The Pragmatic Bookshelf

PragPub

The First Iteration

IN THIS ISSUE

Issue #16
October 2010

Contents

DEPARTMENTS

Except where otherwise indicated, entire contents copyright © 2010 The Pragmatic Programmers.

Feel free to distribute this magazine (in whole, and for free) to anyone you want. However, you may not sell this magazine or its content, nor extract and use more than a paragraph of content in some other publication without our permission.

Published monthly in PDF, mobi, and epub formats by The Pragmatic Programmers, LLC, Dallas, TX, and Raleigh, NC. E-Mail support@pragprog.com, phone +1-800-699-7764. The editor is Michael Swaine (michael@pragprog.com). Visit us at http://pragprog.com for the lowdown on our books, screencasts, training, forums, and more.

ISSN: 1948-3562

Up Front

The Easter Eggs of October

by Michael Swaine

This is our October issue, and for many of us, October is a time of witches and spirits and too much candy. Scary stuff. In that spirit, we've scattered October and Halloween references throughout the issue, some of them more overt than others. Consider them our October Easter eggs.

Moving right along...

We've got another Guru Meditation by Andy Hunt that I think you'll find memorable. And this issue introduces a new column by The Agile Samurai [U1] author Jonathan Rasmusson. It's called "Way of the Agile Warrior" and the first one takes on the ten questions you'd be crazy not to ask at the start of any project.

Brian Hogan, author of HTML5 and CSS3 [U2], looks at new features in HTML5 that can make it easy to make your sites more accessibile. Jeff Langr and Tim Ottinger, authors of our upcoming Agile in a Flash [U3] card deck, talk about what Agile is not. Hew Wolff offers a spirited defense of the thesis that good code tells the truth; Dan Wohlbruck has another history article, this one focusing on the invention of the telegraph; and Jeremy Bingham gets into the spirit of the season by taking us on a shopping trip.

We also have the usual quiz, events calendar, and collection of interesting bits harvested from the fields of tweet. Oh, and John Shade finds a kindred spirit in Linus Torvalds, only to lose him again before column's end.

So welcome to this issue. We hope if finds you in good spirits.

Next issue, along with the technical articles, we'll be tackling a subject close to our hearts: writing. We'll talk with Susannah Pfalzer, who took on the job of Managing Editor for The Pragmatic Bookshelf earlier this year. We'll ask her what it's like to write a pragmatic book for us, and how you can get your foot in the door. We'll delve into what we think makes a good technical book, and offer some writing advice. Who knows, maybe we'll even include a little fiction or poetry. Watch for it on Wednesday, November 3.

External resources referenced in this article:

[U1]	http://pragprog.com/refer/pragpub16/titles/jtrap/the-agile-samurai
[U2]	http://pragprog.com/refer/pragpub16/titles/bhh5/html5-and-css3
[U3]	http://www.pragprog.com/refer/pragpub16/titles/olag/Agile-in-a-flash

What Agile Is Not

Dealing with Cowboys, Dogmatists, and Authoritarians

by Jeff Langr, Tim Ottinger

No two Agile implementations are alike, and no two Agile teams are identical. Jeff and Tim share their experiences in dealing with widely different types of teams.

In determining the core set of cards for our Agile in a Flash [U1] deck, we tried to cover all Agile bases, including prescriptive step-by-step lists. This month's article is about going beyond the cards.

Step-by-step descriptions of Agile practices are useful. The steps we provided are proven and there is no shame in following them precisely. Many highly Agile, highly successful projects strictly follow the exact steps we outline. The only problem is that focusing on practices risks a mechanical and unenlightened interpretation.

Like snowflakes, no two Agile implementations are alike. Scrum and XP differ dramatically in focus, implementation, and feel. Add in some of the Lean and ToC (Theory of Constraints) concepts, perhaps some Domain-Driven Design, and it gets pretty complicated trying to describe the kind of procedure that might be considered Agile.

We've visited and worked in dozens of allegedly Agile shops, ranging from rigid and sterile team rooms to wild west cowboy-coding ranges.

Cowboys

In the wild west shops, we promoted team collaboration over rogue hero mentalities. We promoted a team standard and collective code ownership. We taught pairing and test-driven development. When these values were embraced, the teams grew together. Not everyone was willing to switch from rugged code-slinger to pastoral code farmer, but those who stayed learned to improve their quality and deliver more useful code. Some say it was a turning point in their careers or the lives of their products.

In the sterile process-driven shops, we introduced retrospectives and incremental improvement to counter rigidity and stagnation. We promoted the Agile manifesto's values and principles, especially the value of empowered teams, over the concept of Agile as a strict process composed of phases and steps.

Did we make a difference? Both of us can relate numerous success and failure stories from our attempts to help teams transition. Even where the overall engagement appeared to fail, we could always find small successes, sometimes as simple as "turning one individual from the dark side." And in even our finest successes, we faced moments of challenge and loss.

Here's one story:

At a small dot-com, one of us coached an XP team of six developers. Their velocity was consistent and fairly predictable after several iterations, but not quite enough to push through the workload expected by the customer. To help, the company brought on two young, gung ho developers, one of whom quickly expressed considerable disappointment at his team's throughput. He felt the team wasn't working hard enough, wasn't putting in enough hours.

The team came in one Wednesday morning to discover that this whippersnapper was still wearing his same clothes from the day before. Uh oh. "I stayed all night to help bring up our velocity." The cowboy honestly thought everyone would look at him as the hero. Thankfully the team knew enough to confront him. In how many ways was this a boneheaded move on the cowboy's part?

- The rest of the team felt the cowboy had tried to make them look bad.
- The team was also upset because he had broken teamwork rules in acting unilaterally.
- The "velocity increase" would only have been sustainable if at least one member of the team were always willing to work overnight for the indefinite future.
- The cowboy's solo, wee-hours-of-the-morning coding skills left the rest of the team struggling with poor code and its impact for the next day and a half. The "velocity boost" was a considerable net loss.

The positive element was that the team spoke up when they had to, and re-asserted their expectations for team behavior. They would not tolerate similar cowboy actions in the future.

We focus on building a team first. More important than specific Agile practices, we seek ways of getting individuals to learn how to collaborate even more.

In another shop, one developer balked at the thought of a shared team coding standard. "I think coding standards stifle individual creativity," he said. Our Agile coach answered, "Yes. That is what they're for." That glib answer opened up a conversation about degrees of freedom and degrees of conformance, teamwork, production, and shared responsibility. The team ended up with a standard they liked well enough, found that it allowed them to work in unfamiliar code more easily, and over time they ended up working together quite well.

Agile is not the cowboy attitude of "do whatever you want because it feels right."

Dogmatists

On the flip side, we've encountered shops where "Agile" is a checklist. The following paragraph, representing a potential set of Agile team rules, is a pastiche of actual strictures we've seen in Agile teams.

Thou shalt run stand-up meetings for precisely 15 minutes, during which every participant will speak only their three sentences, and during which uncommitted parties may not speak. Iteration planning meetings are every other Monday for 180 minutes, and retrospectives are every other Friday for 120 minutes. A story must be in the form "As a user, I want to be able to do something, so that I may accomplish some business goal." Every story card must be formatted with the point total on the upper left corner, and the initials of the person working it on the lower right. Only the project manager may touch cards in these columns on the peg board. All teams must submit a burn-down chart to upper management daily. Pairing is mandated between the hours of 9 am and 4 pm.

There is nothing wrong with these rules, and they might be helpful in many circumstances. On the other hand, one might follow each and every such rule and yet never begin to realize the benefits of Agile development. Done strictly, these processes are merely ceremonies bolted onto a command-and-control structure. [Insert standard warning about teaching pigs to sing here.]

We have both followed all of these strictures and structures and found them good. They are all based on relevant concerns. Yet we've also broken almost every such Agile "rule." Agile development is demanding, insisting on continual delivery in the face of significant challenges, and no rule can withstand every contingency.

A forewarning, however: Each practice is based in one or more Agile values and principles. By discarding a practice, you may be opposing your own success. They are not to be discarded lightly or without measuring the result.

Agile is not a set of rules.

Authoritarians

Perhaps surprisingly, the worst shops we encountered were neither free ranges nor houses of unyielding Agile dogma. The worst we saw were those who wanted to maintain old-style waterfall practices with command-and-control mentality. Often in these cases, authoritarian sorts viewed Agile practices as a way of getting lowly programmers ("coders") to finish their work faster.

Many of the individuals in these shops lived in fear, both before and after the arrival of Agile. The sad result was that they ended up seeing Agile as just another means for their bosses to cow them into submission.

For an Agile team to succeed, the involved parties—management, sponsors, customers, and the development team—must work as a team toward a common goal. The stated Agile goal is to frequently and continually deliver high-quality software to meet the demands of the marketplace. If management introduces Agile but the development team resists it, Agile will not take root. If the development team adopts Agile as a grassroots effort, but the customer or other parts of the organization undermine the process, Agile will not take root.

Agile is not a command and control structure.

Start with the Values

The advice from Kent Beck and Ron Jeffries for successful adoption of Agile Software Development has always been to start with the values. So we did. Our first section in the Agile In A Flash deck is called The Idea and introduces values and concepts first. The second section deals with The Plan, which helps focus and direct business decisions. The Team section teaches more about living out the values in a community of software developers. Finally, The Code gives specifics on building products in a dynamic environment, covering testing, coding, and quality.

The Agile In A Flash cards are fronted by lists to help you quickly understand Agile ideas, concepts, and practices. We ask that readers view each process card as a starting point. Lessons learned from practicing as-given will trigger other ideas on how to help your team succeed. Remember that you can substitute and evolve practices, provided you remain true to the values.

Next: We'll talk more about how to grow your Agile team using shu ha ri as a maturity guideline.

About Jeff

Jeff Langr has been building software for over a quarter century. He is the author of *Agile Java* and *Essential Java Style*, plus more than 80 articles on software development and a couple chapters in Uncle Bob's *Clean Code*. He runs Langr Software Solutions [U2] from Colorado Springs and happily builds software as an employee of GeoLearning.

About Tim

In addition to developing the Agile in a Flash [U3] card deck with Jeff, Tim Ottinger has over 30 years of software development experience including time as an Agile coach, OO trainer, contractor, in-house developer, and even a little team leadership and management. He is also a contributing author to *Clean Code*. He writes code. He likes it.

Send the authors your feedback [U4] or discuss the article in the magazine forum [U5].

External resources referenced in this article:

[U1]	http://www.pragprog.com/refer/pragpub16/titles/olag/Agile-in-a-flash
[U2]	http://langrsoft.com
[U3]	http://www.pragprog.com/refer/pragpub16/titles/olag/Agile-in-a-flash
[U4]	mailto:michael@pragprog.com?subject=Agile-cards
[U5]	http://forums.pragprog.com/forums/134