Software Testing Techniques

Getting Started with Software Testing

Beth Schechner

Elementool

Software Testing Techniques

Getting Started with Software Testing

Software bugs are so prevalent and so detrimental that they cost the U.S. economy an estimated $59.5 billion annually, or about 0.6 % of the gross domestic product.[i] An estimated $22.2 billion could be saved annually in the U.S. just by implementing an improved testing infrastructure to enable earlier and more effective identification, and removal, of software defects!

These are large numbers, for sure. You're probably thinking that since this cost is spread out over the entire U.S. economy, that the impact isn't all too staggering. However, think about what 0.6% of your company's annual revenues are! That's probably a fairly large number; now imagine that you could prevent the loss of almost half that number simply by improving your software testing process. Testing is what ensures that your software application is of the high quality that your customers want. It's what makes sure that the product you send out the door is consistent with your business goals. This is a HUGE piece of the software development lifecycle (SDLC); testing typically consumes 40 – 50% of any development project, and even more effort is needed for systems requiring a higher level of reliability.

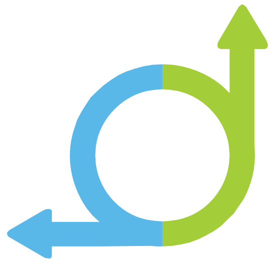
What is Software Testing?

Software Testing falls under the umbrella of Quality Assurance (QA), which is the monitoring and perpetual improvement of the entire software development process. QA is responsible for making sure that all standards, processes, and procedures are followed. It also has the job of ensuring that any problems, and potential problems, in the software system are found and appropriately dealt with.

Every software product or application has a specific target audience. If you're developing video games, financial management applications or hospital administration software, to name a few examples, your audiences are going to be vastly different. Gamers are going to be looking for an entertaining experience, while your bankers and nurses are going to be judging your applications based on very different criteria and may have an entirely other idea of success. When your organization is developing or investing in a software application, you need to make a judgment on whether or not your application is going to meet the needs and expectations of your target end users and customers, purchasers and your business partners. Software testing is the process in which you attempt to make this judgment.

Getting Started

Once your developers have generated the source code, you need to test your software in order to find and fix as many errors as humanly possible before delivering the product to your target customers. Your goal should be to create a hermetic process that prevents bugs from slipping through the cracks using a series of test cases that will test every aspect of the product and make sure that everything is properly covered - but how should you get started?



These testing techniques will point you in the right direction towards building your testing practice and discovering bugs, before your application is released to customers! These techniques will act as a structured guide for creating test cases that work with the internal logic of software components, and the input and output domains of your application to find errors in the function, behavior, and performance of your application.

Spreading the Word

Any new process is going to come as a shock to an organization, especially if there is already an established way of handling software development and testing projects. How to introduce a new process for testing all depends on how big your company is, and what unique risks are involved. For large organizations with high-risk projects (meaning many people, groups, budgets and investors involved), management needs to seriously get on board and approve of a formalized QA process. Testing, after all, is a way of affirming that your project is meeting business goals, and getting there in a way that doesn't meet those goals is counterproductive.

In companies with a lower risk level, management approval and the overall implementation of your QA program may happen at a slower, gradual pace. Which process you choose, and how you go about implementing it, will depend on management, feedback to and from developers, and reliable communications among your customers, managers, developers, and testers!

Having good requirements management processes will also put you one step ahead when you start with your software testing, letting your team hit the ground running and maintain their pace when testing. Starting with clear, complete, testable requirement specifications that have been written into requirements or design documentation, will make it that much easier to test. Knowing what needs to be tested and how you're doing in comparison to your business goals will save you both time and money during the course of the SDLC.

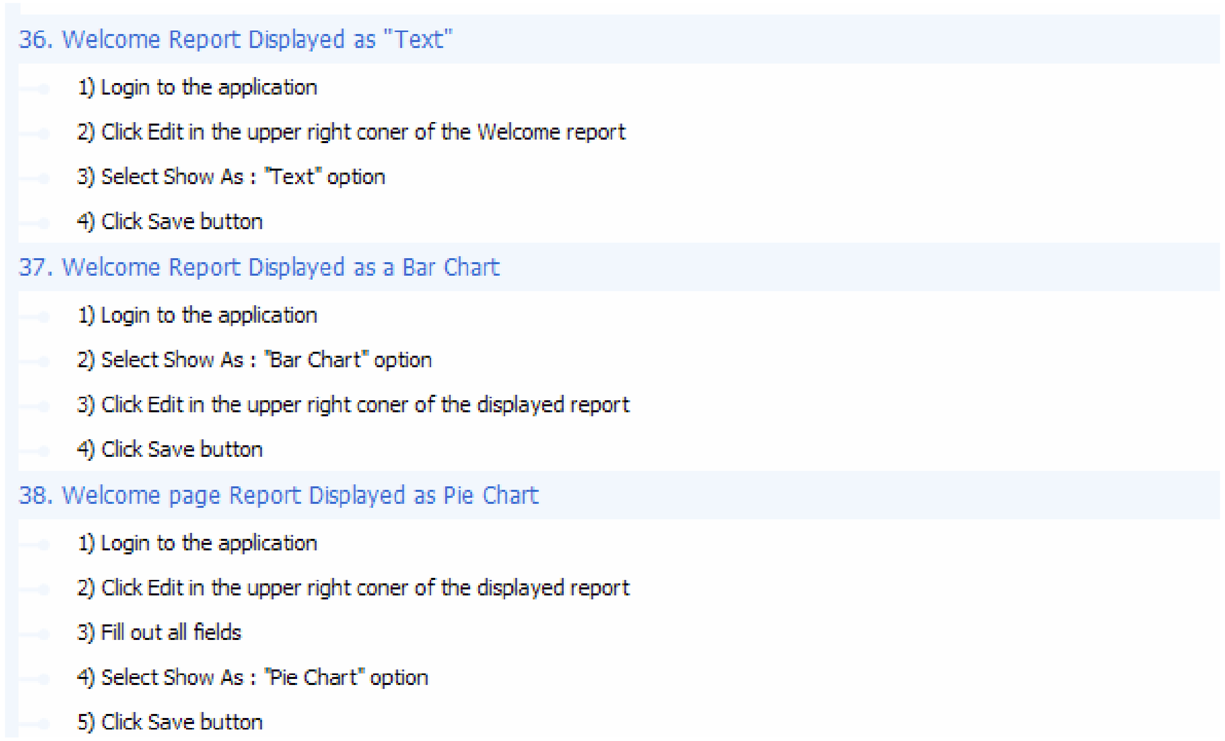
Planning It Out

Now that you've started laying out your company's software testing strategy, what next? Well, you'll need a *software test plan*, and a series of *test cases*. When you're building your test plan, you'll want to lay out the objectives, scope, approach, and goals of your software testing effort. Building your test plan lets you see the entire project written out, and figure out what actions you'll need to take to accomplish your business objectives. Your software test plan, when finished, should let all those involved, outside of the testing team, understand the 'why' and 'how' of your testing effort. Think of your test plan as a roadmap for testing, and in your roadmap, you're going to need specific directions.

Those specific directions will take the form of *test cases*. Each test case should describe a specific set of test steps of a function or feature of your application, along with the expected results. This will ultimately help you to determine if each feature of your application is working correctly. A test case can include things like a test case name, objective, test conditions, steps, input data requirements and of course, your expected results. Each organization and project will have its own particular requirements for building a test case; in some cases there may be just the bare basics, or a whole load of detailed information. Obviously for a more involved function or feature, more details should be included.

Building test cases should be one of the first steps of your testing process, since they lay out what needs to be tested and what should be expected. However, developing your test cases not only gets the testing process started, but it can also help find problems in the requirements or design of an application, since building the test cases forces you to think through the entire application.

At this point, you may want to consider a management tool to assist in tracking your test cases. This tool can also help communicate the resulting bugs to your development team, so that they can be resolved or otherwise handled and then sent back to QA for re-testing. A management or tracking tool will also help with your reporting process, so that the whole team and business management is involved, and can know what stage the testing process is at, at any time.


36. Welcome Report Displayed as "Text"

1) Login to the application
2) Click Edit in the upper right coner of the Welcome report
3) Select Show As : "Text" option
4) Click Save button

37. Welcome Report Displayed as a Bar Chart

1) Login to the application
2) Select Show As : "Bar Chart" option
3) Click Edit in the upper right coner of the displayed report
4) Click Save button

38. Welcome page Report Displayed as Pie Chart

1) Login to the application
2) Click Edit in the upper right coner of the displayed report
3) Fill out all fields
4) Select Show As : "Pie Chart" option
5) Click Save button

Test Case List Example

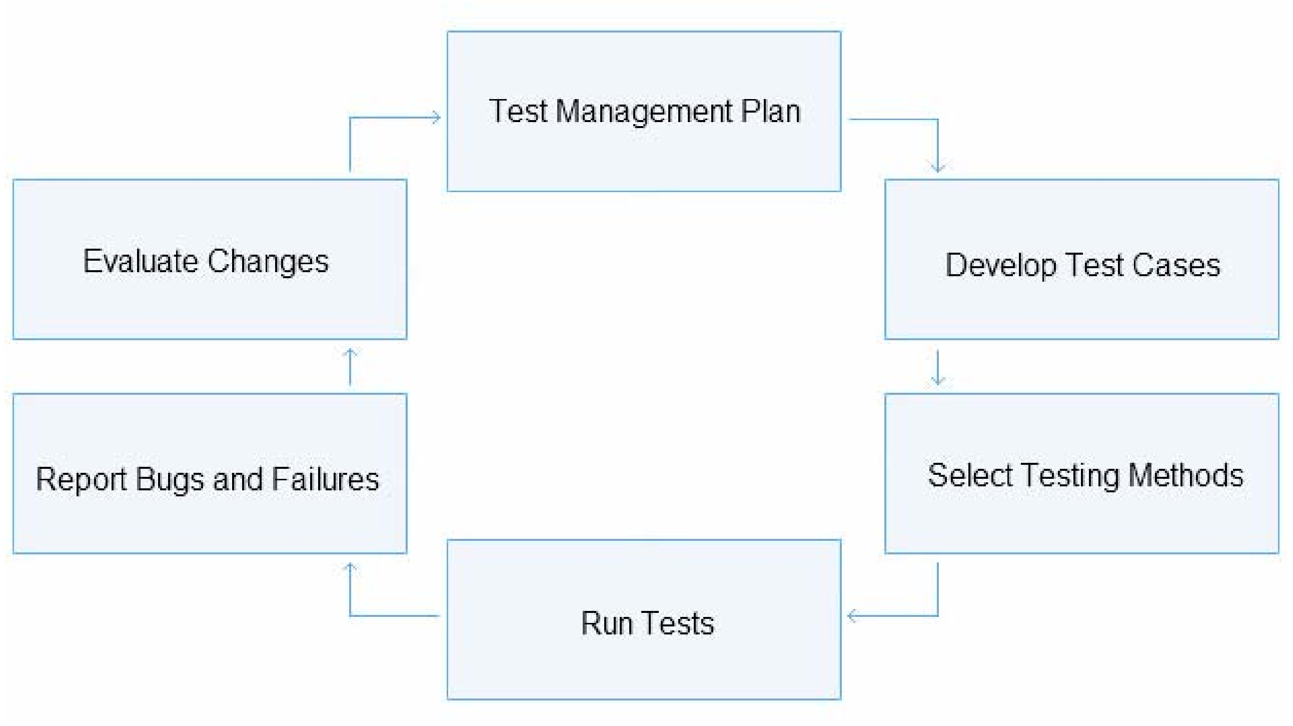
Methods

There are many, many different types of software testing, and which method or methods you choose to employ depend on your product features, goals, existing processes, management and preference. We'll cover just a few here; the most simple or straightforward testing methods are *Functional* and *System Testing*. Both Functional and System Testing are *Black Box* testing methods (see below); Functional Testing simply tests the separate features of your application, based on your previously defined requirements. System Testing has the same end goal, but tests all of the different pieces of your application and how they work when combined.

Mentioned above, *Black Box* and *White Box* Testing are less specific tasks and more an approach to your testing, and require slightly different skill sets. Black Box testing is not based on any knowledge of the software application's internal code or design, but rather on the overall requirements and functionality. White Box testing, on the other hand, is based on coverage of paths, conditions, branches and code statements. When using White Box testing, the tester or QA manager is aware of and familiar with the internal code of the application being tested. As a result, it's not uncommon for White Box testing to be done by developers.

For a more inclusive testing approach, you'll utilize Integration and / or Incremental Integration Testing. The former of these methods tests to see whether the different pieces of your application function together, as planned. The different pieces could be individual applications, different functions or even applications that will need to work together on one network. The latter, Incremental Integration, tests the application consistently each time a new functionality is added. This method is particularly useful when you need to make sure that various functions of your application can work both independently and as a whole.

Performance Testing, also known as Load or Stress Testing is also important to run on your application, particularly if you are building a website or a web-based application which multiple users will expect to access and actively use at the same time. Load Testing is when you put unusually high demand on your system or application in order to measure its response, and determine what amount of traffic or use will cause a system error. For example, if you're building a multi-player game, you'll test to see what might happen if 1,000,000 people are accessing your game online and 25000 users are creating characters, 25000 users are chatting, 25000 users are completing level 3 and 25000 users are adjusting their preferences. Can your system handle any combination of these processes? What if there are 6,000,000 users online at once? Performance Testing is highly important for producing a successful web-based application.

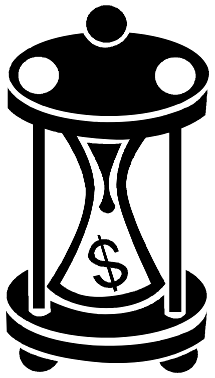


Software Testing Lifecycle

These different methods can be practiced independently, or in any combination, to make up your application testing process.

Conclusion

IT workers spend more than 34% of their time just fixing software bugs[ii]. 34% of the average 40 hour workweek is 14 hours! That's almost 2 full days out of every week spent fixing software bugs. However, if you lay out your testing process, employ organized and well thought out testing methods, and enforce them throughout the SDLC, than you can save your team **time** and your business **money**. And isn't that what every testing manager wants to be known for?



More eBooks can be found on our website at: www.elementool.com/ebook

[i] NIST: http://www.nist.gov/public_affairs/releases/n02-10.htm

[ii] Accenture: http://www.accenture.com/Global/Research_and_Insights/Outlook/By_Alphabet/InformationFail.htm