March 2011

testing experience

The Magazine for Professional Testers

Testing @ Domains –

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

iqnite DEUTSCHLAND 2011 Conferences Special

printed in Germany
print version 8,00 €
free digital version
www.testingexperience.com
ISSN 1866-5705

The key principle of testing "Testing is context dependent"

by Nisha Dhiraj Kale

Today's test teams have to cope with the increasing complexity of the systems under test, while often schedules, budgets and team sizes have not been scaled accordingly. Not every part of the tests can be automated and, due to limitations in time and budget, only a small fraction of all use cases can be covered in manual testing. This article describes the domain knowledge necessary for a software tester and its advantages, particularly with regard to helping the testers manage the complexity issues and select and prioritize the tests according to the application under test.

Software Testing Principles:

Software testing is an extremely creative and intellectually challenging task. When testing follows the following principles given in the ISTQB® Foundation Level syllabus, the creative element of test design and execution rivals any of the preceding software development steps.

Testing shows the presence of bugs

Testing an application can only reveal that one or more defects exist in the application. However, testing alone cannot prove that the application is defect free. Therefore, it is important to design test cases which find as many defects as possible.

Exhaustive testing is impossible

Unless the application under test (AUT) has a very simple logical structure and limited input, it is not possible to test all possible combinations of data and scenarios. For this reason, risk and priorities are used to concentrate on the most important aspects to test.

Early testing

The sooner we start the testing activities, the better we can utilize the available time. As soon as the initial products, such the requirement or design documents are available, we can start testing. It is not uncommon for the testing phase to get squeezed at the end of the development lifecycle, i.e. when development has finished, so by starting testing early, we can prepare testing for each level of the development lifecycle.

Another important point about early testing is that when defects are found earlier in the lifecycle, they are much easier and cheaper to fix. It is much cheaper to change an incorrect requirement than having to change a functionality in a large system that is not working as requested or as designed!

Defect clustering

During testing it may be observed that most of the reported defects are related to a small number of modules within a system that contain most of the defects found. This is the application of the Pareto Principle to software testing: Approximately 80% of the problems are found in 20% of the modules.

The pesticide paradox

If you keep running the same set of tests over and over again, the chances are that no further new defects will be discovered by those test cases. Because as the system evolves, many of the previously reported defects will have been fixed and the old test cases do not apply anymore. Anytime a fault is fixed or a new functionality added, we need to do regression testing to make sure the new changed software has not broken any other part of the software. However, those regression test cases also need to change and reflect the changes made in the software, in order to be applicable and hopefully find new defects.

Testing is context dependent

Different methodologies, techniques and types of testing are related to the type and nature of the application. For example, a software application in a medical device needs more testing than a games software. More importantly a medical device software requires risk based testing, be compliant with medical industry regulations and possibly specific test design techniques. By the same token, a very popular website needs to go through rigorous performance testing as well as functionality testing to make sure the performance is not affected by the load on the servers.

Absence of errors fallacy

Just because testing didn't find any defects in the software, it doesn't mean that the software is ready to be shipped. Were the executed tests really designed to catch most defects, or where they designed to see if the software matched the user requirements? There are many other factors to be considered before making the decision to ship the software.

In software engineering, domain knowledge is knowledge about the environment in which the target system is to operate, and every application applies different methodologies, techniques and types of testing which are related to the type and nature of the application. Different software products have varying requirements, functions and purposes, so the same test cases and strategy cannot be applied across all applications. For example, a software application in a medical device needs more testing than games software. More importantly a medical device software requires risk based testing, be compliant with medical industry regulators and possibly specific test design techniques. By the same token, a very popular website, needs to go through rigorous performance testing as well as functionality testing to make sure the performance is not affected by the load on the servers.

Testing as a job requires one to be proficient with technical knowledge and experience of the development life cycle, functional domain, and technology and tools used for testing. In addition, it requires individuals to demonstrate a high capability for logical analysis, deduction, reasoning, communication, and presentation skills.

Business domain knowledge is important as it ensures that the tester understands the perspective and needs of the users for which the software is being built. Good business domain knowledge ensures that testers are more likely to identify missing requirements and incomplete requirements or stories that might need extra details; they can identify relevant test requirements/scenarios and create effective test cases. Communication between end-users and software developers is often difficult. They must find a common language to communicate in. Developing enough shared vocabulary to communicate can often take a while.

When we test a particular product, we have two kinds of axes where Y axis denotes the technology and where the X axis denotes the domain. Domain is common for all technologies. Domain experts can write the business driven scenarios to execute the actual business, whereas the technology experts will be able to write scenarios on the technology, which makes their scope narrow.

The advantages of domain knowledge:

1. Reduces the training time: A person can be productive quicker than a person that has zero domain/industry knowledge. So it adds value to project/product.
2. Good knowledge of the functional flow (workflow), the business processes and business rules: It will help to better understand the product requirements.
3. Good knowledge of UI features: It will help the person in improving the look & feel of the UI as well finding more bugs at an initial stage at the UI front-end.
4. Good knowledge of back-end processing: Knowing how effectively/efficiently the data/code is handled.
5. Domain knowledge is also important for defect triage: Knowing how the application will likely be used and how it is expected to perform, will tell QA whether a given defect is trivial, significant, or in fact critical.
6. Terminology: Reporting in the language of the business, resulting in good rapport with the business team(s).

Are you going to test the BFSI applications (Banking, Financial Services and Insurance) just for UI or functionality or security or load or stress? You should know what the user requirements in banking, working procedures, commerce background, exposure to brokerage etc. are, and you should test the application accordingly. Only then you can say that your testing is enough. This is where domain experts are needed.

Let's take example of my current project: I am currently working on a stock market application, where I need to know the basic concept and terminology used in stock markets. If I know the functional domain better, I can better write and execute more test cases and can effectively simulate the end user actions. This is a distinct advantage. As testers we need to apply our domain knowledge in each and every step of the software testing life cycle.

While testing any application you should think like an end-user. The user who is going to use your application may have a good understanding of the domain he is working in. You need to balance all these skills so that all product aspects will get addressed.

If you have the above skills, you will certainly have the advantage of domain and testing experience, and therefore have a better understanding of different issues and deliver the application better and faster.

Case Study: Testing BFSI applications (Banking, Financial Services and Insurance)

Testing has always been challenging in the software development life cycle of a product. Testing financial products requires more domain knowledge, and testing in general is very critical in the life cycle of any software development activity. Compared to other domain products, financial products pose significantly more challenges due to their complexity in design, development and testing, impact on end users, and safety characteristics. This in turn means that the test engineers face multiple challenges while testing the financial software to deliver high quality to the customer.

Some of the challenges faced while testing in the financial domain are:

Id	Challenge	Explanation
1	Domain and System Knowledge	Lack of domain and system knowledge will result in inadequacy of testing. Test engineers with testing knowledge will not only suffice testing the product as a whole but also must know the complete functionality, usage scenarios, workflows, environment, standards used, configurations, regulations, interoperability, domain tools etc.
2	Application Workflow	Testers must have good understanding of usage and workflow use cases of the product as used by the end users at there sites. Bugs raised on invalid use cases may be rejected which leads to waste of effort in finding and reporting the bugs.
3	Test data/ datasets	Lack of availability of standard test data/datasets during testing will lead to insufficient test coverage. Test engineers must have standard datasets classified based on various parameters stored at a shared central repository and make it available for testing to ensure adequate coverage of tests at various test levels.
4	Financial domain standards	Insufficient knowledge of domain standards will result in poor quality of testing.
6	Specialization tests i.e., Image quality, Performance, Reliability, interoperability, Concurrency, Hardware etc.,	Financial products have to be tested for some specific specialized tests to meet the quality goals of the product. Test engineers must be experienced fully trained and possess special skills to perform these tests.
7	Process Compliance - ISO, CMMI	Financial devices are compliance to certain standards to certify that they are fit for intended use. Test engineers must be trained on these standards to perform their job well in qualifying the product during testing.
8	Exposure to end users/Application specialists and Site.	Lack of exposure to end user use cases, interactions with Application specialists and non-awareness of sites will result in insufficient knowledge of product and poor quality of tests.
10	Unstable releases	Developers check in software and test team is made responsible for building the software and verifying it. Developers did not do sufficient unit testing before checking in. The code checked in was also not reviewed in many cases. Test engineers were unable to proceed on testing as per their plan as they faced many issues, which should have typically been caught during reviews and unit testing.

Conclusion:

Domain knowledge plays an important role in software testing, as one of the software testing principles says "Testing is context driven". Testing is always done differently in different contexts. Domain expertise is important in software testing because the person who has domain knowledge can test the application better than others.

› biography

Nisha Dhiraj Kale

As an ISTQB® certified testing professional I am currently working as Associate Consultant at Capgemini Pune for their Client CLSA. I have 5 years of testing experience in software testing. During this time, I have worked on different projects ranging from client server to web-based applications. I have knowledge of the lending, capital market and banking domains. I am an engineering graduate and also hold a diploma in Business Management from Symbiosis Pune.