REGRESSION TEST SELECTION TOOL

SIEMENS

Sabancı Üniversitesi

FACULTY OF ENGINEERING AND NATURAL SCIENCES

Student(s)

Cihan Şentürk
Hakan Büyüktopçu
Metin Berkay Ataklı
M. Tarık Demir

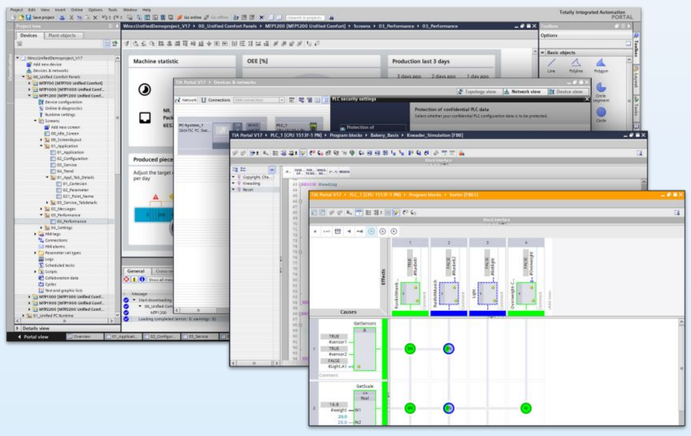
Faculty Member(s)

Anıl Koyuncu
Hüsnü Yenigün

Company Advisor(s)

Ceyda Baycan
Eren Samaner
Caner Reşber

ABSTRACT

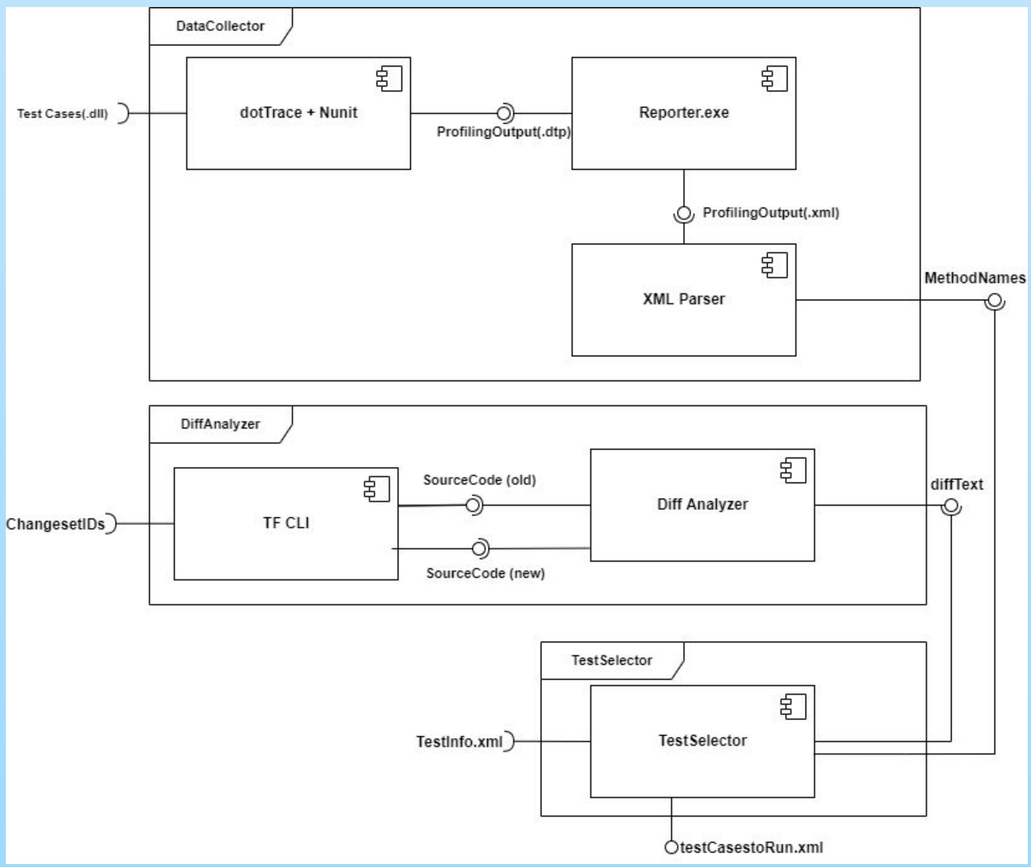


Regression testing[1] is the testing process that ensures the changed parts of software operate as intended and do not yield unexpected results. Running regression test suites for a large software is a problem since it takes too much time. Our project aims to provide a safe regression test selection tool for the software named TIA Portal owned by Siemens. It has a very large codebase and running regression test suites takes too much time: 12 hours for daily and 15-16 hours for biweekly regression testing. Product development team wanted to reduce this time, and our product managed to decrease the amount of tests and the time required by filtering out the redundant test suites.

OBJECTIVES

The main objective of the project is to reduce the duration of the regression testing process by using a test selection[2] method which will not miss any of the erroneous cases. The resulting product should be easy to use, give the desired output to be utilized by the tester and also be suitable for further improvement considering the continuous development of the company product.

PROJECT DETAILS



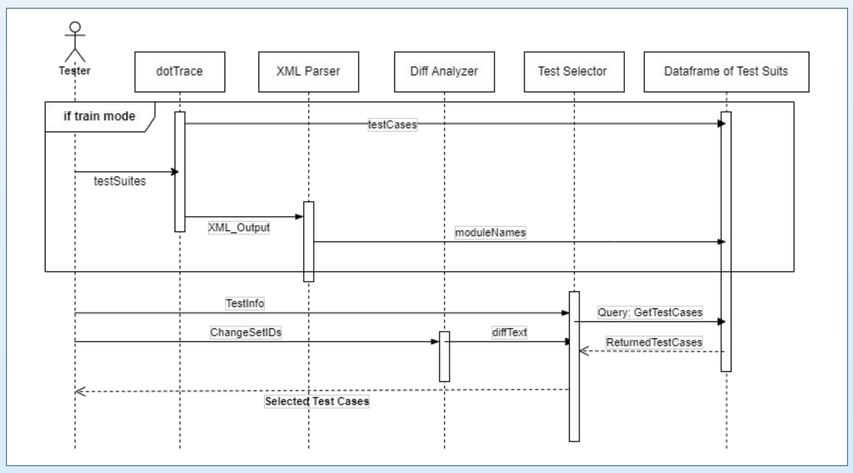
Component Diagram of the Software

The final software product is designed to consist of three main parts: namely ***DataCollector***, ***DiffAnalyzer*** and ***TestSelector***. DataCollector and TestSelector are implemented in Python programming language, whereas *DiffAnalyzer* is implemented in C# (ISO/IEC 23270:2018).

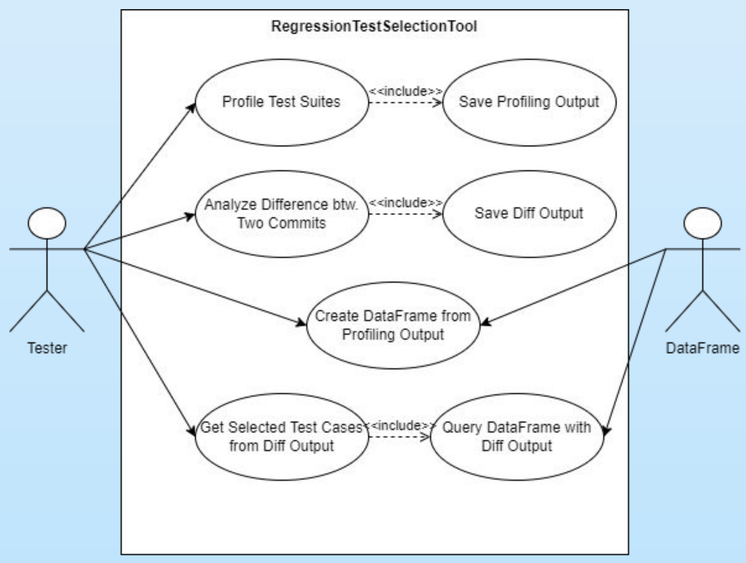
DataCollector is the initial component of the tool that mainly utilizes an external tool called dotTrace owned by JetBrains to obtain profiling output, namely snapshots, for each test case. Then, it converts snapshots to a text document in order to be used by *TestSelector* to form a mapping between test cases and the parts of the code (classes or methods) that are exercised by the test cases.

DiffAnalyzer provides changed methods and classes. First, using the command-line tool of TFS (Team Foundation Server), the list of changed files in a given changeset are found. Both old and new versions of these files are obtained. Finally, the difference between two versions of a file is analyzed by comparing the syntax trees of these files to find exactly which classes and which methods are modified.

TestSelector handles the creation of the mapping of methods and classes with their corresponding test cases and the selection of the test cases from given methods or classes utilizing this mapping. Basically, a test case is selected into the test suite for a changeset if and only if it exercises a part of a code that is modified in that changeset.



Sequence Diagram of the Software



Use Case Diagram of the Software

OUTCOMES

Before mentioning the outcomes, it should be stated that development and basic testing of this product was completed using similar open source software applications and tools. Also, the final tests and coverage calculations were performed on company environments by company advisors.

The tool was tested with the weekly test suites that the product development team uses and some sample weekly changesets. The results can be seen in the table below, using both class and method level differences for test selection. Method level drastically improves the reduction made by the tool for more than half of the changesets when compared with class level.

Using method level differences; although in some cases the number of tests that are reduced is noticeably slim, in more than ⅓ of the changesets the reduction rates go up to ⅔ of all test cases. On the other hand, the reduced time is relatively less when compared to the number of tests. This is because the remaining tests consist of longer tests, which keeps the time spent for testing higher than expected.

Change Set ID

#	Class Level	Time	Method Level	Time
38615	127	15:17:30	91	14:43:11
46594	46	12:38:56	46	12:38:56
49788	46	12:38:56	46	12:38:56
56098	127	15:17:30	122	15:04:59
78715	96	14:50:40	47	12:40:42
80004	0	-	0	-
84007	96	14:50:40	96	14:50:40
97963	127	15:17:30	107	14:48:01
101945	127	15:17:30	122	15:10:01
106308	122	15:10:01	120	15:02:00
115878	127	15:17:30	46	12:38:56

Total Test Cases: 128, Total Time: 15:40:32

REFERENCES

1. Memon, A. M., Pollack, M. E., & Soffa, M. L. (2001). Hierarchical GUI test case generation using automated planning. IEEE Transactions on Software Engineering, 27(2), 144–155. https://doi.org/10.1109/32.908959

2. Skoglund, M., & Runeson, P. (2005). A case study of the class firewall regression test selection technique on a large scale distributed software system. 2005 International Symposium on Empirical Software Engineering. https://doi.org/10.1109/isese.2005.1541816