WK1

Keynote

Wednesday 11/12/2008 8:45 AM – 9:45 AM

Seven Years Later: What the Agile Manifesto Left Out

Presented by:

Brian Marick
Exampler Consulting

Presented at:
Agile Development Practices 2008
November 10 - 14, Orlando, FL., USA

SOFTWARE QUALITY ENGINEERING

330 Corporate Way, Suite 300, Orange Park, FL 32043
888-268-8770 ▪ 904-278-0524 ▪ sqeinfo@sqe.com ▪ www.sqe.com

Brian Marick

Brian Marick (marick@exampler.com, www.exampler.com) was a programmer, tester, and team lead in the ‘80s, a testing consultant in the ‘90s, and is an agile consultant in this decade. He was one of the authors of the *Manifesto for Agile Software Development* and is a past chair of the board of the Agile Alliance. Brian is the author of two books—*The Craft of Software Testing* and *Everyday Scripting with Ruby*—and a number of articles. His consulting concentrates on blending formerly-independent test teams into agile projects, the use of executable examples (a.k.a, tests) to drive creation of products, and helping programmers learn their craft by pairing with them.

Seven Years Later:
What the Agile Manifesto Left Out

Brian Marick, Exampler Consulting
marick@exampler.com
www.exampler.com

When you see the presentation, you'll know that it's not in a format that can be reproduced on paper. These slides are a quick summary of the key content.

They were made on September 5, 2008. It is possible – even likely – that what I emphasize will change between now and the date of the presentation.

The Unspoken Values

- Discipline
- Skill
- Ease
- Rhythm
- Exhibitionism
- Joy

Discipline

- Agile requires more discipline than conventional software development.
- The main driver of discipline is the requirement that the team deliver working software to the business at frequent intervals.
- Internal discipline is required when the business isn’t demanding enough.

Skill

"I've also been tired for years of software people who seem embarrassed to admit that, at some point in the proceedings, someone competent has to write some damn code." – anonymous

- "Soft skills" – standups, retrospectives, etc. – are important. They are not enough.
- Hard skills – coding, testing – are harder to obtain and require more attention.
- Teams have a lot of learning to do, so they should "go slow to get fast".

Ease

If you’re used to hammers, when you drive a nail, you don’t think about the hammer, you think about hammering. It’s only when there’s something wrong with the hammer that you become conscious of it.

The same should be true of the activities of making software. There should be no “friction” getting between you and your task, no tools that divert your attention from your goal.

There should be a feeling of ease and comfort at work.

Exhibitionism

- Show everyone your code, status of all tasks, status of the iteration, progress at fixing a problem identified at last retrospective, etc.
- Don’t merely provide visibility when someone thinks to look. Make a more active effort to spread information.
- People should err on the side of making too much information available.

Rhythm

- A steady tick-tick-tick of activity.
- Repetition, with enough variation to keep people engaged.
- At all scales, from micro (test-code-refactor-repeat) to macro (the steady pace of release-ready delivery).
- Note that a regular rhythm makes it easy to notice departures from the norm.

Joy

When I first became involved in Agile, people would tell me “This is the best project I’ve ever worked on!” (including people from the business side). Nowadays, I’m much more likely to hear “At least work doesn’t suck as much as it used to” (actual quote).

This is wrong. There is a particular kind of joy in doing work well and being appreciated for it. Teams should act to increase joy, even if there is no obvious ROI in doing so.