Preparation

Best Practices of Software Testing

- Rob van Steenbergen

Continued from the last edition of Testing Circus...

After years of experience in the testing profession I have heard a lot about the 'best practices' for software testing. Sometimes people just want to share their best ideas, so you can use them too. Some stories I've overheard and noticed:

Waiting for the specifications

After making your test plan, you will have to wait until the functional specifications are ready. At that moment you can enjoy a period of rest, because without good detailed specifications you can't begin on working on your test scripts. During this time you can apply the "George Constanza" method to look as if you're working very hard. George Constanza is that "little man" from the "Seinfeld" series. His method is simple and effective: Start practicing on a look as though you are thinking of something difficult and stare just straight ahead. Create occasional variation by walking back and forth in your office with your hand against your chin to make it really look like you are thinking. And outward gaze is also good, but keep looking like you have a hard time with a difficult thing. Try it, it really makes you people think you are very busy.

Reviewing specifications

When you are asked if you want to review the specifications, accept that offer with pleasure. Especially early versions of this kind of documentation are a real laugh! You can find so many mistakes that it's just fishing in a pond full of hungry fish. Make a list in MS-Word and use a nice review template so that it looks professional. Please explicitly take notice on grammar and spelling errors and report them as quickly as possible.

The development of test scripts

If the specifications are ready, you can start creating the test scripts. For consistency in your test scripts, use the same testing technique for everything. The best testing technique is the decision table, because you can really use it for everything and this technique creates the most test cases. It is good to make an in-between progress report on how many test cases you have created per functionality and how much in total. A project manager likes to see as much test cases as possible for the test trajectory.

Test environments and the issue database

If you're still waiting for the specifications, you can think in this stage about the test environment and a database for bugs. With any luck, these things already installed and configured. In that case you just have to fill in the necessary forms and do the necessary requests to you IT department to get it ready for your project. If these are not standard out-of-the-box tools yet at your company, you should be thinking how to test in production. Setting up a test environment is an impossible task for a tester. If you are working on a new

product, there will be no users working with the product while you are testing. In this case you need no test environment

If you are testing a new version for an existing product and there's no test environment yet, it is more difficult and you will still want to use a test environment. Require this to be setup by the developers by requesting this via the project managers. In your test plan you made the assumption for this project that there is a fully working test environment available for testing. The test plan is approved by the project manager so this will be an official document for that.

If there is no bugs database in you company, don't do anything about this. Just create an Excel sheet for yourself where you keep track of the bugs and communicate the bugs by email. It is important that people solve bugs and a database is not really a goal or needed for that. You can communicate bugs by email and then you've got a nice archive automatically in your mailbox. If someone does not solve a bug you can always find your email back for proof if people start complaining that the software is not properly tested.

Always use priorities for bugs. The best way is to a clear term for the most dangerous bugs: "Showstopper" or "Blocking". For the rest of the priorities you can use vague terms such as: "normal", "serious", "priority 3", et cetera. And if the bug is not important to you, just use the priority: "cosmetic".

What did you just read?

Did you recognize parts of the stories above? Could you agree with them or even some parts of them? Please not that those parts will be the areas where you might want to study some more on testing. Maybe a colleague is telling some kind of story as you've read in this article. Run away or help him or her, because this kind of stuff is really killing our professionalism.

The message is: Keep critical of your own work and professionalism. Of course, everyone has a bad day or a project that is not going well. The main concern of the tester is to stay sharp and deliver valuable information about the quality to the stakeholders.

Have you got your own story (own experiences and learning) or did you hear about something comparable? Please write me at rob@chickenwings.nl

Rob van Steenbergen is an independent software test consultant from the Netherlands. He has over 15 years of experience in software testing with several organizations. Currently Rob is working for ThiemeMeulenhoff, an educational book publisher. The domains he has experience in are banking, software houses, government, embedded software, infrastructure and public transport. Besides projects with clients, he is involved with a Dutch test magazine as an editor, runs a website about test events (www.testevents.com), has his own blog about testing and writes articles for national and international magazines, such as the Testing Circus. Website – www.chickenwings.nl - Rob tweets at @rvansteenbergen