June 2011

testing experience

The Magazine for Professional Testers

printed in Germany

print version 8,00 €

free digital version

www.testingexperience.com

ISSN 1866-5705

Improving the Test Process

Plan, do, test and act

by Bert Wijgers

Testing is an activity that can lead to improvement of the software creation process. We have to go one level deeper in order to learn how to improve the software testing process. We can test the products of the testing process. It is difficult to assess process quality with metrics, but we can assess the craftsmanship of individual testers. This is done by testing them.

An obvious starting point for any quality improvement activity is what is commonly known as the Deming circle: plan, do, check and act. First you make an improvement plan and then you execute it. Then you check whether the plan has worked. If it didn't, you make a new plan. If the plan worked more or less, you act, that is, you make adjustments to make it work better. If the plan worked really well in the first place, you can start to make a new plan to improve even further. Whichever way, the circle starts again at some point and this should continue indefinitely.

Deming always referred to this circle of quality improvement activities as the Shewhart circle. Deming (1986) made only one modification; he replaced "check" with "study". Unfortunately, this modification did not catch on, probably because "plan, do, study and act" doesn't sound as good as "plan, do, check and act". Therefore I propose an alternative that is in line with Deming's idea that the third activity is much more than just checking and that has the same rhythm and rhyme as Shewhart's circle: plan, do, test and act.

Software testing is part of software creation. In agile projects this is made explicit; testers work in multidisciplinary teams with developers and designers. In waterfall projects test teams seem to have a more independent position, but still they are part of the bigger picture. Software cannot be built without a plan or at least an idea, it cannot be tested before it has been built, and it cannot be released before it has been tested and defects are solved. Even in test driven development, where the tests are designed before the software is built, eventually the software has to run in order to pass the test. When testing, building and designing become intimately interwoven, the quality improvement circles become shorter. However, the same activities remain; plan, do, test and act.

The quality improvement circle can be identified at different levels. So, if we want to improve the test process, we have to make a plan and do it. After that, we test and, if necessary, we act upon the test results. The approach is always the same, but it can be applied at a different levels.

Circles within circles

Dynamic testing, that is, testing software by executing it, is part of the software quality improvement circle. First a plan is made that we can call a functional design; in fact, any kind of formal or informal documentation that is meant to guide developers can be considered as a plan. Based upon this documentation the software is built; the plan is executed. Then it is our turn.

Part of dynamic testing is checking. Whenever we do tests to confirm that information systems behave according to specifications, we are checking. But testing is more; another part of testing is non-confirmative. These are the kind of tests that are meant to find defects. This is a very different goal that requires different means. Testers who do non-confirmative testing need not only be good observers, they also need to be imaginative. Both kinds of testing, confirmative and non-confirmative, are valuable and they complement each other.

Static testing, in the form of documentation reviews, is part of the documentation quality improvement circle. Since documentation is used as the plan of the software quality improvement circle, reviewing it is a very cost effective type of testing.

In short, testing is the critical examination of products from the design activities and the building activities. If we happen to find a defect, we dive into it to pinpoint the cause and to estimate the effects. Through testing we contribute to the quality of the software only indirectly; the defects we find will be fixed, which makes the software or the documentation better.

Products and preconditions

Products from the testing activities have to be critically examined as well. These products include test scripts and issue reports. Together these products give an idea about the quality of the products under test. To assess the quality of test scripts and issue reports, we have to turn to the users of these products. Product

quality only manifests itself when a person interacts with the product. Test scripts are used by testers, issue reports are used by developers and designers.

When the products are up to the expectations of the users or when the goals of the user can be achieved, quality is perceived. Even when a product is well made, the quality can be perceived as low, for example, when the product is used by someone who expects something different or by someone who chose the wrong tool to achieve his goals. On the other hand, when a product is not well made, high quality will never be perceived, no matter which person uses the product. Testing is a way of finding out whether a product is well made. The quality of a product can only be judged by the actual users, when they use it.

One way to assess the quality of a process is to assess the quality of its products, but we also have to keep an eye on the clock. Since testing can never be complete, the optimal subset of all combinations of inputs and circumstances has to be tested. An important part of the quality of the testing process is in the choices that are made. Unfortunately, these choices can only be judged after the software is extensively used in the production environment. Defects in the production environment are input for test process improvement activities.

Apart from a sufficient amount of time, a tester, like any other professional, needs good materials. One of the things testers need to do their job is knowledge of the software and its use. This knowledge is transferred by means of formal and informal communication. More often than not this communication is documented to some extent. The quality of this documentation is very important for the quality of the testing process. It is good practice to review the documentation that goes with the software. Since documentation is never complete, all other forms of communication about the software and its intended use are important as well.

Another thing that testers need is a good set of tools. Management is responsible for a well equipped test environment. Without a good test environment the quality of the test process will falter. The most important testing tool, however, is the mind of the tester. A tester's mind needs to be challenged on a daily basis, otherwise it may turn blunt. The negative impact of repetitive tasks on the sharpness of a tester's mind should not be underestimated. Tooling can be used to free testers from this burden.

People over processes

There are big differences between testers with regard to efficiency and effectiveness. Given the same test object and documentation they will produce very different test scripts and issue reports. In order to guarantee the quality of the testing process, you want to hire the testers that find the most defects and that find the most important defects first. This can be done quite simply by testing them. Give them a piece of working software and some sort of documentation to go with it and see what happens. Make sure you know this particular piece of software and the defects it contains very well. If you need a tester to do mainly confirmative testing, you should look for good reading and observational skills.

Once testers are part of the team, they will not be judged and rewarded by the number of defects they discover. This would only motivate them to go for easy bugs and to log variations of the same bug separately. Measuring the performance of individuals and teams is a tricky business because you may end up with beautiful numbers and ugly results. When testers are too busy with their bug counts, they will not undertake any process improvement activities.

Bug counts can be used in coaching sessions, but only to help testers to become better at their job. To make clear that bug counts have nothing to do with rewards, it is advisable to coach testers on the basis of their performance in a controlled setting that is not part of their daily working space. Testers should be tested at regular intervals and make improvement plans based upon their performance. Another way to improve the performance of individuals is to let them work in pairs. In doing so, they will learn to reflect on their way of working.

Individual professional growth and test process improvement require roughly the same activities: plan, do, test and act. One thing to keep in mind, though, is that better is not always good. Testers need some freedom to follow and develop their intuition. In doing so, they will inevitably make mistakes. As long as testers are allowed to make mistakes and learn from them, the test process as a whole can improve as well.

Reference

Deming, W.E. (1986), Out of the crisis, MIT Press, Cambridge MA, USA.

> 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 had 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.