A Software Acceptance Testing Technique Based on Knowledge Accumulation

Yi Yu, Fangmei Wu
Dept. of Telecommunication Engineering
Shanghai Tiedao University, P. R. China
E-mail:yuyi@shtdu.edu.cn

Abstract

System acceptance testing in general relies on the specification of system requirements, but for complex systems, especially for complex safety systems, the issue whether system requirements specified by users are complete should be considered. This paper presents a software acceptance testing technique based on knowledge accumulation, which can help to expose the software faults caused by the lack of knowledge. A software test tool using the technique for the railway signaling computer interlocking systems and some tested results are also introduced in this paper.

Keywords: software, acceptance testing, knowledge accumulation, railway signaling, interlocking

1. Introduction

As a commonly recognized procedure in software engineering, the acceptance testing of a software system is conducted after its integrated testing has been completed. The object of an acceptance testing should be an integrated software system, and there should be generally no error on its interface. The task of the acceptance testing is to verify the operational availability of a software system being tested or to ensure that all functions of the software system are in conformity with what are expected by users. Ideally the software system functions required by users should be specified precisely in a user requirements document at the user requirements analysis phase. However in practice it is very difficult to describe the user requirements accurately and completely, especially for the development of a new software system or a complex safety system. It is true that the user requirements document can be revised and substantiated continuously in the whole life-cycle of a software development. But on account of the lack of knowledge or the one-sided understanding, for example, users do not understand the characteristics of a software product and a software engineer does not have some specific knowledge, mistakes in a user requirements document or misunderstanding of the document can not be completely avoided. To identify the software faults due to this kind of the lack of knowledge and the inaccurate knowledge representation, it is necessary to pay attention to the knowledge acquirement and accumulation in the acceptance testing phase.

2. The Software Acceptance Testing Model

Software acceptance testing adopts essentially the black-box testing technique and is user-oriented generally. The software acceptance testing model can be simply shown by the following function.

$$Y = f(X)$$

Here, $f(.)$ is represents a software system to be tested and can be regarded as a black-box; X is a test case input set for acceptance testing; And Y is the output set.

The task of software test is to expose faults in a software system under test, but the technique based on black-box can only find the failures of a software system under test. The following issues must be addressed when a software acceptance testing is carried out using black-box testing technique:

- Determining relation between X and Y;
- Selecting test case generation strategy;
- Judging test results;
- Acquiring and accumulating knowledge.

2.1 Determining relation between X and Y

From the external angle, a software system can be classified as either an input determined system, or an output determined system, or an input-output all determined system or an undetermined system.

Assuming that X is an input set of the software system under test, Y is an output set of the software system, and $R = \{\{x, y\} | x \in X, y \in Y\}$ is the 2-element relation between the X and the Y, a software system is termed as an input-output all determined system

if R meets all following conditions (1), (2), (3) and (4), as an input determined system if R meets only the conditions (1), (2) and (4), as an output determined system if R meets only the conditions (1), (3) and (4), and as an undetermined system if not.

Condition (1): R does not contain the element $\{x, y_i\}$ and the element $\{x, y_j\}$ at the same time, where $y_i \neq y_j$ and $y_i, y_j \in Y$;

Condition (2): $\{x\} = X$;

Condition (3): $\{y\} = Y$;

Condition (4): R is sole.

Condition (1) rules out the undetermined outputs of a system under test against an input x. Condition (2) means that the system has an output or several outputs against each input x of the input set X. Condition (3) means that each output y of the output set Y of the system is activated by an input x. Condition (4) means that the relation R between X and Y of the system is sole. The cause-consequence diagram [1] may be a good method to show the relation R.

A software system to be tested must be a determined system (either an input determined system, or an output determined system or an input-output determined system). In the railway signaling field, the railway station signaling control system is a safety multi-dimensional N-step sequence system [2]. Its interlocking software should be a determined software system.

2.2 Selecting test case generation strategy

There are many strategies and methods to generate test case at present, for example, the object-oriented software test strategy[3], the knowledge-based test planning[4], the automated test case generation method[5], the generating functional test cases in-the-large[6], the strategy based on the safety relation between the input and the output of system[2], and so on. Unfortunately, no single testing method can guarantee to give accurate results in every circumstance. For a complex safety system, such as the railway signaling computer interlocking system, more than one methods or strategies of generating test case should be considered at the same time. The TAS (Test and Assessment System) for the safety critical softwares of railway signaling computer interlocking systems introduced later uses simultaneously the knowledge accumulation strategy and the strategy based on the safety relation between the input and the output of system.

2.3 Judging test results

In the process of acceptance testing, the basis for passing a judgment on test results is generally the specifications. But if the software under test is a complex system or its specifications are informal, the judgment is not to be based on the specifications alone, and some supplementary criterions should be considered. In the TAS, the safety and efficiency criterion is the supplementary criterion.

2.4 Acquiring and accumulating knowledge

Acquiring and accumulating knowledge in the acceptance testing phase is to make up for the incompleteness of user requirements document or specifications, to increase the test efficiency and to reduce the test cost. Increasing the test efficiency means to find more faults by using lower test cases.

3. A Software Acceptance Testing Technique Based on Knowledge Accumulation

A software acceptance testing technique based on knowledge accumulation is shown in Fig. 1.

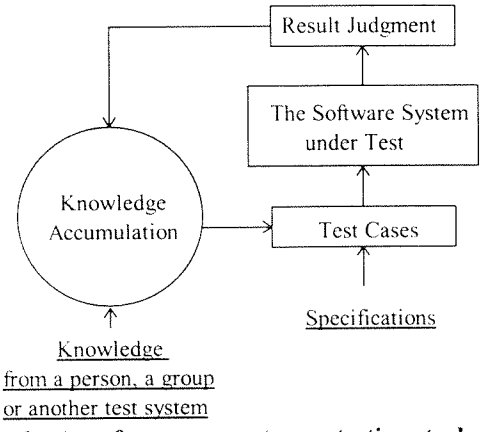

Figure 1. A software acceptance testing technique based on knowledge accumulation

The key to this technique is to establish methods of acquiring and accumulating knowledge. This paper introduces two knowledge accumulation methods, a vertical knowledge accumulation method and a horizontal knowledge accumulation method. The first method means that knowledge is accumulated by a person, a group or a knowledge system. The longer time is, the more knowledge is accumulated. But the second method means that knowledge is accumulated by exchanging between different persons, different groups or different systems. It means that the more different persons, different groups or different systems there are, the more knowledge is accumulated.

The TAS for the safety critical softwares of railway signaling computer interlocking systems has used the technique based on knowledge accumulation, which includes the vertical knowledge accumulation method and the horizontal knowledge accumulation method.

4. A Software Test Tool: the TAS for the Safety Critical Softwares of Railway Signaling Computer Interlocking Systems

A railway station signaling control system is a safety critical multi-dimensional N-step sequence system. At the present moment there is not any appropriate knowledge representation available to describe its sequence logic accurately and completely. Therefore it is too difficult to describe its user requirements accurately and completely when the software is used to realize the sequence logic functions, the reliability and the safety of the system. To make up for the incompleteness of the user requirements document, a special tool — the TAS for the safety critical softwares of railway signaling computer interlocking systems has been developed by the Testing Center of Railway Computer Interlocking System of Ministry of Railways in P. R. China. The TAS has used the software acceptance testing technique based on the knowledge accumulation. It consists of some basic software T&APs (Test and Assessment Platforms). Each basic software T&AP is composed mainly of the following nine modules. They are a basic operation set module, a basic case model module, a case model list module, a case generating module, an operation selection module, a control software module, a result analysis module, a recording operation module and an abstracting case model module. A basic software T&AP is shown in Fig. 2.

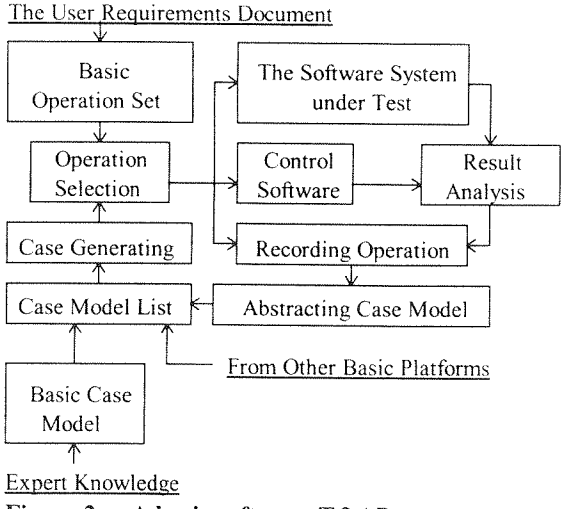

Figure 2. A basic software T&AP

The basic operation set module essentially consists of the operation sets generated according to the user requirements document of a software system being tested. The basic case model module consists of the case models abstracted from the expert knowledge. The case model list module is practically a test case model list with the statistics ratio. It derives case models not only from the basic case model module, but also from an abstracting case model module that can be the module of the basic platform itself or the module belonging to other basic platforms. The structure of getting case models from other basic platforms is the horizontal knowledge accumulation method. Others belong to the vertical knowledge accumulation method. The case model list module is the link of the TAS for the safety critical softwares of railway signaling computer interlocking systems. It links all basic platform of the TAS. The function of the case generating module is to change the case models into test cases which are suitable for the software system under test. The operation selection module selects operations from the basic operation set module or converts the test cases of the case generating module into operations according to the test task and the test strategy. The selected operations are sent to the software system under test and to a control software module at the same time. Here the control software module includes a subjunctive interlocking software system. It is an expert system in fact. Its output serves as a part of standard of distinguishing whether the output of tested software system is right or not. The result analysis module is designed to compare the output of the software system under test with the output of the subjunctive interlocking software according to the safety and efficiency criterion. If the comparing result is that the safety of software system being tested is lower than that of the control software or a critical efficiency failure of the software system under test emerges, a signal to record the input operation is sent to a recording operation module. After having received the signal, the recording operation module records immediately the operations corresponding to the output. The abstracting case model module can abstract a test case model from concrete operations. The abstracted test case model is added to the case model list.

5. Results and Conclusions

Up to now, seven interlocking software systems have been tested by means of the TAS for the safety critical softwares of railway signaling computer interlocking systems. The results are summarized in Table 1.

The test results are classified as the safety failures, the critical efficiency failures and the other failures

according to the safety and efficiency criterion. If an output of the interlocking software system under test is unsafe or risky, the output is counted as the safety failure. If an output seriously reduces the availability of the interlocking software system, it is counted as the critical efficiency failure. The interlocking software systems listed in Table 1 are developed by seven different developers. The tested software 1 in Table 1 is the first software system tested with the TAS. The tested software 2 and 3 are the second group of software systems tested with the TAS. And the other four tested softwares are the third group. The later an interlocking software system is tested, the larger the possibility of exposing its failures is. The tendency of average increasing expresses that the technique based on knowledge accumulation is effective in acceptance testing phase. Although the effect of the software acceptance testing technique based on knowledge accumulation is excellent, there are some problems that need to be further solved . For example, the problem of how the test case models are abstracted from the concrete operations automatically.

Table 1. Test results

GROUPS	THE TESTED INTERLOCKING SOFTWARE SYSTEM	PLATFORM	RESULTS			
			SAFETY FAILURES	CRITICAL EFFICIENCY FAILURES	OTHER FAILURES	TOTAL
1	Tested software 1	platform 1	3	1	6	10
	Average		3	1	6	10
2	Tested software 2	platform 1	7	0	17	24
	Tested software 3	platform 2	10	0	24	34
	Average		8.5	0	20.5	29
3	Tested software 4	platform 1	7	1	20	28
	Tested software 5	platform 2	11	3	31	45
	Tested software 6	platform 3	9	1	14	24
	Tested software 7	platform 4	12	7	22	41
	Average		9.75	3	21.75	34.5

References

[1] Anne Cartier and Marie-Christine Lartisien, Reliability, Availability, Maintainability and Safety Assessment, John Wiley & Sons Ltd., England, 1992

[2] Yi Yu and Fangmei Wu, "A Black-Based Safety Test Strategy for High-Speed Railway Interlocking Software", Journal of the China Railway Society, Vol.19 Suppl., November 1997, pp. 95-100

[3] Jin Lingzi, "Progress in Testing Object-Oriented Software", Computer Research & Development, Vol.35, No.1, Jan.1998, pp.6-13

[4] Dolly Samson, "Knowledge-Based Test Planning: Framework for a Knowledge-Based System to Prepare a System Test Plan from System Requirements", J. Systems Software, 1993, pp.115-124

[5] W.T.Tsai, D.Volovik, T.F.Keefe, "Automated Test Case Generation for Programs Specified by Relational Algebra Queries", IEEE Transactions on Software Engineering, Vol.16, Iss:3, March 1990, pp.316-339

[6] Sandro Morasca, Angelo Morzanti, Pierluigi SanPietro, "Generating Functional Test Cases in-the-large for Time-critical Systems from Logic-based Specifications", Software Eng. Notes, Vol.21, No.3, May 1996, pp.39-52