What is Agile?

By Allan Kelly, allan@allankelly.net, http://www.allankelly.net

When the history books are written 2010 might be the year that Agile went mainstream after 10 or more years as the alternative. In truth much that we call "Agile" has been around for a long time, Agile was simply the umbrella that gathered all these good techniques together.

Or maybe Agile was something new. Gathering all those good techniques together, making them work together, filling in the blanks and constructing a lining narrative was, is, something new.

Then again, maybe Agile is just Lean by another name.

Before launching into Agile and Xanpan in too much depth it is important to define what one means when using the term "Agile." Unfortunately I don't believe there is a single, short, concise, answer to this question.

Rather how you define Agile depends a lot on who you are and why you are asking. How you see Agile, as something old, something new, something borrowed or something blue depends on where you are standing and what you are looking for.

Purists might argue that the Agile Manifesto tells us exactly what Agile is. Yet this document is over ten years old, in that time Agile has changed and expanded and the context has changed: Web 2.0 has been and gone, SaaS is here, Google and Apple have replaced Microsoft as technology leaders, the world has been through a major recession, and so on. (The manifesto and the 12 principles are included below as an appendix.)

About the only thing everyone can agree on is that Agile is not the Waterfall. But defining Agile as not the Waterfall is not very helpful, in fact, Agile is not not the Waterfall, it is both more and it is less.

With all that in mind there are several perspectives one might take in answering the question "What is Agile?". Each perspective might be the right answer for you at some given moment but each one is a valid answer.

The Historic perspective

Agile is defined by the a manifesto and 12 principles written in 2001 - see side boxes. The manifesto is now (2013) a historic document, it was written at a time when heavy-weight development processes (e.g. SSADM and V-Model, and most things which were ISO-9001 or CMM approved) ruled software thinking. The term "Agile" was coined to group together a set of so called "light weight" methodologies.

The manifesto is, like the US Constitution, a document one can read all sorts of meaning into it. Depending on who you are, and what you want to read, you

read different things in both. Unlike the US Consitution there is no Supreme Court to arbitrate on whats-what. So Agile is what I say it is - or maybe, like art ("art is what artists do"), Agile is what Agile advocates say it is.

The Not the Waterfall perspective

Traditional software development was largely rooted in a mindset which said "Define what you want, design it, build it, testing it and deliver it; in sequence with little or no feedback from later stages." While (in my experience) development teams didn't usually manage to follow that model they tried to, and when they didn't they considered it a failure.

I think a lot of the time "Agile" is defined colloquially as "anything other than the waterfall model" but Agile is more than "not not the waterfall". Indeed, as Laurent Bossavit described in The Leprechauns of Software Engineering (Bossavit 2012), Royce's Waterfall (Royce 1970) model wasn't even discussed in literature much until Barry Boehm started using it as a counter example to his Spiral Model (Boehm 1986).

"Not the Waterfall" is a poor, if understandable, definition. Unfortunately, this means that any process that doesn't follow the classic waterfall strictly can be considered Agile. Adding to the confusion Waterfall itself can cover a number of different approaches, stage gate models like DoD 2167 and 2168 and all encompassing methods like SSADM.

Although commonly done it is a mistake to define Agile working as "not the Waterfall." Agile is not the Waterfall, but there are many other things which are not the Waterfall but are not Agile either. If we simply define Agile as "not the waterfall" then every failed attempt at the Waterfall was by definition Agile.

An aside: Agile get out of jail free card

In companies where strong, documentation centric, procedures have been hoisted on development teams Agile is sometimes seen as a "get out of jail free" card. Simply saying "this project is Agile" is seen to exempt work from company procedures. Unfortunately, this card is also used as a cover for cowboy development.

Just because an team or organization declares themselves Agile does not mean they are Agile. The true test of whether a team is Agile or not should be results based, not process or tool based. A team may follow Scrum slavishly but if they the result is not a responsive, nimble, productive team then they are not Agile.

Unfortunately there are many developers, teams and organisations out there all too willing to excuse their lack of process, lack of rigour, lack or documentation and even lack of results as "We are Agile."

> Should you ever find yourself facing a team who seem to be playing "Agile" as some sort of *get-out-of-jail-free card* look beyond the label and find out what practices they **are doing** which they think qualifies them as Agile.

The Agile as Better Perspective

Similar to the The Not the Waterfall Perspective, people who hold this view want a improvement over their current (usually poor) development processes and techniques. The motivation is simply to have fewer late projects, fewer unhappy users, fewer bugs and so on.

On the one hand this view is cynical, it holds limited promise; on the other hand this is perhaps the most work-a-day view. People holding this view are looking to make their lives better and their company's use of IT better.

In some cases Agile as Better links the Historical perspective and the Not the Waterfall perspective. I've seen companies that have tried hard to apply traditional waterfall based heavy-weight development approaches. In doing so companies tie themselves in knots trying to enforce a development model which doesn't match the problems they are trying to address. For these companies the year is still 2001; just adopting a mindset which is not grounded in Waterfall thinking is itself a massive improvement.

The Toolkit Perspective

I often find that when I talk to developers and testers in an company they implicitly view Agile as a toolkit of techniques which might be applied - The Toolkit Perspective. Ask them "Are you Agile?" and they will answer by referencing the practices which they use or don't use.

The Toolkit Perspective is best explained together with two other perspectives: The Methods Perspective and The Toolkit Perspective. The relationship between these threes is illustrated visually.

The State of Agile Perspective

Conversely, talk to managers - especially senior managers - and they are concerned with company Agility. Their perspective is the end result, - The State of Agile Perspective - they don't care whether a team do stand-up meetings, iterations or not, they care about the Agility of the company. Indeed, some Agile practices - e.g. Test Driven Development - may appear downright unAgile to them.

How you define the "state of Agile" depends on what the company is trying to do. In general it is something about being responsive to customers - which

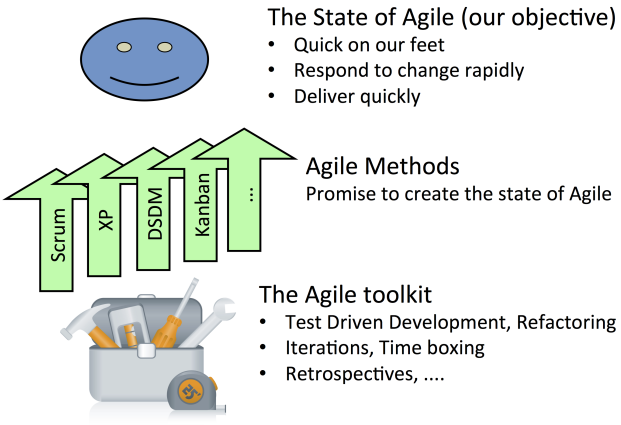

Figure 1: State, Method and Toolkit perspectives

implies you actually know what customers want in the first place so that must be included too. It is something about delivering fast and generally being quick on your feet.

MIT Professor Michael A Cusumano says:

> "I think there are two things that managers should pay attention to. . . .
>
> One is agility. It comes in different forms, but basically it's the ability to quickly adapt to or even anticipate and lead change. Agility in its broadest forms affects strategic thinking, operations, technological innovation and the ability to innovate in products, processes and business models." (Cusumano and Hopkins 2011)

He continues to explain:

> "I can't think of anything more important than building an agile company, because the world changes so quickly and unpredictably — there can be catastrophes like that tsunami in Japan or there can be once-in-a-century innovations like the Internet or there can be smaller levels of disruptive innovations like mobile computing and wireless technologies."

The Method Perspective

Talk to another group of people - middle managers mainly - and The Method Perspective appears. Those who take the Methods view are concerned with

which Agile Method (Scrum, XP, DSDM, etc.) the company is following. These methods are largely composition from the toolkit, ready made combinations of tools. According to this view following one of the methods will deliver the state of Agile.

The Learning Perspective of Agile

Almost last but definitely not least there is The Learning Perspective of Agile: I continue to believe that Agile is fundamentally an implementation of Organizational Learning concepts (Senge 1990„). The idea that all teams, departments, companies learn and those who are able to harness positive learning and change in technology/solution, problem/business and process domains will succeed. He who learns fastest wins.

Organizational Learning is present in one form or another in all Agile and Lean approaches but it is seldom front of stage. Indeed to bring it too far forward would be self-defeating. In working with clients I find that taking the learning perspective of Agile allows me understand and approach the most difficult issues.

For me organisational learning provides the theory and underpinnings of Agile. I should stop there, I could carry on but I'd need a book to do this perspective justice - fortunately I wrote just such a book a few years ago: Changing Software Development: Learning to be Agile (Kelly 2008).

The Marketing Perspective

Given these different perspectives the question we need to ask is: "what is not Agile?"

In many ways this is a more difficult question to answer because while any self-appointed authority can define what is Agile nobody has the authority to declare things unAgile, there is no Supreme Court of Agile. Personally I find it hard to see how business process re-engineering, CRM and ERP systems, virtualisation technology and domain specific languages can be regard as Agile but I have seen all of these cited as Agile tools or enablers.

Which perhaps leads us to the final and most cynical perspective on Agile: The Marketing Perspective.

Agile has grown and changed since it first appeared nearly 12 years ago. It has grown and changed in that time, and with no-one to police the brand it has become all encompassing. Come up with a good idea now and marketing demands you brand it Agile and put it under the umbrella.

Beyond the label

There is no one right answer to the question "What is Agile?". Each of these seven perspectives are good answers and they are not, on the whole, incompatible.

My advice is: when someone says they want to "be Agile" or "adopt Agile" or anything else "Agile" it pays to look beyond the label and ask: "what is Agile to you?" and "what are you looking to achieve?" Unless you know the context "Agile" is meaningless.

Appendix

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Source: http://agilemanifesto.org/ 2001

Twelve Principles of Agile Software

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity–the art of maximising the amount of work not done–is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Source: http://agilemanifesto.org/principles.html 2001

References

Argyris, C. 1999. *On organizational Learning.* Oxford: Blackwells.

Boehm, B. 1986. "A spiral model of software development and enhancement." *ACM SIGSOFT Software Engineering Notes* 11 (4): 14–24. http://dl.acm.org/citation.cfm?doid=12944.12948.

Bossavit, L. 2012. *The Leprechauns of Software Engineering.* LeanPub.

Cusumano, M. A., and M. s Hopkins. 2011. "How to Innovate When Platforms Won't Stop Moving." *MIT Sloan Management Review* 52 (4).

de Geus, A. P. 1997. *The Living Company.* Nicholas Brealey Publishing.

Kelly, A. 2008. *Changing Software Development: Learning to Become Agile.* John Wiley & Sons.

Royce, W. W. 1970. *Managing the development of large software systems: concepts and techniques.*

Senge, P. 1990. *The Fifth Discipline.* Random House Books.

(c) Allan Kelly 2013 allan@allankelly.net

This essay is a work in progress. The author welcomes comments and feedback at the address above. April 2013

About the author

Allan Kelly has held just about every job in the software world, from system admin to development manager. Today he works as consultant, trainer and writer helping teams adopt and deepen Agile practices, and helping companies benefit from developing software. He specialises in working with software product companies and aligning products and processes with company strategy.

He is the author of two books "Business Patterns for Software Developers" and "Changing Software Development: Learning to be Agile", the originator of Retrospective Dialogue Sheets (http://www.dialoguesheets.com), a regular conference speakers and frequent contributor to journals.

Allan lives in London and holds BSc and MBA degrees. More about Allan at http://www.allankelly.net and on Twitter as @allankellynet (http://twitter.com/allankellynet).