HOW **RISK-BASED TESTING** OVERTAKES **EXHAUSTIVE TESTING** IN PRESENT INDUSTRIAL SCENARIO

Risk-based testing gains prominence in today's fast paced world where the pressure to roll out the software is the order, and there is virtually no time to conduct extensive all-encompassing tests to cover all functionalities. Testing is a process motivated by RISKS, and risk-based testing supplements this fact well with prioritizing risks and negating them on time. Read on to know how.

INDEX

WHAT IS RISK-BASED TESTING?

There are a number of possible tests that can be done on the features and functions of any software. But it would take great amount of time and resources to complete all such tests. Project timescales do not allow for extensive tests, especially in today's hyper competitive and fast paced business environment, marked by short shelf-life, where speed is of critical essence. The onus is on the testing team to select a finite number of tests that could be accomplished within the available time.

This is where risk-based testing helps. **Risk-based testing** pays considerable attention to balancing risks with budget, time-schedules, quality, and feature-sets. It tests only those functionality, which have the highest probability of failure, and therefore have more impact. This kind of testing negates any undesirable event that could possibly lead to the product or system failing to deliver on expectations.

METHODOLOGY

The **underlying principle** behind risk-based testing is to test software selectively rather than exhaustively. It prioritizes the various available tests based on the importance of the functionality tested in the overall scheme of things, and the severity of impact in the event of failure of the functionality.

The **risk-based approach** to testing focuses on unearthing bugs that would leave a negative impact on the business. These bugs may be further prioritized depending on the magnitude of their impact. For instance, bugs that lead to customers being unable to work with the software, and cause them to lose time and money would be ranked as the most severe, and are given high priority. Test for bugs where customers can still work with the software, albeit with a minor inconvenience, would be given relatively lower priority, and would be undertaken only if time and budget permits; ideally, after testing for all the high impact bugs.

Risk-based testing provides insights on the **test strategies** to be used. With this approach, testers evaluate the existing level of system quality and risk, select the most relevant test objects and test cases, execute the tests, and report test results related to the evaluated risk.

A major misconception prevalent in the software industry is equating software quality as a function of testing. Contrary to the prevailing assumption, testing an application extensively neither renders more stability, nor does it provide any added value to the customer. Testing is not only related to stability, but it also ensures the proper functioning of the software as per the user requirements. Risk-based approach to testing does away with this misconception and brings the focus of testing to the core functionality or what the software is supposed to deliver, rather than its technical merits.

ADVANTAGES OF RISK-BASED TESTING

Risk-based testing might seem inadequate or inferior to the conventional comprehensive testing, for it does not cover all bases. But this is far from the truth. Risk-based testing offers several advantages and benefits over conventional exhaustive testing.

How risk-based testing enables you in smart decision making?

The direct and most apparent **benefit of risk-based testing** over exhaustive testing is considerable savings on time, money, and effort, and improving the efficiency of the tests. With risk-based testing, the testing team may optimize use of their time, making the best use of the resources. It also allows greater control over the testing process, especially providing a good basis to define when to stop testing. It also offers a method to prioritize tests against deadlines.

Prioritizing the tests to be done, as long as it is done correctly, offers the high likelihood of detecting defects in the order of their severity and provides the ability to avert catastrophe. This allows testers to spend maximum effort on finding the formidable problems first. Organizations would be aware of the residual level of risks, and can make smart release decisions accordingly.

Is exhaustive testing a misnomer?

Exhaustive testing is all-encompassing, and entails subjecting the software to extensive tests to deliver a product that fulfils the laid down specifications. However, in today's highly competitive and fast-paced economy with short shelf-lives for software products, spending time and resources on such extensive testing may be unnecessary and counter-productive. Risk-based testing allows delivering the project to market faster, and keeps the testing budget under control.

While exhaustive testing sounds good in theory, risk-based approach to testing is better when considering the practical situation on the ground. It ensures intelligent effort allocation within practical constraints.

The term 'exhaustive testing' is a misnomer anyway, for rarely would anyone have enough time to run the entire gamut of tests, to cover all possible eventualities. Testers have always used risk-based testing, albeit in an ad-hoc fashion. Risk-based testing refines it to a finer level, introducing methodological risk-assessment methods, to test the most critical functionality or elements, picking and choosing the most relevant of tests. It ensures testing of all critical functions of the application that actually matter.

Since risk-based testing encourages prioritization, it brings the focus on actual business needs over functional perfection. It offers an opportunity for the client and test manager to negotiate the testing requirements, and thereby strike a balance between end user requirements and technical perfection. In the absence of such requirements, testers tend to veer towards making the technical side perfect, while ignoring the business or end user side.

One criticism against risk-based testing is that it increases the effort required for the test process, for there is the added process of identifying relevant tests in the first place. The investment in making quality risk analysis upfront is worth its while in reducing the actual testing time and resources considerably. Once the initial analysis is over, only periodic updates and maintenance of traceability are required.

Risk-based testing has become relevant in today's fast paced world where the pressure to roll out the software is the order, and there is virtually no time to conduct extensive all-encompassing tests to cover all functionality. By placing the focus on the business scenario, customer satisfaction improves.

THE PROCESS

There is no documented method to **undertake risk-based testing.** It is a concept or an approach rather than any specific type of test. Nevertheless, any risk-based testing approach would invariably entail four steps or processes:

1. **Drawing up the testing requirements in terms of risk involved in the project.**
2. **Prioritizing the testing requirements in terms of magnitude of the risk.**
3. **Planning and defining tests according to requirement prioritization.**
4. **Executing tests according to prioritization and acceptance criteria.**

Drawing up the testing requirements

There is no hard and fast methodology to draw up all risks associated with a project. The usual methods adopted to increase the precision of risk-based testing are:

1. Conducting brainstorming sessions with stakeholders associated with the project. The possible stakeholders include the technical developers, project managers, infrastructure providers, representatives from the business team, system maintenance team, and end users.
2. Gut feelings, instincts and expertise of the testing team, who would be familiar with similar cases.

The potential risks are generally evaluated in terms of:

- Compromise in quality of service delivered by the software
- Loss of reputation and goodwill for the business owing to malfunction
- The extent of resources, including human and monetary resources, to set the glitch right
- Possible legal costs and compensation
- Safety of the people and property connected with the system
- Security of the system, and the entire eco-system

Prioritizing the Testing Requirements

Since the fundamental approach to risk-based testing is selective testing, success depends entirely on identifying the functionality or attributes which matter most, or which have the most significant impact on the project. Prioritizing wrong tests would lead to ineffective testing, and increase the risk of the software failing at a critical point, with all its consequences.

The common method to prioritize the requirements is by holding a quality risk analysis session, to identify quality risk items and assess the likelihood and impact of the same.

There are several ways to calculate risk and prioritizing the various risk factors, based on the worst-case outcome. Some of the ways are:

- Ad-hoc method, relying on programmers' and stakeholders' experience and expertise to prioritize sub-systems based on their criticality, high use, or complexity.
- Applying the **Pareto principle** or the 20:80 rule. When applied to software testing, the rule holds that 20% of the feature set covers 80% of the user requirements. The risk-based approach seeks to identify bugs in the 20% feature set that allows user to do 80% of the work.
- Using a top-down design tree to identify 'vital' functions. For example, if a login page of a website does not work, the user cannot access any other page, and all such other pages working perfectly give us no benefits. The possible critical functions that may be considered will include system downtime, lost transactions, data corruption, need for rework, maintenance costs or efforts, implications of access denied to customers, regulatory compliance, and more.
- Using cause-and-effect tree to identify critical functions, in the same way.
- Interviewing end users to get a perspective on what really matters to them. The factors to be prioritized may be diverse, depending on the stakeholder, and may include, among other things, sponsor preference, end user ease, regulatory requirements, or more. Assigning a score on a poitnts scale for each risk-factor and plotting the points for each factor on a histogram is one easy way to rank the various risks.

There is no rigid rule for which factors matter most in ensuring accuracy. What is important and what is not depends on the business and the specific circumstances related to the software and use case.

Regardless of the factors selected as priority for testing, it is important to develop consensus on the priority among the stakeholders, and update the same on Functional Requirements Document (FRD) in unambiguous terms, so that the testing team is clear on what tests need to be executed, and their order of execution. If the stakeholders are from diverse backgrounds, training them on the rationale of risk-based testing, and how to approach it, helps in developing such consensus.

Planning and Defining Tests

Planning and defining the tests as agreed upon by the stakeholders is relatively easy and straightforward. This is the job of the testing team, and usually tests are done in a sequential manner. The best practice is to sort test cases in the ascending order of magnitude, and start running the test cases in this order until time and effort extends.

Executing Tests

The best laid plans of men and mice can go for a toss. Executing tests based on the defined schedule may seem straightforward enough, but many things can go wrong. The production schedule may be delayed, market conditions may have undergone change forcing a revision of timelines, or anything else.

The deadline given to the customer remains sacrosanct in most cases, and if there is little time than scheduled, the onus is on test managers to either do whatever tests that rank top until time runs out, or apply the Pareto principle to finalize the scope of testing in the reduced timeline, to ensure least risk while maintaining the highest quality.

KEY CHALLENGES

Risk-based testing poses **several challenges** along with the numerous advantages and benefits that it offers. Organizations need to understand such challenges and avail sufficient steps to overcome the same, in order to improve the accuracy of their risk-based tests.

The success of risk-based testing depends on understanding the risks, or getting the risks right, and prioritizing them correctly. There is the ever present danger of unrecognized risks and risks being assessed too low, and such dangers could subvert the entire process. It requires a high degree of competence to have a proper understanding of the latent risk factors, and such competency requires adequate investment in training.

While risk-based testing attempts to bring in a methodological approach to identifying and prioritizing testing, the core activity of risk assessment is still subjective, for the simple reason that there is lack of reliable objective criteria. There is no workaround to trust experts' judgments. The only way to overcome such challenge is to identify and select the right stakeholders for the risk assessment exercise, and this is easier said than done.

Not all stakeholders would be enthusiastic to participate in the exercise. Involving the customer in the risk assessment sessions may not influence customer satisfaction or how the customer appreciates the supplier. It requires convincing the stakeholder that participating in the testing process is a win-win approach, and make explicit how such participation would benefit them.

Again, when prioritizing risks, what is "most important" is often subjective, and depends on the stakeholder's perspective. For instance, malfunctioning of a banking software may not really cause an inconvenience from the bankers' or maintenance team's perspective, but it will pose a huge risk to the end users or customers who use the software. The malfunctioning may have the potential to create huge losses in terms of customer defections, diminished goodwill, decline of market share, and other implications. The success of risk-based testing depends on identifying such latent risks, and for this, it is critical to undertake a comprehensive exercise to draw up the testing requirements.

Finally, even after getting the potential risks right, it may still be difficult to attach a test to an identified risk, especially when risks are described in abstract terms. It is essential to describe risks as concretely as possible, and also consider what types of tests to run for the risk.

ABOUT SUYATI

Suyati provides marketing technology and integration services for companies that wish to combine the best of breed solutions and create a unified approach to customer acquisition. This unified digital marketing approach requires system integration between various CMS and CRM platforms, and a slew of ecommerce, Marketing Automation, Social Media Listening, email and social marketing, and customer service systems. Our specialized knowledge in Salesforce, open source and .Net based systems enables us to build effective custom integrated solutions for our clients.

Suyati's custom technology solutions have been deployed in companies in the US, Western Europe and Australia, and have helped many enterprises leverage the web/cloud/mobile technologies to acquire customers through integrated digital marketing. Suyati is based in Chicago with product engineering capability out of the US and India.

www.suyati.com **services@suyati.com**

Reference:

1.http://homepages.cs.ncl.ac.uk/felix.redmill/publications/1A.R-BT%201.pdf
2.http://istqbexamcertification.com/what-is-risk-based-testing/
3.http://www.rbcs-us.com/images/documents/risk-based-testing.pdf
4.http://www.softwaretestingclass.com/what-is-risk-based-testing-in-software-testing/#sthash.GORzRdSA.dpuf
5.http://www.softwaretestingclass.com/what-is-risk-based-testing-in-software-testing/
6.http://www.stickyminds.com/article/strategy-risk-based-testing?page=0%2C1
7.https://www.ictstandard.org/article/2011-10-25/concept-risk-based-testing-and-its-advantages-and-disadvantages
8. https://en.wikipedia.org/wiki/Pareto_principle#In_software