13

March 2011

testing experience

The Magazine for Professional Testers

printed in Germany

print version 8,00 €

free digital version

www.testingexperience.com

ISSN 1866-5705

Testing @ Domains –

How does Finance, Automotive, Medical etc test?
Do we have to take care of the domains?

iqnite DEUTSCHLAND 2011 Conferences Special

Handicap of a head start

by Bert Wijgers

In order to test effectively we have to know how the application under test will be used. Domain knowledge from prior experiences can help us to determine probable scenarios and associated risks. It can, however, also be misleading. Domains change over time and every organization does business in its own way. Therefore we should know about the business rather than about the domain.

Does the testing of software differ from one domain to another? And if so, how important is domain knowledge in dealing with those differences? In this article, I will investigate the validity of the assumption that domain knowledge is an advantage for a software tester.

Software as a tool

In some cases software is the product. This is true for games. The product is a collection of files that delivers value directly to the user. In other cases software is part of the product. This is true for mobile telephones, navigation systems and pretty much every other electrical device that is produced nowadays. Often, however, software is not part of the product, but it is only there to support the delivery of a service. This is true for information systems in banks, insurance companies and public organizations. Every thinkable organization uses software to support its business. This kind of software delivers its value directly to the professionals and only indirectly to the customers. This kind of software is a tool.

In the remainder of the article, I will talk about testing software that is used as a tool. It is this kind of software that is used by professionals in different domains.

Domain knowledge

It helps to have experience with specific applications. You know the menu structure. Perhaps you've made a few mistakes in the past from which you have learned. In other words, you are past the steep bit of the learning curve. Some of these applications are domain specific, which means that domain knowledge is built into them. However, having knowledge of such an application is not the same as having knowledge of the domain.

Domain knowledge is knowledge about facts and rules in a distinct line of business. To what extent do we need domain knowledge to test software?

Testing and checking

Regardless of the testing methodology, there are two basic styles of software testing, confirmative and non-confirmative. The goal of confirmative tests is to show that software does work as it is supposed to. The goal of non-confirmative tests is to show that software does not work as it is supposed to. These are very different goals that require different means.

Confirmative testing requires documentation of some sort. This documentation tells us how the software should work under specific circumstances with specific inputs. We check this. This testing style is often done in a scripted way. Since confirmative testing is based on specifications, it should be done objectively. Prior experience in the same domain is more likely a hindrance than an advantage.

Confirmative testing is done in the earlier stages of software development. Once the software does more or less what it is supposed to do, testing can take a more non-confirmative character.

Non-confirmative testing requires imagination. We imagine what could happen and observe the way in which software reacts to different inputs and circumstances. Then we judge whether this reaction is adequate. In order to do this, we need an understanding of the way in which the software is used. This testing style is often done in an exploratory way. In order to be effective at non-confirmative testing which involves thinking up and executing plausible scenarios and correctly judging the software's reaction, some domain knowledge is needed.

This knowledge can be acquired by talking to business representatives from the organization. It is dangerous to rely on knowledge acquired in previous assignments.

Facts and rules

Let's suppose you have worked as a tester for an insurance company. During this assignment you have acquired some knowledge about the insurance domain. Your next assignment is in another insurance company. Because of your domain knowledge you were the one that got the job. After a while you realize that a lot of things you knew about the old insurance company don't apply to the new one.

An insurance company offers numerous different insurances with different conditions. Someone working for the insurance company should know as much as possible about all the different insurances, especially in a role where this knowledge is needed in communication with clients or colleagues. When testing the software used in this insurance company, it might be handy to have extensive knowledge about all the different insurances. However, since you're not in direct contact with clients or insurance professionals, it is probably enough that you know where to find the descriptions of the different insurances and their conditions. Knowledge about the specifics of insurance policies will not help you if you need to test software in another insurance company with different insurances.

Before testing software you should explore the requirements, and not only the ones that are written down. You should try to get to know your user group. Although there are similarities between companies in the same domain, there are also differences. Even within companies there are differences between departments. Specific knowledge, based on user groups you have encountered before, might cloud your vision.

The facts and rules of a domain are not fixed. At a specific level they vary between, and even within, organizations. At a more general level they may change over time. Regulations, and even laws, are not written for eternity. Therefore you need to verify domain knowledge you have acquired in previous assignments. In doing so, you just might learn something new.

Software quality

Testing can never be complete. When software is actually used by professionals in their daily work, combinations of circumstances and inputs will occur that have not been tested. In order to test the optimal subset of all possible combinations, we need a genuine interest in the software and its users, some imagination and, of course, our testing skills.

Logically, we can never reach the level of domain knowledge that the actual users of the software have. This is because we have our own domain, which is software quality. Software quality manifests itself in the interaction between users and software. Knowledge about the use of software is part of our domain. We need to know what users do and how they do it.

As software testers we operate in between business and technology. To do our job well we have to know them both to some extent. We need to know more about the technology than business people and more about the business than technical people. There is only one sort of domain knowledge that really makes us better at our job and that is knowledge of the software testing domain.

› biography

Bert Wijgers
is a test consultant with Squerist, a service company in the Netherlands. Squerist focuses on software quality assurance and process optimization. Its mission is to inspire confidence through innovation.

Bert holds a university degree in experimental psychology for which he did research in the areas of human-computer interaction and ergonomic aspects of the workspace. He has worked as a teacher and trainer in different settings before he started an international company in web hosting and design for which a web generator was developed. There he got the first taste of testing and came to understand the importance of software quality assurance. Bert has a special interest for the social aspects of software development.
He uses psychological and business perspectives to complement his testing expertise.

In recent years Bert has worked for Squerist in financial and public organizations as a software tester, coordinator and consultant. He is a regular contributor to Testing Experience.