Determining The Black-Box Component's Failure

Ibrahim Hilal
abehelal@go.com.jo

Aman Jantan
manj70@tm.net.my

School of Computer Sciences
University Sains Malaysia
11800 Penang, Malaysia

Abstract

Many papers propose different approaches to evaluate the reliability of black-box components based on component's success. This paper determines or evaluates component's reliability based on its failure. Since even reliable components might fail, we argue that component's acquirer should evaluate the component's reliability based on its failure and not on its success. In this paper we propose a new approach towards determining the component's failure. Both the operational profile and the appropriate test cases are needed to support our approach.

Subject Indexing

Software Engineering, Reusable Software, reliability

1. Introduction

Software reliability is defined as "the probability of failure-free software operation in a specified environment for a specified period of time" [2]. Also it is defined as the ability of the software systems or components to perform their required functions under stated conditions for a specific period of time [1].

The software component is a piece of software that is used to build a more complex software system. Integrating Software components in a software system can reduce the time and the development's cost of the software systems. The reliability of a Software component can determine the reliability of the software system. Unlike hardware (where hardware can fail by time due to usage) , software components do not fail due to wear out or age. Hardware failure is time dependent, while software component is not. When the source code of the component is supplied or available then it is called White-box component. Black-box component is called so because the source code is neither supplied nor available; only the object code is. The internal implementation of the black-box component is usually hidden from the component's acquirer. The component's acquirer needs only to know the functionality, and how it interfaces with its environment [4]. Lyu's handbook contains a survey of many software reliability models This work presents our approach to determine the reliability of the black-box components.

Fig. 1 shows the steps that are required to determine the failure of the black box components.

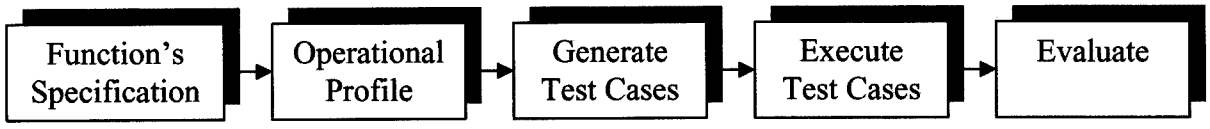

Fig. 1: The processes to determine the component's failure

This paper is organized as follows: section 2 contains related work, section 3 contains two steps that are needed prior to evaluate the component, section 4 contains our approach, and section 5 contains future work.

2. Related Work

The models that are available to compute the reliability of components use complex mathematical formulas. They required the user or who performs the testing strong background knowledge in mathematics and statistics. In addition, it requires time to generate the necessary parameters. Lyu's handbook contains a survey of many software reliability models

3. Operational Profile and Generating test cases

Woit defines the Operational profile as follows:" is a description of the distribution of inputs that is expected to occur in actual software operation" [9]. Operational profile is the set of operations and their probability of the occurrence [8].

Operational profile is an essential part for generating test cases that we need to determine the failure probability of the component. After we identify all the component's functions (or operation) and determine their specifications we can build their operational profile. An operational profile has to be made for each function of the component before we generate the test cases for this function.

We are concern with the test cases that make the component fail and not the test cases that make the component pass. So our testing goal would be, to make the component fail by running as many tests as possible.

Testing requires to place the software in an environment similar to its real usage environment [7]. We should try as much as possible to make the test cases that we generate closer to the actual use of the functions that the component performs. The closer the test, the higher the accuracy of the evaluation is. Test cases should cover all possible functions invocation orderings. After we generate those test cases, we execute those tests, and then we evaluate the result of those tests.

Generating different test cases will be based on the operational profile that we make or create.

Many operational profiles should be made for each user in a multi-user system. Also we should make many profiles for every user that uses a multi- mode system.

How to make an operational profile and how to generate test cases are beyond the scope of this literature.

4. Our approach: Determining the Component's Failure

Software reliability is defined as "the probability of failure-free software operation in a specified environment for a specified period of time" [2]. "software failure occurs when the behavior of the software departs from its specification...during its execution" [3].

Many literatures determine the software system reliability by determining the reliability of its constituent pieces of software or components. Likewise we can determine the software system failure probability by determining its constituent components failure probability; consequently we can determine the component failure if we identify its functions (or operations) first, and then determine the failure probability of each one of those functions that the component can perform.

The component's supplier should indicate to the component's acquirer all of the functions (or operations) that the component can perform with no exclusion. (i.e. calling another component, perform database access, establish internet connection, transfer data or file, ...etc.).Once we identify those functions, we can run several tests for each one of those functions based on the operational profile we imply from each function's specification or definition. By doing so, the failure probability of a particular function is equal to the number of the recorded tests that executed and failed divided by the total number of executed tests whether failed or succeeded (terminated successfully or not) for that particular function.

Two values are obtained as a result of executing test cases. The first one is the component's function failure probability (as shown above) and the second one is the relative occurrence of this function within the other component's functions. Once we obtain both the failure probability and the relative occurrence for every function (or operation) that the component performs, then we can determine the component over all failure probability by using the Total Probability theorem [10]:-

$$P(F) = \sum_{i=1}^{n} P(\text{Function}_i \cap F)$$

$$P(F) = P(\text{Function1} \cap F) + P(\text{Function2} \cap F) + \ldots\ldots P(\text{Function}_n \cap F)$$

$$P(F) = P(\text{Function1}) * P(F \mid \text{Function1}) + \ldots. + P(\text{Function}_n) * P(F \mid \text{Function}_n)$$

Example:-

A certain component performs 3 functions: Insert, Update, and Delete from a certain date base file system. The probability of failure and the relative occurrences of each of those functions is shown in table 1.

Function name	Failure Probability	Relative Occurrence
Insert	0.1	0.5
Update	0.15	0.3
Delete	0.20	0.20

Table 1

To compute the failure of this component will base on the above equation:-

$P(F) = (0.1 * 0.5) + (0.15 * 0.3) + (0.20 * 0.2)$

$P(f) = 0.135$ is the failure probability

How to create an operational profile, how to generate test cases, how to execute those test cases, and how to evaluate the test's results are beyond the scope of this literature.

5. Conclusion and Future Work

In this paper we have presented an approach to determine the reliability of the black-box components. Since even reliable software components can fail, determining the reliability of the components based on its failure is more important than determining the reliability of the components based on its success. Also we have showed the importance of the component's operational profile and generating test cases to support this approach. Our work enables the component acquirer to determine the reliability of the black-box. using simple and quick method. Our future work would be toward how to generate more accurate test cases to obtain more accurate evaluation; also how to develop an algorithm that is used to obtain an operational profile of the component.

References

[1] IEEE, IEEE Standard 610.12-1990 *IEEE Standard glossary of Software Engineering Terminology*, February 1991.

[2] Lyu, M. R., Handbook of Software Reliability Engineering, *IEEE Computer Society Press*, 1996.

[3] Pai, G. J., A Survey of Software Reliability Models, A Project Report. CS 651: *Dependable Computing*, 2002.

[4] Ravichandran, T., Rothenberger, M., Software Reuse Strategies and Component Markets, *Communication of ACM46, 8*, 2003.

[5] Alzamil, Z., Application of the Operational Profile in Software Performance Analysis, *wosp 04*, January 14-16 Redwood Shores, CA, 2004.

[6] Hamlet, D., Mason, D., Woit, D., Theory of Software Reliability based on components *In proceeding of23rd International Conference on Software Engineering*, pp.361-370, Toronto, Canada 2001.

[7] Musa, John D. More Reliable Software Faster and Cheaper, *Authorhouse*, September 2004.

[8] Musa, John D. Software reliability Engineering. *McGraw-Hill*, New York 1999.

[9] Woit, Denise M., A Framework For Reliability Estimation, *Proc. 5th IEEE International Symposium on Software Reliability Engineering (ISSRE'94)*, November 6-9, 1994. pp. 18-24.

[10] Montgomery, D., Runger, G., Applied Statistics and Probability for Engineers, *John Wiley*, September 2002.