Comparative Study of Regression Testing Tools Feature on Unit Testing

Asri Maspupah1, Ani Rahmani2, Joe Lian Min3

Departement of Computer Engineering and Informatic, Politeknik Negeri Bandung

1asri.maspupah@polban.ac.id, 2anirahma@jtk.polban.ac.id, 3joelianmin@jtk.polban.ac.id,

**Abstract**. Regression testing is one of the essential activities in software development. Regression testing ensures that the software is still working properly and new software modifications do not cause unexpected defects. These modifications can be caused by changes in requirements or repair of defects. Most of the software industry carries out regression testing on the system under test (SUT) by utilizing tools automatically rather than being done manually which requires a long time estimation and ineffective process. Currently, there are many regression testing tools and variations that make software practitioners need additional time and money to analyze the capabilities of the tools. This paper is a comparative study of the capabilities of tools in carrying out the regression testing process in the java programming language. Each tool has experimented with user input data in the form of a SUT program and a test case that has the same characteristics. Furthermore, the tool's performance is observed based on the feature evaluation criteria. The criteria used are support platform and technology, test case selection, and functionality. The results show how the tool's performance in conducting regression testing so that it can help users to choose regression testing tools that suit their needs.

1. Introduction

The software industry has shown growth in line with the development of software development technology over the past few years [1]. The next challenge is to produce high-quality software products to support end-user needed [2]. Regression testing is one type of software testing to find defects in parts of the software that have function appropriately as a result of software changes [3]. Regression testing is needed to ensure that these changes do not interrupt parts of the software that are not affected by the changes [4] [5]. Generally, the method of running regression testing is to run the retest to check whether there are new defects or fixed defects reappearing [5].

Regression testing in the software industry uses manual and automated testing. Although manual testing takes a lot of time and resources, is not reusable, and prone to tester errors due to the repeated testing process, 60% of companies are satisfied with the manual approach [6] [1] [7]. Meanwhile, in automatic testing, companies tend to choose regression testing tools that do not suit their needs due to a lack of knowledge [6]. Where automation testing should be able to improve test coverage and reliability test [8] [6]. In the industrial era 4.0, there are various kinds of regression testing tools that are available free of charge through open source or sold commercially with various features. Therefore, the company requires initial set-up costs, training, and time analysis tools to find out which tools suit the needs and environment of the software developer [6].

Meanwhile, the previous research discussed many comparative studies on the ability of automation testing tools. However, rarely does research discuss regression testing. Research in [2], [9], [10], and [11] used a comparative study of automation testing tools using feature evaluation parameters as the basis for comparison. Meanwhile, [12] and [3] conducted a comparative analysis of regression testing tools. On the other hand, the implementation of a comparative study can use an exploratory study approach through testing tools based on data input by answering feature evaluation parameters or based on a questionnaire survey to respondents by providing questions form the basis of comparison.

Based on the above phenomena, this paper focus on a comparative study of the capabilities of tools for regression testing, especially for the tools that support automatic testing in software development using the Java programming language. The comparative study uses evaluation feature parameter analysis that includes the software environment, test case selection, functionality, and usability. Each tool will be tested by using input data in the form of a SUT program and a test case that has the same characteristics. The research result is a feature tool matrix.

The systematics of this paper is organizing as follows: Session 1, the background and objectives of the research. Session 2, related work under comparative study tools. Session 3, the methods used in this research. Session 4, the discussion of this research results. Finally, session 5 discusses conclusions.

1. Method
	1. Selection of Regression Testing Tools

The selection of regression testing tools was conducted by research reviews on several market places that provide automation testing tools, both open-source and commercially sold [2]. The research review using literature reviews on several scientific publications and tool provider sites. Determination of regression testing tools based on criteria;

1. Automation testing tools can support regression testing techniques including selection, minimization and prioritization. A description of the regression technique is shown in Table 1.
2. Automation testing tools can provide testing for software development in java programming language.
3. Automation testing can provide testing at the unit level.

**Table 1.** Regression Testing Technique [12] [13]

|  |  |
| --- | --- |
| Regression Testing Technique | Description |
| Regression Test Selection (RTS) | Identification of test cases that are relevant based on the program has been changes. |
| Regression Test minimization (RTM) | Eliminate redundant test cases based on criteria. |
| Regression Test prioritization (RTP) | Sort test cases based on their priority level with relevant test cases based on a certain approach |

Based on these criteria, we chose six regression testing tools that are,

1. RTSTester is a java tool for automatic regression test selection in open source. TRSTester uses two test case selection libraries (Ekstazi and STARTS) that can be selected [14].
2. STARTS is a static regression test selection for programs that use the Java Maven-based project. STARTS performed a static dependency analysis to find test cases affected by the change [15] [16].
3. babelRTS is a multi-language tool for carrying out regression test selection by selecting test cases based on the code changed at the time of the modification program [17].
4. Ekstazi is a java library for regression test selection based on file dependency [18] [19].
5. TMP is a regression testing tool that applies the elimination of test cases to a set of test cases using a linear programming approach [20].
6. Total & Additional Statement Coverage is a regression testing tool for sequencing a set of test cases using the total and additional statement coverage approach [21] [22].
	1. Parameter studies feature evaluation

To know the differences and similarities of features in regression testing tools, we identified feature evaluation parameters as a means of observation during a comparative analysis [2], [9], [10], [11] and [23] that to determine whether regression testing tools are appropriate, we can consider several key points, namely support to platforms and technology, test case selection, and functionality. The definitions of feature evaluation parameters are shown in Table 2.

**Table 2.** Parameter Feature Evaluation

| Parameter Evaluation | Description | Aspects studied | Reference |
| --- | --- | --- | --- |
| Cross Platform | operating system supported | supported platform | [2] [10] [11] |
| Cost | the cost of providing tools | free or license | [2] [9] [10] |
| Regression testing method | type of regression testing technique | Type of regression testing method: RTS, RTM, or RTP | [12] |
| Classification of test cases | how to arrange a test case | reusable, retestable and obsolute test case | [12] |
| Functionality | a function provided by tools for running regression testing | regression testing implementation and number of line code | [10] [11] [23] |

* 1. Comparative Studies

The comparative study was conducted by an exploratory study method based on experimental trials [24]. Each tool is tested by running tools for regression testing by providing the same input data. The characteristics of the input data are shown in Table 3.

**Table 3.** Characteristics of experimental input data

|  |  |  |
| --- | --- | --- |
| Input Data Type | Description | Input Data |
| System Under Test (SUT) | program code to be tested | JodaTime |
| Test Case | a set of test case scenario data created in the form of a program at the unit testing level | Script test joda time |

Experimental testing is done by testing the program when a feature modification occurs so that the test is carried out twice. First, testing after the program has been implemented to find out defects. Second, testing when the program code changes to find out whether the program code is still functioning appropriately. Furthermore, the process of observing behavior and outputs regression testing tools is dependent on feature evaluation parameters as experimental results. Finally, the comparative analysis process is to determine the similarities and differences in the features of regression testing tools.

1. Result and Discussion

All regression tools are compared into several feature evaluation parameters as shown in Table 4. The table can be used as a comparative review in selecting tools that suit your needs and objectives.

Tabel 4. Comparative Review of regression testing tool

|  | **RTSTester** | **STARTS** | **Ekstazi** | **Total & Additional Statement Coverage** | **babelRTS** | **TMP** |
| --- | --- | --- | --- | --- | --- | --- |
| **cross platform** | Windows, LinuxOS | LinuxOS, MacOS | Windows, LinuxOS | windows | Windows, LinuxOS | Windows |
| **cost** | free | free  | free  | free  | free  | free  |
| **regression testing method** | RTS | RTS | RTS | RTP | RTS | RTM |
| **functionality:** |  |  |  |  |  |  |
| ability | run tests on selected test cases | run tests on selected test cases | run testing test cases based on priority | run tests on selected test cases | provide the relevant test cases to run | provide the relevant test cases to run |
| use JUnit | Yes | Yes | Yes | Yes | No | No |
| test case criteria | It depends on the selected RTSTester library, namely Ekstazi or STARTS | The test case is relevan with program code changes | The test case is relevan with program code changes | sort test case relevan with program code changes | The test case is relevan with program code changes | Complete test case coverage and minimal execution time |
| How work test case choosen | dependency graph | dependency file by invoke method base on checksum function | Sorting test case base on statement coverage test case | “Change” statue on each program file has been a change | calculate linear programming on each script test |
| run tool | run the RTSTester class | run the tool from the command line | run the tool from the command line | run the tool from the command line | run the tool from the command line | run the tool from the command line |
| Running regression testing | run based on the selected test case | run based on the selected test case | run under test case priority order | run based on the selected test case | not run | not run |
| Output  | test log containing the number of test cases, number of failures, execution time, and build status | a list of selected test cases for regression testing |
| number line of code | 973 line of code | 883 line of code | 4,230 line of code | 25,236 line of code | 151 line of code | 1,027 line of code |
| **classification of test cases:** |
| reusable | It depends on the selected RTSTester library, namely Ecstasy or STARTS | dependency distance that is far enough through the program method which is connected | not available | not available | not available | not available |
| retestable | direct dependency | The checksum values for the file dependencies are the difference | Test cases are relevant with changing program | test case with program code status changed | The result of the calculation of the linear programming formula is 1 |
| obsolute | has no dependencies | The checksum values for the file dependencies are similarly | Test cases not relevant with changing program | test case with program code status not changed | The result of the calculation of the linear programming formula is 0 |
| **source code** | [14] | [16] | [19] | [21] [22] | [17] | [20] |

Based on table 4, parameter feature evaluation as a basis for the comparison of regression tools. In the results of the comparative analysis, three tools run on both platforms, Windows and Linux, namely RTSTester, Ekstazi, and Babel. While Total & Additional Statement Coverage and TMP can only run on Windows, STARTS can only run on Linux. Thus, most of the tools can be run on the Windows operating system.

On another aspect, its ability to perform regression testing, there are four of the six tools capable of carrying out test execution after a get selected test case. Meanwhile, babelRTS and TMP are only able to provide a list of test cases that are relevant to the regression testing process criteria without running a test case. The shown that use of the JUnit library as a prerequisite for running tools to regression testing.

One of the main aspects of carrying out regression testing is the test case sorting criteria and the method used to obtain the selected test cases. Changing the program code is one of the most commonly used test case sorting criteria. As many as five of the six tools that use these criteria, TMP uses the test coverage criteria and run time execution criteria. Meanwhile, from the method used to get the test case selected, all tools have different ways.

On the other hand, there are aspects of test case classification that affect the sorting of test cases, namely reusable, retestable, and obsolete. Classification of test cases based on reusability has one tool that provides this feature, namely STARTS. Meanwhile, the test case classification is based on retestable and obsolete used in all tools. The method used in the classification depends on the test case selection method so that all tools have different test case classification methods.

1. Conclussion

This study discusses the analysis of the different features of regression testing tools such as RTSTester, STARTS, babelRTS, Ekstazi, Test Minimization Project, and Total & Additional Statement Coverage. Since regression testing in unit testing is part of what software companies need to do, the results of comparative analysis can help users as a reference for selecting tools.

Based on the results of the comparison of the feature comparisons of all tools, each tool has similarities and differences that can be viewed from the evaluation parameters used. Thus a comparative study of feature evaluation can use cross-platform and technology, cost, regression testing method, classification of test cases, and functionality.

In future studies, research can add other feature evaluation parameters such as usability, drawbacks/strengths / interesting findings, environment tools, and performance efficiency as a basis for comparison when conducting a comparative study.

Reference

1. Silva, Roshali. Perera, Pulasthi. Perera, Indika, and Samarasinghe, Kithsiri. “Effective use of test types for software development,” *in 17th International Conference on Advances in ICT for Emerging Regions, ICTer 2017 - Proceedings*, pp. 7–12, Sept 6-9 2017.
2. Gamido, Heidilyn V. and Gamido, Marlon V. “Comparative review of the features of automated software testing tools,” International Journal of Electrical and Computer Engineering (IJECE), vol. 9, no. 5, pp. 4473–4478, October 2019.
3. Zarrad, Anis. “A Systematic Review on Regression Testing for Web-Based Applications*,” Journal of Software*, vol. 10, no. 8, pp. 971–990, Agustus 2015.
4. Kandil, Passant. Moussa, Sherin and Badr, Nagwa. “A methodology for regression testing reduction and prioritization of agile releases,” *in 2015 5th International Conference on Information and Communication Technology and Accessibility, ICTA 2015*, pp. 1-6, Dec. 21-23, 2015.
5. Rosero, Raul H. Gomez, Omar S. and Rodriguez, Glen. “An approach for regression testing of database applications in incremental development settings,” *in 2017 6th International Conference on Software Process Improvement (CIMPS), 2017*, pp. 1–4. Okt. 18-20, 2017.
6. Minhas, Nasir Mehmood. *Regression Testing Challenges and Solutions*, Sweden: Blekinge Institute of Technology, 2018.
7. Sutapa, F. A. K. P. G. Kusumawardani, Suning. and Permanasari, A. E. “Automated Test Suite for Regression Testing Based on Serenity Framework: A Case Study,” *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, pp. 138–144, March 13-15, 2019.
8. Engstrom, Emelie and Runeson, Per. A “Qualitative Survey of Regression Testing Practices,” *11th International Conference on Product Focused Software Development and Process Improvement (PROFES ‘10)*, pp. 1-12, Jun. 21-23, 2010.
9. Rana, Kritika and Yousuf, Mir Mohammad. “Comparative Analysis of Testing Tools,” *Journal of Emerging Technologies and Innovative Research*, Volume 5, Issue 10, pp. 585 – 591, Oktober 2018.
10. Kannan, M. and Lokeshwari K. “Comparison Of Software Testing Tools With Respect To Tools And Technical Related Parameters,” *International Journal of Advanced Research in Computer Science (IJARS*), volume 8, no. 9, pp. Dec.70-75 2017
11. Satheesh, Arjun and Singh, Monisha, “Comparative Study of Open Source Automated Web Testing Tools: Selenium and Sahi,” *Indian Journal of Science and Technology*, vol 10(13), pp 1-9, April 2017.
12. Kandil, Passant. Moussa, Sherin and Badr, Nagwa. “A Study for Regression Testing Techniques and Tools,” *International Journal of Soft Computing and Software Engineering [JSCSE]*, vol. 5, no. 4, pp. 64-84, April 2015.
13. Rosero, Ra´ul H. G´omez, Omar S. and Rodriguez, Glen. “15 Years of Software Regression Testing Techniques – A Survey,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 05, pp. 675-689, July 2016.
14. Shen, Yukang. “RTS Tester source code,” Github, Dec 24 2018, [online]. Available on: https://github.com/JohnsonSYK/RTSTester. [Accessed: September 29 2020].
15. Legunsen, Owolabi. Shi, August. and Marinov, Darko “STARTS: STAtic Regression Test Selection,” *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 949-954, Oct. 30 - Nov. 3, 2017.
16. Gyori, Alex. “STARTS source code,” Github, April 16 2019, [online]. Available on: https://github.com/TestingResearchIllinois/starts. . [Accessed: September 29 2020].
17. Maurina, Gabriele. “babelRTS source code,” Github, Agustus 30 2020, [online]. Available on: https://github.com/GabrieleMaurina/babelRTS. . [Accessed: September 29 2020].
18. Gligoric, Milos. Eloussi, Lamyaa. and Marino, Darko. “Ekstazi: Lightweight Test Selection,” *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. pp.713-716, May 16-24, 2015.
19. Gligoric, Milos. Eloussi, Lamyaa. and Marino, Darko. “Ekstazi: Lightweight Test Selection Source Code,” Website, 2014, [online]. Tersedia: http://www.ekstazi.org/index.html. [Diakses: 29 September 2020].
20. Irenelizeth, “Simple Test Suite Minimization Source Code,” Github, June 23 2017, [online]. Available: https://github.com/irenelizeth/test\_minimization. . [Accessed: September 29 2020].
21. Vyas, Anand. “Repository creating-prioritized-testclass-list,” Github, May 2 2017, [online]. Available on: https://github.com/anand-vyas/creating-prioritized-testclass-list. . [Accessed: September 29 2020].
22. Vyas, Anand. “Repository automated-test-prioritization,” Github, May 2 2017, [online]. Available on: https://github.com/anand-vyas/ automated-test-prioritization. . [Accessed: September 29 2020].
23. Sandin, Easter Viviana. Yassin, Noraniah Mohd. and Mohamad Radziah. “Comparative Evaluation of Automated Unit Testing Tool for PHP,” *International Journal of Software Engineering and Technology*. vol. 3, no. 2. pp. 7-11. Dec. 2016.
24. Edgar, Thomas W. and Manz, David O. “Research Methods for Cyber Security.” Elsivier, 2017.