ISSN:2249-5789
Harsh Chincholi et al, International Journal of Computer Science & Communication Networks,Vol 2(3), 343-346

Developing Framework to do the Reliability Testing on webMethods Product Suite

Harsh Chincholi [a], Sandhya S [a], Rameshchandra V[b]

[a] *Computer Science & Engineering, R.V. College of Engineering, Bangalore*
[b] *Software AG Bangalore Technologies Private Limited.*

Abstract

Over the past few years, use of integration server (IS) in critical situations has been increasing and need for high dependable integration server is necessary. Now, reliability as the primary characteristic of dependability is a key factor to evaluate integration server .So a Reliability framework is developed which is an automatic testing tool for integration server. Reliability framework provides facility to identify method parameter types, types returned by methods, subtyping relations and visibility constraints and fragments of code (test cases) are constructed which will be used to check the public methods which are under test .Programs which are under test are crashed when bugs are detected by framework and throws an exception.
Keywords: Integration server, reliability, test cases.

1. Introduction

webMethods product suite is an integrated set of design tools, run-time servers, registry/repositories, and Internet browser-based user interfaces that enables to develop and run integration solutions, create and manage a business-to-business integration network, develop and run composite applications, design and run business processes, develop and govern a service-oriented architecture, monitor and improve the performance and efficiency of business activity. Integration Server which is part of the webMethods Product Suite is one which receives requests from client applications and authenticates and authorizes the requesting users, invokes the appropriate services and passes them input data from the requesting clients, receives output data from the services and returns it to the clients and supports a wide range of established and emerging standards so that users can interact with virtually any business partner that is connected to the Internet.

Since integration server used in critical situations need for reliable integration server has been increasing. To demonstrate reliability appropriate test cases has to be generated and to evaluate these test cases [1]. So to build reliable software application understanding reliability of integration server at the architectural level is necessary. Recent approaches verified software reliability at architectural level and focus on system level reliability. These approaches assume that reliabilities of the components in a server are known [2]. Since these approaches assume the reliabilities it is not clear how the reliability of server or services is known. Reliability should have been randomly guessed and none of these is solution are satisfactory.

This paper provides remedy to the shortcomings of previous approaches by proposing a framework for predicting reliability of software components at architectural level. This is intended to be complementary to the existing on system-level reliability prediction. The purpose of the application is to implement a platform to do reliability testing on integration server across organization. The application provides a framework for all the teams working on various web methods products suite to generate test cases of integration server code, run the tests cases, which generates HTML or XML test results reports, which contains class name, number of tests checked, execution time, number of failures and customizes the reports generated.

2. Need for a New Reliability Framework for IS

The purpose of reliability framework is to discover problems with in the design and to the meet system reliability requirements as early as possible [3].

Testing all system requirements is not always good since some systems are expensive to test, may generate large number of test case which may be not useful all the time. In such situations different testing approaches can be adapted. In reliability testing statistical confidence is also extremely important. It can be increased by increasing the number of items tested. One of the important aspects of reliability is how to explain failure. In some circumstances it is not known whether failure is due to system. So scoring process can be used to discover the failure [4].

3. Current State Challenges

Integration server is web bases application it will influence our relationship with customers and it is important to the company. If the failure of the integration server would result in business loss and corporate world is asking 365 days of reliability.

An enterprise application is a collection of hardware, operating system services, software components, and (usually) human processes that are intended to cooperate to provide the expected business services. As reliability depends on the different components of the system, failure of one component affect the remaining components also. Failure of application may happen because if analysis of the application is not made properly, if code written is not up to the mark or the input given is incorrect.

Software reliability is to provide services, provide good results and should have ability to discover errors. Currently functional testing frameworks are present which doesn't check reliability of integration server like lack of resilience, environmental failure at architectural level [5]. In order to overcome the functional testing frameworks disadvantages this framework is developed which analyzes methods, produces test cases and generate test report.

4. Methodology

The Reliability Framework can be modularized in to following modules as shown in figure 1: Data type definitions, Heuristics, Random test capability, Test case generator, Tester, Result monitor.

Data Type definitions:

Pre-defined data type information will be provided by data type definitions like information about how to conduct random tests, providing base test cases and templates for use in assertion generation [6]. Thus, information from the interface declaration is provided in terms of defined data types.

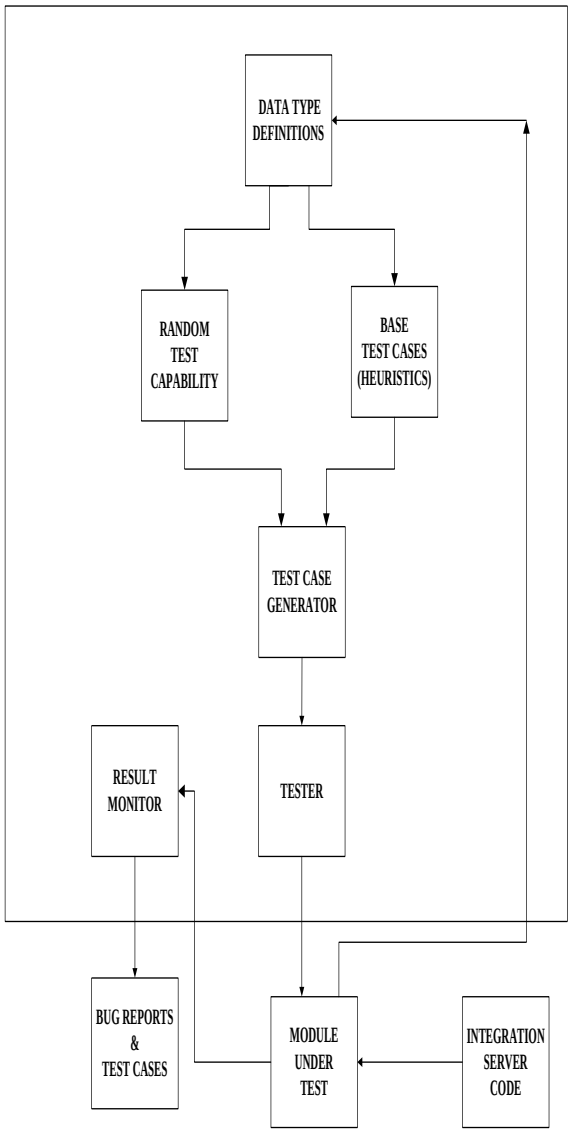

Figure 1: Modularized Reliability Framework

In data type definitions, class loader reads the class bytecode from the file system, analyzes it using Java reflection. And also java reflection is used identify visibility constraints, method parameter

types, types returned by methods and subtyping relations for each method declared in a given class.

Heuristics:

Heuristics is second module which consists of base test case information which will check a definite data type. If any discontinuities present in the response space then it is found in effective manner.

Random test capability:

Random test capability does the parameter-graphing, test case selection and execution. The parameter-graph is an abstract representation of the parameter-space of the program's methods. By traversing the graph for a method under test different parameter combinations are created. This parameter-graph representation is useful since it allows to easily bound the depth of method chaining.

A different combination of its parameter-types' value-range will be present in method which is provided by test cases. Result monitor can produce two or more test cases producing the same value. This may occur if a returning method returns the same value for different parameter combinations, or two returning methods have overlapping return-value ranges. If more parameter combinations are present in the method then that test case is tested frequently and more parameter combinations produce a bigger variety of parameter state. And for complicated method large numbers of test cases are generated. In the parameter-graph void-returning methods are currently excluded. Since parameter-space is represented implicitly-via the parameter-graph, result monitor efficiently computes the number of test cases for a given search depth.

Each generated test case consists of one or more methods or constructor invocations present in a single block. And the Exception thrown by these test cases is caught and passed to the runtime. The runtime either considers the exception a bug of the code under test or considers it an ill-formed test case because it may be failed to provide legal inputs and hence suppresses the exception and heuristics take into account the type of the exception and the method call history that led to the exception.

Test Case Generator:

Base Test cases and number of test cases generated in random test capability are collected by test case generator and reported if bugs are found.

Tester:

The tester is a one that executes the module under test with a particular test case. Required data structures are initialized by the tester, executes the module under test, and then the remaining data structures are erased after the test.

Result Monitor:

The result monitor will be searching the status of the module under test and it will be looking for crashes. Results with other than graceful behavior can be reported as bugs.

5 .Conclusion

Integration server is used in critical situations, need for high dependable integration server has been increasing. Reliability as the primary characteristic of dependability is a key factor to evaluate integration server. Some of the existing tools focus on system-level reliability and think that the reliabilities of components are known and this assumption is unreasonable, making reliability prediction an important missing ingredient in the current tools. Since many uncertainties associated with components under development prediction of component reliability is a challenging problem.

The work carried during the course of this project focuses on developing a framework which checks Reliability of integration server which provides the platform to transitively analyzes methods of integration server, determines the size of each tested method's parameter space, and selects test cases for testing, produces test files and generate report for test files which contains number of failures, which helps to fix failures.

6. References

[1] Roberto Pietrantuono, Stefano Russo and Kishor S. Trivedi "Software Reliability and Testing Time Allocation: An Architecture-Based Approach", IEEE transactions on software engineering, vol. 36, no. 3, may/june 2010, pp.323-337.

[2]Apiletti Leslie Cheung, Roshanak Roshande, Nenad Medvidovic, Leana Golubchik, "Early Prediction of Software Component Reliability", *ICSE' 08*, May 10 - 18, 2008, Leipzig, Germany, pp. 111-120.

[3] Yumei Wu, Yongqi Zhang and Minyan Lu "Software Reliability Accelerated Testing Method Based on Mixed Testing", Reliability and Maintainability Symposium (RAMS), IEEE 2010 Proceedings - Annual.
[4] Shuanqi Wang, Yumei Wu, Minyan Lu and Haifeng Li "Software Reliability Accelerated Testing Method Based on Test Coverage" Reliability and Maintainability Symposium (RAMS), IEEE 2011 Proceedings - Annual.
[5] Erqiang Feng, Chang Liu and Jun Zheng "Software Reliability Accelerated Testing Based on the Combined Testing Method", Reliability, Maintainability and Safety (ICRMS), IEEE 2011 9th International Conference, pp. 651 – 656.
[6]Philip Koopman"Ballista Design and Methodology", Institute for Complex Engineered Systems Carnegie Mellon University, Hamershlag Hall, Pittsburgh.