Keep the Baby

by Joe Schofield

Please don't tell me you're doing "agile." Really, what does that mean to anyone who actually knows something about agile? At least twelve different development approaches now claim a spot under the "agile" umbrella.[1] Many of these are incompatible. Many of these defy articles of the Agile Manifesto. Some of these are as heavy-weighted as any traditional methodology.[2] As comforting as those words might be to conventionalists, they portray a dichotomy with those who claim "if it's written down, it's not agile."

What about effectiveness of "agile" projects? An update in 2014 for *Scoring and Evaluating Software Methods* by Capers Jones rated Team Software Process TSP℠ and Personal Software Process PSP℠ projects more favorably than "agile" in producing fewer defects, with higher defect removal efficiency, and with fewer defects delivered. "Waterfall" based projects were rated lower than both TSP / PSP and "agile."[3]

Method: Metric:	FP/PMM	Defect/FP - Pre-test	DRE %	Defect/FP Delivered
TSP/PSP	11	3.6	96	0.14
Agile	10	3.8	92	0.30
Waterfall	8	4.5	87	0.59

FP/PM = Function points per Person-Month; Defect/FP = Defects per Function Point; DRE% = Defect Removal Efficiency

"People" are often at the center of "agile" projects gone wrong: Scrum Masters acting out of role, Product Owners not engaged, sponsors with spotty support, team members unable to stay focused, lack of agility in team members as they get locked into specific roles, team inexperience, and inadequate team training. Mismanaged sprints and technical debt accumulation also contribute to "agile" flops.[4]

Any reason to doubt that a survey of organizations using "agile" found 85 percent of those organizations had experienced some level of failure with "agile"?[5]

At the beginning of a recent workshop, I asked participants to define the word "it.[6] The brief exercise revealed some expected (and unexpected) results. A similar exercise on defining agile yields similar results with a range of variation under the canopy of practices known as "agile." Thus the use of the "agile" today in quotes throughout this article.

Culture and attitude also afflict "agile" projects. Misaligned values, inability to transition, unwillingness to try "agile" processes, and pressure to "do it the old way" ("Scrumfall") were cited by 39 percent of respondents as the cause of their tribulations.[4]

Wait. Don't throw in the towel just yet; maybe there's a baby in that bathwater. Of course, there it is. Call 9-1-1 —save the project, and the baby. We need the baby. We need much of the freshness of thinking and the new perspectives that come with "agile." Here are a few of my favorites:

- **Optics / visualization.** "Agile" is often cited for its transparency and visual cues. The product roadmap gives us an overall glimpse of the work. Task boards provide more of a micro view of the status of tasks. Burndown charts remind us of what's left to be completed. Velocity charts give us a clue as to how fast we are getting there. When completed with accurate data, each of these provides a snapshot that we've needed in the past.
- **The role of the product owner.** How many of us have longed for that perfect person to make a call on the business side, the "go to" person, the person that's always there for you? Not all product owners live up to their role, but those who do are an extravagance to a development team.
- **Definition of done.** "I thought you meant this when you said that." "No, I meant this instead of that." Been there? Felt that? Why not demarcate the finish line before you start the race? Defining "done" does exactly that. Defining "done" is a responsibility of the product owner, and while some "agile" proponents don't want to admit it, it is a "contracted" outcome. That doesn't nullify its value; in essence, it may increase it.
- **Daily stand-ups.** Daily (and I emphasize the "daily" in "daily"), short, focused, direct, strictly formatted, and team-driven are some of the characteristics of these meetings that provide timely visibility to teams and tasks. Not everyone gets to participate or talk—just the right folks. No management questions, at least not here. Daily stand-ups are definitely preferred to the days when you don't see fellow team members for weeks at a time, aren't sure of the status of dependencies, can't find the customer, or aren't clear as to which priorities are next.
- **"As a" statements** for story creation and the use of personas. Repeatable and structured practices drive lean and effective processes. These words aren't always "welcome" in "agile" circles, yet, "AS a *PERSONA*, I WANT / NEED SO THAT . . ." is a powerful construct for epics (high level requirements) and stories (lower level requirements). The persona eliminates the developer guessing approach to requirements often interjected when desperation intersects with schedule pressures. It also minimizes the "us vs. them" conflict when development teams don't have sufficient access to stakeholders and their needs.
- **Sprint demos / reviews.** Once again "agile" and most definitively Scrum, asserts a disciplined engagement among the development team and the product owner. (Discipline

and structure aren't the characterizations embraced by the entire "agile"; though they seem to be gaining increasing acceptance.) Feedback, interaction, closure, re-focus, and usage of "done" are employed during these cyclic events. Planned and time-boxed (similar to the stand-ups) these meetings are necessary to keeping the team informed and the customer current on progress.

- **Retrospectives.** Another MEETING! Don't scream. All three of the meetings in this list add value to the stakeholders, the product, or the process. Retrospectives provide improvement value to the process and the development team who are its only participants. Again, these meetings are planned events in the rhythm of iterations. Retrospective meetings are targeted at identifying enhancements addressing what and often the how. Don't skimp on retrospectives.
- Strong **focus on teaming.** No process description for any methodology prescribes poor team commitment, weak relationships, and disjointed ownership (though feature-driven development is based on individual code ownership). Team rooms, war rooms, and bull pens exemplify some of the synonyms for team facilities. But "agile" makes successful teaming conditions explicit. "Agilists" BEWARE—these techniques work for traditional methodologies (dare I say "waterfall") also; it's just rare that we get to benefit from following potent teaming principles.
- Using recent **performance to estimate** work being planned. No doubt influenced by the team's velocity, teams limit their optimism based on recent productivity. Is this approach not intuitive? This principle cautions teams that promise to "catch up later" or to rely on increased momentum going forward while discounting the uncertainty of the future. Another way to think about this "bounded estimation" is "slippage debt"—teams mitigate credibility risk by not over-promising.
- **More definition and discipline** than advertised. Most of these "favorite things" imply (impose) a degree of discipline which some "agile" advocates would consider anti-agile. Sorry. It's just how I'm wired and how I roll.

From the earliest days of the "agile" movement, "agile" was used as the anti-venom for waterfall reinforcing an "anything but that" mentality. Clearly not all the "agile" world intended that anything non-waterfall would fall into the "agile" bucket. Jim Highsmith once noted:

"The Agile movement is not anti-methodology, in fact, many of us want to restore credibility to the word methodology. We want to restore a balance. We embrace modeling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes. We plan, but recognize the limits of planning in a turbulent environment.[7]

When reviewing the list of practices, or lack thereof, for "agile" teams that have experienced turbulence, lost altitude, even crash-landed, one could argue that failed "agile" projects failed by not being "agile" projects at all. An explanation could be proffered of more "traditional" projects; in actuality, some of those failed because they did follow a disciplined process that just didn't mesh well with the needs of the product development and its stakeholders.

Today we have more choices. That doesn't guarantee us that the right choice will be selected; knowledge and thinking are still required. Some of us may refer to this as "adult supervision."

Excuse my sensitivity to the use of the word "agile"; clearly it's overloaded and in dire need of disambiguation. Many organizations are getting it; some will get it right. Learning organizations will get it right sooner. Until then, **keep the baby!**

References and further readings:

1 retrieved Wikipedia, Agile Software Development, 8/30/2014: Adaptive Software Development (ASD); Agile Modeling; Agile Unified Process (AUP); Crystal Methods (Crystal Clear); Disciplined Agile Delivery; Dynamic Systems Development Method (DSDM); Extreme Programming (XP); Feature Driven Development (FDD); Lean software development; Kanban (development); Scrum; Scrum-ban

2 Crystal (maroon) and DSDM, as examples

3 Capers Jones; Scoring and Evaluating Software Methods, Practices, and Results; Version 10.0; July 23, 2014. The data for the scoring comes from observations among about 150 Fortune 500 companies, some 50 smaller companies, and 30 government organizations. Negative scores also include data from 15 lawsuits. The rankings are based on about 20,000 projects that span 50 industries and 24 countries.

4 Common Agile Pitfalls, retrieved and summarized from Wikipedia, Agile Software Development, 9/4/2014

5 VERSIONONE; 8th Annual State of Agile™ Survey; 2014

6 Transitioning to Agile Workshop; Schofield; September, 2014

7 History: The Agile Manifesto; Jim Highsmith; 2001; agilemanifesto.org

Joe Schofield

CFPS, SMC, AEC, CSQA, CSMS, Certified CMMI Instructor
joescho@joejr.com, http://joejr.com/bio