ORTS: A Tool for Optimized Regression Testing Selection

Sheng Huang, Jun Zhu, Yuan Ni
IBM China Research Lab, Pudong New District Shanghai 201203, China
{huangssh, zhujun, niyuan}@cn.ibm.com

Abstract

This paper presents ORTS, a tool for facilitating testers to generate optimized regression test suite for commercial Java applications. The aspects emphasized by the demonstration are: (1) how to help testers capture runtime traces of test execution; (2) how to identify change points during the build update; (3) how does ORTS improve the efficiency of the regression testing and reduce the cost by generating optimized regression test suite. The whole design strategy is lightweight, making the regression test selection process more automated and effective, and scalable to commercial regression testing scenarios with resource and time constraints.

Categories and Subject Descriptors D.2.5 [**SOFTWARE ENGINEERING**]: Testing and Debugging – Testing tools.

General Terms Verification.

1. Introduction

Regression testing is the process of validating the modified software to provide the confidence that the changed parts of the software behave as intended and the unchanged parts of the software have not been adversely affected by modifications [1]. A variety of strategies [2] have been proposed to select a subset of regression tests for execution to verify the modified program in academic research. However, most of these test selection approaches are source code based. For a large commercial system, it is hard to build the data flow or control flow required in these approaches [2]. In addition, only considering the changes in programming language level is not enough in large commercial systems. To the best of our knowledge, ORTS is the first tool that targets at regression testing selection for commercial Java applications. ORTS pursues the following three unique features which are not well addressed by existing approaches, but required in regression scenarios of commercial Java applications。

Feature 1: Scalable runtime profiling. We have done a survey on current commercial Java applications, and it shows that most of them are Web applications. Existing regression tools for java application such as Contest [4] lack of support to popular IT artifacts used in Java Web Applications such as JavaScript (JS), JSP. ORTS has the capability of capturing Java method invocations, JavaScript method invocations and JSP loading events. In this manner, ORTS guarantees that no IT artifact traversed during the execution is missed. In addition, the run time profiling incurs little overhead to test execution. Therefore, ORTS is scalable for runtime profiling of commercial Java applications.

Feature 2: Build Oriented change identification. Current testing services are likely to be distributed in different regions as the testing team is separated. The testing team always has no access to source codes. An alternative solution by decompiling binary files [3] is proposed, but it is illegal. In ORTS the change points are derived by analyzing the builds (EAR/WAR/JAR) of two versions. Besides the logic changes caused by modifications to Java/JS/JSP, the configuration level changes, such as frameworks for Inverse of Control, Data Access Object, etc, are also taken into consideration. This feature gives ORTS the ability of identifying complete change points of commercial Java applications.

Feature 3: Optimized regression test suite. Similar to the safe regression strategy in [2], only test cases traversing change points would be selected out in ORTS, consequently lots of unnecessary test cases would be avoided. Even in this way, the test suite size may still exceed the deliver pressure. To address this problem, ORTS prioritizes the regression test suite in terms of risk and guides the rerun schedule under the time pressure.

In the next section, we provide an overview of ORTS by describing main components and demonstrate how the above three features are addressed in the design and implementation of ORTS. An effectiveness evaluation is presented in Section 3. Section 4 describes the demonstration plan.

2. Overview of ORTS

As illustrated in Fig. 1, the ORTS tool consists of two main components: ORTS Agent and ORTS Server. The testers could login to ORTS Server from the Web to consume the testing service. ORTS Agent needs to be downloaded from ORTS Website and executed with target applications for test profiling.

How to enable scalable profiling? The Weaver of ORTS Agent is used to run static instrumentation to the targeted applications. The Weaver utilizes AspectJ [5] to instrument binary java code and executes a customized Ajax like instrumentation over

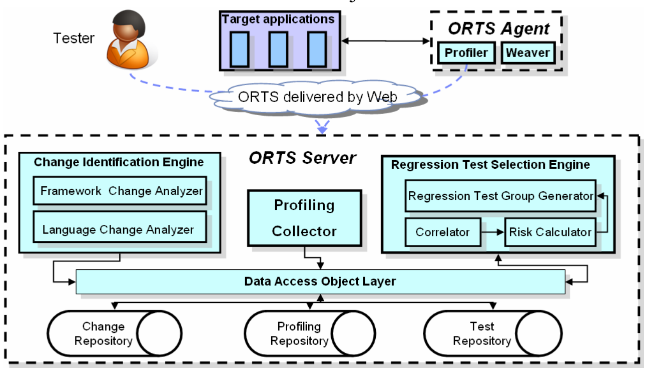

Figure 1. Tool overview

Copyright is held by the author/owner(s).
OOPSLA 2009, October 25-29, 2009, Orlando, FL, USA.
ACM 978-1-60558-768-4/09/10. 10.

JSP/JS to collect JSP loading events and JS method invocations. With the lightweight instrumentation approaches, the Profiler could profile applications without incurring obvious latency proved by pilot evidence. After test execution, the Profiling module would upload profiling data of tests to ORTS Server through the Web.

How to enable build oriented change identification? After extracting IT artifacts from the build, the Language Change Analyzer extracts the Java method changes by analyzing binary Java class directly and runs a syntax analysis to identify method level changes of JS and page level changes of JSP. Furthermore, the Framework Change Analyzer parses the entire configuration files in the build such as configurations of spring, struts, ibatis, etc., and maps these configuration changes to Java/JS/JSP changes which are linked to the test case profiling data.

How to generate optimized regression suite? Firstly, the Corrlator derives all the test cases traversing change points and passes them to Risk Calculator. Secondly, the Risk Calculator utilizes a normalized metrics of integrated factors to measure the risk of test cases. The integrated factors include the number of change point excised by a test case, the change types of test cases, the invocation counts of change points per test case, the count of IT artifacts traversed by a test case, bug history of test cases, and business value. Finally, the Regression Test Group Generator utilizes a greedy algorithm to generate a series of test groups. Each group covers all the change points. The groups are ranked by normalized risk metrics. In addition, as the time consuming of test execution in previous version gives accurate rerun time estimation, tester could select Top-K test groups to cover the most risky test cases within limited time.

3. Efficiency Evaluation

The tool has been piloted over an IBM internal project to select regression test suite during version iteration. Meanwhile, the testers identify changes and select regression test suite manually. As illustrated in Fig. 2, both approaches could identify 100% change points, while ORTS is more accurate in locating the change impacted files since updates causing no syntax differences considered in manual approach are ignored in ORTS. Compared with 113 test cases selected by testers, only 21 test cases are selected out by ORTS to rerun. Both approaches could reveal 5 defects. Based on the IBM best practice, one experienced tester could select out 20 test cases per day or execute 8 test cases per day, 5.5 person days (PD) are needed for test selection, and 14 PD are needed for test execution in manual approach. On the other hand, test selection process of ORTS only takes several minutes, and only 2.5 PD is required to run the test suite. Consequently, ORTS could significantly cut 85% cost by saving 17 PDs.

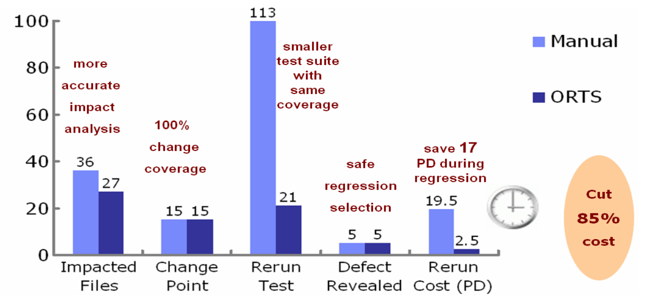

Figure 2. Tool pilot evaluation evidence

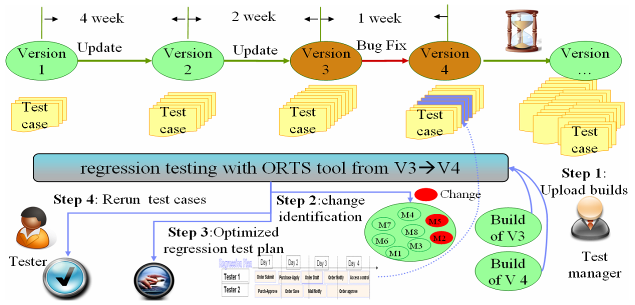

Figure 3. Regression process in ORTS

4. Demonstration

The goal of demonstration. Through the demo, we plan to show the following three aspects of ORTS. For a commercial java applications : (1) how to help testers capture runtime traces of test execution;(2) how to identify change points during build update;(3) how does user improve the regression testing efficiency by running regression test suite generated by ORTS. Figure 3 Regression process in ORTS

The way of demonstration. In the demonstration session, we plan to let participants interact with the tool directly, especially experience the regression process in ORTS as illustrated in Fig. 3. A sample commercial Java Web Application Scenario will be provided and deployed on the demo client computer. Some bugs are reported from customer after version 3 is delivered, expected to be resolved in one week. The testing team gets a new build v4 (Claimed to have fixed the bugs in v3) from the develop team. The build is about 24M, including 156 JSP files, 89 JS files, 369 Java files and 74 JARs. The demonstration mainly includes the following steps: First of all, with the help of regression wizard, we would like to let participants upload builds of v3 and v4 to ORTS and run build change identification. A report showing the change points of Java/JS/JSP and frameworks from v3 to v4 would be generated only in several minutes. In the next step, the participants could generate the regression plan automatically. Only 10% of test cases need to be rerun to check if the changes have caused adversely impact. Finally, the participants would be motivated to act as testers to rerun the test cases with and without the monitoring of ORTS agent. We would demonstrate that no obvious latency occurs under monitoring mode and various IT artifacts traversed are captured, including JS/JSP/JAVA codes.

Besides, we hope to discuss with participants to see how the regression process should be improved and what additional features are needed in ORTS to refine the design and implementation.

References

[1] M. J. Harrold, J. A. Jones, T. Li, D. Liang, A. Orso, M. Pennings, S. Sinha, S. A. Spoon, and A. Gujarathi. *Regression test selection for java software.* In Proceedings of 16th OOPSLA, 2001, 312-326.

[2] G. Rothermel and M. J. Harrold. *Analyzing regression test selection techniques.* Software Engineering, IEEE Transactions on Volume 22, Issue 8, 1996 , 529-551.

[3] J. Zheng, B. Robinson, L. Williams, and K. Smiley. *Applying Regression Test Selection for COTS-based Applications*. In Proceedings of 28th ICSE, 2006, 512-521.

[4] http://www.alphaworks.ibm.com/tech/contest

[5] http://www.eclipse.org/aspectj