

Improving the Efficiency of Regression Test Case Prioritization for Large Scale Workflow SOA

GT RAMYA

Department of CSE, JNTUCEA(Autonomous) Ananthapuramu

Abstract— *Large software system development is a difficult and error-prone process. Any level of development may have errors, thus it's important to find and fix them as soon as we can to prevent further development and lower verification costs. To define necessary characteristics and evaluate their influence on the development process, quality engineers must be involved in the process from the very beginning. So, software testing provides accuracy and quality of the software product and services under the test. Many web services are service-oriented workflow applications with different functions. Web Service Business Process Execution Language (WSBPEL) has become the standard architecture for all service applications online. These applications often suffer from failures or defects, especially during the service evolution of service composition. Although costly, regression testing is a crucial part of software maintenance. Regression testing verifies altered software and makes sure that the added features did not create unforeseen faults. In the existing system, WSBPEL activity dependencies such as correlation and synchronization dependencies are proposed. For analyzing the internal structure changes module dependency technology is used and modification impact analysis is used for test case Prioritization of service oriented applications. Program slicing techniques are the foundation of the majority of regression test selection methods. In the existing system slicing technique is used for prioritizing regression test cases. But, it does not support large scale workflow applications. It is important to propose an LSBPEL (Large Scale Business Process Execution Language) technique after studying the demand for large scale and many service-oriented workflow applications. In comparison to conventional approaches that address single service test case priority, the suggested solution is more efficient.*

Indexed Terms— *Regression Testing, Test Case Prioritization, BPEL, Large Scale Applications.*

I. INTRODUCTION

Change is inevitable in any engineering discipline. In software Engineering it is even more important when it comes to the quality. To provide good quality of the

product, software testing plays a vital role in delivering such. Software Testing validates whether a product fulfills clients needs or not. Large-scale programming is now an established technology in service computing. All service applications in the internet space now use WSBPEL as their standard design. The previous ten years have seen significant changes in software design and development activities. Traditionally, software systems were created to function in a known, unchanging environment [1].

A maintenance lifecycle (the "design, development, and deployment" of a new Version) was used whenever software had to be upgraded in order to enhance its quality or satisfy new needs [1]. Costly maintenance activities and an unacceptable time to market were the results of this strategy. The same consolidated testing techniques that have been used for years on conventional systems also apply to service-centric systems. Primarily, the idea that a combination of unit, integration, system, and regression testing is needed to gain confidence that a system will deliver the expected functionality [2]. Software maintenance is becoming very important and more expensive day by day. When the software is modified during maintenance phases, retesting is performed and this process is nothing but the Regression Testing [3]. It aims at detecting potential faults caused by software changes, and is the de facto approach. It reruns test cases from existing test suites to ensure that no previously working function has failed as a result of the modification. Although many researchers point out that frequent executions of regression tests are crucial in successful application development, rerunning the regression test suite for large and complex systems may take days and even weeks, which is time-consuming again [3]. When running the test suite and identifying issues, it is preferable to identify failures as soon as possible to save expenses. Regression

testing methodologies must be used effectively as a result [5].

Various application sources involved in various projects are taken into one unit and built as a single application. Once the application is ready for testing, the test suite will be prepared and types of testing to be performed will be listed out i.e., Black Box testing, White Box testing, Unit Testing, Regression Testing, Retesting etc.,. When the test selection is done, then priority of the selection to be done. Post prioritizing the test selection, execution takes place based on priority [5]. So, test case Prioritization is important in regression testing as shown in following Fig. 1.

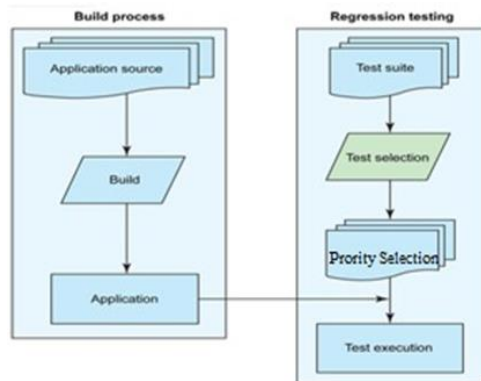


Fig. 1: Test Case Prioritization in Regression Testing

II. SOFTWARE TESTING

Running an application with the goal of identifying software problems is known as software testing (errors or other defects). Demand for software applications has elevated the level of quality control for newly generated software. It has been regarded as the phase of the software development life cycle that is the most crucial. Software items can be examined through testing to determine the discrepancy between real and desired circumstances and to evaluate the software's attributes.

III. RELATED WORK

The issue of addressing changes regarding the non-functional behavior of software services controlled by external organizations—and hence regarded as black-box artifacts—is addressed in this study by Epifani, I., et al. The author presents a statistical method for

identifying change points as well as a description of the execution trail brought on by client invocations. The author additionally created a change-point analysis tool as a component of the KAMI framework "a toolset" and used simulations to test the approach. As a consequence, the relationship between the length of the trace and the range of probabilities included in the models as well as the distance between various change-points (in a multiple change-point scenario) is carried out [1]. Numerous prioritizing strategies arrange test cases according to the specific programme statements they cover.

IV. PROPOSED WORK

In this proposed work, scheduling approach for the test case Prioritization is used. The proposed grid based computing application is more effective than traditional which covers single service test case priorities. It has been determined in this study that regression testing is the kind of testing used to find software defects that arise as a result of modifications made to the product. After reviewing the various test case priority methods, it was determined that the BPEL algorithm was the most effective method for Prioritization. The BPEL method will be improved in the future to increase the accuracy of mistake identification during regression testing. Software testing is a crucial step in ensuring the accuracy, completeness, and utility of user requirements and specifications [1]. A technique is used to ensure that the programme is error-free [2]. Regression testing is one of the approaches used in software testing. It is a technique that keeps track of new software updates and ensures that the upgrades won't have an impact on the functionality of the product as it now exists [3]. This requires ensuring that none of the software modifications interfere with the already-existing functions.

The process of regression testing is started with the first release of the product, so it is said to be a maintenance activity in software development process models. Any change request, software, or service update or next release of the same product leads to the regression testing [4]. The change in requirement leads to the code modifications. After the source code is modified, regression testing is commenced; any regression error during regression testing is looped

back to code modification [3]. The regression errors include the bugs like software enhancement problems, configuration mismatches, substitution errors, structural failures in code, and service unavailability or failures. After the completion of regression testing, the new version of software has been released as shown in the fig. Likewise, test case Prioritization technique segregates the test cases based on some adequacy measures results to lower testing costs and improve testing process efficiency, but TCP does not exclude test cases from actual test suites. Test case selection technique eliminates repeated test cases from test suites. The primary goal of test case reduction techniques is to reduce the number of test cases in the test suite, which raises the price of regression testing. The goals of selection and prioritizing are the same, but Prioritization techniques organize test suites.

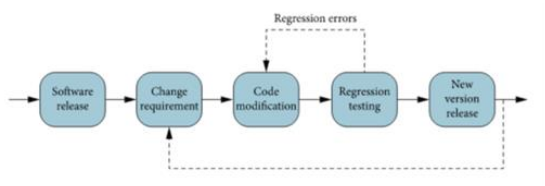


Fig. 2 : Maintenance Process Model

The above Fig. 2 shows the Software Coding and Testing Stages through Maintenance Process Model where the objectives of regression test case Prioritization techniques were widely characterized in the literature as N-release development, continuous quality upgrades, continuous development, and continuous integration [8].

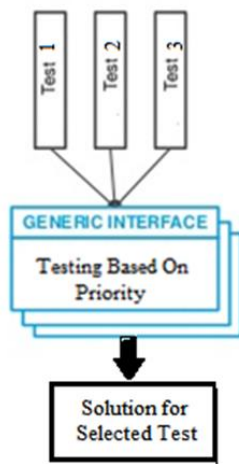


Fig. 3: Proposed Testing Architecture

Advantages:

- A large number of service-oriented workflow applications are provided.
- Provides priority-based testing choices
- First, high priority selection is made throughout the testing procedure.
- Selects the top priority items to be handled first.

V. METHODOLOGY

The priority problem is to map every $i \in T$ onto some $j \in S$ to achieve minimum execution cost and complete the workflow execution within the deadline D . We solve the scheduling problem with priority concept by following the divide-and-conquer technique and the methodology is listed below:

- Step 1. Discover available services and predict execution time for every task.
- Step 2. Group workflow tasks into task partitions.
- Step 3. Distribute users' overall deadline into every task partition.
- Step 4. Query available time slots, generate optimized priority plans and make advance reservations based on the local optimal solution of every task partition.

Algorithm:

```

Algorithm 1
1  request processing time and price from available services for  $\forall T_i \in \Gamma$ 
2  convert  $\mathcal{Q}$  into task partition graph  $G(V,E,D)$ 
3  distribute deadline  $D$  over  $\forall V_i \in G$ 
4  Repeat
5   $S \leftarrow$  get unscheduled task partitions whose parent task partitions
6  have been scheduled
7  for all  $i \in S$  do
8    compute ready time of  $i$ 
9    query available time slots during ready time and sub-deadline
10   on available services
11   if  $i$  is a branch then
12     compute an optimal schedule for  $i$  using BTS
13   Else
14     compute an optimal schedule for  $i$  using STS
15   end if
16   make advance reservations with desired services for all tasks in  $i$ 
17   adjust sub-deadline of  $i$ 
18 end for
19 until all partitions have been scheduled
  
```

Methods Used:

The goal of test case Prioritization is to enhance the likelihood and rate of fault detection if the test cases are in order during regression testing, in accordance with specified performance goals and determinants. Prioritizing test cases can help risk-based testing's risk

analysis process. The following objectives can be addressed via test case Prioritization:

- The testers seek to raise the chance of finding flaws early in the regression testing and to boost the rate of fault identification.
- Increases the identification of high-risk problems and finds them early in the regression testing process.
- Increases the probability of finding regression errors when the code changes during the first regression testing phase.
- Increases the pace at which they test the system's coverable code coverage.
- Increase the speed to develop their confidence in the reliability of the system.

Method	Mean	Variance
Non-prioritized	51.5	576.4
Adaptive test-case prioritization	86.9	82.8
Similarity-based test quality metric	47.5	410.4

Table 1: Table Showing Non Prioritization and Prioritization Ratios

VI. EXPERIMENTAL RESULTS

Experimental setup for this study is discussed in this section. Experimentation is performed by using the system with the following specifications: Intel Core i7 processor with 8 GB RAM, and Java as a language. Selected datasets with details are used for showing test case priority.



ID	Project Name	Bug	Priority
1	food	food is worst	1
2	food	long booking problem	1
3	ticket	ticket not booking	1
4	ticket	tickets not refunded	1
5	ticket	tickets not refunded	1
6	ticket	tickets not refunded	1
7	ticket	tickets not refunded	1
8	parcel	parcel not booking	1

Fig. 4: List of Bugs with their Priority Levels

In Fig. 4 showing various bugs listed in working projects with priorities, based on project and bug

priorities, solutions will be worked out by programmers in rectifying. Figure showing project name, bug and priority columns which are noted in various levels.

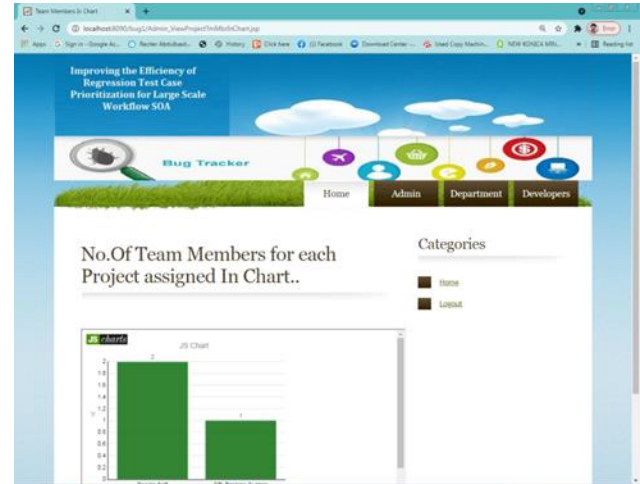
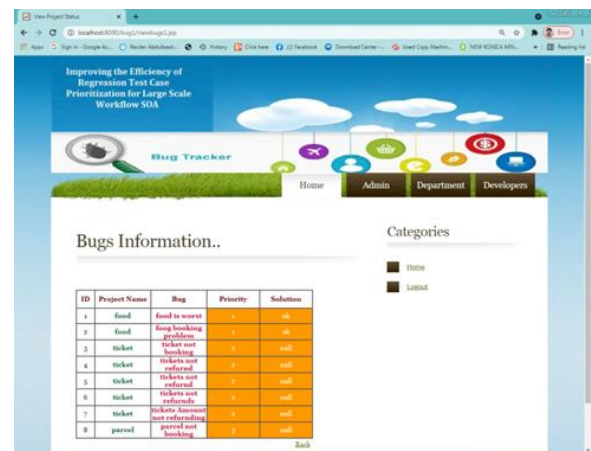


Fig. 5: Chart Showing Various Team Members with Bug Management

In Fig. 5, chart showing various team size noted in various projects based on the team size and bugs noted, priorities will be assigned for faster and better access.



ID	Project Name	Bug	Priority	Solution
1	food	food is worst	1	ok
2	food	long booking problem	1	ok
3	ticket	ticket not booking	1	not
4	ticket	tickets not refunded	1	not
5	ticket	tickets not refunded	1	not
6	ticket	tickets not refunded	1	not
7	ticket	tickets not refunded	1	not
8	parcel	parcel not booking	1	not

Fig. 6: Report Showing Solved and Unsolved Bugs on Priority

In Fig. 6, showing various bugs listed in working projects with priorities and status showing solved and pending bugs, based on project and bug priorities solution is solved and worked out by programmers. Figure showing project name, bug, priority and solution columns which are noted in various levels.

VII. CONCLUSION AND FUTURE WORK

Test case prioritizing is a technique used in regression testing to increase the effectiveness of error detection in changed service-oriented workflow systems by switching the order in which test cases are run. This study suggests a novel test case prioritizing methodology for regression test cases in grid-based computing systems as well as a workflow scheduling approach for service-oriented workflow applications. Instead of using the FIFO concept, which requires important projects to wait in order to fix bugs, the proposed grid-based computing applications are more effective than the conventional ones. In the suggested strategy, each project is given a priority, and critical projects whose flaws affect single service test case priorities are fixed first.

REFERENCES

- [1] N. B. Ellison, V. Jessica, G. Rebecca, and L. Cliff, "Cultivating social resources on social network sites: facebook relationship maintenance behaviours and their role in social capital processes," *Journal of Computer-Mediated Communication*, vol. 19, no. 4, pp. 855–870, 2014.
- [2] R. Kazmi, D. Abang Jawawi, R. Mohamad, and I. Ghani, "Effective regression test case selection: a systematic literature review," *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–32, 2017.
- [3] J. Ahmad and S. Baharom, "Factor determination in prioritizing test cases for event sequences: a systematic literature review," *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 10, no. 1–4, pp. 119–124, 2018.
- [4] R. D. Adams, "Nondestructive testing," in *Handbook of Adhesion Technology*, Springer, Berlin, Germany, 2018.
- [5] M. Khatibsyarbini, A. M. Isa, D. N. A. Jawawi, and R. Tumeng, "Test case Prioritization approaches in regression testing: a systematic literature review," *Information and Software Technology*, vol. 93, pp. 74–93, 2018.
- [6] E. Cruciani, B. Miranda, R. Verdecchia, and A. Bertolino, "Scalable approaches for test suite reduction," in *Proceedings of the 41st International Conference on Software Engineering*, Montreal, Canada, May 2019.
- [7] G. P. Sagar and P. Prasad, "A survey on test case Prioritization techniques for regression testing," *Indian Journal of Science and Technology*, vol. 10, no. 10, pp. 1–6, 2017.
- [8] Q. Luo, K. Moran, and D. Poshyvanyk, "A large-scale empirical comparison of static and dynamic test case Prioritization techniques," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, New York, NY, USA, November 2016.
- [9] JUnit, Java Testing, 2016, <http://junit.org/junit4/>.
- [10] V. Neethidevan and G. Chandrasekaran, "Database testing using Selenium web driver—a case study," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 8, pp. 559–566, 2018.
- [11] Z. Sultan, S. Nazir Bhatti, S. Asim, and R. Abbas, "Analytical review on test cases Prioritization techniques: an empirical study," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 2, 2017.
- [12] B. Jiang and W. K. Chan, "Input-based adaptive randomised test case Prioritization: a local beam search approach," *Journal of Systems and Software*, vol. 105, pp. 91–106, 2015.
- [13] F. Harrou, Y. Suna, B. Taghezouitb, A. Saidic, and M.-E. Hamlatid, "Reliable fault detection and diagnosis of photovoltaic systems based on statistical monitoring approaches," *Renewable Energy*, vol. 116, pp. 22–37, 2018.