Qentelli
imagine { } automate

REGRESSION TESTING

Definition | Methods | Tools | Best Practices

Sridhar Jayaraman
VP of Engineering, Qentelli

Summary

Software Release in 21st century is like rowing a boat. Fixing about 80% of the holes isn't good enough for the ride. Testing is a significant part of Software development today. It's almost impossible to survive through the turbulence of sudden overlooked Regressions. That is exactly why the teams started testing for the potential hiccups or misroutes each time they develop something new. But,

01 How often do you perform Regression Testing on your code-base?

02 What is the recommended frequency of Regression Testing?

03 How to ensure Test Coverage while picking the Test suits?

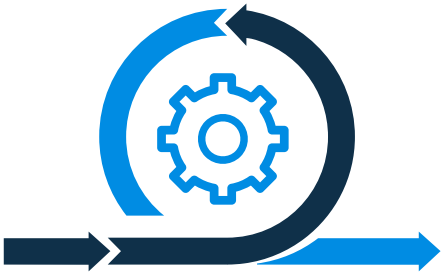
04 What is the correct technique for Regression Testing?

This e-Book intends to answer all these questions and more about Regression Testing.

Regression Testing - 101

As Software Development moved from Waterfall to Agile, the one aspect that encountered difficulties to cope up was Testing. When fast-paced Agile and DevOps kicked in, it is even more stressful for the businesses to deal with the speed vs quality war. In a generation where it is important to keep the customer's experience spot on to stay relevant; companies can't afford to have software regressions. That's how it became a practice to run two kinds of tests each time there is a significant change - when the Testers started doing selective retesting to detect faults introduced during modification of a system or software component.

> "... a testing process which is applied each time after a software component is modified"



The software applications are growing in terms of their sizes and complexities. Regression Testing is to make sure that the modifications have not caused any unintended adverse effects. Also, to verify that a modified system or software component still meets its specified requirements. The new functionality must work fine with the rest of the system. So, all the previous test cases would be re-run to ensure the application is still doing what it's supposed to do and there are no cracks because of the recent change.

How to decide the right test cases for Regression Suite?

Regression Testing has been a part of SDLC for ages, it almost feels like there is not much to learn more about it. But it is a tricky area that is often set aside, and many organizations are challenged to deal with it during strategic Test planning. The objective of regression testing is to uncover broken functionality that may have caused by the modification on existing code base, hence it is recommended to identify core functionality test cases from the existing Test Bank and based on the below factors:

- Test cases that cover critical/core functionalities
- This test case is part of the functionality that is directly impacted by the change performed
- Test cases that have frequently generated bugs and defects
- Test cases part of the functionality which is visible to users
- All the Complex and Integration Test Cases
- Boundary Value Test Cases

Categorizing the Test Cases:

RE-USABLE TEST CASES

These are the Test cases that can be repetitively used in succeeding regression cycles.

Automating these kind of test cases comes handy during executing new builds.

OBSOLETE TEST CASES

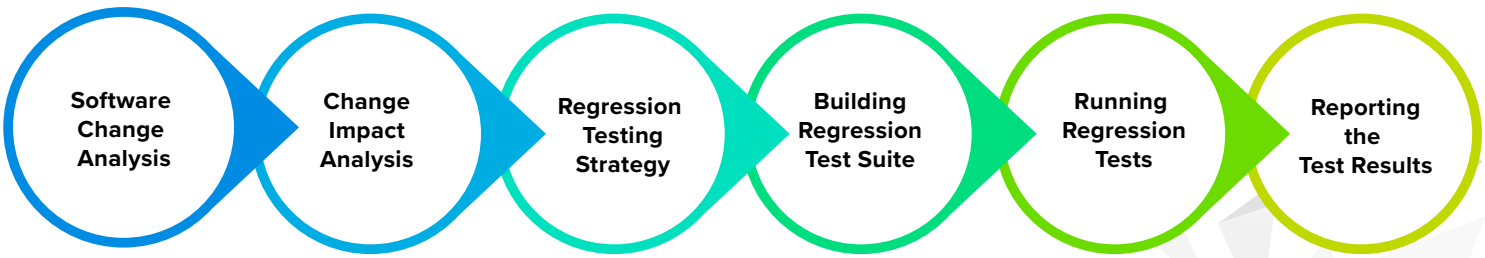
These are bug-specific Test cases that cannot be used in succeeding regression cycles.

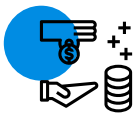
Automating these kind of test cases is almost useless.

They can only be used when the respective bug occurred.

Classifying the Regression Test Cases based on the recurrence of the defects or risk exposure is important. This division must happen right at the beginning of the project and verified at the closure. A detailed change impact analysis and bug history can help the core teams to identify better and efficient test cases.

A typical progression of Software Regression Test would look like this:





Automating Regression Tests: Benefits and Challenges

Test often, Test smart!

Nearly every test case that is written under Release Specific Test is expected to be a part of Regression Suite, which makes it broader with each new release consisting of newer versions of code. Manually testing all those cases can be daunting, especially to align them with tight time constraints can force testers to ignore certain sets. Test cases that cover core functionalities stay same throughout the continuance of the application until they are replaced with new functionality. So, automating test cases like these can save good amount of time.

Things to remember before Automating Tests:

DESIGN TEST CASES	Always design test cases with detailed scenarios before automating the tests. That can help in identifying the defects and improve the scope of the test case.
REMOVE UNCERTAINTY TO KEEP IT LEAN	If a test passes in one run and fails in the next with no changes done to the script, it indicates a flaw in test environment or in the test code itself. It is important to analyse the results and stay consistent.
ALWAYS REVIEW FOR VALIDITY	As the new features are added and the old ones are taken down, running test cases related to an old or unstable functionality is useless and often misleading. It's always important to check automated test cases to ensure they are current.
AIM FOR FASTER FEEDBACK	To enable a quick feedback loop, the tests need to be automated at component or API layer without depending on the UI. Activate loops for App performance, log management, error tracking, dev tracking and code profiling – depending on your app requirement.

Having said that, observations from many practices revealed - automating tests without really understanding the context can only make it aimless. Applications with multiple layers (Unit, API, Service, GUI, etc.) has to be tested according to that layer's purpose. Considering the complexity of applications, it is next to impossible to achieve 100% test code coverage. The major business cases, scenarios, and subsets of the inter-related ones shall be automated. But the cases that are complex in nature, that needs downstream system checking are better left for Manual Testing.

Also, if the QA practices aren't strong enough, adding automation to chaos will only result in Automated Chaos!

Automating Regression Tests

ADVANTAGES	CHALLENGES
ROI	Coping with the requirement changes
24*7 Availability	Tool and Framework Selection
Early Bug Detection	Test Coverages
Proven Reliability	Scaling of application with each iteration`
Continuity and Transparency	Recognizing Automation requirements

How much of Exploratory Testing should be included?

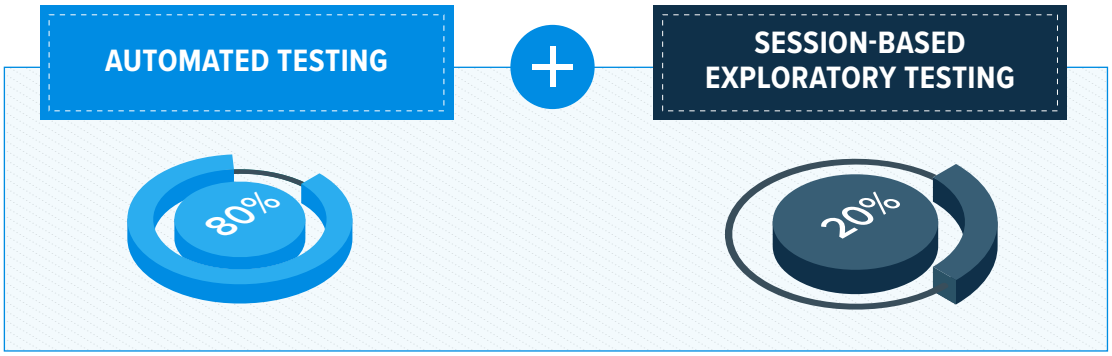
Introducing Test Automation doesn't mean it is an end to Manual Testing but is a scope for the traditional resources to re-skill themselves and make the most of general intelligence. When developers make too many changes to the Software too frequently, it often leads to instability of the software, higher test failures, a lot of bugs and unknown defects. Weaving an exploratory testing phase into the regression testing process can greatly improve the quality of the software. On many occasions, a human can catch an unknown issue, even if there is no test case indicating the existence of such issue.

Things to remember before Automating Tests:

> "Because Automation can't think, predict, suggest, refine or collaborate"

It is beneficial to perform exploratory testing before every major release. This approach usually reveals failures that scripted tests may not discover, improves QA coverage, and even indicate points to improve test suite, that would significantly alter the end-result of the software. Practitioners say that Exploratory Testing isn't a methodology to follow but a mindset of testers that can really take the game of Testing a few notches up.

An ideal Agile Testing Strategy (to achieve maximum coverage) includes:



To make the most of Exploratory Testing:

- Divide the application into modules and further divide them into various pages. Start testing from the pages and go upwards. That will maximize the scope.
- Maintain a checklist of the characteristics and Bug taxonomy.
- Initiate with one essential situation and then steadily improve it to add on some extra aspects to test it.
- Test all the response areas and negative situations for error messages.
- Check GUI (Graphical User Interface) against the setup standards.
- Check for the complex business rationales.

Let us take an example

Consider a simple, yet common application – an online shopping portal. During the purchase process, the application validates the address provided by the user. On the surface, this validation is independent of the payment type used (Credit/Debit/Bank Transfer, etc). However, there is a bug in the code for payment through a gift card (address validation is not performed!).

Because gift cards are not commonly available in Test Environments and it was considered that payment methods don't have any impact on address validation, the bug was deemed low priority. Only later when the bug surfaced in production, it was found that the address validation code was an older version causing another defect.

This type of "masked defect", where one defect hides another can cause cascading issues due to dependencies in the software design. These types of issues can be problematic for the application development process. In cases where defects alter the behaviour to such a degree that it is effectively hidden from the developers, it's common for test cases to fail to notice these issues. Hence, it is recommended to perform a combination of Automated regression and Exploratory testing in every major release.

What should be the frequency of Regression Testing?

Regression Testing is usually performed after any changes are made to the code base. The frequency of regression testing may vary from project to project based on the complexity of the software and/or change made to the system.

Regression Testing should be performed whenever:

- A new release of the packaged software has been put out to production
- The application is enhanced or revised for any reason
- The software is being integrated with another product
- There are changes to the configuration or changes are made after a testing stage is completed
- Of course, bugs!

It is recommended to perform core Regression Testing, nightly or alternate days and complete regression suite execution, once a week.

In short,

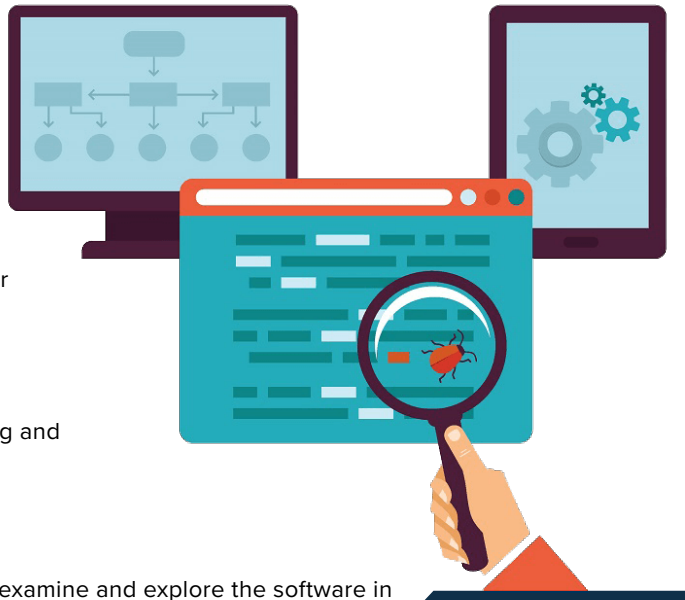
> "Regression Test must be performed as frequent as the changes in the application software"

Industry Best Practices in Regression Testing

Though it feels like a costly, time-consuming and somewhat nagging practice, Regression Testing is a very crucial and inescapable part of modern software testing life cycle. Setting up a strategy can strengthen the objective of the testing exercises. The strategy must understand the product, fully cover the product needs and assure product quality at an optimal cost. The practices mentioned below may help to make the most of Regression Testing practices.

Automation

Automate as many test cases as possible and eliminate effort for repetitive validation. Complement this with validation from a Business Expert to ensure the automation process did not miss any operational scenarios.



Analyze and Prioritize

Analyze the Regression Testing requirements and identify test cases that are absolutely required in final validation of the software, then prioritize them based on functionality and criticality. This way, the teams can stay prepared for the impact of sudden changes to the applications.

Test Data Management

The best way to gain most profitable ROI. Ensure you have a way of generating and cleaning Test Data for the suite.

Exploratory Testing

Exploratory testing empowers individual Tester, responsibility and freedom to examine and explore the software in various ways that are not documented.

About Qentelli

Qentelli, a technology company based out of Dallas, TX accelerates digital transformation and cloud transformation journeys through the implementation of DevOps, Automation, Agile transformation, AI and Deep Learning. Qentelli is known for its tailored orchestrated engineering efforts to safeguard companies while they commence their transformational journey. Forrester recently recognized Qentelli's efforts at using AI and ML in augmenting human testers.

Author

Sridhar Jayaraman
VP of Engineering, Qentelli

Sridhar Jayaraman is a VP of Engineering at Qentelli. With experience in Delivery, Consulting and Solution Engineering, he helps potential and existing clients to solve their business problems through an Engineering mindset.