

Lessons We've Learned in Software Testing

A summary of points from our informal and interactive discussion about the lessons that we have learned during our QA career; We can all benefit by the trials and errors and the experience of others, so here is your chance to partake of the collective experiences and knowledge of your peers...

- Define a standard set of terms to use as common terms. This helps avoid ambiguity.
- QA needs to be end-to-end and included in each phase of the Software Development Life Cycle.
- Other IT Managers do not need to be told that QA is lacking or needed—it's well recognized. As such, selling QA should be easy.
- Rather than adding cost and time to a project, QA should actually reduce costs and shorten the development and timeline—in addition to increasing quality.
- It is important to have a good working relationship with developers and with the business units.
- When a new project is started, it is important to have members from all areas of expertise to discuss all the ins and outs of the requirements and expectations.
- Don't rely on the client to test, even if they say they are going to be testing.
- Automate smoke tests.
- Get developers to do unit testing.
- Excel can be a great tool for testing.
- It is important to track every bug you find—even if you can't recreate it initially. If it occurred once in the system, it is going to happen again.
- It is important to really learn the business you are working in.
- It is important to use a team approach.
- In addition to following test scripts, exploratory testing can be beneficial too.
- Early in the project, define your testing scope and get sign off from key players: QA Leads, Business Leads, Developer Leads, Project Manager, etc.

- Make sure to incorporate navigation of screens into your test cases so that any one can execute them.
- When determining what automated tool to use, do not bring in an intern to do the research/recommendation. If you do, have people review the recommendation before proceeding with the purchase.
- Requirement tracking tools are only as useful as the requirements you put them into.
- Developers and business users should work together to negotiate functionality.
- When you do your initial pass of tests on a new piece of software, don't linger too much on the "honey", the happy path tests. Unless you work in a very broken software department, the happy path will probably work—it is where the programmer spent the most time. After you understand the workflow, move directly to bad inputs, trying to exit the workflow abruptly, doing things out of order. These are the things the programmer thought about less, so to scare out the bugs, do these early.
- When you get a handoff of a build of software, ask the programmer how he is feeling about the project and how it went. A lot of the time the answer is "fine" or "okay", but every once in awhile, the programmer will say, "I'm a little worried about X feature and how it will fit in with the legacy code. It was hard to figure out." Put these things high on your list to test.
- Watch out for items released in a build that are not yet ready to test, i.e. online help. You may want to test them anyway and log items as a bug as a reminder in case they are released into production.
- The QA team should limit their communications about business requirements to the business team, not the development team. Remember, QA is testing how development interpreted the requirements. The best approach is to have communications between your development lead and your QA lead and the business analysts. This also allows developers more time to code instead of answering questions.