

Final project for Manual Testing

SEMINAR PAPER

***“Regression testing as a way to stabilize the software application after the change or interfaced with other software”***

Fanka Shundovska

Skopje, December 2023

* “Testing is to Software as Experiment is to Science”[[1]](#footnote-1)

If epistemology is about the way we know things, ontology is about what things fundamentally are, and ontogeny is about the history of changes that preserve the integrity of something. What does this have to do with testing? A test / examination as evaluation/ - we detect it as measure knowledge, skill, or classification in many other topics, verbally, on paper or on computer, for the things that we have to know.

As we’ve explore the history of testing, we’ve found that the ancient Chinese government conducted standardized testing to help select candidates for government jobs. The earliest evidence of examinations in Europe date to 1215 or 1219 in Bologna. These were oral in the form of a question or answer, disputation, determination, defence, or public lecture. The imperial examination system was known to Europeans as early as 1570. The examination system was disseminated broadly in Europe. During the 18th century, the imperial examinations were often discussed in conjunction with Confucianism, which attracted great attention from contemporary European thinkers such as Gottfried Wilhelm Leibniz, Voltaire, Montesquieu, Baron d'Holbach, Johann Wolfgang von Goethe, and Friedrich Schiller. Evidence of written examinations do not appear until 1702 at Trinity College, Cambridge. Germany implemented the examination system around 1800. In 1806, England adopted this method for its civil service positions as well, although standardized testing did not make it to universities until the 1850s. Like the British, the development of the French and American civil service was influenced by the Chinese system.[[2]](#footnote-2)

In 1935, IBM introduced computerized scoring in an effort to produce more reliable results. In the 1970s and 1980s, computers moved into other areas of testing beyond scoring. They were used in the design, construction, delivery, reporting of results and more. In the mid-1980s, computer-based testing was introduced simply as an electronic version of the traditional pencil-and-paper method. Many institutions and certifying bodies used online proctoring as the next step in web-based testing .[[3]](#footnote-3)

* What is the history of regression testing?

The evolution of testing was to make the exam process more convenient and comfortable while still protecting the integrity of our certifications. The term "regression" was coined by Francis Galton in the 19th century to describe a biological phenomenon. The phenomenon was that the heights of descendants of tall ancestors tend to regress down towards a normal average (a phenomenon also known as regression toward the mean). Regression testing means rerunning test cases from existing test suites to build confidence that software changes have no unintended side-effects. The “ideal” process would be to create an extensive test suite and run it after each and every change. Unfortunately, for many projects this is just impossible because test suites are too large, because changes come in too fast, because humans are in the testing loop or because testing must be done on many different hardware and OS platforms. [[4]](#footnote-4)

Researchers have tried to make regression testing more effective and efficient by developing regression test selection techniques. Developers are using Regression Test Selection techniques, because RTS try to maximize average regression testing performance rather than optimize aggregate performance over many testing sessions. Regression testing is an expensive and frequently executed maintenance process used to revalidate modified software. To improve it, regression test selection (RTS) techniques strive to lower costs without overly reducing effectiveness by carefully selecting a subset of the test suite. In practice, testers work around this by prioritizing the test cases and running only those that fit within existing constraints. [[5]](#footnote-5)

* What is Regression Testing?

Regression testing is a type of software testing applied to code immediately after a code update to ensure that the update introduced no new bugs, as effective way of monitoring the effects of change. This is because new code may bring in new logic that conflicts with the existing code, leading to defects. The goal is to assure that the changes have not unintended consequences on the behaviour of the test object. QA teams have a series of regression test cases for important features that they will re-execute each time these code changes occur to save time and maximize test efficiency. If the project does not have a version control system, it can be tricky to locate the exact component that produces the bug. With regression testing, we know exactly where the bug comes from, enabling better troubleshooting. It is essentially a periodic health check-up for the software. Due to its highly repetitive nature, regression testing is a great candidate for automation testing.

In a software development environment which tends to use black box components from a third party, performing regression testing can be tricky, as any change in the third-party component may interfere with the rest of the system.

* Good and bad reasons for using Regression Testing and benefits

If we analyse, the good reasons for using Regression Testing is that bug fixes often break other things the developer isn’t concentrating on. Sometime bug fixes don’t fix the bug. Checking software with Regression testing still runs after making a change in the infrastructure. It enable in discovering faulty localisation, to detect errors in the build process (wrong parameters) and it conforming to standards or regulations.

Regression testing is performed when changes are made to the existing functionality of the software or if there is a bug fix in the software. Regression testing can be achieved through multiple approaches; if a *test all* approach is followed, it provides certainty that the changes made to the software have not affected the existing functionalities, which are unaltered.

The bad reasons while using Regression testing are the arguments in terms of replicability of results and in terms of quality in analogy with a production line (manufacturing analogy).

* Risks of change

The risks of change are:

- bug regression testing while checking that a bug fix has removed the symptoms of the bug that have been identified

- Old fix regression while checks that a new fix has not broken an old fix: refactoring should limit this as old fixes are refactored into the code

- Incremental regression testing as we develop

- Localisation testing to test if a product has been correctly localised for particular market

- build testing has an error been introduced in the field that means the system will not build correctly

* Examples of Regression Testing

Regression testing is a practice and method. There isn’t a testing tool for regression testing. Any test that was developed at the initial launch of a certain feature, brought up to run at later releases, is already a regression test. You can have regression tests by application attribute and type:

Attribute: visual, functional, performance, security

Application layer: UI and API

Application type: web, mobile, API and desktop

Granularity: end-to-end, integration and unit tests (test pyramid)

To perform regular check-ups on existing code, regression testing basically involves:

* Testers writing and building automated test suites

Testers and business analysts filtering out release-specific tests to re-run on affected areas from new releases. After the new feature is written, regression testing is run to make sure that the new recommendation feature does not affect the existing feature. The test plan might involve running manual or automated regression tests to check the functionality of the existing code and if there are any conflicts between 2 versions. If any bugs are discovered, they are fixed and the regression test is run again until all tests pass.

* Why is regression testing important in Agile?

Sometimes a mild modification can cause a domino effect on the product’s key functions that will take tons of effort to reverse. Practicing regression testing aligns with the Agile testing methodology in continuously iterating, integrating and testing new code. Further analysis can also alter the scope to pinpoint optimization opportunities. Regression testing is not strictly about functionality. Visual regression testing is employed to detect visual bugs that may occur due to changes in the codebase. Instead of manually reviewing every single UI across hundreds of devices and browsers, QA professionals can simply execute a suite of automated regression tests to locate those visual bugs. If the product undergoes frequent modification, regression testing will be the filter that ensures quality as the product is improved. Regression testing is applied under these circumstances:

A new requirement is added to an existing feature

A new feature or functionality is added

The codebase is fixed to solve defects

The source code is optimized to improve performance

Patch fixes are added

A new version of the software is released

When changes to the User Interface are made

Changes in configuration

A new third-party system is integrated with the current system

All of these occasions involve a restructuring or adjustment of the current code, which may result in unexpected behaviours, hence the need for regression testing. With test driven development, each new feature should come out with its own set of tests. There are always several crucial steps that should be followed.

* *Regression testing steps:*

1. Detect source code changes

Identifying the impact and risk of the latest code change is key to building a solid regression test.

2. Prioritize impacted areas and test cases

QA teams discuss which changes should be extensively tested and which could bear with adequate testing. Modifications that have an impact on core features, or those that significantly alter how the application works, should always be the top priority. Defining priorities heightens in importance when the size of the codebase is bigger. The number of tests and time they take to complete could take up to months or an entire sprint.

3. Determine Entry Point and Exit Point

Once the team agrees on which changes should be examined, they can select the tests that need to be performed, have to execute only the relevant ones in each regression testing session.

4. Categorize regression test cases

In this step, we deep-dive into the plan formulated in step 3 and categorize them by various factors. A few common test categorization criteria includes: Manual vs automated regression tests.

5. Test Environment Preparation

As new code are being developed constantly, environments need to be stable and ready-to-test to not interfere with the planned testing schedule. In additional, poor environment setup can give tests more reason to fail, missed defects and false positive/negatives.

6. Schedule and Execute Tests

At this stage, all test cases are ready for execution. Teams can schedule test cases to run based on the plan.

7. Measure regression testing success

This stage gives important insights for future test runs. Test reports can reveal weak points in the application for in-time adjustments for the development team.

* *Top Regression Testing Tools:*

1. Katalon Platform

Katalon Platform is a comprehensive end-to-end AI-augmented automation testing platform that can take your regression testing to the next level. It is the all-in-one regression testing tool for the website, web services, desktop application, mobile application and even API testing. Katalon Platform also supports running scripts on a wide range of devices, browsers, and testing environments, allowing QA teams to perform a lot of testing activities in just 1 place instead of having to spend time configuring environments and constantly switching tools.

- StudioAssist: Leverages ChatGPT to autonomously generate test scripts from a plain language input and quickly explains test scripts for all stakeholders to understand. Manual test case generator: Integrates with JIRA, reads the ticket’s description, extracts relevant information about software testing requirements, and outputs a set of comprehensive manual test cases tailored to the described test scenario.

- SmartWait: Automatically waits until all necessary elements are present on screen before continuing with the test. Self-healing: Automatically fixes broken element locators and uses those new locators in following test runs, reducing maintenance overhead.

2. Selenium

It remains among the top open-source solutions for browser-based and cross-platform regression testing. Automation testing scripts written with Selenium have to be continuously revised as code changes occur, which is time-consuming.

3. Watir

Watir, or Web Application Testing in Ruby, is an open-source library using the Ruby programming language, eliminating the need for an external server. Watir allows users to easily create code without needing to read extensive documentation.

4. IBM Rational Functional Tester

Rational Functional Tester, or RFT, is a tool for software test automation from IBM. It offers automated testing for functional, regression, GUI, and data-driven testing, and is compatible with web-based, .NET, Java, Siebel, SAP, terminal emulator-based applications, and PowerBuilder. It provides a recorder of user actions, multiple customization options, script maintenance capabilities, and the ability to share functional tests with other team members and run them on hybrid environments.

5. Apache JMeter

Apache JMeter is an open-source Java application for testing load, performance, and functional behaviour of web applications. It has been expanded to include testing other functions, such as the efficiency and concurrent user request handling of a server. Can be run on any environment that supports a Java virtual machine, including Windows, Linux, and Mac. It is a great tool for functional performance and regression testing on different technologies.

* *Regression Testing Techniques*

Regression testing has three most prominent implementation methods, including retesting all, regression test selection, and test case prioritization.

1. Retest All

In this technique, regression testing is applied to all existing test suites. It is easy to see that although it is the safest way to ensure all bugs are detected and resolved, this method requires substantial time and resources. That is why the complete regression approach fits better in certain contexts

2. Regression Test Selection

With this approach, QA teams may select the relevant parts that can be affected by the changes and perform regression testing on these chosen parts only. By picking out the related areas, you can apply limited and relevant test cases to reduce both the time and effort invested in regression testing. This approach is suitable for more complex or large-scale applications, whose number of test scripts to be executed is higher.

3. Test Case Prioritization

This approach is essentially prioritizing test cases that must be included and performed first in the regression testing process. These test cases should be prioritized based on several criteria:

- Failure rate

- Business impact

- Gradually used functionalities

- Test cases to check the functionality of new features

- Customer-centric features

- Security-related features

There may be many other aspects to consider depending on the specifics of your business and organizations. It is important to understand how business requirements translates into the features of your application for better decision making.

4. Corrective Regression Testing

Corrective regression testing is simply re-running all current test cases when no code change has been made, yet. This practice is to double-check on whether the current code functions fine, and whether we can reuse the existing test cases or not. If the test results are positive, QA teams can be confident that their test cases are up-to-date. At this stage, testers can start the test planning and prioritization process while the new code is added.

5. Progressive Regression Testing

With progressive regression testing, testers acknowledge that changes to the code may call for changes in the test suites themselves, so they update the test scripts to align with the new requirements. This approach is employed when there is a change that impacts product vision.

* Retesting vs Regression Testing

The two terms retesting and regression testing can be confusing for test automation novices. They might sound similar but are in fact entirely different from each other.

Retesting literally means “test again” for a specific reason. Retesting takes place when a defect in the source code is fixed or when a particular test case fails in the final execution and needs to be re-run. It is done to confirm that the defect has actually been fixed and no new bug surfaces from it.

Regression testing is performed to find out whether the updates or changes had caused new defects in the existing functions. This step would ensure the unification of the software.

In a typical software development pipeline, retesting is performed before regression testing practices. Retesting solely focuses on the failed test cases while regression testing is applied to those that have passed, in order to check for unexpected new bugs. Another important note is that retesting includes error verifications, in contrast to regression testing, which includes error localization. Moreover, automation is a crucial feature in regression testing, allowing you to make the most of your test case capabilities. Additionally, regression testing eliminates all underlying side effects caused by code changes in the most cost-effective way possible.

* Agile Regression Testing

With the Agile development approach, teams can gain numerous benefits and values, such as accelerated time-to-market, customer support, and product improvements. This comes with a significant challenge of balancing between sprint development and iterative testing to avoid conflicts as the product matures. Agile implementation of regression testing plays a key role in aligning the existing and updated functionalities, avoiding all possible rework in the future.

Agile regression testing ensures the business functions are stable and sustainable. Regression testing helps developers to focus their efforts on building new functionalities for the application rather than keep on returning to check for defects in the old features. Applying regression testing reveals the unexpected risks in software builds, helping developers to respond more quickly and efficiently.

In agile software development—where the software development life cycles are very short, resources are scarce, and changes to the software are very frequent—regression testing might introduce a lot of unnecessary overhead.

* Final Conclusion

Regression testing is key to improving the overall quality of the product and user experience and a tools in helping to provide stability in the face of code change. Regression testing provides a tools for managing change. The right regression testing tools can be used throughout the lifecycle and significantly identify all surfaced defects and eliminate them early in the pipeline. In addition, regression testing in Agile offers a host of technical and business advantages. Standards and regulation often require regression testing. Therefore, the more your organization invests in planning and performing regression testing, the more control you will have over the budget, process, and error mitigation of your product. It can reduce the cost of applying tests (by storing expected result).[[6]](#footnote-6)

Regression testing is a crucial process for stabilizing software applications after changes or interfacing with other software. It involves previously conducted tests to ensure that new updates or integrations have not adversely impacted the existing functionality. By doing so, regression testing helps to identify and fix any issues that may arise due to changes in the software.

*References:*

Online Database:

- Prioritizing test cases for regression testing - university of Nebraska ... Accessed December 24, 2023. <https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1017&context=csearticles>

- Regression Testing - University of Texas at Arlington. Accessed December 24, 2023. <https://heracleia.uta.edu/~sharifara/5321/9_Regression-testing.pdf>

- (PDF) an overview of regression testing - researchgate. Accessed December 24, 2023. <https://www.researchgate.net/publication/321243328_An_Overview_of_Regression_Testing>

- Regression testing - school of informatics, University of Edinburgh. Accessed December 24, 2023. <https://www.inf.ed.ac.uk/teaching/courses/st/2011-12/Resource-folder/11_regression.pdf>

Websites:

1 - Andrew. (2015, April 30). *Testing is to software as experiment is to science*. Andrew Bancroft. <https://www.andrewcbancroft.com/2015/04/29/testing-is-to-software-as-experiment-is-to-science/>

2- Wikimedia Foundation. (2023, December 21). *Exam*. Wikipedia. https://en.wikipedia.org/wiki/Exam <https://en.wikipedia.org/wiki/Exam>

3 - McCraw, Debra B. “A Brief History of Testing: We’ve Come a Long Way!” CompTIA, October 6, 2023. <https://www.comptia.org/blog/a-brief-history-of-testing-we-ve-come-a-long-way!#myself>

4 - Regression testing. Accessed December 24, 2023.

<https://www.cs.umd.edu/users/aporter/html/currTesting.html>

5 - Katalon. “What Is Regression Testing? Definition, Tools, Examples.” katalon.com, August 3, 2023. https://katalon.com/resources-center/blog/regression-testing. <https://katalon.com/resources-center/blog/regression-testing>

6 - A history-based test prioritization technique for ... - IEEE xplore. Accessed December 24, 2023. <https://ieeexplore.ieee.org/document/1007961>

7 - Regression testing. Accessed December 24, 2023. <https://www.cs.umd.edu/users/aporter/html/currTesting.html>

1. Andrew. (2015, April 30). *Testing is to software as experiment is to science*. Andrew Bancroft. https://www.andrewcbancroft.com/2015/04/29/testing-is-to-software-as-experiment-is-to-science/ <https://www.andrewcbancroft.com/2015/04/29/testing-is-to-software-as-experiment-is-to-science/> [↑](#footnote-ref-1)
2. Wikimedia Foundation. (2023, December 21). *Exam*. Wikipedia. https://en.wikipedia.org/wiki/Exam <https://en.wikipedia.org/wiki/Exam> [↑](#footnote-ref-2)
3. Wikimedia Foundation. (2023, December 21). *Exam*. Wikipedia. https://en.wikipedia.org/wiki/Exam <https://en.wikipedia.org/wiki/Exam> [↑](#footnote-ref-3)
4. Regression testing. Accessed December 24, 2023. ttps://www.cs.umd.edu/users/aporter/html/currTesting.html. <https://www.cs.umd.edu/users/aporter/html/currTesting.html> [↑](#footnote-ref-4)
5. Katalon. “What Is Regression Testing? Definition, Tools, Examples.” katalon.com, August 3, 2023. https://katalon.com/resources-center/blog/regression-testing. <https://katalon.com/resources-center/blog/regression-testing> [↑](#footnote-ref-5)
6. Efficient regression testing based on test history: An ... - IEEE xplore. Accessed December 24, 2023. https://ieeexplore.ieee.org/document/7332496/. <https://ieeexplore.ieee.org/document/7332496> [↑](#footnote-ref-6)