Thesis Title: Automated Regression Testing Process Model in Agile Environment.

CPSC543: Software Maintenance

Instructor: Dr. Song-James Choi

By,

Deepak Prasanna: dee7@csu.fullerton.edu

Date of Submission: 11/4/2018

Abstract

Regression Testing is a type of software testing used to find whether any new problems occur as a result of software changes. Before applying a change in a program, a program is first tested. After a change is applied, the program is retested in selected areas to determine whether the change has created new bugs or issues, or if the actual change has achieved it’s the planned objective.

To create a plan for an effective regression testing we should first identify what test-cases must be executed, what must be improved and changed in the test-cases, when it is necessary to execute regression testing of the system, what and how should be automated, what is needed to perform the automated testing, how to analyze the outcomes of the regression testing.

Whether it’s a mature testing process or just a start, some steps are necessary to ensuring that we have a quality regression test suite for each business domain. It’s important to update our test suites to maintain the precise and detailed representation of the business domain, to review test suites with business stakeholders, development and testing teams to achieve the high coverage for the required business domain.

We should create out test suites based the priority. Regression prioritization should be based on the importance of the test cases and whether these cases can detect and identify possible defects in the product. But there are some challenges that organization face while performing the regression testing in a software project such as complexity of the software due to addition of more and more functionalities, time consuming due to execution of all the tests cases. We can overcome these challenges by applying some best practices/approaches in our software testing process.

Table of Contents

1 Introduction....................................4

1.1 Problem Statement..................................5

2 Current Approaches....................................5

2.1 Test Case Prioritization..................................5

2.2 Reusable and Obsolete Approach..................................6

2.3 Include All Complex Test Cases..................................8

3 Our Approaches/Improvement Suggestions....................................9

3.1 Categorization of Test Cases..................................10

3.2 Automate the Regression Testing..................................11

3.3 14

3.4 14

4 Conclusion 14

5 References....................................16

1 Introduction

In Software projects, half of the total cost of the product is due to maintenance. As most of the defects are found after the release or if new requirements arise from the users after every release, such as adding new features or making improvements to an existing feature. Once we perform the new changes to the code it is vital to evaluate the impact of change on existing features. Instead releasing a software with improper evaluation can cause system failure or some critical bugs in production environment. Hence, regression testing is recommended after adding new features or after fixing bugs.

Regression Testing is one of the successful testing method which analyzes the impact of recent changes made to the existing features or new features of the software. Regression testing mainly checks whether the newly added features has modified the existing behavior of software. During regression testing we retest previous test cases to assure all the existing features are working as expected.

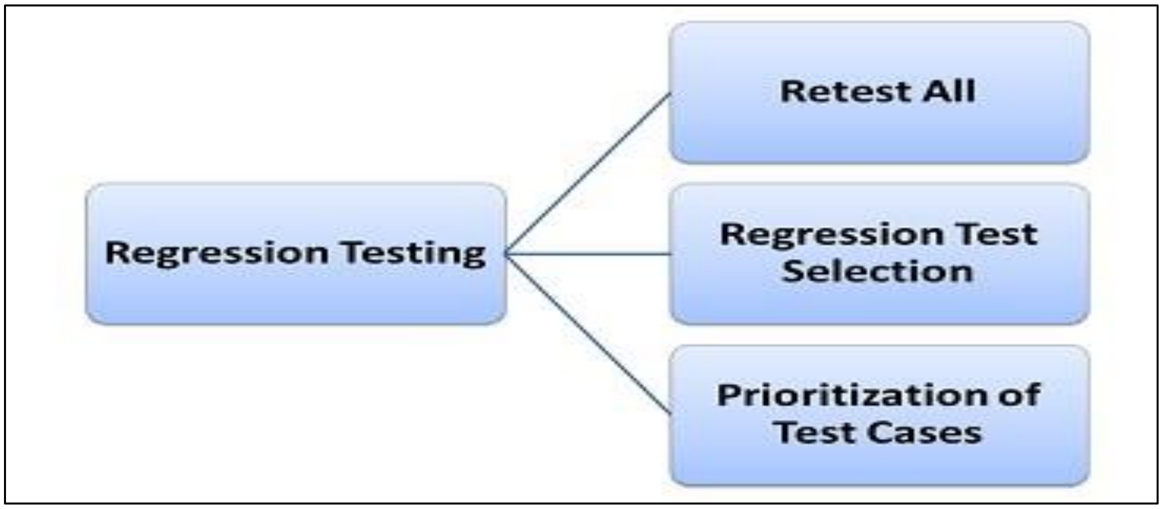


Fig 1 – Regression Testing Approaches

1.1 Problem Statement

Regression tests can be performed manually on small projects and in agile projects regression testing strategy is considered endangered as it consumes more time than other testing strategy such as smoke tests. However, regression testing strategy improves the quality of the software and helps the professionals to revisit test cases after the completion of each sprint. In this document, we recommend best practices to the companies to adapt regression testing in Agile environment. We recommend two approaches

1. Risk Based Testing fused with Regression testing strategy.
2. A 5-process model to implement automated regression testing. In addition to the 5-process model, we also recommend companies to use Testing Whiz tool to implement automate regression testing effectively.

2 Current Approaches

1. To effectively carry out regression testing we need to firstly prioritize the test cases as per the criticality and frequent usage of features.

2. Selected test cases should be re-executed in place of executing all the previous test cases. The selected test cases should be divided into two categories as Reusable and Obsolete. Only reusable test cases should be used in subsequent regression cycles.

3. Include All Complex Test Cases.

2.1 Test Case Prioritization

Test case prioritization helps in improving the performance of the software testing. One of the objective of prioritization is to detect the Fault rate which is a measure to find out how fast the faults are detected in the software testing process. If we have an improved rate of fault detection during testing, it can provide quick feedback on the system being tested. It further enables software engineers to start fixing the faults very early which is otherwise not possible.

Prioritize the test cases as per the business impact, criticality and frequently used functionalities. It is always beneficial if an evaluation is completed to check which test cases are relevant. One approach is to assign priorities to the test cases based on importance and customer use. Below are the categories given in which test cases can be prioritized.

- Priority 0: Sanity test cases check for basic functionality (as per the SRS of the application) and are run to verify pre-system acceptance and ensure functionality after an application under test goes through a major change. These test cases deliver high project value.
- Priority 1: This includes the test cases that test the essential functionalities for delivering high project value.
- Priority 2: These are executed as a part of the system test cycle and are selected for regression testing as-needed. These test cases deliver moderate project value.

Pros:

- Prioritization technique can significantly improve the rate of fault detection.
- The selection of test cases based on priority will greatly reduce efforts spent on regression testing.
- Prioritization enables software engineers to start fixing the faults early.

Cons:

- Project scope is limited.
- Cost reduction is not significant.
- Prioritization for fault rate detection is not much efficient.

2.2 Reusable and Obsolete Approach

In large software projects the regression testing is difficult to perform due to continuous increments/patches to the system. Therefore, we need to execute only few significant test cases to save the cost of testing and execution time. In such situations we need to categorize the test cases into two categories which are given below.

a) Reusable Test Cases:

These are the test cases which can be repetitively used in regression testing. These test cases should be automated to ease the execution of tests cases on every new build. Instead of writing test cases for an entire application, the test cases should be written only for individual features or functions. In order to test the whole system, we can link test cases for related features and write test cases only for unique features.

The reusable test cases should be designed using the technique which distinguishes the common part and specific part. A test suite is divided into parts common and specific. The common part includes the features or elements which are always the part of a software being tested. Whereas, the specific part includes the elements that may change or permanently removed.

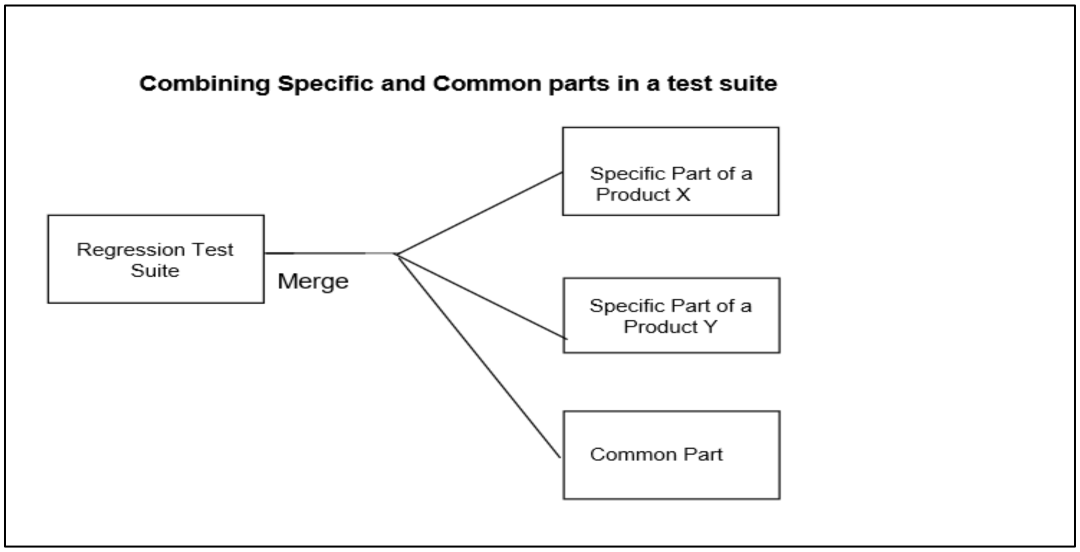

Combining Specific and Common parts in a test suite
Regression Test Suite
Merge
Specific Part of a Product X
Specific Part of a Product Y
Common Part

Fig 2.2- Reusable Test Case Technique

Pros:

- Reusable test cases can be used in succeeding regression cycles which will save the testing time.
- Reusable test cases can be used in multiple applications which have similar features.

b) Obsolete Test Cases:

These test cases are written for specific defects and should not be used repetitively in succeeding regression testing cycles. The good way to use them is when related fault occurs. Since it's not possible to re-execute all the test cases in large software project while performing regression testing, therefore, QA team should periodically clean up the regression test suite by removing obsolete test cases which are no longer valid e.g. to delete cases related to outdated requirements.

Pros:

- This approach can help a QA team to set correct priorities for test cases and utilize their time on executing the required test cases only.

Cons:

- If this approach is not applied carefully it can lead to the removal of some essential test cases and due to which many previous bugs cannot be detected.
- It can't be used in succeeding regression cycles.

2.3 Include All Complex Test Cases

A test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. A test case shall cover the complexity of a business logic so that the code is efficient to run on any device. In complex business logic situations, it is recommended to use decision table testing for regression testing which is an easy and confident approach to identify the test scenarios for complex business logic.

Few functionalities of a software system can only be accomplished by following some complex sequence of GUI events. For example, to open a file a user may have to click on the File Menu and then select the Open operation, and then use a dialog box to specify the file name, and then focus the application on the newly opened window. Due to increase in the number of possible operations the sequencing problem exponentially increases. This can become a serious issue even if one of the steps is not working then the whole functionality does not work. Hence all such complex test cases should be included in regression test suite.

Pros:

- Complex test cases that are designed using decision tables will help to detect various combinations of conditions that would otherwise not have been found and therefore not tested or developed.

Cons:

- In order to write complex test cases, decision tables are created first to clarify the complex business logic which is very time-consuming activity.

3 Our Approaches/Improvement Suggestions

Nowadays more with less has become has becomes a principle for most of the IT organizations. If there are more projects, more is the competitive pressures and greater failure risk which needs to be managed with less resources with tight schedule. Due to such constraints, there is no scope to do compromise on quality and stability of business applications in today’s competitive world.

Instead of risking late projects due to increased cost or low quality, we need to find the best approach to achieve better with less. The objective of software testing should be to reduce the risk of failure as well as ensuring the quality and stability of the business applications.

3.1 Categorization of Test Cases

In order to accomplish this objective, we should apply the principle of Risk Based Prioritization of tests, known as Risk-based testing (RBT). The goal of Risk Based testing approach is to make sure that appropriate testing activities are determined and prioritized based on risk. The basic role of risk-based testing is to optimize available resources and time without affecting the quality of the product. RBT approach mitigates the risk of failure in business and increases customer satisfaction.

Risk based testing helps to quantify and mitigate risks in the lifecycle of applications and prioritize tests more effectively. Using RBT approach, we can create Optimized Regression Test Suite based on Business Severity and Priority. The Success criteria for regression testing will be determined by the ability to identify high risk defects in software and ensure that they are fixed.

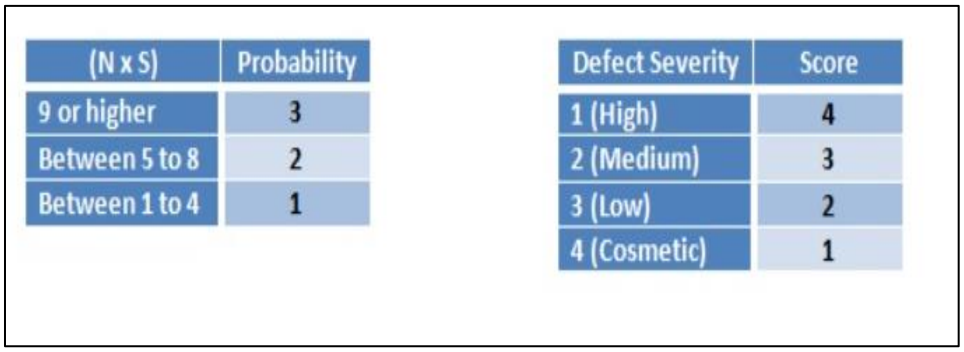
We must categorize the test cases during the inception of the project and must also be validated at the end in order to perform the regression testing. This categorization should be done on the basis of their risk exposure.

Supporting Proof:

We can calculate the risk exposure using below formula:

Risk Exposure (RE= R*P) = Requirements Risk (R) * Probability for Defect (P)

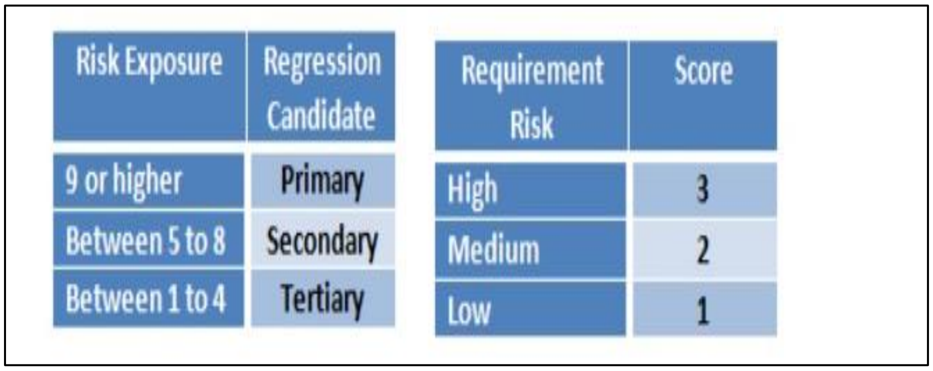
Probability for Defect (P) = Number of Defects (N) * Average Severity of Defects (S)



(N x S)	Probability
9 or higher	3
Between 5 to 8	2
Between 1 to 4	1

Defect Severity	Score
1 (High)	4
2 (Medium)	3
3 (Low)	2
4 (Cosmetic)	1

Fig 3.1- Probability of Defect (P) & Av. Severity of Defects (S)



Risk Exposure	Regression Candidate
9 or higher	Primary
Between 5 to 8	Secondary
Between 1 to 4	Tertiary

Requirement Risk	Score
High	3
Medium	2
Low	1

Fig 3.1.1- Risk exposure (RE) & Requirement Risk (R)

3.2 Automate the Regression Testing

In most of the organizations, software testing is carried out manually by creating test scripts with a testing tool based on the project dependence. However, when an organization practices a "test and trash" methodology, they do not re-use the test scripts often. This technique leaves little means to justify the cost of their initial creation, but it is worse, when the test script developer leaves, on-going support from others suffers as they learn the tool and script language to maintain the tests. Test automation is an investment that must be supported to have a return over time.

With the evolution of many open source testing tools such as Selenium, Appium, Cucumber, etc. we could easily implement regression testing for a software. Though Regression testing strategy fits best to overcome this issue, it is much and more effective to automate regression testing which helps us to save lots of time and cost. Automated regression testing has thus become a requirement for success in agile environment where the user specification keeps on getting updated all the time with every product release.

Due to constant update in the product specification, we are cornered to recheck the test cases for all the previously implemented functionalities. Without automated regression testing, an organization adds the amount of time and resources needed to manually smoke test and regression test the features that have been implemented to the new functionality in each release. The overall new release is initially tested and debugged to assure the quality of the new features. Then we add the new test cases to the list of regression testing necessary in the current and future releases.

We recommend following the below practices to successfully implement automated regression testing:

1. Creating the inventory of the workflow present in the application
2. Transform the workflow into test scripts
3. Maintain the version-control of the test scripts as software assets
4. Automate the operation of test scripts based on changes to the application
5. Use the test results as inputs to management's business risk analysis

Supporting Proof:

Testing Whiz:

This approach of automated regression testing is best carried out using software testing tool 'Testing Whiz'. Testing Whiz provides solutions to automate system integration testing that confirms that any system under test works in synchronization with associated systems and tools and exchanges data seamlessly after every integration or upgrade. An illustration of regression testing automation for a change in specification is carried out using a 5-process model:

1. **Gathering the changed item from the product backlog:**

   Based on the analysis of the current version of the product, the management decides to modify an existing item in the product backlog. Once the change has been implemented for that item, we mainly focus on testing that particular item in the product backlog.

2. **Updating the test cases:**

We shall specify test cases for that modified item according to the current functionality. We can also update the change scripts if necessary.

3. **Select Test cases:**

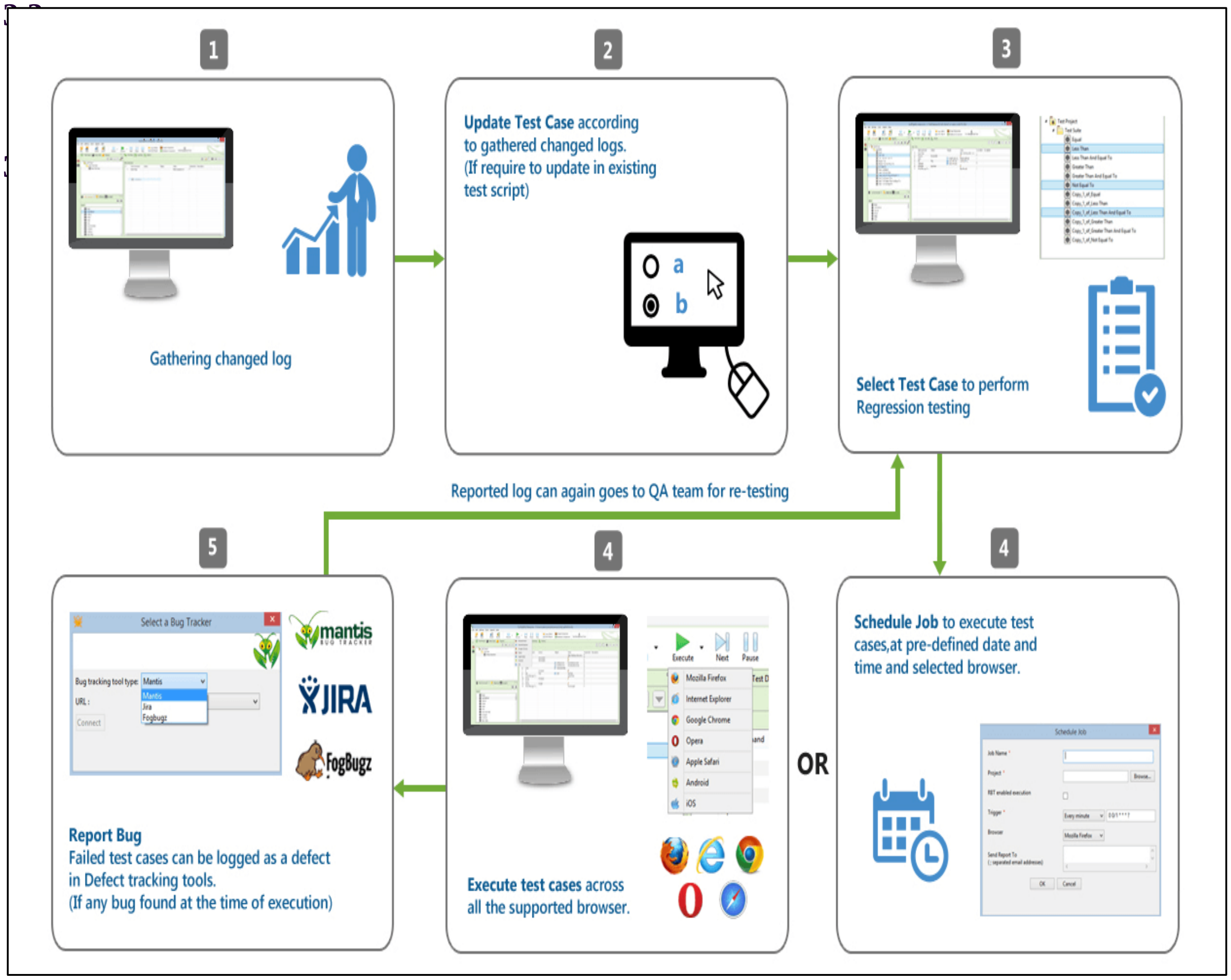
Testing Whiz gives us an option to select test cases to perform regression testing. Based on our proposed approach, the selected test cases should be prioritized, sorted according to frequently used functionality, check for failing test cases, consider core functionality test cases.

4. **Schedule the job:**

Testing Whiz allows a feature to run a background job which runs all the test cases. Background does not require memory partition as Testing Whiz tool automatically allocates memory for the background job. The user needs to schedule the background job based on his convenience. In general, background jobs are scheduled during the non-business hours of the organization, where there is a minimal CPU utilization. Testing Whiz tool triggers an email to the owner of the background job and generates the status of completion. The bugs in the test case execution will be intimated to the user as a detailed document in PDF format.

5. **Logging the defect:**

The bugs in the test case execution will be logged as defects on the Defect tracking tool such as JIRA, ServiceNow, Mantis or FrogBugz based on the organization. The bugs will be reported to the development team requesting to work on the issue. The overall process of Testing Whiz is shown in Figure as below:


1
Gathering changed log
2
Update Test Case according to gathered changed logs. (If require to update in existing test script)
3
Select Test Case to perform Regression testing
4
Schedule Job to execute test cases,at pre-defined date and time and selected browser.
OR
4
Execute test cases across all the supported browser.
Execute
Next
Pause
Mozilla Firefox
Internet Explorer
Google Chrome
Opera
Apple Safari
Android
iOS
5
Report Bug
Failed test cases can be logged as a defect in Defect tracking tools.
(If any bug found at the time of execution)
Select a Bug Tracker
Bug tracking tool type:
Mantis
Jira
Fogbugz
URL :
Connect
mantis
JIRA
FogBugz
Reported log can again goes to QA team for re-testing

4 Conclusion

The objective of software testing is not only to find a defect but also to help in fixing it. Typically, bug fixing causes additional bugs in the software. This the reason that is every software company performs regression testing. It plays an important role in a software project to retest the unchanged parts of an application.

Identifying regression test cases is a critical task and requires thorough knowledge on an application or product under test. Change impact analysis and a history of defects both play a major role in the identification and prioritization of test cases.

Three main reasons to perform regression testing are following:

- When developers fix a bug, add a new functionality, or modify an existing one, they always change a program code. Even a little change may result in a bunch of new bugs. Regression testing is required to identify those bugs.
- In case of changing operating environment, a software tester can reveal and localize undesirable side effects by performing of regression testing.
- Effective regression testing is a key to the successful integration testing. A software tester should keep in mind that new bugs and side effects may appear after fixing bugs that were found during regression testing. However, a high-quality software product cannot be built without complete and systematic regression testing.

Regression testing is to be performed many times during the development and testing phases. If this testing is not performed adequately all the previous testing effort will be wasted. Regression testing should be performed every time when a new function/feature is added to the application and when previously discovered software bugs need to be fixed.

The software testers often get confused on how much regression testing should be performed. It depends on the scope of the new added functionality/feature. If the scope of the fix for a feature is huge then the large application area would be affected and due to which the testing needs to be thoroughly conducted including all the application test cases. But this can be easily decided when the tester gets input from the software developer about the scope, nature and amount of change.

The goal of regression testing is to detect bugs that occur due to recent changes made to the software system. It is vital to detect and fix the bugs as soon as possible and make sure that the software works as per the requirements

While performing regression testing, the first priority of a software tester should be to identify the functions with high risk and the functions that are frequently used. After satisfying these two conditions, the other functionalities may be included in test suite. Regression should be performed using automation tools/frameworks instead of manually executing it. It is a good approach to automate the regression testing for high risks of a program. Automation regression testing for websites, desktops or mobile applications would reduce the completion time, cost and effort.

5 References

1. https://pdfs.semanticscholar.org/3b20/14862c6269b54c6d701bfb016e2dced2b9a8.pdf
2. https://www.softwaretestinghelp.com/regression-testing-tools-and-methods/
3. https://stackify.com/what-is-regression-testing/ - Challenges
4. http://sam.cs.lth.se/ExjobGetFile?id=707 white paper
5. https://www.testing-whiz.com/blog/4-ways-to-select-the-right-software-test-automation-tool (strategy to choose right automation software)
6. https://dzone.com/articles/5-exceptional-regression-testing-tools-excluding-s
7. http://www.dsrminc.com/whitepaper/automating_regression_testing_a_dsr_case_study.pdf
8. https://www.guru99.com/regression-testing.html
9. https://www.scnsoft.com/blog/regression-testing-in-agile-development
10. https://pdfs.semanticscholar.org/de18/137df9ddb5cfbfc3922e516524b7750f413c.pdf
11. https://search.proquest.com/openview/a0c1d16b9f54a39c7ae722d790fba334/1?pq-origsite=gscholar&cbl=1606379
12. https://ieeexplore-ieee-org.lib-proxy.fullerton.edu/document/8004401/ https://ieeexplore-ieee-org.lib-proxy.full
13. https://www-sciencedirect-com.lib-proxy.fullerton.edu/search/advanced?docId=10.1016/j.jss.2016.01.018
14. http://di gitalcommons.unl.edu/cgi/viewcontent.cgi?article=1031&context=csetechreports
15. https://www.sciencedirect.com/science/article/pii/S0950584916304888

16. https://dl.acm.org/citation.cfm?id=505468
17. https://www.capgemini.com/2017/01/best-practices-in-identifying-test-cases-for-regression-suite/
18. https://pdfs.semanticscholar.org/1601/3e34d8b1ff09c2271195c14fdf31557cb2ec.pdf
19. https://www.vtt.fi/inf/pdf/publications/2000/P406.pdf
20. https://pdfs.semanticscholar.org/c08b/eaa6e584e9ceb83e380edd84783cd2f59062.pdf
21. https://www.capgemini.com/2017/01/best-practices-in-identifying-test-cases-for-regression-suite/
22. https://www.360logica.com/blog/importance-regression-testing-software-development/
23. http://blog.qatestlab.com/2016/04/01/conduct-regression-testing/
24. http://blog.qatestlab.com/2013/10/01/how-and-when-to-perform-regression-testing/
25. http://thesai.org/Downloads/Volume8No2/Paper_39
Analytical_Review_on_Test_Cases_Prioritization_Techniques.pdf