Lean and Agile.
Marriage made in heaven, or oxymoron?

David West
Transcendence Corporation
College of Santa Fe
profwest@fastmail.fm
dwest@csf.edu

Yes, I am deliberately trying to provocative. And yes, in a very specific way, I believe the latter half of the question to be true.

This might be surprising since LeanAgile has seemingly become one word in the world of software development. So perhaps some caveats are in order.

I have known Mary and Tom Poppendieck for more than a decade. We were friends and colleagues in the world's largest object user group – in Minneapolis. Tom was my student at the University of St. Thomas (also in Minnesota, not – very unfortunately in the winter – in the Virgin Islands). Tom was one of the technical reviewers of my book on objects.

Except for the fact that their book outsold mine a hundred to one (and for which I will never forgive them), I have nothing but respect and admiration for its contents and its ideas. Their book is a masterful treatise and its contents should be learned and applied by software developers everywhere. Including, maybe, agile developers.

So what's the problem?

In a nutshell: Lean and Agile are grounded in fundamentally different world-views and therefore will inevitably find themselves in opposition on critical points.

In the following paragraphs I will try to show the opposing world-views, illustrate one point of conflict, and then suggest how the two viewpoints might be reconciled.

Lean Worldview = Production

Some quotes from *Lean Software Development, An Agile Toolkit*[i] to introduce my assertion that the Lean worldview is a production worldview.

Jim Highsmith's foreword, *"... Mary and Tom Poppendieck take lean industrial practicesto a new level – they tell us how to apply them to software development*." And, *"... provides a wealth of information about applying lean techniques from an industrial setting to software development*."

Some phrases from Ken Schwaber's foreword, "*industrial process control*," "*agile processes*," "*her [Mary's] background in manufacturing and product development*."

From page xxii of the introduction, "*While recognizing the hazards of misapplied metaphors, we believe that software development is similar to product development and that the software development industry can learn much from examining how changes in product development approaches have brought improvements to the product development process*."

Production and process vocabulary and metaphors are pervasive throughout the entire book. Although there is a clear rejection of 19th century ideas about production (e.g. Taylorism) there is an equally clear adoption of enlightened production models (e.g. the Toyota production model).

Specific agile practices are evaluated from the perspective of contribution to production. If a specific agile practice is seen to be in conflict with the lean production process model, that practice must be modified or eliminated.

Lean is not entirely about process. For example; of the seven principles of Lean,

- Eliminate waste
- Amplify learning
- Decide as late as possible
- Deliver as fast as possible
- Empower the team
- Build integrity in
- See the whole,

only the first is unequivocally connected to process, to production.

These principles suggest a way to transcend the production worldview evident in every other aspect of Lean. We will return to this possibility in the last part of this article.

Agile Worldview = Theory Building

I have had the great privilege to be associated with, and count myself among the friends of, most of the inventors and advocates of Agility. I have discussed the following idea about the philosophical foundations of Agile and found them to be in agreement. Only one, Alistair Cockburn, has put the idea into print.

Appendix B, pages 227-239, of Cockburn's[ii] *Agile Software Development* contains a reprint of an article written by Peter Naur in 1985, titled "Programming as Theory Building[iii]."

Naur makes the argument that the act of developing software has **mistakenly** been taken as an act-of-production – production of "a program and certain other texts." He cites several examples of empirical data inconsistent with the production model of development; including, the fact that documentation of arbitrary completeness and exactitude does little, if anything, to convey an understanding of a program to those not involved in its original creation.

Theory building, ala Naur, is the individual and collective effort to:

- Understand the World
- Understand how the software is shaped by the World and how it will integrate with that World
- Understand the essence of the software and how best to articulate (code) that essence
- Understand if you have gotten the first three understandings right.

The observable activities associated with theory building include telling a lot of stories, exploring ideas, trying things to see if they work, testing your understanding, populating your physical space with evocative reminders of your understanding, and doing these things iteratively in increasingly comprehensive increments.

Looks a lot like an agile environment, but bears little resemblance to a production environment.

Except for citing Ryle's ideas about the possession of a theory, Naur does not explicitly lay out a set of underlying principles for theory building. If he had, they almost certainly would have been consistent with XP's values of Simplicity, Communication, Courage, and Feedback.

Worldviews in Conflict

Lean views software development as a process for moving from conception to product. It wants to optimize that process, albeit in a radically different way and with radically different values than traditional (e.g. Taylorism) attempts at optimization.

Agile views software development as a process for building a consensual theory of the world: with an artifact being a byproduct – an expression – of that theory.

Because the fundamental worldviews of the two sides are dramatically different, it is inevitable that there will be conflicts. These conflicts will usually manifest themselves at the level of tools and practices.

For example: the product backlog.

Agile is premised upon the idea of modularizing work on the basis of user stories[iv]. A user story is, "one thing that the customer wants the system to do." (Kent Beck)

User stories originate from the users, *aka*, the customers or the business. Stories can be produced far faster than they can be implemented, especially at the beginning of a large-scale project, or when the goal is to "agilize" the entire enterprise.

Almost all agile projects establish a product backlog, a set of stories to be implemented. This set can be quite large. I have seen projects with a product backlog of hundreds of stories.

Lean looks at the product backlog and sees 'inventory' and 'waste.' Mary Poppendieck is on record suggesting that the product backlog should be eliminated or, at minimum pared to a size more evenly matched to the collective velocity of the teams'.

Agile sees the product backlog as a snapshot view of an emerging theory. Even if that snapshot view is physically manifest as a wall full of story cards, it is not an inventory! The cards on the wall serve as a form of external memory, with each card evoking (recalling to mind) detailed conversations and understandings of how things work.

Agile works best when there is a huge product backlog and when there is a large amount of churn in the composition of that backlog. Churn results from conversations about story essence; story priority; feedback from developers about stories not understood; stories that turned out to be easier or harder to develop than expected; stories that had to be refactored to make them more understandable or more tractable to development; and feedback from user acceptance of stories completed.

Eventually, churn diminishes and the product backlog becomes stable. The addition of new stories reduces to a trickle and the prioritization of stories changes only nominally. At this point it becomes even more tempting to consider the product backlog as an inventory.

Resist the temptation. The backlog is still a physical manifestation of the theory. It provides absolutely essential context for all of the development work being done.

The backlog provides the same kind of critical support for software development as continuity editors and technical advisors to for movie production. When attention is focused on a single scene, it is easy to forget that the hero was wearing a blue shirt, not a red one, in the last frame of the preceding scene. It is easy to forget that you can't hear an explosion in space. Similar errors occur when working on story implementation and it is the context – the product backlog – that provides continuity and correction of misunderstandings of essence.

In all of the agile projects I have coached, I insisted on having the product backlog on the wall in the form of story cards adjacent to the sprint tracking information. Daily stand-ups were conducted within easy view of both the stories of immediate focus and the product backlog stories. The product backlog was reviewed and discussed in detail during every planning game and every retrospective – simply to refresh the minds of everyone involved with the state of our collective theory.

The point of this example: your worldview necessarily colors your interpretation of "things." A simple artifact, like a product

backlog, has very different realities, purposes, values, and functionality based on your worldview perspective. In this case the 'production worldview' results in an interpretation that is actually harmful to agile software development.

I realize I am making a very general argument and offering a single example to support that argument. This is an artifact of space limitations, not a lack of examples.

Reconciliation

Marriage is bliss – except for the misunderstandings, arguments, and conflicting goals. Lean and Agile make such a beautiful couple. Surely this marriage can be saved?

Of course it can but there are three prerequisites, one for Lean, one for Agile and one for both.

Lean needs to take off the "production glasses" and look at Agile and elements of the agile development process from a holistic perspective that includes all seven of the Lean principles. If the product backlog had been evaluated from more than the 'eliminate waste' principle, its contributions to "amplify learning, decide as late as possible, empower the team, build integrity in, and see the whole" would be obvious.

Agile practitioners must, somewhat ironically, to the exact same thing. When you listen to agile practitioners talk about what they do – their vocabulary, metaphors, and implementation of the practices reflect the perspective of agile as an alternative mode of production. Ask most agile folk about Naur and theory building and you will get a blank look.

Both Lean and Agile must stop applying, in a literal and rote manner, the tools and practices. Tools and practices are nothing more than expressions of values, principles and philosophy. They are not the only possible expressions and may not even be the best expressions. Neither side will be able to realize their respective founders' admonition to "use, adapt, and transcend" until and unless they come to understand why the practices and tools are what they are.

AgiLean (think tabloids and bennifer or brangelina) was a case of love at first sight. The honeymoon was an exhilarating interval of finding new ways to merge ideas. But that time is past. If this marriage is to survive, both parties need to get past the superficial attractions – because at that level conflicts will inevitably arise.

---

[i] Poppendieck, Mary and Tom. *Lean Software Development, an Agile Toolkit*. Addison-Wesley. 2003.

[ii] Cockburn, Alistair. *Agile Software Development*. Addison-Wesley. 2002.

[iii] Naur, Peter. "Programming as Theory Building," in *Computing: A Human Activity*. ACM Press. 1992.

[iv] The essence of a user story is the terse sentence or two on a 3x5 card – but the expression of a story, even in agile, can be a complicated use case or an elaborated and annotated story card, *ala* Cockburn.