Believe the Territory

Change is inevitable. Don't rely on documentation alone—be alert for signs you need to change course.

by Markus Gaertner | mgaertne@googlemail.com

The Swedish Army has the following dictum [1]:

> *When the map and the territory don't agree, always believe the territory.*

This dictum applies to software development in a number of ways.

Requirements

When the requirements document and the requirements don't agree, always believe the requirements.

On projects, the most fragile document is the requirements document. This occurs for several reasons. First, your customers may not completely know what they want. This is normal, since they are trying to create a mental picture of their future. While the customers might have a mental picture of what they need, transmitting it to others can be problematic. Mental models are translated into written and verbal language, transmitted over noisy communication channels where relevant information may be lost, interpreted by the receiving side of those channels, and, finally, incorporated into some software. And then you wonder why you did not meet your customer's requirements.

> "... if you cannot adapt to the changing requirements of your customer, you, too, may go out of business."

The second problem with requirements is that they change over time. For competitive advantage, customers must be able to react to their particular business and markets. If they can't react, they will go out of business. Therefore, if you cannot adapt to the changing requirements of your customer, you, too, may go out of business. Since it's nearly impossible to think about all the changes that may come up during projects, you should leave the code and the tests as flexible as possible.

Third, requirements change not only for the sake of your customers' competitive advantage but, over time, the perception of "high quality" software may change. A new operating system may launch with a new user interface look and feel, and your customer may want to adopt it. Human interaction models change and while some of these changes may be predictable, many others are not. Some changes may creep in slowly; others, like changes in regulations and law, may occur month to month.

If the document that holds an interpreted snapshot of the requirements at some point in time is not able to cope with these changes as they occur, we had better believe the requirements rather than the document.

Test Ideas

When the test plan and the tests don't agree, always believe the tests.

There can be a big difference between the test plan document and the ongoing activities on a software test project. The test plan document is an output of the planning activities surrounding testing. Therefore, it lists some testing activities that were thought to be necessary at some previous point in time. This might be early in the project, when little knowledge about the product to be built existed. Therefore, when we run into a situation where the test plan lists an activity different from what the testers should be doing right now, reflect on whether the test plan defined a different activity based on lack of feedback on the actual progress, experience gained, or whether the tester is going off on a personal tangent.

If the document was written with some particular situation in mind and that situation unfolds in a different manner, testers need the right to change their course of action. When testing is not adapting to the context in which it is executed, we may miss critical bugs and information about the software that should be shared with our stakeholders.

Projects

When the project plan and the progress don't agree, always believe the progress.

In chapter ten of *Quality Software Management: Congruent Action* [2], Jerry Weinberg describes the Addiction Cycle. When the actual progress made falls behind the planned progress, a project manager is put under pressure to catch up. Giving in to the pressure, the project manager deliberately chooses to skip the review process and the test activities so that the product can be delivered faster. The short-term effect of his decision is some pressure relief for him. Of course, in the long term, the problems and bugs that were not addressed during the skipped practices are revealed and put the project under even greater pressure—calling for more shortcuts and skipping other prac-

tices. The only way out of the Addiction Cycle is to believe the progress being made and take necessary adaptations to the course, renegotiating the plan as needed.

Processes

When the process description and the process don't agree, always believe the process.

According to Alistair Cockburn, the process description defines which roles your team will fulfill [3] and explains how you hand off work products to another team member. But, processes seldom fit their written descriptions. That's why a temporary snapshot of the processes in the form of a process description is not likely to reflect the actual working habits in all projects everywhere in your company. Process descriptions may help new employees learn how a team works, but as the team gets through its norming and storming phase [4], these descriptions are likely to become out of date. Team members get into a habit of working together and sometimes decide to do things differently depending on the day's circumstances. Therefore, when the process description does not reflect the actual process followed, it's not necessarily bad.

Instead of constraining highly effective software teams by process descriptions that are not followed for long, team members must be able to adapt their process to their particular needs. This involves regular reflection over the course of the project and small adaptations to the way work actually gets done. If a process description does not give the opportunity for a team to adapt on the actual project, team members will feel they are on a death march and, in the end, may even boycott the project.

Investigate

So, what should we do with all this documentation? My advice is to document everything that has value to your situation. But, when looking something up in a document, be suspicious; be very suspicious. Like a private investigator, dig into the available documentation and let it guide you, rather than making you inattentive. Keep your eyes and ears open for any changes that may have occurred after the document was written or reviewed. Compare the document with what is happening and act accordingly. Last but not least, make sure you update the document when a serious difference jumps out at you. **{end}**

For more on the following topic go to www.StickyMinds.com/bettersoftware.

- References

index to advertisers

Better Software (USPS: 019-578, ISSN: 1553-1929) is published six times per year January/February, March/April, May/June, July/August, September/October, November/December. Subscription rate is US $40.00 per year. A US $35 shipping charge is incurred for all non-US addresses. Payments to Software Quality Engineering must be made in US funds drawn from a US bank. For more information, contact info@bettersoftware.com or call 800.450.7854. Back issues may be purchased for $15 per issue (plus shipping). Volume discounts available.

Periodicals Postage paid in Orange Park, FL, and other mailing offices. POSTMASTER: Send address changes to Better Software, 330 Corporate Way, Suite 300, Orange Park, FL 32073, info@bettersoftware.com.