IRON ROOSTER

Software Testing

The IronRooster Way

WHITEPAPER

SOFTWARE TESTING

THE *WHY'S* AND *HOW'S* OF TESTING...

I have been managing testing for more than 10 years. Through the years I came across all kinds of projects – from simple to complex, from agile to old-school, from 100-person to one-man-band, from 1-month to 2-year projects – I have seen them all. Most of them have one thing in common:

The lack of general understanding of the purpose and importance of testing and therefore desperately trying to save time and money by cutting down on testing activities.

In the followings I will try to answer the most common questions that any member on a development project would ask about testing hoping that I can convince you that testing is just as important that the development itself and one cannot exist without the other.

WHY SHOULD YOU READ THIS?

If you are reading this you are probably thinking about testing your software and you already have some idea about the concept of testing.

There are many misconceptions about software testing in general so before you dive into organizing your testing you should have some answers to some basic questions and what testing involves.

In the followings you can find my answers to some questions that I collected that were most frequently asked by clients during my years in testing.

TABLE OF CONTENTS

3. Why do I need testing?

3. What happens if I leave out testing?

4. What activities does testing involve?

5. When should I start testing?

6. What do testers need for testing?

7. What is a good tester like?

7. Why do we need testers to do the job?

8. My team has the best developers. Do I still need testers?

9. Is there any way we can cut down on testing?

10. Why don't the developers just do a better job?

11. Is there any way to save some cost?

12. Is my system bug-free at the end of testing?

12. How many test types are there?

16. Can test automation eliminate manual testing?

16. What are the outputs of testing?

17. There are many testing solutions on the market. Which one should I choose?

WHY DO I NEED TESTING?

If you search on the term software testing you will find that its main goal is to find bugs. In my opinion having organized testing has a more important goal apart from just finding bugs, and that is to be in control. By planning and organizing testing the goal is to be aware of most of the issues in your system.

If you know your issues you can match the impact (of not fixing), the cost (of fixing) and other measures to them so you can make a confident decision about whether you want to fix them or not. Timing of the fix is also an important factor when it comes to its cost; fixing a bug in a function when it is still in development is the cheapest to do while having a patch release to fix a bug in a system already in production can come out very expensive.

Having planned testing has a huge impact on when the bugs are found and how expensive your fix is therefore a well-planned test process will make sure you are in control of your cost.

WHAT HAPPENS IF I LEAVE OUT TESTING?

Let’s assume that you leave testing out of your process and deploy your solution into production without testing it. You can be sure that the users will find those bugs that are in your application. The question is, do you want to be in control and know about these before they reach your users or do you want them to get annoyed with many times easy-to-fix issues.

Users tend to get frustrated with bugs that seem minor to a developer – things like hard-to-reach functions, misplaced buttons, poorly designed workflows or unresponsive functions. Let’s not forget that in most cases whatever tool the users use they want it to solve their problem efficiently with the least response time.

If they are using it for business purposes throughout their day, they also expect a consistent and steady service (high uptime). Unless you are lucky enough to not have many competitor applications on the market it is very likely that if your users get too annoyed with these small issues they will turn to some other solution and not wait for you to fix the issues. If they are patient enough then you may get away with releasing patches every couple of weeks.

After a couple of patch releases you will ask yourself: Would have I been better off spending a little more time to do testing instead of releasing patches? Would have I had more customers by now?

WHAT ACTIVITIES DOES TESTING INVOLVE?

A well-planned testing on a project consists of several phases and types of activities from analysis and planning to test execution and reporting.

During the project planning phase, the strategy of testing is defined with high-level goals, focus and general circumstances of testing. Together with planning the development of the functions, the test plan is defined with more details about testing; planned coverage, estimated timeline, milestones, baselines of testing and depth of each testing level. When function design comes to a stage when there is a solid agreement on how the workflow will work then testers can start planning the test scenarios (or test cases) that will be executed during the execution phase.

When a development of a function is complete and the baselines of testing are updated to match the expected behavior of the system the testers can start checking the functionality. When they find an issue, they may need to analyze it to make sure that it is not due to working with wrong data, environment or checking against outdated baselines.

Testers often need to communicate with business analysts or developers at this point to get further understanding on how the function is designed to work. After each bug-fix phase testers go back and re-check the functionality again and they even run some regression tests to make sure that new bugs were not introduced with the fix to already working functions. On large-scale projects the testing team often produces reports on the progress of testing including the amount and types of bugs they found.

The purpose of this is to have an overview of the general quality of the development and to ensure that the testing progress is aligned with the planned timeline. In case the number of bugs found are higher than planned insight of the types of bugs helps the project management see which phase introduced the majority of the bugs and make changes to the development process, communication or how information is shared if necessary.

WHEN SHOULD I START TESTING?

Testing is not a segregated activity but has to be done in sync with other tasks of software development. Testing has an effect on all parts of the development process; project management, business analysis (or design) and software development itself and vice versa. A bug found in the system may have an effect on the projects outcome (timeline for example), how another function is built up or how it's queued back to development to fix. Also, a change in the project's timeline or how a function is designed or what solution is used in development affects how testing is done.

This is why testing should start when the project starts and should be planned and built upon the development activities involving testing into major decisions and having regular communication between testers and other team members.

WHAT DO TESTERS NEED FOR TESTING?

Although experienced testers will be able to do some high-level testing on your application with very little information about its purpose and business background, if you want to be aware of the quality of your software you need to provide more information to your testing team – commonly called baselines of testing.

Generally, apart from the application's purpose, testers will need to know the expected behavior of what they are testing and what the system should avoid happening. The more detailed information you provide the more thorough your testing can be.
The baselines of testing include the circumstances of how the user will use a function (environment, data), the steps that the user is expected to take in your application to complete their task (workflows) and how the system should react to certain interactions (error handling, security handling, data coming from or going to interfacing applications).

If you want to focus testing on how your users will like your solution (acceptance, usability, workflow, functionality testing) you need to provide a list of user or business requirements or workflow designs that lets testers know what the users and marketing wanted to achieve with the software in the first place.

If you want your testing to start earlier so that you can keep your bug fixes at a low cost you also need to provide a detailed description on how each function is designed to work. This is commonly called specification or detailed design.

If your application is made up of several modules or needs to get information from or provide information to other applications then integration testing comes in play and for that the testers will need to understand the system architecture.

Apart from these the best thing to do is to provide a way for the testers to communicate with designers and developers and ask questions about what they are testing.

WHAT IS A GOOD TESTER LIKE?

Test Engineers can have several areas of expertise each requiring different types of personalities. There are testers specializing in manual testing, test automation, UX/UI testing, test analysis, performance, security testing, compliance or hardware testing. Most test engineers have an IT related degree and a software testing certification. They are commonly good communicators and creative enough to come up with ways to explore the system's functionality and possible flaws.

Since testers will often need to organize their testing tasks to make sure nothing skips their attention many testers are very systematic in the way they organize their work. A key quality of a tester may also include empathy towards the users of the system. Experience is also a key in how efficient a tester can be with his/her testing. Some experienced testers are able to predict where the most bugs lie with very high chance.

WHY DO WE NEED TESTERS TO DO THE JOB?

Since testing requires some skills to be most efficient it is best if the testing is done by certified and texperienced test engineers. There are also some specialization areas of testing; like test automation, where other certification or coding experience is required to do the job.

There are certain types of activities that can be done by other people in the team and there are tasks which are best done by experienced test engineers. For example, unit tests are usually done by developers while many parts of acceptance testing can be done by the end-users or business analysts (or designers, product owners). However, test analysis and planning, test case scripting, test automation, exploratory, performance or security testing is best done with expertise.

If your team is very short on test engineers you can also reduce their workload by having developers test each-other's unit right after development. They may not find as many bugs as a tester would but major issues can come to light early and therefore can be fixed with the lowest cost. A higher-level testing (functional testing, integration testing) can be done by the test engineer(s) when the major issues are fixed to make sure that the functionality is aligned with user and business requirements.

MY TEAM HAS THE BEST DEVELOPERS. DO I STILL NEED TESTERS?

Software development is a very complex process with many parts that can go wrong. By statistics most bugs come from parts of the process where there is much communication involved – like the user and business requirement definition or when the functionality design is handed over to developers.

Therefore, even if you have the best developers on your team things can still go wrong so you cannot eliminate testing.

Starting with defining the business or user requirements there can be misunderstanding on what the user or marketing wants. When the designers, business analysts or product owners detail the function design there can also be mistakes in the business logic or workflow. Many mistakes come from usability issues which are again not the developers' fault but rather design flaws that can make your product uncomfortable for the user to work with. Finally, performance or security issues derive from architecture design flaws which in most cases cannot be blamed on the developer of a function.

IS THERE ANY WAY WE CAN CUT DOWN ON TESTING?

There are ways to make testing more efficient and eliminate parts that involve lower risks. In order to do this, you need to understand your system's risks (the likelihood of bugs in your system and what cost they may hold).

For this you need to analyze the changes and enhancements that will be done in your application and define the focus and priorities of testing. By prioritizing you will be able to reduce the number of tasks when timeframe becomes the limit of testing on the project. Any time there is a major change in the design of a function you will need to re-assess the focus and the priorities to make sure you test high-risk areas of the application.

Based on your test plan it is also important to produce a planned timeline and to monitor the testing progress against this timeline. If you implement regular progress monitoring into your project you can be aware of a rising workload in testing and can react to it by re-assessing focus and priorities or analyzing the cause of the higher demand.

You can also reduce the workload on your testing team if you build up your testing process in a way that allows allocating other resources to some testing tasks. For example, you can ask your end-users (or designers, business analysts, product owners) to do a regular assessment of the usability of your finished modules or functions.This way you can focus your testing resources on tasks that are best done by testers (like system testing, security, performance testing or test automation).

When it comes to cutting down on testing, I have to mention test automation as it is often referred to as a way to reduce efforts. Although it definitely has different approach to testing and is a very popular choice, I would rather call it reorganization of the resource efforts to enhance the depths of testing than actually reducing the workload.

First of all, test automation is most often not a stand-alone solution to testing therefore it has to be combined with manual testing. Test scripting has a much higher cost compared to manual test writing and sometimes even manual test execution so it is very easy to multiply the resources needed to do the job than to actually reduce it.

WHY DON'T THE DEVELOPERS JUST DO A BETTER JOB?

As mentioned earlier it is not only coding issues that lead to bugs in your system however many issues can be avoided if the developers do quality coding. Here are some tips to enhance the quality of development that can reduce the number of bugs in the system:

- Follow coding guidelines, conventions and clean coding standards
- Include code reviews into your development process
- Ensure that the developers do unit tests before handing over their code
- Keep your baselines (requirements, designs, specifications) up-to-date
- Make sure you hand over finished units to testing as early as possible

IS THERE ANY WAY TO SAVE SOME COST?

Yes! Definitely! That is the whole purpose of planning testing. With a good process and planned focus, we can find most issues early and avoid the high cost of fixing something that is already in production. The more rigorous the planning and testing the less cost there is in fixing.

If you know what bugs you have in the system, you may not want to fix them all – you may only want to fix what hurts (costs) the most. The key is to be aware of them. Another way to reduce cost is to have parts of testing done by other members of the team so testers can focus on tasks that are best done by them. Another common way to reduce the time spent is test automation.

There are many articles and websites offering test automation services claiming that test automation will reduce your testing time and cost. Based on my experience this is not always true. The efficiency of test automation depends on the type of the project, the technology they use, the type of testing they target, how frequently these tests will be executed and other factors.

Also, test automation has to be planned even more thoughtfully than manual testing as writing the test scripts has a much higher cost than writing (and sometimes even running) manual tests. However, if you implement test automation with the right focus using the appropriate technology on the relevant project, they can definitely save you time and money especially if they are combined with manual testing.

IS MY SYSTEM BUG-FREE AT THE END OF TESTING?

In short, no. First of all it is impossible to test everything therefore it is impossible to find all the issues in a system. However, you can guide your testing to focus on the most important areas of your product to make sure that the functions that matter the most work as expected. Secondly you may not want to fix all the issues that testing finds. The issues that have too much cost compared to their impact are often left in the product without a fix.

But again, the primary goal of testing is not finding all the bugs but being aware of the quality and the impact of the issues.

HOW MANY TEST TYPES ARE THERE?

There are a lot of test levels, types, methods and even specific test techniques used to enhance testing. Here are a few phrases you are likely to hear often from test engineers:

Functional Testing

A test type that focuses on checking the functionality that users are likely to be using your system for. Common non-function al test types include: performance, security or qualification testing.

Unit Testing

This type of testing often consists of automated scripts embedded into the code and is usually expected to be done by the developers. This type of test focuses on the smallest functions and methods in the code.

Acceptance Testing

This type of activity checks the changes or enhancements that the user or the marketing team wanted in the system. Most of the test cases are written for this level of testing as these can be re-used to re-test the existing functionality in future releases. The customer acceptance of a release is often based on the result of the acceptance tests.

Regression Testing

Regression testing is running a sub-set of high-level tests (called regression test cases) to check that existing functionality works after a change (bug-fix, enhancement, configuration change, database change, etc.) is implemented in the system.

It is a good practice to write test cases for this type of testing as they are often re-used throughout several sprints or releases of a system. This is why the regression test suite is a constantly growing test case collection and it is the job of the test manager or test analyst to define the sub-set of test cases that are needed to be executed within each sprint or release.

Smoke and Sanity Testing

These two phrases are often confused and used interchangeably. Although both of them are done to avoid waisted time on deeper testing, they have a couple of differences. Smoke testing (also called Build Verification Testing) is carried out after a build is completed and is aimed to test previously working core functionality of the system.
This activity usually involves running a sub-set of acceptance test cases (or an automated test suite). Sanity testing (also called Tester Acceptance Testing) comes right after or together with Smoke Testing.

It focuses on changed functionalities and is checking if the expected requirements are met with the newly added changes. These tests are usually not scripted and testing is done only on a high level (to see if the changes "make sense") without actually going into too much detail. The results of both test types are used to decide whether the system is stable enough and added changes work roughly as expected to carry on with further testing.

API Testing

In modern systems the back-end and front-end layers are separated and Application Programming Interface (API) is available to allow interactions to functionalities of the back-end. API testing is commonly used to test the functionality, reliability, performance and security of the system without using the User Interface.

This testing requires specific testing tools and usually involves some scripting in writing the tests. These tests are often the part of the test automation suite because APIs serve a primary interface for the business logic and writing automated scripts for the User Interface requires a more frequent maintenance.

Web Testing

This type of testing focuses on testing requirements of the web application through a number of test types, including functionality, usability, accessibility, reliability, performance and security testing.

Security testing is one of the key areas that testers need to focus on when it comes to web testing because web-based systems are especially prone to security vulnerabilities.

Depending on the supported environments web tests also involve compatibility testing of the application functionality in multiple operating systems, devices and browsers. There are several tools that testers use when testing web applications; cross-browser testing tools, mobile testing tools, performance testing tools, security testing tools, etc.

UX/UI Testing

UX (User Experience) and UI (User Interface) testing focuses on usability, accessibility, functionality value and other measures on how the user feels when interacting with the system. This type of test often involves the monitoring of how the end-users use the system by having them use pre-defined tasks or letting them explore the system and provide feedback on their experience.

Compliance Testing

Compliance testing (also known as Conformance testing) focuses on validating whether the system complies to the required standards or regulations required for the industry.

For example, medical systems and devices produced for the US market are regulated by FDA therefore must undergo compliance testing for FDA requirements and testing results have to be documented.

CAN TEST AUTOMATION ELIMINATE MANUAL TESTING?

Many companies focusing on test automation services claim that with automating testing they can eliminate manual testing efforts. This can be true for systems or functions where there are no real user interactions – like batch processing systems or data calculation functionalities.

For products that a human will be interacting with will always need manual (human) testing. The other question when it comes to automation is its cost. Automating tests are in most cases a lot more expensive that running the test once manually. So unless you have unlimited resources and budget for automation you will have to consider the value of automation against manual execution.

WHAT ARE THE OUTPUTS OF TESTING?

Depending on the size of the project and the purpose of testing there are a number of outputs that are produced by the testing team. If you are creating a testing process for your project one thing to keep in mind is that the outputs are only valuable if they hold information that is needed and somebody actually reads them. Do not just create outputs for the sole purpose of documentation because they are only useful if they are up-to-date so the more documentation you have to more you have to keep updating.

The most important output of testing is the bug ticket which contains the full description of a found issue. Depending on the way the project is doing its bug tracking there are several tools on the market that can be used for reporting and tracking bugs. Another type of output is test cases which are most useful when several future releases are expected for the project. They also come in handy when new team members have to be trained for the project. These can also be managed with specific tools designed to manage testing.

If you want to know the functionality coverage of your testing first of all you have to have some ways to connect your functionality to your testing. This is fairly easy to do if you have a list of business or user requirements and written test cases to connect them to. Most test management systems on the market support requirement coverage analysis so it is even easier to do if you use one of these tools.

For a large-scale project, it is common to document the test strategy along with the test plan to make sure that the whole team is on the same page about the goal and focus of testing. These outputs are often re-evaluated based on major changes to the scope of the development or function design solutions.

THERE ARE MANY TESTING SOLUTIONS ON THE MARKET. WHICH ONE SHOULD I CHOOSE?

It depends on the product or solution to be tested. There are many technologies, software solutions and usage circumstances and they all require different focus in testing. The future release plans and used development model also have an impact on how testing is built up on the product.

For example, medical solutions are often created using waterfall development model which testing has to conform. Medical products often require special (compliance) testing specified by regulations and rigorous risk, security and performance testing.
A modern low-complexity web app on the other hand, is often developed with agile methodology and involving continuous releases. A solution like this will most often require a lot of test automaton experience and cross-browser testing.

SOFTWARE TESTING

There are projects that not only require test engineers but they also need thoughtful planning in order to focus test efforts on the right activities and avoid unnecessary testing. Proper planning of testing can be quite a challenge on some projects and so requires an experienced test manager. A test leader with relevant background is able to ask the right questions, focus and prioritize testing by recognizing risky areas of the system saving time and money for your project.

In case you are a manager of a development team and are just looking for a team to join your team to cover the project's testing you will need experienced testers in planning testing and expertise in the area that testing needs to focus on. In case you are a product owner looking for a development team to create the solution you will in most cases get an all-in-one service that includes development and the right sized team and expertise.

Contact Us

Ironrooster.io