

Challenges in Testing Without Requirements

White Paper - Version 1.0

Abstract

The title of this article may seem a paradox i.e. one may always argue that “How’s testing possible without testers having any knowledge of requirements”. But, the reality is that most of the times, testers have to perform their jobs with minimum documented requirements or in worst case, no requirements at all. Such a situation not only makes the tester’s work difficult but also makes planning and estimation almost impossible and considerably reduces Return of Investment from QA department. In this write-up, we try to explore the situation of testing without requirements, its impact on whole Software process and how effectively a tester can perform his work.

Reasons for not producing Requirements document

With ever expanding teams, temporary resources, a large system may not be documented well enough or sometimes not at all, for the tester to form complete scenarios from the functional specification (if present) document.

This can pose a great risk to the testing of the system. Requirements will never be clearly understood by the test team and the developers will adopt an 'I am king' attitude, citing incorrectly working functionality as the way the code or design is supposed to work. This can confuse end development and you may have a product on your hand that the business doesn't relate to, the end users don't understand and the testers can't test.

Dealing with a system that needs to be tested without functional specifications requires good planning, smart understanding and good documentation - lack of which caused the problem in the first place.

Testing without requirements specifications

What type of testing will be done on the system– functional / performance / regression?

Is it a maintenance project or is it a new product?

From where have the developers achieved their understanding of the system?

Is there any kind of documentation other than functional specification?

Is there a business analyst or domain knowledge expert available?

Is an end user available?

Is there a priority on any part of the functionality of the system?

Are there known high-risk areas?

What is the level of expertise of the development team and the testing team?

Are there any domain experts within your test team?

Are there any known skill sets in your team?

Are you considering manual/automated testing?

All of the questions above should help you identify the key areas for testing, the ignorable areas for testing, and the strong and weak areas of your team. Once you ascertain the level of skills and expertise in your team, you have choices available to choose from. Depending on the comfort level in your team, you can choose one from the options given below.

Theoretical Approach

The theoretical approach will ensure that you have good comprehensive documents for the system at the end of the process.

Use Case Based Approach

Traditionally use cases have been used for requirement gathering. There has been a considerable shift to use them for testing, primarily because they model user role-play, movement, access rights and the interaction of one module to another – which is essentially what a test plan aims to do.

A use case defines the following –

1. Primary actor or actors
2. Goal
3. Scenarios used
4. Conditions under which a scenario occurs
5. Scenario result (goal delivery or failure)
6. Alternative scenarios or flows

Business Rules testing

Every system operates on business rules. If one were to document the business rules on which a system is based – it would be an exhaustive list to write down and track. Most business rules become implicit in the design and mind. Business rules are typically implemented separately from presentation, application and data logic. To add to this, they are under constant change due to market requirements, user needs and software upgrades. Business Rules testing will help validate that your system is performing as expected, thus enforcing correct business practices on the user and the system code.

A typical way to gather business rules for your system is by talking to the business analyst and trying to understand what the analyst expects the system to do or how he/she expects the system to behave. A meeting with a developer will be helpful to understand how he/she thinks the system should behave. You may get conflicting ideas here, and it is important that these are sorted out before you have begun testing.

Test Strategy

Test team will be loaded with many tasks at hand. As testers will be working under the pressure of meeting critical deadlines and ever-increasing pressure of finding bugs early, it is of supreme importance to prioritize the efforts. There may be the cases when tester is required to get clarifications, understand the product's features, document test cases and testing all at the same time. This is where the judgment of tester comes to play. There has to, be a balanced approach between all tasks at hands and due importance should be given to testing the product. The whole idea is to uncover more and higher priority bugs as early as possible and be more productive.

Precise documentation of test cases is again subject to availability of time. But the testers needs to be ready with checklist which consists of –

- a. All the tests to be conducted.
- b. Proper sequence in which to execute the test cases.

The correct flow will help to focus the thinking of tester at the time of execution of test cases. The testing approach in this situation is often seen as Adhoc testing. In such form of testing, the tester randomly tests the functionality more on the gut feel. The problem is that one cannot always rely entirely on such form of testing as one is never sure of – “How much has been testing?” and “How much is remaining to be tested?”

This is where the Exploratory testing has a role to play. It is the more organized approach to testing. The features of software are explored thoroughly by the, tester’s thought process and more meaningful tests scenarios are tested and documented maintaining the proper flow of execution. The tester can best gain for such form of testing when he explores a particular functionality until he has tried all the ideas and then move on to next areas to test. This will help the tester to build the confidence in testing. It is very important to document all the ideas that were tested, as it will help in post analysis later on. Another thing that can be of great help is what I would term as Group

Impact of testing with no requirements

Let’s consider an ideal case in which a tester gets Software requirements well on time i.e. at Design and Analysis stage in the Software Life Cycle. In this situation the tester’s job will begin with reviewing the requirements. A lot of defects are uncovered at this stage as requirements are verified for ambiguity. This forms the base for creation of test cases. In the next stages, the test cases are derived from requirements and then execution of test cases takes place followed by analysis of results. But the irony is that, this situation does not always exist for a tester. Testing is still considered to be simply as a separate phase in Life cycle rather than being accepted in each phase of life cycle. In this case, testing starts at the tail end of the Development process just few weeks before actual release. Testing in these conditions often makes life of QA department miserable and causes the below listed effects-

- As overall product knowledge is not documented in this case and is confined only to a few people,
- QA department’s role in overall product life cycle becomes quite less.
- Defect Migration increases
- Estimation of efforts, resources and planning in general becomes difficult.
- Most of the times, testers and Developers are out of sync regarding the functionality
- The number of “AS-DESIGNED” defects in the defect database increases considerably.
- Automation of test cases becomes infeasible.

Conclusion

In the end, all one requires is careful thought, planning and mapping out of tasks before one actually begins testing. Testers and Test Analysts usually react by pressing the panic button and begin testing – which ultimately leads to complete chaos and immeasurable quality of the product. Clear thinking with valid goals will help you and your product achieve desirable quality. You might end up discovering defects or even requirements your business analyst never thought possible!

About STC

STC ThirdEye Technology (India) Pvt Ltd is India's largest Independent software testing organization providing End-to-End testing Services. We build and operate dedicated India-based testing centers for our customers with the latest computing and data communication technologies, and deliver our services, with high standards of security and confidentiality. Consistent qualities of deliverables under compressed time schedules enable us to get repeat business. We help Fortune 500 ERP, BFSI, Healthcare, Gaming and Telecom solution providers We are ISO 9001:2000 certified organization. For more details, please visit us at www.stcthirdeye.com

Disclaimer

The Whitepaper series presents report on subjects in the sphere of activities of STC ThirdEye Technology (India) Pvt Ltd that are to be considered in the interest of wider public. These papers are part of the ongoing studies and authors will be glad to receive your comments. The views expressed in these papers are to be regarded as those of the author and should not be interpreted as reflecting the views of the management of STC ThirdEye Technology (India) Pvt Ltd. STC ThirdEye Technology (India) Pvt Ltd assumes no responsibility for any actions taken by Anybody based on the information provided in this paper