Becoming a Software Testing Expert – Part 3

Posted by: *Huib Schoots*
Date: Jan 4, 2012

Category: Columns

This is the third and last part on the theme "how to become a software testing expert." Part 1 "You can learn testing!" can be found here and Part 2 "What makes a good tester?" is here.

In part one I said that if you want to be an expert software tester, you must qualify yourself in many skills. That through feedback, actively seeking and making mistakes, helps you find out where you can improve. And that coaching is a great way to learn faster. In part two I stated that passion and attitude are most important. Then the skills, competencies and qualities such as proactivity, communication, social / emotional development, collaboration, fast learning, curiosity, but also the ability to apply test techniques. Knowledge to me is the least important. A good tester has test, technical and domain knowledge (in that order of importance).

Passion and attitude
"Without passion no change and no progress" said Ferry Bezem of Twynstra Gudde in this Dutch Article. "How passion for your job can lead to success" is one of the other many articles I found when I googled "passion in your work". To become succesfull, you need to find out who you are and where you get energy. To become good, you need to have passion for your profession. This article explains how to "develop" passion. In addition attitude or mentality are important as well. With the right attitude, you will be successful for sure! And don't forget: you can change your attitude!

Critical thinking
Testing for me is questioning a product in order to evaluate it. A tester does this by learning everything there is to learn about the product and analyze this information. Critical thinking is an essential competency of a tester. But how many testers train it? On a well known Dutch book site I searched for books on critical thinking. What struck me was that there are several books on critical thinking for medical personal to find. And that sounds actually quite logical: medics diagnose patients, testers do the same for their "patients": the test object!

Waste

We write comprehensive test plans that are never read. We use extensive templates to make sure nothing is forgotten (and we do not have to think too much). We do product risk assessments but subsequently we don't do anything with the risks identified. We often do not use any test techniques. Why?! I think because many people simply don't know how. I often ask why test techniques are not used. The most common answer is: because there is not enough time!? But these techniques should make our testing more effective and efficient, don't they? As a tester you should be dreaming these techniques. You should OWN them!

Adaptive?
Every novice tester in the Netherlands starts with TMap or ISTQB. With that fact in itself is nothing wrong. A three-day class where the basics of software testing are taught is a good start. But too often that's it. We like to guide our testing with process models: ISTQB and TMAP are full of them. But the reality is often more complex and testers get in trouble when the situation is slightly different as they are used to. The choice of a particular tool, technique or method depends on the context. TMap Next claims to be an adaptive method, but do you remember how much time was spent on this topic in your class? And how adaptive are you yourself? Do you use the same template over and over? Do you simply copy the test plan for the next release using a search and replace?

Learn, practice and training
I learned very much by critical reading. There are many free software test magazines and there are many great books on software testing. The internet is an excellent source of (often free) information. There are many videos available of presentations at meetings and conferences. On my blog I keep a list of great resources for testers. Lynn McKee has a similar list. Or start reading blogs, twitter can inform you about new interesting blogs and other things worth reading.

Try weekend testing or a testing dojo to practice your skills. Join a local software test community or organize your own gathering with a group of testers who want to meet and learn. At several employers I introduced intervision meetings for testers. I also founded DEWT with a group of passionate colleagues: like-minded people who like to challenge and inspire each other. We spend many hours discussing our profession and we certainly do not always agree. We discuss and practice in a safe environment and learn from each other.

Try something different
Have you been a software tester for years and are you planning to spice up your resume with another certificate? Then maybe the BBST training is something for you. Let me warn you in advance: this training will give you no certificate and it requires you to study and do exercises almost daily for a month. If you really want to learn, you need to invest a lot of time. Malcolm Gladwell claims in his book Outliers that the key to success in any field is practicing 10,000 hours. Try a training that changes your view on testing: rapid software testing for example. This training has changed my view of my profession and inspired me tremendously. Visiting conferences can be very inspiring and instructive. Join TestNet or another (online) testing community like The Software Testing Club and get inspired. You have plenty of choice!

Learning from others
I enjoy working with others. As I observe, I ask questions or explain my view on the situation. We still can learn a lot from our colleagues in projects … and vice versa! And I'm not just talking about fellow testers. A programmer can build a tool or script in only a few hours that can save testers days of work. By doing unit tests together, we gain insight in what developers test and how they do it. Try working together: pair programming or pair testing can be a powerful tool. Unfortunately, it has a negative image, because it seems to double the effort.

But I think it can be beneficial. The added value is mastering the details of a job easier, fewer mistakes, faster learning, team building, cooperation and fun.

Writing and presenting
Blogging, writing columns or giving presentations forces you to structure your thoughts. In addition, your stories or presentations will trigger reactions by others. By taking this feedback seriously, can you sharpen your thoughts and learn again.

Finally
Ask yourself what you can do better often. Evaluate! Ask for reviews on your products regulary. Also ask for feedback from your colleagues on the process, your skills and your performance proactively. Furthermore, my advice is simple: you need a lot of practice to become an expert. So what are you going to do tomorrow?

Huib Schoots sees himself as a context-driven tester. He is team manager testing at Rabobank International and board member of TestNet. He is a member of DEWT (Dutch Workshop on Exploratory Testing), the student Miagi-Do School of Software Testing and maintains a blog on http://www.magnifiant.com/