The Pragmatic Bookshelf

PragPub

The First Iteration

IN THIS ISSUE

Issue #17
November 2010

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

PragProWriMo

by Michael Swaine

If you've always wanted to write a book, or you're already writing a book or some shorter-length work, but you've hit an iceberg and are now treading water, this issue is for you. You'll find several articles here on writing—and they're just the tip of the icebreaker. We're supporting your writing this month with a dedicated writing forum, a Twitter feed, and a challenge. You can read all about our PragProWriMo program below.

What We've Got for You

If you're old enough to remember *Byte* magazine from its heyday, or more recently, *Apple Directions* magazine from Apple's developer support program, you know Gregg Williams. He introduced *Byte* readers to the IBM PC and the Apple Macintosh and the Commodore Amiga, and his writing in *Apple Directions* was the true technical voice of that publication. Lately he's gotten into Clojure, and this month he shares with us his experience with that JVM-based Lisp dialect.

Our columnists cover a wide range this month. Andy Hunt writes about estimating and how constraints foster ingenuity. Jonathan Rasmusson writes about production readiness. I ramble on about the ebb and flow of writing. And John Shade follows the advice of Jonathan Swift.

In an information-packed interview, Pragmatic Bookshelf managing editor Susannah Pfalzer talks about what it's like to write a book for the Bookshelf. In May 2009, Chris McMahon organized the first Writing About Testing conference. That experience and his background as a professional musician have shaped his views on his software development work, and he writes in this issue about software development as performance. Agile in a Flash [U1] authors Jeff Langr and Tim Ottinger discuss the Japanese martial arts concept of Shuhari and how it applies to agile development. And Dan Wohlbruck is back with another tech history article, this time looking at the history of radio.

And of course we do the wheat/chaff triage thing on the tweets we follow in Choice Bits, and we report on all the good stuff coming up in our Calendar. Ah, but I promised to tell you more about PragProWriMo.

PragProWriMo

Last year we tried an experiment in writing. It worked well enough that we thought we'd do it again.

We call it PragProWriMo, which means Pragmatic Programmers Writing Month. We were inspired by (we ripped off) NaNoWriMo, National Novel Writing Month, which you can read about here [U2].

PragProWriMo is all about helping you write that book, whether "that book" is a technical book you'd really like to write but for some reason haven't been able to get started on, or the novel you've always threatened to write but never seem to be able to get your teeth into. Whatever the book, we have one question for you:

What are you waiting for?

We invite you to decide that you're really going to write that book. No more putting it off. The time is now. Oh, and you're going to write it in one month.

We're not kidding. Last November I took the NaNoWriMo challenge and succeeded. That program works like this: You decide that you're going to write a 50,000-word novel in the month of November. You don't revise, you don't worry about quality, you just write, every day. About 1700 words a day. The organizers and other participants provide a lot of encouragement and some help, but you do all the writing. And on November 30, if you've kept up the pace, by golly you've written a novel. Over 100,000 people participate in NaNoWriMo every year, and about 15% of them finish their novels.

Our PragProWriMo challenge is a little less daunting.

We'll provide supporting materials and encouragement and all you have to do is to write 60 pages toward that book during the month of November. If you're already writing a book for us and you're averaging two pages a day, then ta-da! you're already a winner.

To help you along, we're setting up a forum and a Twitter account. Follow us on Twitter at @pragprowrimo [U3] to stay up to date. Join the forum at forums.pragprog.com/forums/190 [U4] for more detailed writing advice, answers to your writing questions, and progress reports from participants. And when you finish your 60 pages, you might even get some special recognition from us.

Of course we'd love for you to submit a proposal for that book to us. But it's your work. You can publish it for free, you can do print on demand, you can hide it from the world and keep it to yourself, or you can take it to another publisher. A great new Prag book may result from this project, or it may not. What we're really trying to do is to help you write the book you've always wanted to write.

Even if you're not ready to take our PragProWriMo challenge and write a book in November, you can still take advantage of the writing advice and support that we're queueing up for November. Follow us at @pragprowrimo [U5] and visit the forum at forums.pragprog.com/forums/190 [U6]. You may learn something useful, and the mere act of joining in will focus your attention on your writing in a new way.

Solution to Last Issue's Quiz

Last month we celebrated 101010 day, the tenth of October, 2010. 101010 is 42 in binary [U7]. 42 was the solution to our 0th quiz, as revealed in our first issue [U8]. It is also the answer to last month's quiz [U9], and the answer to so much more [U10].

Why 42 is the answer to last issue's quiz is simple. Each number in the list is an Easter egg number [U11] with special meaning for a director or writer. And 42 is perhaps the most famous of these, or at least the most famous such number that fits numerically between 37 and 47. The list, explicated:

- 19 crops up regularly in Stephen King novels.
- 27 often appears in videos and songs by "Weird Al" Yankovic.
- 37 is frequently used in Kevin Smith movies.
- 42 is the ultimate answer to life, the universe, and everything, according to *The Hitchhiker's Guide to the Galaxy*.
- 47 appears in many Star Trek shows.
- 113 appears in the form A113 or A-113 or A1-13 in movies from CalArts grads, including John Lasseter and Brad Bird. It refers to a classroom at CalArts.
- 114 appears in Stanley Kubrick films in the form CRM114.
- 1138 is the number that George Lucas likes to hide in his movies, although sometimes it's not so hidden [U12].

Coming Attractions

Next issue we've got some fine articles lined up, including Adam Goucher's piece on testing for the cloud. Watch for it Wednesday, December 1.

External resources referenced in this article:

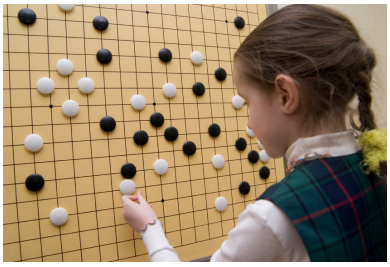
[U1] http://www.pragprog.com/refer/pragpub17/titles/olag/Agile-in-a-flash
[U2] http://www.nanowrimo.org/
[U3] http://twitter.com/#!/pragprowrimo
[U4] http://forums.pragprog.com/forums/190
[U5] http://twitter.com/#!/pragprowrimo
[U6] http://forums.pragprog.com/forums/190
[U7] http://www.fortytwoday.com/
[U8] http://pragprog.com/magazines/2009-07/the-quiz
[U9] http://pragprog.com/magazines/2010-10/the-quiz
[U10] http://en.wikipedia.org/wiki/42_(number)
[U11] http://en.wikipedia.org/wiki/A113
[U12] http://www.imdb.com/title/tt0066434/

Shu Ha Ri

Learn, Detach, Transcend: Steps to Agile Mastery

by Jeff Langr, Tim Ottinger

Our well-meaning agile coach's maturity model was based, either deliberately or subconsciously, on Shuhari, a Japanese martial arts concept involving three stages of mastery: "first learn, then detach, and finally transcend."



Many well-meaning agile proponents have devised various scales to measure agile maturity, not unlike the Capability Maturity Model (CMM). A Google search for "agile maturity model" returns a fair number of results, including a blog post whose title, "Yet Another Agile Maturity Model," suggests their unwillingness to disappear. The intended goal of such a model is to help managers, developers, and coaches assess the effectiveness of their company's transition to agile methods, and to highlight which practices need further development.

We have no problem with the notion of a team wanting to understand areas in which they need improvement. But we also think most teams have a sense of what they're doing poorly. A good coach can spot these same problems in short order, although it usually takes longer to understand root causes.

Practice-Driven Scale vs Goal-Driven Scale?

At one company, an agile coach introduced a five-level maturity model, very much like the CMM. The scale was backed by a checklist oriented toward goals, not practices. At the lowest maturity level, the goal was basic education and understanding of agile. At the highest level, the expectation was that the team continually assessed and improved themselves. Well, that's agile, right? All practices, principles, and values aside, if your team isn't continually seeking improvement, you're not agile.

While the intention for the model was noble, unfortunately it ended up being used for nefarious ends. Although it was dreamed up by a respectable agile coach in one organization, an executive in another organization appropriated the scale and began to use it to grade teams. Insidiously, the perception of agile shifted from a team-centric mechanism to a top-down governance scheme.

The thing that worries us most about agile maturity models is the idea of consistent scale. Standardization is a good idea, and indeed we always seek some level of standardization within an agile team (see our Coding Standards card for an example). But across teams, not so much: it's simply pointless to seek the same progress on the same scale with different people all working to the same end state.

It is hard to imagine that software developed for different types of products, by teams with varied experiences, for deployment in different regulatory environments, or for different markets should operate in precisely the same way or conform to the same metrics. It is rather unlikely that two different teams would optimize the same way, let alone all of the teams within a corporation.

Still, it is not wrong for a company to want to measure the accomplishment of its people. What if we rate according to proficiency at agile practices like TDD or pairing? That measurement certainly has more meaning than counting lines of code per week or "stays later than his coworkers." But it can also be damaging if, for example, you unfairly pit the Python team doing greenfield development against the C++ team struggling with a large legacy code base. It may also be meaningless: A team might have produced the best possible code using TDD and yet still fall down when it comes to delivering on customer expectations.

The most useful measurements of a team's ability are necessarily customer facing. Look at Running Tested Features[U1] (RTFs), defect counts, consistency of delivery, and customer satisfaction metrics, all directly related to how well the team can deliver quality software. Measure the trends over several releases. And remember that it's still meaningless to compare these numbers across teams—it provides no value to the customer. It will definitely not improve the team's capacity, and may even demoralize them.

Instead, to understand your maturity, look inward. Your goal is to improve yourself and your team, not keep up with the Jones's team. The empowered, energized team is best characterized by the self-awareness of need and the desire to improve both team and product.

Shuhari

One scale that seems to have some meaning (though no certification body and no way to standardize) is Shuhari [U2], a simple—but romanticized and mythologized—concept. Our well-meaning agile coach's maturity model was based, either deliberately or subconsciously, on Shuhari. Both scales are focused on stages of learning.

Shuhari is a Japanese martial arts concept (also encountered among players of *Go* and is actually built upon three Japanese words, thus three stages of mastery, that roughly correspond to the phrase "first learn, then detach, and finally transcend."

Shu

At the *shu* stage, the student copies the master's moves as perfectly as possible. The mentor corrects the student if the movement is imperfect, inefficient, or ineffectively performed. A student really has very little self-expression, but is being schooled in the fundamentals of the art. When the student has reached a solid level of mimicry, he has learned what to look for in form and balance and style and posture. He is competent enough as a novice, yet his knowledge is more memorization and even muscle memory. He is aware of his movements and knows he is "doing them right."

This is how we often began agile transitions. We started with strict workflow, strict practices, scheduled pair programming, rules, and strictures. Stand-up meetings were strictly three questions, and we sometimes limited attendance to only those producing the software (pigs v. chickens). We may have placed restrictions on hours worked per week. We programmed in pairs and conducted workshops on refactoring, TDD, planning, etc. The goal of the training and

coaching was to deliver fundamental theory and guide students to proper form and practice.

When a team produces consistent results using rigid methods, they have completed their shu stage. It is better to reach shu than not to reach it. It is better still to go beyond the kinds of things that can be measured with a practice-oriented Agile Maturity Model.

Ha

A budding martial artist will move from the shu to the *ha* stage. In this stage, he or she understands the principles well enough to depart from rigid mimicry and try different variations. Because he knows the accepted forms and their purposes, he may notice the difference in effect between his experimental way and the way he was taught. If his departure works less well, he returns to the way of his teaching. If his variation works better for him, he may retain it. This stage involves experimentation and observation, and in time the techniques become his own.

In our agile transitions, we try to quickly reach a point where the team can retrospect and make measured improvements on its own practices. If those changes don't pan out, they are abandoned or modified. A coach still provides value at this stage by helping keep the team mindfully measuring their changes and not falling back on old habits. The more a team becomes self-directing, the less they need a full-time coach.

In the ha stage, teams often will start to innovate in the way they close their feedback loops, the data they use in their big visible charts, and the way they design their software. They may adopt different tools and different protocols in dealing with each other. The ha stage will include some successes and some failures, but the team will be wiser with each experiment.

Meaningful retrospectives are critical if the team is to reach ha. Some teams will reach shu, then abandon retrospectives and cling to the rules. Such teams are practicing Scrum or XP (or what-have-you) mechanistically, and will rarely move forward. This stagnation may require a coach change to re-trigger the team's interest in improvement and self-management.

As the team matures in the ha stage, it's perfectly reasonable for them to depart from purely Agile concepts. They may adapt more techniques and concepts from the Theory of Constraints or Lean. They may adopt the Pomodoro technique [U3] or features from Getting Things Done [U4]. They may create or adopt new build and test tools. They may begin to adopt portfolio management and prioritization techniques they had never considered before. They may automate more of their processes.

Some leaders may take advantage of increased freedom to push their teams to return to pre-agile practices, all the while declaring themselves to be "post Agile" without ever internalizing the principles. They will abandon practice and principle alike. A coach can tell progress from abandonment.

Ri

In the *ri* stage, the martial artist has reached a point of mastery. Her reflexes and thinking are well-formed and she requires little conscious guidance to

carry out combinations of movements. She is largely free from slavishly following old forms, in that the forms have become second nature (tacit knowledge). She is able to teach and to adapt to new situations easily.

At the ri stage of an agile transition, the team becomes capable of abandoning rules and strict forms. They can produce quality work, groom the code base, and satisfy customers because it is their collective will to do so, and because they have built and tested their own body of rules and practices. They may discover ways to transform their work system that their coaches never considered.

The ri stage is the goal of agile team, coach, and company who have sought to honestly pursue this type of work system.

Shuhari is the progression through these three stages. The Shuhari progression helps a team develop its ability to deliver value, but it makes the very term "agile" start to look pretty nebulous. Is a team that no longer follows the practices non-agile, or post-agile, or just advanced agile? How do we measure relative agility of one team against another? How do we reward teams who have become agile? How do we correct those who don't get it?

Measurement and comparison, as we've mentioned, are not very important. The real questions are whether the quality of delivery is improved, and whether the organization as a whole is delivering value to the customers in an effective way. Far from blurring the term "agile" to mean merely "good," Shuhari defines agile as one way to do good things well. When we find better ways, you'll hear from us.

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 [U5] from Colorado Springs and happily builds software as an employee of GeoLearning.

About Tim

In addition to developing the Agile in a Flash [U6] 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 [U7] or discuss the article in the magazine forum [U8].

External resources referenced in this article:

[U1] http://xprogramming.com/articles/jatrtsmetric/

[U2] http://en.wikipedia.org/wiki/Shuhari

[U3] http://www.pragprog.com/refer/pragpub17/titles/snfocus/pomodoro-technique-illustrated

[U4] http://www.davidco.com/

[U5] http://langrsoft.com

[U6] http://www.pragprog.com/refer/pragpub17/titles/olag/Agile-in-a-flash

[U7] mailto:michael@pragprog.com?subject=Agile-cards

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