Whitepaper

When *Good Enough*
is *Not an Option*

Scenario Testing

www.BayTechServices.com

Contents

What is Scenario Testing?

Scenario testing is a way for system testers to put themselves in the shoes of the users to simulate real-world user experiences or "scenarios". Scenario testing is used to create large test cases that model real use of the application by covering multiple requirements. Scenario testing helps find important bugs that other techniques may not be able to detect.

Why do we need Scenario Testing?

Although most organizations test individual functions thoroughly, complete testing of all possible interactions between functions is virtually impossible. Reasons include:

- Domain of possible inputs is too large.
- There are too many data combinations to test.
- There are too many possible paths through the program to test.

Many test techniques aim to prioritize testing efforts to focus on important areas. For example, boundary value analysis and state transition testing aim to find the areas most likely to be defective. And risk-based testing prioritizes testing of functions based on risk they represent and likelihood of failure.

Another way to increase the ROI of your testing effort is to develop more 'powerful' test cases. Scenario testing helps create and prioritize powerful test cases based on scenarios that represent real use – both common usage and special cases.

How does Scenario Testing Relate to Functional Testing?

Functional testing is typically performed at low levels of granularity. For example it might test for functions such as user commands, data manipulation, searches, business processes, user screens, and integrations. But each function is tested independently. And this testing cannot test for the interactions between multiple functions.

Scenario testing is a type of functional (or black box) testing that tests applications at a higher level of granularity. It tests your application for complex interaction between functions. Scenario testing detects bugs that are difficult to detect by testing individual functions.

How does Scenario Testing Work?

Scenario testing tries to test the product by creating test scenarios that replicate real usage. A test scenario ensures that business process flows are tested from end to end. A test scenario is a hypothetical story that helps think through complex usage situations of a product. Test scenarios are made to address both typical and unusual usage of the product. Test scenarios are independent tests, or a series of tests that follow each other, where each of them is dependent upon the output of the previous one.

Test scenarios include:

1. Test cases and/or test scripts and
2. The sequence in which they are to be executed.

Example

Hospital ABC uses a Hospital Management System (HMS) to manage patient activities with the hospital. The system includes features to manage outpatients as well as inpatients. An outpatient is someone who consults a doctor, is prescribed medication and leaves the hospital the same day. An inpatient gets admitted to the hospital.

One of the features of HMS is "Patient History", which stores patients' history, both for outpatients and inpatients. The HMS helps doctors in better diagnosis and treatment by providing them access to important patient information. Individual functions of the HMS application have been well tested and each feature works properly. And yet, under scenario testing, the HMS failed the following test. Here is the scenario it failed.

Bob, a middle-aged man goes to a hospital with pain in his arm. He is asked to wait for the relevant doctor and his details are entered as an outpatient. While waiting to see the doctor, he starts experiencing a severe pain in his chest. He is immediately admitted in the hospital and his status is changed to inpatient.

In the test scenario, Bob's history was initially entered as an outpatient, and later in the day his status was changed to inpatient. After making the entries in the HMS, the tester logged into the application as a doctor to review Bob's Patient History – however, the history page for Bob was blank. This led to the discovery of a bug that *when Patient History is first entered as an outpatient, and then the patient status is changed to an inpatient the same day, the Patient History does not transfer over.*

Characteristics of a Good Test Scenario

Good test scenarios test something that is useful and reflect interesting real-life situations. Good test scenarios are proxy for users. Specifically, a good test scenario (or story) should accomplish the following:

- The story involves the usage of many features of the product. The aim is to find problems that arise during the interaction of various features of the products.
- All testing techniques find bugs but not all bugs are fixed - many of these bugs are ignored as irrelevant or unimportant. A good scenario clearly shows the impact of a bug not being fixed. It motivates the stakeholders to fix the bugs.
- The story of the scenario is credible and the stakeholders believe that not only could it happen in the real world, but it probably will happen.
- The test results are easy to evaluate. This is valuable for all tests, but is especially important for scenarios because they are complex.

Ten Tips to Create Good Scenarios

Writing good test scenarios is part science and part art. Besides product knowledge, writing powerful test scenarios requires an understanding of the business context and empathy for the user. Ideas for effective test scenarios come from observing users and competitors' products, looking up customers' data and feedback and learning business rules. But a good scenario writer also has to be a good story teller. Writing interesting stories is an art and improves with practice. The following best practices will provide some ideas for the writing good scenarios:

1. Identify the key players or objects in the application and write their life histories. For example, in a library management system, one of the objects is a book. The life history of a book includes purchase of the book, getting added to inventory, getting placed in racks, getting issued, getting lost, and getting damaged.
2. Analyze possible users, their motivations and compulsions while using the application. For example, in an application for police complaints management, a police officer may want to record fewer complaints to indicate less crime. On the other hand a citizen may want to log even trivial complaints. Knowledge of user motivation will help you imagine stories for different users.
3. Consider users who may want to abuse the system? For example an accountant may want to embezzle and siphon off money while using an accounting system.

This will help you create a story of embezzlement by looking for loopholes in the accounting system.

4. Most software applications are 'event-response' based. An event is triggered by either users (e.g. click of a submit button) or generated by the system (e.g. triggering of an alarm at a particular time). A response is what the application does in response to an event. Identify important business events of a system, for example when due date of submission of insurance premium is over, or when a person applies for a new insurance product. Now you can create scenario stories around these events and the business rules for responses.
5. Identify special business events related to the application. For example, when a purchase order is cancelled after the goods are dispatched but not yet delivered. These special business events need special business rules and special processing. Creating stories around special business events can help you find bugs in the underbelly of the application.
6. Identify the benefits that your organization promises to the user or the benefits the users perceive they will get from the application. For example, a marketing brochure may claim that your word processing software has features similar to Microsoft Word. Then the user expects it to have a 'grammar checking feature' even though that feature is not in yours. Check for such benefits and then create stories to test them.
7. Interview users about famous challenges and failures of similar applications or predecessors of the application.
8. Work alongside users to see how they work and what they do. Close interaction with users and observing them in their work environment can give you good insights into how a users could use the application.
9. Read about what similar applications are supposed to do. Knowledge about the domain by reading related books, articles, and internet surfing also goes a long way in getting deeper insight into the application.
10. Study complaints about the predecessor to this system or its competitors. Database of user complaints provides good insight into various user scenarios.

What to be Careful about?

Just as with any other technique, scenario has limitations – and it is important to understand them. Although scenario testing can detect bugs that other techniques may not find, here are some things that scenario testing cannot do:

- You cannot run scenario testing when the application is still under development. If the application still has major bugs at the individual feature level, then it will be difficult to execute test scenarios. In an ideal situation, scenario testing should be done after all lower level tests have been completed.
- Scenario tests try to identify problems. The focus of scenario tests is to be creative and create a credible story that can find important problems from a user perspective. But scenario testing is heuristic in nature and is not intended to capture all problems.
- Scenario testing requires special skills and is difficult. It almost requires the skills of a business analyst, in addition to that of a tester. Also, it takes much more effort to create and document complex stories of scenarios compared to simple test cases that test individual functions.

Summary

Scenario testing is one of the techniques that helps find important bugs at the application level. It aims to find bugs that arise due to the interaction of multiple features of the product. The technique requires the tester to empathize with different users of the product and visualize different scenarios in which users could use the product. Implemented right, scenario testing has shown to vastly improve the quality of your product.

When *Good Enough* is *Not an Option*

BayTech Services

45 Coventry Lane

North Andover, MA 01845

Phone: 978-852-7019

Send comments to comments@BayTechServices.com

www.BayTechServices.com