Use Case Set Optimization in Android Mobile Platform Considering Energy Consumption

Peixing Yang[1], Di Zhou[1]

1.College of Computer Science and Technology , Nanjing University of Aeronautics and Astronautics

Key Words: Use Case Set, Optimization, Energy Consumption, Android

Abstract. Along with the increasingly mature computer technology, various computer systems with complex structures and strong functions have been widely applied to such key safety fields as aerospace, communication & transportation, medical treatment & public health and nuclear power energy. In case of system failure, human life and property will suffer from heavy losses or the environment will be seriously destroyed. Therefore, people pay more and more attention to software quality. Although many technologies and methods have been proposed for the problem about how to ensure software development quality, the software quality is still mainly ensured through software testing. Therein, software testing is an important but expensive process in the whole software development process, so it is significant to reduce software testing cost and improve testing efficiency.

Introduction

Generally speaking, the test requirements are proposed for achieving test objective and namely refer to the test items needed by an project. Specifically, all activities in a test process can be traced back to the test requirements. For example, when making a test plan, it is necessary to clearly define the following basic elements: firstly, clearly propose the test requirements, namely the contents of the test objective; then, decide how to test and what test method shall be adopted; then, evaluate how long does the test take and how many test personnel are needed, namely test scheduling; finally, clearly define the test environment. Additionally, test skills, test tools, corresponding professional background knowledge, possible test risks, etc. are also included in the test plan. In a word, all the above contents are the basic elements of the test plan. The important basis for making the test plan is namely the test requirements and all contents in the test plan can be traced back to the test requirements, so the test requirements are regarded as the basis and the key point of the test plan. Similarly, test scheme, use case and relevant contents shall be based on the test requirements. Since the test requirements are mapped from software requirements, so the detail level thereof is closely related to that of the software requirements. Under the precondition of ensuring the consistency between the test requirements and the software requirements during the compilation process, it is necessary to pay high attention to accurate and detailed expression in order to avoid omitting and misunderstanding relevant test requirements. Meanwhile, the test cases compiled thereby shall cover all test requirements, wherein the test requirements are converted from the software requirements and the execution results of all test cases will be finally traced back to the software requirements, so the test cases shall be compiled mainly on the basis of the software requirements. Additionally, it is necessary to abide by relevant compilation rules, standards, etc.

Test Case Set

The test requirements are sourced from the software requirements, and one (or several) test requirement(s) is (are) corresponding to one software requirement. If one software requirement can be converted into one or several test requirements, then the test requirements can be regarded to have covered all software requirements, namely: the coverage rate of the test requirements is 100%. However, the coverage degree of the test requirements cannot be regarded as 100%, because the

common software requirements only clearly indicate the significant functions and characteristics rather than directly present the latent functions and characteristics (which are not clearly defined but shall be provided). Actually, such requirements shall be also included in the test requirements, so it is necessary to analyze the significant and latent requirements when analyzing the test requirements, or supplement or optimize the test requirements according to the defects found in the actual test process and then update the test cases, thus to improve the coverage degree of the test requirements.

Test Case Energy Consumption Modeling

In this article, symbol definition and function relationship are adopted to formally describe the energy consumption of Android memory system by the sequence from whole to parts, thus to gradually establish the energy consumption model of the memory system.

The energy consumption of the whole system is composed of the memory energy consumptions of the system and can be expressed as follows:

$$E_{SYS} = \sum E_{DA} + \sum E_S + \sum E_{SWB} + \sum E_{IBSW} \quad \text{Formula 1}$$

Formula 1 is further decomposed to expand the constituent parts thereof as follows:

$$E_{DA} = E_{DA-base} + E_{DA-workload} \quad \text{Formula 2}$$

$$E_S = E_{S-base} + E_{S-workload} \quad \text{Formula 3}$$

$$E_{SWB} = E_{SWB-base} + E_{SWB-workload} \quad \text{Formula 4}$$

The basic energy consumption of the memory is composed of the energy consumptions of the extensible modules thereof:

$$E_{DA-Base} = \sum E_{Disk} + \sum E_{DAE} + \sum E_{CON} + \sum E_{Head} \quad \text{Formula 5}$$

Android basic energy consumption can be expressed as follows through the energy consumptions under idle state:

$$E_{S-Base} = E_{S-Idle} \quad \text{Formula 6}$$

$$E_{SWB-Base} = E_{SWB-Idle} \quad \text{Formula 7}$$

The energy consumption of the equipment is changed along with the load and can be expressed as follows through the function relationship:

$$E_{DA-Workload} = DPF(IOPS, PPIO) \quad \text{Formula 8}$$

$$E_{S-Workload} = DPF(CWS, PETC) \quad \text{Formula 9}$$

$$E_{SWB-Workload} = DPF(N_{Port}, PEPT) + DPF(N_{D-Port}, PEDT) \quad \text{Formula 10}$$

Meanwhile, the equipment connection relationship can decide the quantity of some extensible components and ports and can be expressed as follows through function relationship:

$$N_{DAE} = CNF(N_{Disk}, C_{DAE}) \quad \text{Formula 11}$$

$$N_{SWB} = CNF(N_{Port}, C_{SWB} - 1) \quad \text{Formula 12}$$

$$N_{Port} = N_S + N_{Head} \quad \text{Formula 13}$$

The energy consumption of Android system can be relatively integrally described through Formula 1 ~ Formula 13, and the above formulae can be integrated in order to obtain the energy consumption model of the whole system:

$$
\begin{aligned}
E_{SYS} &= \sum E_{DA} + \sum E_{S} + \sum E_{SWB} + \sum E_{IBSW} \\
&= \sum \left(E_{DA-base} + E_{DA-workload} \right) + \sum \left(E_{S-base} + E_{S-workload} \right) \\
&\quad + \sum \left(E_{SWB-base} + E_{SWB-workload} \right) + \sum E_{IBSW} \\
&= N_{Disk} \times E_{Disk} + CNF(N_{Disk}, C_{DAE}) \times E_{DAE} + N_{CON} \times E_{CON} \\
&\quad + N_{Head} \times E_{Head} + \sum_{1 \to N_{Head}}^{i} DPF(IOPS_i, PPIO_i) \\
&\quad + N_{S} \times E_{S-Idle} + \sum_{1 \to N_{S}}^{j} DPF(\frac{CWS_j}{10\%}, PETC_j) \\
&\quad + CNF((N_S + N_{Head}), C_{SWB} - 1) \times E_{SWB-Idle} + DPF((N_S + N_{Head}), PEPT) \\
&\quad + DPF(N_{D-Port}, PEDT) + \sum E_{IBSW}
\end{aligned}
$$

Formula14

Coverage Rate Analysis of Test Case

Good test cases shall cover all test requirements. In the example of system function, the test cases include function points and business processes. Specifically, the designed test cases shall cover the function points in all requirements, and besides the normal test cases, the test cases used for exceptions shall be also designed, and the test cases used for exceptions shall occupy 20% ~ 30% of all test cases. Similarly, the test cases for the business processes shall also include normal processes and exceptional processes.

Test case optimization and selection can effectively solve the existing coverage problem of the test cases, but we still find the incomplete coverage problem in the practical test process: some functions of the new version cannot be covered by the existing test cases. Like this, the testing personnel need to develop new test cases to test the new functions, but this is not practical for these testing personnel, because it is difficult for them to find out the untested parts in the codes according to their test experiences. Therefore, it is important for us to discuss how to better obtain the test coverage rate during the test process and the test coverage rate of the whole software after the test, thus to be favorable for the testing personnel to design new test cases for the untested parts. In this article, we particularly focus on the updated but uncovered parts during the version updating process.

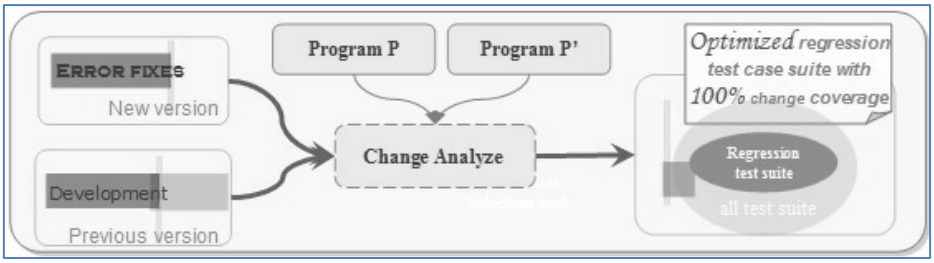

Figure 1 Test Case Optimization and Selection Problem

During the version updating process, the influenced test cases include Test Case 1, Test Case 2 and Test Case 3 which cover 12 nodes. Specifically, 4 updating points are included during the version updating process, wherein 3 updating points are covered and 1 updating point is not covered, so the updating coverage capacity of the existing test case set is 75%. In this way, we can know that the test cases designed for previous version are not enough and some program modules have not yet been covered by any existing test case. Code updating most probably causes program bugs and even causes system crash, so it is necessary to add new test cases to ensure the updating coverage and reduce program running risk.We detail the node coverage rate of the software to function granularity and obtain the correspondingfunction through analyzing Binary codes of the software deployment. Since the source codes are not needed, the software coverage rate can be easily tested

and analyzed. In other words, the test case coverage can be tested and analyzed at a relatively low costs, thus to not only save time and labor, but also ensure high test quality.

Conclusion

The execution results of the test cases can be analyzed and inspected from the aspects of coverage rate, execution rate, passing rate, etc., wherein the coverage rate of the test cases refers to the specific value of the functions covered by the test cases and the functions specified in the test requirements; the execution rate of the test cases refers to the specific value of the executed test cases and the total test cases; the passing rate of the test cases refers to the specific value of the successfully executed test cases and the total test cases. Practically, the coverage rate of the test cases shall reach 100%, in other words, the test cases must cover all test requirements; or else, the test case design is incomprehensive and cannot ensure test quality, and the test cases shall be correspondingly supplemented or redesigned. The execution rate of the test cases is the factor used for measuring the testing efficiency, and generally speaking, the execution rate of the test cases shall reach 100% after the test is finished, but the test may be interrupted due to some special reasons and the test cases may not be completely executed; in such case, we need to analyze according to actual conditions. The passing rate of the test cases is the factor used for measuring the design quality of the test cases and the quality of the tested software, and for the unsuccessfully executed test cases, we need to analyze whether the use case design error or the tested software error causes such execution failure.

Reference

[1] Su T, Lv Z, Gao S, et al. 3d seabed: 3d modeling and visualization platform for the seabed[C]. Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on. IEEE, 2014: 1-6.

[2] Lu Z, Lal Khan M S, Ur Réhman S. Hand and foot gesture interaction for handheld devices[C]. Proceedings of the 21st ACM international conference on Multimedia. ACM, 2013: 621-624.

[3] Zhang X, Xu Z, Henriquez C, et al. Spike-based indirect training of a spiking neural network-controlled virtual insect[C]. Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on. IEEE, 2013: 6798-6805.

[4] Dang S, Ju J, Matthews D, et al. Efficient solar power heating system based on lenticular condensation[C]. Information Science, Electronics and Electrical Engineering (ISEEE), 2014 International Conference on. IEEE, 2014, 2: 736-739.