of software architectures, on the assumption that the parallelization index may be applied differently according to various software architectures forwardly.

5. References

[1] RAFA E. Al-Qutaish. Measuring the Software Product Quality during the Software Development Life-Cycle: An International Organization for Standardization Standards Perspective. Journal of Computer Sceience 5 [J], 2009, PP:392-297

[2] TOSHIAKI Kurokawa; MASATO Shinagawa. Technical Trends and Challenges of Software Testing. Science & Technology Trends, 2008.10, PP:34-45

[3] HUANG Chin-yu; LO Jung-hua; KUO Sy-yen; LYU Michael R. Software Reliability Modeling and Cost Estimation Incorporating Testing-Effort and Efficiency. Proceedings of the 10th IEEE International Symposium on Software Reliability Engineering, 1999, PP:62-72

[4] YAN Jia-her; MAZUMDAR Mainak. A Comparison of Several Component-Testing Plans For A Parallel System. IEEE Transactions on Reliability [J], 1987, PP:419-424

[5] TOSHINORI Hosokawa; HIROSHI Date; MASAHIDE Miyazaki; MICHIAKI Muraoka; HIDEO Fujiwara. A Method of Test Plan Grouping to Shorten Test Length for RTL Data Paths under a Test Controller Area Constraint. Asian Test Symposium, 2003, PP:130-135

[6] JIN Li; LIJUAN Shen; LEI Zhao. Success ratio sequential test plan using development test data, ICRMS 2009, 2009, PP:370-373