Testing, What is it Good For? Absolutely Everything!

An overview of software testing and why it's an essential step in building a good product

Beth Schechner

Elementool

www.elementool.com

Testing, what is it good for? Absolutely Everything!

An overview of software testing and why it’s an essential step in building a good product

In August of 2006 a U.S. government student loan service erroneously made public the personal data of as many as 21,000 borrowers on its web site, due to a software error. The bug was fixed, eventually, and the government department subsequently offered to arrange for free credit monitoring services for those affected. [i]

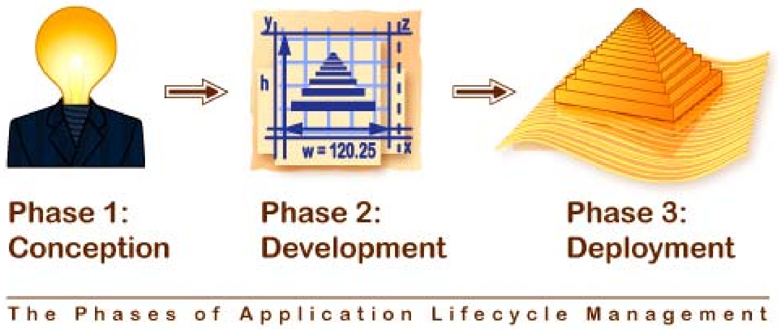
Could this error and the resulting cost to the U.S. government, have been prevented if proper software testing had taken place prior to the release of the system used by the U.S. Government student loan service? According to a 2008 research report conducted by the Research Triangle Institute (RTI) in North Carolina, on behalf of the Department of Commerce's National Institute of Standards and Technology (NIST), an estimated $22.2 billion could be saved annually in the US, by implementing improved testing infrastructure that enables earlier and more effective identification and removal of software defects.[ii]

This example, and the cost incurred to the U.S. economy, should drive home the idea that software quality is essential to business success. For end users who are consuming and using the software application, success means that it delivers what they want, when they want it. For the development team tasked with creating the application, success is achieved when they’ve addressed all of the factors that may somehow affect performance. These include, but are by no means limited to functionality, reliability, accuracy and usability.

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, and for ensuring that any problems, and potential problems, in the software system are found and adequately 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 an organization develops or invests in a software application, it must judge whether the application will meet the needs and expectations of end users, target audience, purchasers and business partners. Software testing is the process in which you attempt to make this judgment.



So software testing is a series of tests to judge the integrity of your application and its ability to function as planned and as marketed. Way back during the initial conception of your application, if you were using Application Lifecycle Management (ALM), you drew out a list of requirements which would ensure that the application you're building will meet your business and technical goals. Well software testing is the process of systematically checking the application against those requirements, to verify whether or not each individual feature is working properly. Or rather, to see exactly how each individual feature is working: properly, improperly, or completely unexpectedly. Software testing also provides business management an unbiased view of the development process status.

To start off the testing process, a tester will execute the program or application in question under controlled conditions, and evaluate the result. A *Test Case* is the set of conditions that the tester uses in their evaluation. The intent, of course, is finding software bugs or issues. The tester is trying to break the system here, to intentionally make things go wrong, so that he can see what things occur when they aren't supposed to and what doesn't happen when it should.   The point of software testing is to detect issues or problems in your application that could cause a fatal error or potentially derail your entire project, before your application is released to the public.

According to the NIST study, software bugs cost the U.S. economy $59.5 billion annually! The researchers at RTI concluded that more than a third of this cost could be avoided if better software testing was performed. Imagine what your company could save by improving software testing.

What Causes Bugs, and How to Create an Effective Testing Environment

Any number of factors can cause a software bug, including programming errors, miscommunication or lack of communication, changing requirements, redesign, rescheduling, and pressures of time and looming schedules. Overlooked or ignored requirements can result in software application bugs - time schedule limitations will occasionally cause development teams to leave out previously defined requirements.

First off, programmers, like anyone else, can make mistakes. When requirements change mid-way through a project, it's not uncommon for there to be a miscommunication over what features are wanted and expected. Juggling the many moving pieces of a software project to figure out scheduling is difficult under the best situations, but when deadlines are looming, it’s almost inevitable that mistakes will be made.

Setting solid requirements is the first step in ensuring a smooth testing process; defining complete and testable requirements that everyone has agreed upon makes it easy to know what needs to be tested and how to go about the testing. In *Agile Development* environments, ongoing coordination with your end-users or clients is necessary to ensure that all requirements, both old and new, are understood by everyone. Fighting for a realistic release schedule will let your team devote adequate time to each step of the Software Development Lifecycle (SLDC), including planning and design, coding, testing, and bug fixing. If your team isn’t under unrealistic demands, they’re less likely to overlook a serious issue or bug because they’re rushing.   This scheduling applies not just to the overall project, but for testing as an individual process as well. If you start testing early on in the SDLC, then you know that there will be enough time to re-test features after fixes and changes have been made.

Sticking to your initial requirements, as much as possible, will make testing easier. Testing an application is difficult enough, but if you change the requirements too often then no one is going to know what needs to be done!

Your goal as a manager is to make it easy for your testers to do their jobs! Encourage questions, and provide the channels for them to be asked – both of management and of team members. Use the tools that you have to make it easy for your QA team to communicate – forums, instant messenger programs, email, web-conferences, internet, email, etc. If your QA team is working together and talking, then they’re going to notice things like duplicate bugs or similar issues and your team is more likely to complete the project within the set timeframe.

How Software Testing Works, and Why it's Necessary

As we've already discussed earlier in this eBook, the primary purpose for testing is to detect software failures so that the defects causing the failures can be found and fixed. Testing won't determine whether a product always functions properly under all conditions, what it will do though is determine whether the application doesn't function properly under specific conditions. Software testing is definitely a negative practice – you are looking to find out and highlight which functions in your application DO NOT work, not the functions which do. Information that you discover while testing, such as why a function isn't working, can (and should) be used during development and coding for fixing issues.

With mobile phones, netbooks, tablet computers and social networking taking a larger role in how we consume technology, new platforms, new opportunities and new competitors are constantly emerging in the field of software development. As a result, developers are under pressure to quickly release new, interesting and useful software applications. This means that development managers need to know what their teams are working on, what bugs have been found during testing and what stage these bugs are currently in, and what (if any) feedback is coming in from clients. We've moved way beyond the days when a basic spreadsheet (or email!) was sufficient for tracking software testing.

The testing or QA manager is the one responsible for finding bugs or issues in your product before it's released. The responsibility is on their shoulders, because no one will remember the hundreds or thousands of bugs that were found DURING the development process. What your target audience, and the media, will always remember is that one critical bug found by the customer AFTER your product has been released. To cover his back, the testing manager should design a seamless testing process that covers everything that needs to be covered, and leaves nothing for chance. Creating test case lists that cover all aspects of each feature that need to be tested, and then assigning them to testers, ensures the test results are tracked, analyzed and reported to the project manager and development team.

These test lists should be dynamic, and change along with product development. When a new feature is introduced or changed, this lets the testing manager change the test case list accordingly. Testing Managers are human too, and occasionally during the testing process, he'll realize that aspects of the product that should be tested have been overlooked. Dynamic lists let him easily make the necessary changes in the test plan, so that every detail is covered.

A testing manager needs to be in near constant communication with everyone involved in the testing process. Ideally, a test case (the conditions or variables set by the tester) should be analyzed by the tester as well as his testing manager. Details and information on past test results, previous testers, and the processes that were used to generate past test results are highly important in comparing functionality of the feature in all its iterations (past, present and future). Once you find a bug or an issue, you're going to need a separate method for tracking and fixing that issue. Keeping track of these issues and their current statuses is necessary for completing the testing process. In an ideal world, any tools used for bug tracking and test case tracking would be integrated, allowing easy transfer of information from one data set to the other – looking briefly at the larger picture, the same statement could be made in reference to all Application Lifecycle Management tools.

If your developers and testers are able to easily communicate and view results from their team members, they can quickly incorporate feedback and issue fixes. Testing managers should want to do everything they can, and use every tool at their disposal, to facilitate communication between and within teams. According to a study by the French computer manufacturer BULL, 40% of project managers cited poor communication as the leading cause of IT project failures.[iii] Ongoing feedback on the status of bugs and issues and individual test case steps, during the testing process, can significantly reduce the amount of time a software application spends in development, and as the saying goes, Time is Money!

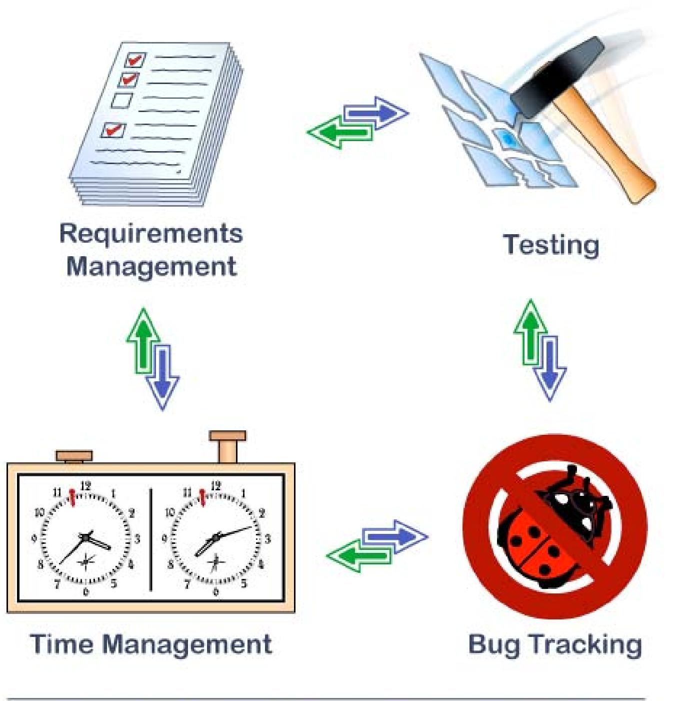
We’ve talked about what testing is and why it’s important, but what sort of data should you have on hand to help things run smoothly? For a detailed and organized test case scenario, it’s useful to have the test case ID and description, expected output, actual output, results, remarks, steps, related requirements, test category, author, and time spent easily available. Since listing the detail of input, expected output, preconditions and post conditions is a time consuming process, testing managers can increase efficiency, and decrease frustration, by having a method or system in place to preserve this information for future and related tests.

Software defaults tend to follow a set process: a programmer makes an error or a mistake, which results in a defect (issue or bug) in the code. If this defect is then executed in the script, it will result in an undesired action, causing a failure. Not every defect will always result in a failure though, and a dormant defect can turn into a failure if you change the environment. Opening a website in Internet Explorer rather than Firefox, for example, or integrating your product with an application from a different vendor can cause an issue where there wasn’t one before. However, a bug isn’t always caused by a programmer error. Issues can also be caused by incomplete requirements, or situations that haven’t been taken into account. For example: your system might crash if a tester submits an alphabetic character in a numeric or date field on a product form. In this case, the requirements should have defined that numeric fields may only accept numeric characters.

Theoretically, test cases can be tracked by keeping a detailed and constantly updated spreadsheet with test case methods, statuses and results. While this is fine for small scale projects, it isn't a reasonable or even feasible method for most development teams. Email is also often used as a more proactive, albeit less organized, method of tracking test cases. Have you ever had to sort through 60 emails just to find the ONE that had the piece of information you were looking for? While both of these methods utilize existing infrastructure and in many cases don’t require any additional cost, once the time spent by developers and testers in looking for files and searching for data has been analyzed, it's obvious that they are inefficient. Since these methods don't allow proper management of the testing process, they aren’t recommended.

A solution for managing software testing that offers a detailed and organized test case scenario, easy archiving and reporting, and fluid communication between groups would obviously offer the most value. Particularly for companies and development teams that can have multiple projects in different phases at various times, with developers and managers spread throughout the globe, a management tool can save precious time and therefore increase the company's bottom line.

Depending on the company, there can be vastly different policies for assigning responsibility for QA and testing. Sometimes one person or group will be solely responsible for QA and testing. In other companies you will find project teams which include a mix of testers and developers working closely together – this mix is something that each company has to figure out depending on your organization's size and structure.



Application Lifecycle Management: Common Steps

Where Software Testing Fits in ALM

Application Lifecycle Management is the process of managing conception, development and deployment of your software application. ALM links business management (looking out for time, money, function, market and practicality) with software development (developing, testing, tracking and fixing). This is all made possible by a suite of tools, applications and processes that facilitate and integrate the two groups. A common collection of ALM tools include requirements management, testing, issue tracking, and time tracking. ALM tools, including perhaps most importantly your testing tools, should encourage communication between all teams involved in both the business and the software development sides.

Testing is the stage where you need to manage and track individual test cases. This is where tools really come in handy. As it can take several tests to determine whether or not your application is working as planned, a Test Case Management tool can be particularly useful in tracking and managing your future, ongoing and completed test cases.

A Test Case, as defined above, is a set of conditions under which the tester can determine whether or not the application is working as originally planned, or conceived. Testing is ***traditionally*** handled by a quality assurance team, whose job is to make sure that the best possible product is deployed at the end of the SDLC, and can take place at any point in ALM and the development process. However, in a traditional development environment testing will start after the initial requirements have been defined and the original coding has been completed. Newer software development methods (models), like Agile, employ test driven development and place a large portion of the testing in the hands of the developer, before it has reached a formal team of testers.

By using a Test Case Management tool, you can keep track of your completed tests, current tests and all the tests that you need to run sometime in the future or delegate to someone else. Reporting functions help tie testing back to business processes, keeping everyone up-to-date on whether or not your application is on track to meet the scheduled release date.

---

[i] Story quoted from the Software QA Test Resource Center: http://www.softwareqatest.com/qatfaq1.html#FAQ1_3

[ii] Software Errors Cost the U.S. Economy $59.5 Billion Annually: http://www.nist.gov/public_affairs/releases/n02-10.htm

[iii] The BULL Survey, 1998: http://www.it-cortex.com/Stat_Failure_Cause.htm#The%20Bull%20Survey%20(1998)