By Derk Masselink

Your way of testing the world of tomorrow

Skills needed by future testers following today's trends

Introduction

We see the prices of hardware, connectivity, and software going down. Meanwhile the number of information systems is increasing. Tooling becomes better and some open source tools are available for free. Computers get a little bit clever by using machine learning and computer vision. What will be the effects on IT and on the role of the tester in the development process? I will try to extrapolate some of the trends and see what they might bring us. What will be the impact on the role of the tester and what skills will he or she need?

Tooling increases maintainability

"We moved to more technical skills and more user friendliness simultaneously"

Last year I was working on an automated software test in JMeter for a new middleware system for an insurance company. We had some quite good tests in JMeter, but it was very difficult to reuse test components and to maintain them. At the same time I was experimenting with Fitnesse. Fitnesse is a wiki-like automated test tool where you program the general behavior of the tests in Java classes and then use this to easily create test cases in a user friendly format on a wiki.

The problem then was that I did not have enough Java skills to make a valuable test; so I stuck with experiments combining Fitnesse and JMeter.

In the meantime, the team started complaining about the difficulties in maintaining the JMeter tests and about the lack of test capacity. So we discussed the possibility of using a easier to maintain test tool like Fitnesse.

After a while, an experienced Java architect and programmer who was part of our team helped to make a proof of concept of a Fitnesse test. In a few days, he made an automated test for one of our transactions. He created some basic java classes that were very helpful and he explained how we could use Fitnesse. In a few weeks, our whole team was up to full speed in using Fitnesse and the maintainability and ease of use of our automated test increased a lot. The advantages are that any team member can play a role in making the test and the tests can be reviewed easily by the business. In fact, because of their wiki character it is even possible to write down the tests and the requirements on the same wiki pages.

So, what can we learn from this? I think the main lesson is that we can learn a lot through cooperating with developers and other people with technical or business skills. Of course, to adapt to new kinds of work we have to acquire new skills and be open to learn them.

Another important lesson is that with the introduction of Fitnesse we made two shifts – one to a more technical type of work by programming classes in Java to execute our tests, the other to a more simple and user friendly type of work by keeping track of our test cases in a wiki in a user friendly way. This made it extremely easy to review and maintain the test cases.

So, the paradox is that we made two opposite shifts by introducing one new tool! Things became more technical and less technical at the same time. And the test role is becoming more technical and less technical at the same time as well! Unless, of course, you decide to split the test role; but that is not a good idea.

One striking feature on both the technical and the non-technical side is the increase in maintainability.

For sure, tooling is a trend that has been around for some years and I expect it to grow in the future. I think that we testers can learn a lot from developers who work with tools to check the code coverage of unit testing and tools to manage their code base, and do nightly runs of the testing. In many situations, testers will be able to profit from tools that have already been introduced by developers and can be used by testers as well.

What other trends do we see today?

Extrapolation of trends and their effect on it and testing

"Meaning and value replace correctness and tests will replace requirements"

Let us look at some possible trends that we can expect in IT and what the effect may be on testing. I cannot foresee the future but I hope you will recognize some of the assertions in table 1.

An analogy

"Testing will take place in a large network of systems"

When you take a look at, you can judge for yourself whether you agree with these trends and their effect on the work of a tester. What is the general picture that arises from this data? Can we find a metaphor for the things we can expect? Well, maybe we can compare it with a beehive. Of course, bees just do what they have to do. But suppose we had to program the bees how to behave? How would we test that? And suppose, just like the bees, the flowers and other parts of nature have to be programmed as well and anyone can program these flowers, trees, and so on.

We would probably define some very specific behavior for one bee and make a test set for that. However, we would not be sure whether that was the best possible behavior. Also, we would define some tests on the group behavior of the bees, such as the passing of information on sources of nectar. And we would define goals and sub-goals in order to judge whether the tested and defined behavior really leads to our goals. In the meantime, flower developers and forest developers have changed the behavior of the flowers, number of flowers, and routes to the flowers. So we have to adjust our test cases. After a few of these changes, we make a tool to generate test cases out of the sub-goals. To be sure everything works fine, we need a test environment that contains flowers and flower simulators. On the other hand, the flower developers need bee simula-

Trend	Why	Effect on IT	Effect on testing
Decrease in transaction costs and costs of software distribution	We see nowadays apps being sold at prices below 1.00 euro. This is possible because the transaction costs of the purchase have evaporated and because software is being sold in large quantities which makes low prices profitable	Many systems will just specialize in one specific task and be part of a larger chain of systems that enable the overall functionality. In some market segments, specialization may increase rapidly.	The chains and networks will become much larger. Functionalities will become redundant and some testing will be executed on many paths through a network.
Increase in the number of software solutions in our activities	Better tooling to create software and due to the decrease in transaction costs	Probably the number of suppliers will also increase. It is not possible to keep track of all these systems. Therefore standards will determine the way systems communicate, and as long as systems meet this standard they can join	Testing over a chain does not mean testing a fixed set of systems but testing a flexible and changing set of systems. At the same time, the number of necessary test cases will increase. This can be achieved by using tools to generate test cases out of principle statements. Many information components and software applications will be outside the direct control of the testers. Quality assurance on the whole chain will be addressed through good relationships and shared practices with suppliers and customers.
Decreased hardware costs and increased use of hardware in consumer products	Just the trend that has been going on for more than 50 years now	Objects like shoes can have their own sensor, computer and network connection. A tremendous amount of information will become available (Big Data)	Test environments have to include simulators for physical objects
Decrease in the functionality life cycle period for released software	Unsure	Risk of incompatibility between systems becomes larger	Overall up-to-date integrated test environments become more important
The amount of information available to software solutions will increase	Ongoing trend	The whole picture of data becomes important and single data elements have no meaning without large sister data or without context. To deal with this data we need advanced data mining and machine learning techniques	Some information is no longer true or false but will, instead, have an extent of usefulness or effectiveness. The meaning and the value of the data become important.
Generated software and machine learning	Although expectations of the past have not come true, this will probably increase, especially for some sub-domains	Setting out principles and modeling become more important than actually writing code	If the functionality has not been written but instead was learned; testing becomes an important tool to fix the functionality, define constraints, and measure quality. In fact, a kind of testing is part of the standard machine learning approach
Moment of test design in the development process	In the past, test-driven development seemed like a miracle. Once we worked with Fitnesse we could actually develop tests in parallel with the software and, on some occasions, the tests were ready first		In the future, test cases may actually become the requirements and be realized before any coding will be done. The responsibility of the tester will shift to business value instead of correctness

Table 1. Trends in IT that may affect the work of the tester

tors in their test environments. In practice, most parties will publish their test environment and a worldwide acceptance test environment will arise. Of course, the quality of the programmed flowers is just as important to us as the quality of our bees. So we will agree on the way we determine the quality with our direct suppliers and customers, and we will continuously publish the measured quality of our bees. Bees cannot do things wrong or right but they can do things more or less effectively.

A new role

"Due to the complexity and velocity of change the tester will have to make decisions without consulting the business"

In the near future, technical skills will become more important. Testers will be programming and will have to understand how to test on system interfaces instead of user interfaces.

In the long term, the role of the testers becomes one of understanding the (changing) model; of attributing meaning to abstract things, and of estimating the value of a solution. Everything is changing all the time, so the most important skill will probably be the ability to learn and help others to learn. After that, understanding and prioritizing based on an estimation of feasible business value become important. There will be no time to discuss complex abstract solutions with the business owner. The tester himself will have to decide what to work on, in combination with tooling that will be developed to support these decisions.

Social aspects

"You will be part of multiple changing teams"

So, do these changes in the environment, the development process, the tooling, and so forth have an effect on the social aspects of the work, and will this require different skills?

Well, if we look at the continuously changing large group of systems made by a changing group of suppliers, then we can imagine that it will be helpful to build a kind of reputation and use that to predict the upcoming changes. The reputations are about groups of systems and suppliers. So the social intelligence needed for this has to deal with a mixture of (groups of) people and systems. Some people working at companies with a large IT department may already today experience the need to navigate through a forest of projects, improvements, and bug fixes where it is important to have a good sense of urgency and priority. Others depend on you and you depend on others. Discriminating between major and minor issues keeps the overall complex healthy. But to know what you have to do today, you have to know what is going to fail and what not, and what is going to change and what will stay the same.

At the same time, you are part of a large complex system. You have a small very specialized task. Your work may become very different from the work of others in terms of concepts and actions. On the other hand, the automation, support by tooling, and possible standardizations may lead to congruence in the kind of work. It is very difficult to predict. The danger of over-specialization and losing touch with other topics can be overcome by actively changing roles or scope, and by activities not directly related to your daily work. Since you will be learning all your life, the costs of changing jobs will not be as high as they might seem.

What about the team you are working in? Since everything is changing all the time, your team will be changing as well. And you can therefore expect to be part of more than one team.

So, you will have a lot of social contacts and possibilities, but you will have to cope with new people all the time unless you take a very abstract position and focus purely on the systems.

Conclusion

"Today is where future and past meet"

So, this article may look a bit futuristic. However, some of the trends mentioned are already practiced by some people today. But let us try to order things in a time perspective. For the coming years, I expect that automated testing will become more and more important to ensure the desired functionality in ever increasing and continuously changing systems. This means testers will have to learn how to automate or they will have to cooperate very closely with developers who can do this for them.

Once you have automated tests, you can clearly publish them in a wiki (like with Fitnesse) and let other people review them. Once this is done, why not use the test cases as the requirements?

Further away, let us say after 20 years, I expect that the tested functionalities will have become so complex and widely spread over a network of interrelated systems that meaning and value will become as important as correctness. Redundancy in paths to achieve a transaction will increase in the same way as you can take multiple routes when you travel to a far-off location for holidays. Like navigation for your car, you as a tester will be probably be one of the first to have navigation over the systems network.

In the end, I expect you will generate test cases out of selected goals that you as a tester will have to guard. ■

> about the author

Derk Masselink was born in Delft (Netherlands). He studied Technical Physics at the Delft University of Technology and finished his master's in the Computational Physics Department. He worked as a software developer in C++, then as a production planner in the food sector, and after that for 15 years in software testing and quality assurance. For the last years, his main activities have been test management and test automation for the insurance and financial sector, as well as the navigation sector.