UNDERSTANDING REGRESSION TESTING

INDEX

INTRODUCTION

The digital space is perhaps the fastest changing field in the world today. Needless to say that this necessitates frequent changes, modification, updates etc. in the software that drives the millions of apps that are in use world-wide. The reasons for change are manifold and include positive ones as well as negative ones. Among the positive reasons, there is technology improvement; need for superior user experience which gets demanding by the day; changes in configuration; change in the requirements of currently used features; additional features being added to stay ahead of competition or comply with statutory requirements; and source code improvements to enhance the app performance. Among the negative reasons is the need to add software patches to plug security issues that have cropped up; or the need to fix escaped bugs that have been detected during use.

Whether positive or negative, the frequent changes along with extremely crunched time, pose a double whammy for software developers and testers. From these extreme challenges were born improved tools and methodologies to address the issues faced in the Software Development Life Cycle (SDLC). Automation Testing is one such tool that came to the rescue of software testers, and Regression Testing is a concept and methodology that helps save valuable time in the SDLC, and simultaneously ensures better app quality. In simple terms, Regression Testing is performed to ensure that the new or modified changes in the code do not negatively affect the app's existing features. It becomes amply clear then, that Regression Testing has an important role to play during the app development and also beyond – when technological and other reasons necessitate changes in the app even long after it has been launched in the market.

This whitepaper is a treatise that will delve into the concept of Regression Testing and cover its various aspects.

ABSTRACT

The software world is driven by immense competition, fast-changing technology and ever-changing user preferences – all of which create an environment in which change becomes a way of life. But then, changes in software need to be carefully thought through because a single altered code can set off a chain reaction, since codes are interconnected. However, the time available to modify or add new codes is extremely limited and this is where Regression Testing gains immense significance.

This paper on **Understanding Regression Testing** is presented in two parts viz.

I) An Overview of Regression Testing and

ii) Regression Testing Process

The Overview of Regression Testing will give the reader insights into the following:

i. The Concept of Regression Testing

ii. Difference between Regression Testing and Re-testing

iii. Role of Regression Testing in the SDLC

iv. Challenges of Regression Testing

The Regression Testing Process section will touch on the following areas:

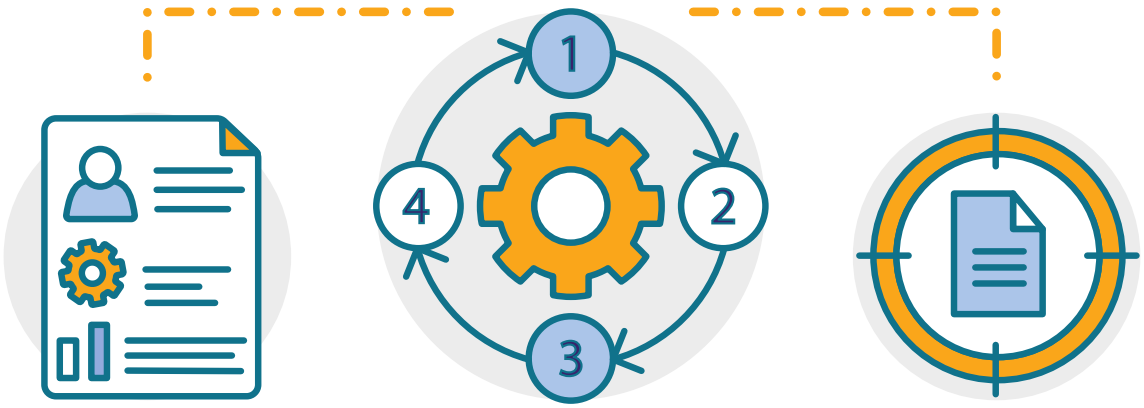
i. Steps for Regression Testing

ii. Regression Testing Techniques

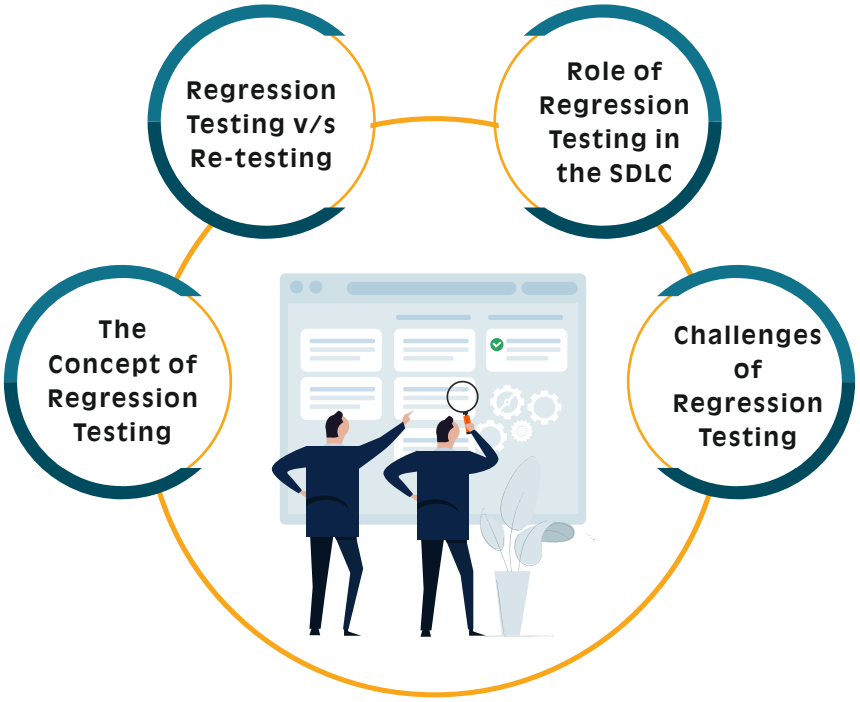
iii. Guidelines for Selection of Test Cases for Regression Testing

iv. Regression Testing Tools

Read on for a more detailed understanding of Regression Testing.



OVERVIEW OF REGRESSION TESTING



The Concept of Regression Testing

Regression Testing is a type of software testing that ensures that new codes or modifications to the existing codes, do not adversely affect the approved codes or program of the app under consideration. This involves a full or partial selection of executed test cases which are re-executed to verify that when they are merged with the existing software program, they perform as intended, and simultaneously combine well with the existing codes, so that the software as a whole functions as it should.

It is true that new codes, modifications to existing ones, or improvement in app features will be thoroughly tested on multiple fronts to ensure proper functionality before it is merged into the app. However, codes are inter-dependent and hence additions or modifications though individually tested and approved, may still create issues when integrated into the application. In fact it may sometimes be a negative catalyst and set off a series of functionality issues with multiple inter-dependent codes being adversely affected. Here is where the role of Regression Testing becomes vital, to ensure that the app's stability and functionality are not compromised when changes are made. Thus Regression Testing reduces the risk of modification failures in an app and ensures that it neither crashes nor throws up even the slightest glitch, so that the app as a whole works perfectly. Regression testing is thus the final gateway or litmus test for code integration.

Regression Testing v/s Re-testing

It is important at this point to clearly distinguish between Re-testing and Regression Testing. Re-testing is a functionality test that is re-done to confirm that the bug or defect is completely fixed and that the new or modified code works as it should. It is also done when a test case fails in its final execution and therefore has to be re-run until it has no defects. Thus re-testing generally precedes regression testing because it has to be done to ensure that failed tests are rectified before they are ready for integration.

Regression Testing on the other hand, involves testing of the software application, when tested new codes are integrated, to ensure that the software as a whole functions as it should. In other words, Regression testing is done to make sure that the modifications and updates do not create defects in the already approved functions and program. Thus Regression Testing will follow re-testing, since only tested and approved codes will be integrated, and these will be further tested to ensure that they do not affect other tested and approved codes.

Role of Regression Testing in the SDLC

Regression Testing by its very nature ensures that integration issues are detected early and reported to the relevant teams for early rectification. This greatly reduces cost of repair and modification. Since codes are integrated in installments, it is easier to detect and resolve the integration defects that show up. Regression Testing greatly supports Continuous Testing through its role in the Continuous Deployment process. Continuous deployment runs on the principal of sending small, frequent, thoroughly tested changes in the software to the production server. The functioning of the app needs to be scrutinized after every deployment and here is where Regression Testing plays an extremely significant role.

Regression testing is essential whenever the app goes through frequent changes, as it acts like a sieve through which new and modified codes blend into the app, ensuring that quality and performance are continuously kept in check.

Challenges of Regression Testing

It must be kept in mind that with every regression test, the test suites become larger and it becomes difficult, time consuming and expensive to execute the entire test suite every time-especially with time and money being of great essence. Hence there is a need to be able to correctly identify all the relevant test cases that are affected by the changes in code and execute only those, so that the right balance is struck between safety, security, time, money and efforts. Since codes are interconnected one small change can sometimes have a ripple effect across the app. Thus choosing all the relevant test cases becomes a major challenge. The next part of this treatise will deal with ways to counter this challenge.

REGRESSION TESTING PROCESS

The prerequisite for Regression Testing is that the new or modified codes are first tested to ensure their proper functionality, and that they are bug-free. Once this is ensured the Regression Testing process can be put into action. Given below are the broad guidelines for Regression Testing, although the actual process may differ in different organizations.

Steps for Regression Testing

01 Identifying and Analyzing Changes in the Source Code

The first step is to identify the changes made to the source code; then determine the modules that are affected by the change; and furthermore check their bearing on the existing features of the app.

02 Prioritizing Changes

The next step is to prioritize the identified modifications and app requirements, to structure the testing process with the relevant test cases and testing tools.

03 Defining the Entry Point and Conditions

Thereafter the entry point is to be decided based on the preset conditions that confirm that the app is ready for Regression Test execution.

04 Defining the Exit Point

The exit point is also to be defined w.r.t. the eligibility criteria decided in the previous step.

05 Scheduling Tests

Once all the test components are identified, the opportune time for execution can be scheduled.

Regression Testing Techniques

There are basically three Regression Testing Techniques which are detailed below:

01 Total or Complete Regression

Under this technique, all existing test suites are re-executed whenever there is a change or modification to any code. Needless to say that this is the surest way to ensure that all defects are identified and fixed, but from a practical standpoint it is too time consuming and expensive. Hence it is important to strike a proper balance and use this method only when the need is inevitable as in cases where the app has to be configured for a new platform or language, or when there is a major update in the operating system.

02 Regression Test Selection

In this technique only test cases that are expected to be affected by the code modifications are selected for regression testing. This technique is more economical in terms of costs and time as only relevant tests are selected for re-execution. It is helpful to group the test cases into two categories viz. Reusable Test cases which can be used in subsequent regression cycles; and Obsolete Tests which cannot be used in further cycles.

03 Prioritization of Test Cases

The third technique of Regression Testing is to prioritize and give preference to the most important test cases while performing Regression Testing. The priority is decided based on failure rate, impact on business, critical functionalities, frequency of use, impact on customer-orientation, and newly added functionalities.

Guidelines for Selection of Test Cases for Regression Testing

As seen earlier, it is not time and cost effective to perform Regression Testing on all the test cases and hence the next alternative is to meticulously determine which tests cases should be selected. A poor selection can result in escaped bugs which is again costly in terms of down time, repair cost, loss of app credibility, embarrassment and disappointed customers. Hence proper selection of test cases for Regression Testing is extremely important. Here is a set of guidelines on the kind of test cases that need attention in the Regression Testing process. Test selection should include the following:

- Test cases that have a high defect or failure rate
- Test cases that confirm the central or core features of the product
- Test cases pertaining to User-facing functionalities
- Test cases of functionalities that have frequent changes
- Test cases of functionalities that have been changed lately
- All integration test cases
- All complex test cases
- Boundary value test cases
- A sample from successful test cases
- A sample from failure test cases

Regression Testing Tools

In cases where there are recurrent changes in the app, the regression testing costs can increase substantially and manual execution increases time as well. This is not a viable option in the current crunched testing time availability and hence the more cost effective option in the long run, is to automate regression testing. The level of automation can be decided based on the number of re-usable test cases that can be reused in subsequent regression cycles.

Listed below are some of the more commonly used tools that help to automate Regression Testing:

01 Avo Assure

This is a codeless test automation tool that is independent of the technology used and can test end-to-end processes across web, desktop, mobile and ERP applications; mainframes; associated emulators, etc

02 Telerik Test Studio

Telerik Test Studio is an automated cross-browser testing platform for web, desktop and responsive applications, supporting functional UI, load and REST API testing.

03 testRigor

This is a tool that helps testers build and execute even complex tests in normal English and can be used for mobile, web and API app testing.

04 Selenium

Selenium is one of the better known open source tools used for automating web applications and is useful for browser-based regression testing.

05 Apache JMeter

This too is an open-source test automation software that can help support various kind of testing including load and performance tests – on a variety of applications, servers or protocols. It also provides a complete regression test suite for end users.

06 Quick Test Professional (QTP)

HP Quick Test Professional is automated software that automates functional as well as regression test cases using VB Script language. It is a data-driven, keyword based tool.

07 Rational Functional Tester (RFT)

RFT is IBM's Java tool for automating test cases – more so for automating regression test cases and it also integrates with Rational Test Manager. It helps in all kinds of testing including functional, regression, GUI, and data-driven testing, and applications (web-based, .Net, Java, Siebel, SAP), and others.

08 Watir

Watir is the acronym for Web Application Testing in Ruby and is an open-source library using the Ruby programming language. Its USP is that it supports user interaction capabilities for website testing e.g. clicking links, filling out forms, and validating texts on a variety of browsers.

09 Eggplant

This is an AI-driven test automation tool that organizes regression testing via prioritization of test cases and minimization of test maintenance and also supports self-healing functional testing.

10 BOTm

BOTm is a codeless, voice enabled, AI and ML driven Automated Testing tool which is a one stop testing platform for all types of mobile app testing – across spectrum. It also allows testers to create reusable test scripts which can be used for multiple handsets for regression testing purpose in SIT or UAT environments, and has many useful features like automator, replicator, regression suites, script editor, video recording and dashboards.

The choice of testing tools will depend on the type of app that is being tested i.e. Web based App, Mobile App, Desktop App, ERP App etc. A fully automated tool that is AI driven is the ideal choice as it will speed up the regression testing process while simultaneously ensuring safety and security and in turn – app quality.

Regression Testing can be a fruitful and extremely useful exercise when the right process, techniques and tools are employed. Time invested in the initial planning and choices will pay great dividends in the final app.

CONCLUSION

Regression Testing is a great time saver and very vital for the SDLC as it tremendously cuts down on the testing time, efforts and long term costs. By ensuring that new or modified codes do not adversely affect previously approved codes or programs, it provides better safety and confidence in the quality of the app as it progresses. Regression testing is also a great boon to apps that are already being used by end users, because it provides timely deployment and integration of new and modified codes.

In the modern software world where speed and accuracy are of utmost importance, it is important to automate regression testing in order to instill confidence in the testing process. Regression Testing tools are a great aid to testers and must be carefully chosen keeping in mind the needs of the app.

For Mobile App Testing, BOTm is an ideal tool as it provides the full gamut of testing requirements along with regression testing, and can be used across spectrum. Further more, being an AI and ML driven automated testing platform it makes testing secure, safe and quick. Experience these benefits first hand by visiting www.botmtesting.com and avail of a Free Trial.