The Pragmatic Bookshelf

PragPub

The First Iteration

IN THIS ISSUE

The Elegance of Scala

Issue #27
September 2011

Contents

FEATURES

DEPARTMENTS

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 Elegance of Scala

by Michael Swaine

This month kicks off a new series on the Scala language and a series of Pragmatic Bookshelf staff profiles.

With this September issue we introduce a new series on the Scala language by Venkat Subramaniam.

Scala has been called the future of the JVM. It's an inspired blend of object-oriented and functional features. Simple but powerful, it's the language the developers of Twitter chose to implement the back end of their system. If you don't know Scala, it's time to get acquainted. If you're already acquainted, there are probably some features you're not all that familiar with.

Either way, Venkat is the guy to make the introduction. He wrote Programming Scala [U1] and if you've attended many conferences, you've probably heard him speak. And you know you'll enjoy his introduction to this important language.

To kick off this ongoing Scala coverage, we're sharing another Scala article, this one by frequent contributor Brian Tarbox.

But wait, there's more.

Jeff Langr and Tim Ottinger are back with another in their ongoing series of agile insights. Here they reveal the only agile tools you'll ever need.

We also have another history article by Dan Wohlbruck. This one is about the origin of that familiar cash-dispensing machine, the ATM.

Ours is a small company, but it can still get confusing for readers to know who's who. Plus, some new people have joined the Pragmatic Bookshelf staff this year. So it seemed like a good time to start cataloguing who does what around here. Our new series of staff profiles kicks off with an introduction to Miles Forrest, who does our screencasts and podcasts. A good man to know.

And of course that's not all. As usual, we culled the collective wisdom of Twitter for some Choice Bits, and at a time when everyone is insisting that they know the real secret of Steve Jobs's success, John Shade admits he has no special insight into Steve at all.

External resources referenced in this article:

[U1] http://www.pragprog.com/refer/pragpub27/titles/vsscala/programming-scala

The Only Agile Tools You'll Ever Need

A Sanity Check on the Use of Tools

by Jeff Langr, Tim Ottinger

The very first value of agile emphasizes "individuals and interactions over processes and tools." But that doesn't mean "don't use tools."

Good value systems change little over time. Ten years on, and the core values of agile [U1] haven't changed one bit. At Agile 2011, the agile manifesto signatories reaffirmed their belief that these values represent the foundation for "better ways of developing software." Agile's underlying value system will be championed far longer than the buzzword *agile* itself.

Of course, values must translate into practices that accommodate the realities of modern software development. Many larger companies (and even some smaller ones) have a couple of oft-legitimate goals that can present challenges to agile adoption:

- They want to hire individuals in remote locations.
- They want to standardize on processes and tools across the enterprise.

A fairly natural response to these goals is to introduce or standardize on a single agile PM (project management) tool (some which are referred to as agile application lifecycle management, or ALM, tools). The thought is that a tool will provide them with immediate visibility across their entire portfolio, and will also help ensure that remote team members have roughly the same experience and level of challenge as local members. So what's the challenge to agile?

The very first value of agile emphasizes "individuals and interactions over processes and tools." This value is bolstered by at least two principles [U2]:

- "Business people and developers must work together daily throughout the project."
- "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."

We're quick to point out that the agile values and principles do not say "don't use tools!" They instead say we must emphasize "individuals and interactions" before we consider tools. The two quoted principles remind us that we want team members to work closely, to converse continually, to collaborate. This value is practical as well as appealing on the personal level, since misunderstanding is a frequently expensive kind of error. Face-to-face communication allows us to iterate and elaborate on our understandings in a way that written text does not.

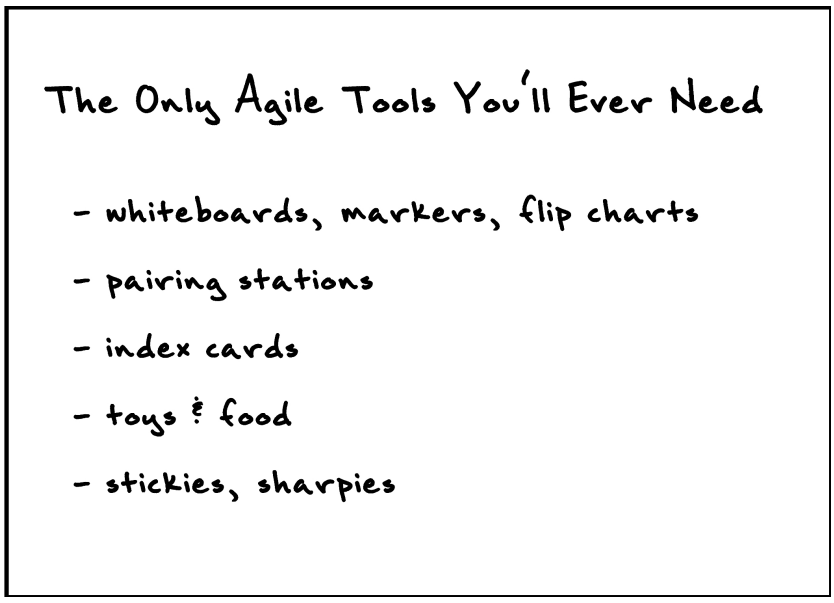
What about agile teams that want to have distributed team members? If you believe the face-to-face principle, distributed teams are immediately less efficient and effective. So does the notion of "distributed agile" make any sense? That's a great question, one that we'll answer in a forthcoming *PragPub* article.

Not All Tools Are Created Equal

We love great tools! We love tools like Hudson/Jenkins that help bring a team together, coordinating and tracking things that absolutely must be done with software. We love tools like Skype that help us communicate cheaply and effectively when we are forced to be remote. We love how Google docs allows the two of us to collaborate effectively as we write this article. We love how Git allows us to safely and effectively distribute and coordinate a source repository across a team of developers. We have favorite test frameworks and text editors and IDEs. We even love measurement tools. Notice, however, that these are not "agile" tools but collaboration and software development tools—the kinds of tools that provide immediate value.

We love whiteboards and large easel pads, which allow us to capture important bits of conversations and broadcast them so anyone can see. We love index cards, which let us capture important bits of conversations no matter where we are, then effectively use them as collaboration and planning tools (and as broadcast tools when tacked onto a corkboard). These tools are high-touch and low-tech. It is trivial to tailor their use to a team's needs.

We love the non-software tools so much that we created an Agile in a Flash card:



The blog entry for the card [U3] stirred a bit of controversy (and also provoked this article). It's a bit tongue-in-cheek—despite the card's title, you will absolutely require additional software-based tools to succeed. We stand by our core point, however: Seek low-tech and immediate tools before you insist on using complex software tools.

The Cost of Using the Tool

There is a famous legend of a sacred white elephant owned by a Siamese ruler. The ruler had a nephew whose wealth and influence posed a threat to the king. The canny king placed the nephew in charge of the royal white elephant. The elephant could not be allowed to come to harm, as it would be seen as scandalous and a bad omen for the kingdom, but the care and feeding of the elephant was amazingly expensive. The honorable custodianship of the realm's

most sacred symbol eroded the nephew's power and wealth so that he was no longer a threat to the king.

A management tool can be of great help, or it can be much like the white elephant: impoverishing the team with many troublesome tasks to distract them from daily work, while having such an organizational emphasis that team members dare not neglect those tedious tasks.

We seek immediacy with our tools. As long as we collaborate in the same room, direct conversation is the most immediate tool possible; whiteboards and index cards are a close second. They are instantly accessible, requiring minimal training, navigation time, maintenance, and cost.

A story from Jeff illustrates how tools can sometimes get in the way of collaboration:

"I attended a team's story-gathering session for a new project. The team had a couple remote members, so they opened a phone conference bridge. They initiated a desktop-sharing session using WebEx to share their project information, which was held within an open-source agile project management tool. For the local attendees, they presented the tool using an LCD projector.

"The team would discuss a story, and the project manager (PM) would attempt to enter it in the tool. As the team moved on to other potential stories, discussions arose around its relationship to already-entered stories. These discussions resulted in changes to other stories—we updated scope or granularity for some, and ended up deleting others.

"The meeting quickly degraded into all of us watching the PM as she struggled to keep up with the discussions. She struggled with the tool itself—'How do I filter the list of stories?' 'How do I make it not require an estimate?'—and we struggled to follow what she was doing. The essence of the stories we were trying to brainstorm was lost in the tool and the constraints of a lower-resolution display. The tool, not the stories, become the focus of attention.

"Twenty minutes in, we had perhaps three stories entered and still tentative, and a couple dozen more to peruse. At this pace, we projected that the meeting would last at least another hour.

"Interestingly, I noted that no one was on the teleconference bridge; for whatever reason, the remote invitees were not in attendance. I asked the PM to dispense with both the bridge and desktop sharing. The remaining team—everyone in the room—moved to a whiteboard, where everyone could see and contribute to the scribbling of candidate stories. Once we were happy, we ended the meeting, and the PM typed them into her tool.

"We were done in less than twenty minutes. We would have wasted significantly more time had we not sought a more collaborative, immediate tool. Granted, more experience with the tool would have helped, but ultimately all acknowledged that the tool was at best a distraction during the meeting."

Tools like Mingle have some ability to support story-related meetings. However, they still pale in the face of non-software tools that have more immediacy and collaboration support. It's not just us: Jon Archer relates that his manager

"rarely looks at [their software tool] now, rather prefers the big visible status on the wall some teams have."

Our recommendation: Any tool you use for collaboration should eliminate any potential distractions to focus. Perhaps the best software tool is an empty text or spreadsheet document, each providing a blank slate and a way to quickly move around, change, or highlight pieces of information.

Charts and kanban walls are so visible, immediate, and meaningful that they inspire action. A team may see that they are not tracking well to the sprint's goals and proactively start rescoping the work or reorganizing the team. Graphs of quality trends urge teams to improve their testing and refactoring. In this way, the physical artifacts have a coercive immediacy. Software-based agile tools tend to require a browser and/or login and some navigation to reach the same information, which robs them of coercive immediacy. Sometimes teams reproduce the tool's graphs in poster form and hang them on the wall in their team space to improve their influence.

Software tool learning curves can be steep. The high-end tools generally attempt to be highly configurable, end-all solutions for agile project and portfolio management. The increased flexibility comes with increased complexity, requiring significant training.

But We Already Paid For It

If your organization has invested significant money in an agile PM tool, we're not telling you to discard it (not yet, at least). Before you proceed, however, you must determine what benefits you expect to derive from use of the tool. Here is a partial list of legitimate goals for organizational investment in a single tool:

- get quick visibility into status for each project
- understand dependencies between projects
- expose and manage constraints that impact projects
- eliminate the cost of multiple tools
- promote common terminology and share valuable information between projects
- roll up meaningful statistics across teams
- realize a price advantage compared to individual purchases of desired tools

Avoid agile tool abuse! Some goals you may have for tools can actually hurt your efforts. Primarily, don't use a tool to help pit one project against another. There is no consistent cross-team measurement of productivity. Due to the endless variables—team composition, skill set, project constraints, technology impacts, and so on—it is not possible to compare a team's productivity to another in a meaningful way. Comparisons provide little value and may actually create resentment within your organization.

Don't allow tools to substitute for collaboration and leadership. No tool is as valuable as maintaining contact with your customers and development team

members. No manager will be a leader who visits their team primarily through management tool reports. A manager must know his team.

Avoid having anyone other than project managers (or individuals acting in that role) do data entry in the tool. Most of the tools provide a simplified interface that allows team members to quickly get in and track their progress during an iteration. But if you're doing agile well, project management is not a full-time job on a single project. In order of desirability:

- The team takes on the responsibility.
- The project manager is simply another team member and contributes in other ways to the project on a daily basis.
- The project manager tackles multiple projects.

In any case, we're suggesting that a good project manager can manage the tool, saving the team members from the effort of re-capturing information that the project manager should already know.

Avoid tracking too much in the tool. The need to track a significant amount of day-to-day activity in a tool is a process smell[U4]. On a given day in a typical agile team doing short iterations, one or two significant things should happen: the team completes a story or tackles a new story. The effort required to track these events in a tool should take seconds only. (A very good team might start and finish a couple stories in a day, doubling the number of seconds required from the PM.) If stories are started and completed in such short order, tracking task breakdowns for the story in a tool becomes a complete waste of effort.

"But we usually have four or five stories open at a time, each spanning the bulk of the iteration." Don't do that. It's not collaborative, and simply compounds all the challenges of waterfall in a short one-or-two-week space. Too much work in process also compels you to track more information in the tool—information that is almost never useful outside that iteration.

Don't allow the tool to dictate your work system. If you can make stories small and digestable, you don't need task tracking. It makes no sense for collaborative teams to assign work to individuals. It is absurd to track activity (rather than accomplishment) to the 15-minute interval in the chaotic world of collaboration. Finally, don't escalate. If the tool doesn't tell you what you want to know, don't beef up the tool. Demanding more governance information will not accelerate delivery or improve quality or provide you with any of the other features that drove you to choose an agile work style.

Wrap-up

Our criteria for tools are that they:

- aid the team (hence the emphasis on development and testing tools)
- add no unnecessary burden to the team
- do not substitute for leadership and management
- provide coercive immediacy to aid the team
- are given their rightful place as aids, not drivers

The first purchase you make for an agile team should not be an agile management tool, and the choice to bring an agile management tool into your practice should be informed and guided by people who are practicing "agile" by the principles and values that define the practice. Do not allow agile novices to be swayed by salesmen and brochures, but carefully pick the tools that will support your practice and your company if you pick any at all.

About Jeff

Jeff Langr has been happily building software for three decades. In addition to co-authoring Agile in a Flash [U5] with Tim, he's written over 100 articles on software development and a couple books, *Agile Java* and *Essential Java Style*, and contributed to Uncle Bob's *Clean Code*. Jeff runs the consulting and training company Langr Software Solutions from Colorado Springs.

About Tim

Tim Ottinger is the originator and co-author of Agile in a Flash [U6], a contributor to Clean Code, and a 30-year (plus) software developer. Tim is a senior consultant with Industrial Logic where he helps transform teams and organizations through education, process consulting, and technical practices coaching. He is an incessant blogger and incorrigible punster. He still writes code, and he likes it.

Send the authors your feedback [U7] or discuss the article in the magazine forum [U8].

External resources referenced in this article:

[U1] http://agilemanifesto.org

[U2] http://agilemanifesto.org/principles.html

[U3] http://agileinaflash.blogspot.com/2010/06/only-agile-tools-youll-ever-need.html

[U4] http://www.langrsoft.com/blog/2008/10/task-tracking-is-agile-smell.html

[U5] http://www.pragprog.com/refer/pragpub27/titles/olag/Agile-in-a-flash

[U6] http://www.pragprog.com/refer/pragpub27/titles/olag/Agile-in-a-flash

[U7] mailto:michael@pragprog.com?subject=agile

[U8] http://forums.pragprog.com/forums/134