Erik van Veenendaal

Back to Basics

Tool Selection and Implementation

Column

The Initial Idea

In preparing for my column, I began writing about changes to the tool landscape as a result of agile. Many new (freeware) tech-oriented tools now exist to support component/unit testing, measure code coverage, and support continuous integration. Further, test automation has been pushed to the forefront of our attention due to the need for regression testing. Finally, new project management tools have become available, providing functionalities such cards, task boards, and burn-down charts. This column was intended to be a sort of follow-up to a previous *Testing Experience* column in which I discussed so-called "forgotten tools" [1].

What Happened

However, while elaborating on the initial idea, I encountered some unfortunate experiences in my projects with (test) tools, each of which I will describe briefly below.

A. A medium-sized local organization was in search of a configuration management tool. Tool requirements were defined only for functionality (nothing on non-functionals, nothing on the vendor, nothing on the platform, etc.). A quick internet search was done by the two-person selection team. When they presented their recommendation, only the cost of tool acquisition was taken into account (no training, consultancy, or maintenance costs were considered).

B. In a medium-sized multinational organization, a test management tool was acquired. The implementation was stated to be very simple by those responsible for the selection – everyone, they promised, would be able to work with this great new tool in 15 minutes. One person set up the tool and off we went (no training, no processes, no consultancy). I then spent hours trying to find and upload documents and emailing back and forth with the tool coordinator, conveniently located in another country. After a while, it became clear I was not the only one wasting effort. Only once an expert was involved could the problem be solved quickly and easy. All security settings had been set up incorrectly, and as we now know, some basic tool training would have helped as well.

C. Finally, in a large multinational organization, requirements (user stories) were still being documented by means of Microsoft Word. Those who understand anything about requirements (and their

attributes) know this is no longer considered a good practice (if it ever was!). Change management was a mess, and it also was very problematic in the transformation from waterfall to agile. There was a clear need and a business case could easily have been established. But instead of doing so and launching a tool selection process, they searched on the internet for a freeware requirements management tool. They quickly decided on one (without any requirements documented upfront). As could be expected, the implementation thereafter was big disaster and total failure.

These are not made-up stories, stories from long ago, or stories from small start-up organizations. On the contrary, these are stories from actual projects today, taking place in large, "mature" organizations. Undoubtedly, this issue of *Testing Experience* is loaded with great papers on tools, how brilliant they are, and how they can be used. Strangely enough, we rarely read papers on failure stories. Am I being negative? Is it just a coincidence that this happened to me in the last few months?

Selection and Implementation

A structured selection and implementation is required – this remains a critical success factor for achieving expected improvements to the testing process. All of the case studies presented above would have largely benefited from this. Although it is taught in many courses and we all know this is the way it should be done, real life is often different from theory. Therefore, using the power of repetition, I will summarize briefly the required activities when performing tool selection and implementation.

Firstly, as part of a tool selection process:

- identify and quantity the problem: is the problem in the area of control, efficiency, or product quality?
- consider alternative solutions: look for alternatives to tools in both the test and development process
- prepare an overall business case with measurable business objectives, considering costs (see Table 1) over the short- and long term, expected benefits, and payback period

Initial Tool Costs	Recurring Tools Costs
Knowledge acquisition	Tool ownership (maintenance, licence fees, support fees, sustaining knowledge levels)
Selection process	Portability
Integration with other tools	Availability and dependencies
Costs for purchase, adaptation, or development	Maintaining and updating test scripts

Table 1. Overview of Tool Costs Categories

- identify and document tool requirements, including constraints and prioritizing requirements. In addition to tool features, also consider requirements for hardware, software, supplier, integration, and information exchange
- perform market research and compile a short list. At this point, you may also consider developing your own tools, but watch out for dependency on individuals and make sure you are considering only long-term solutions
- organize supplier presentations: let them use your application to prepare an agenda on what you should expect to see
- formally evaluate the tool: use it on a real project, but allow for additional resources
- write an evaluation report and make a formal decision on the tool(s)

Once the selection has been completed, the tool needs to be implemented in the organization. Some critical success factors for the implementation of the tools include:treating it as a project: formal testing may be needed during the pilot, e.g. integration testing with other tools and the environment; incremental rollout: remember, if you don't know what you're doing, don't do it on a large scale; adapting and improving the testing processes based on the tool; providing training and coaching; defining tool usage guidelines; performing retrospective meetings and being open to lessons learned during its application (this may even result in improvements to the tool selection and deployment process – for example, following the causal analysis of problems during the first large-scale applications of the tool); monitoring tool use and its benefits; beware – deployment is a change management process.

Shelfware

Don't get me wrong – I'm all in favor of tools. In today's agile world, there is just no way one can survive without tools; they should always be considered. But instead of writing this column about sexy new tools, I ended up writing to (again) raise awareness on tool selection and implementation, which somehow to my own surprise is apparently still needed. Test tools are implemented with the intention of increasing either test efficiency, control over testing, or the quality of deliverables. Implementation of testing tools is not trivial, and the success of the implementation depends both on the selection of a tool to address the required improvement and the tool's implementation process. Once a thorough selection and implementation process has been carried out, an adequate test tool (suite) will support the test process. Although many tools still end up as shelfware, following my suggestions will help ensure this doesn't happen in your project!

References

[1] Van Veenendaal, E. *"Forgotten Tools." Testing Experience*. Issue 04/10: 12.2001.

[2] Van Veenendaal, E. *The Testing Practitioner*. UTN Publishers: 2002.

> about the author

Erik van Veenendaal (*www.erikvanveenendaal.nl*) is a leading international consultant and trainer, and a widely recognized expert in the area of software testing and quality management. He is the founder of Improve Quality Services BV (*www.improveqs.nl*). He holds the EuroSTAR record, winning the best tutorial award three times! In 2007 he received the European Testing Excellence Award for his contribution to the testing profession over the years. He has been working as a test manager and consultant in various domains for more than 20 years. He has written numerous papers and a number of books, including "Practical Risk-Based Testing: The PRISMA Approach" and "ISTQB Foundations of Software Testing". He one of the core developers of the TMap testing methodology and a participant in working parties of the International Requirements Engineering Board (IREB). Erik is also a former part-time senior lecturer at the Eindhoven University of Technology, vice-president of the International Software Testing Qualifications Board (2005–2009) and currently board member of the TMMi Foundation. You can follow Erik on twitter via *@ErikvVeenendaal*.