Agile – where's the evidence?

From time to time someone asks "Where is the evidence that Agile works?" My initial reaction, usually when it is posed inside a company is: "this is a diversion". Although it sounds like a reasonable question the person asking wishes to stall moves towards Agile. The question is aimed at derailing or side tracking conversations.

In truth my reaction is overly defensive. This is a rational question, aimed at making a rational decision and Agile advocates should be able to put up evidence to support their position. However doing so isn't very clear-cut.

If we are to really reach a rational decision surely we should examine the alternative. If the team don't do Agile, what are they doing?

For the sake of brevity in let us assume it is "The Waterfall." Now we need to ask: *Where is the evidence for Waterfall?* - as far as I know there is none, although there are plenty of cases of failed projects.

Not that Agile is immune from failed projects of course. And a rigorous study would also need to define success and failure but lets keep moving forward and see if we can find any evidence.

Agile?

Lets start with "Agile." How do we define Agile? Scrum? XP? Where does Agile start and Lean end, or vice-versa? Or is, as I believe, Agile a form of Lean?

We have a definition problem. Agile is poorly defined. I e-mailed a friend of mine who is undertaking a PhD in Architecture and Agile and asked him: "*Can you please point me at a literature review?"*

(Of course if I was a serious researcher I would closet myself in the library for days or weeks and do the literature review myself. I'm not and I need to earn money so I didn't.)

His answer: "that's an interesting question because a lot of research simply assumes that agile (as a whole) is better, without any evidence for it." He also pointed out there is a context issue here. Is it better for embedded systems? For web development? In finance? Health care? Modern software development is a varied field, it is not homogenous.

And how do you define better? Better architecture? Better requirements? Better code quality?

And Better compared to what? Chaos? Good waterfall? Bad waterfall? CMMI level 1? 2? 3? 4? 5?

Any serious evidence for Agile would need address all answer these questions. This it is unlikely to come to a conclusion that covers all cases.

And Scrum

Despite this one study claimed Scrum resulted in productivity improvements of as much as 600% (Benefield, 2008). I've even heard Jeff Sutherland verbally claim Scrum can deliver 1000% improvement in 2008. To be honest I don't believe these figures.

After posting this in a blog one commenter said he had contacted Jeff Sutherland for the collaborating evidence; Jeff pointed him to someone else, who pointed him onward. The trail eventually lead back to the beginning.

If these figures are possible then I think it says something about chaotic state the organizations started in more than the power of Agile. In these cases my guess is following any process or practice would be an improvement. Standing on one while coding would probably have generated a 50% improvement.

Better?

There is a trap for Agile here. Much traditional work has defined better as: On schedule/time, on budget/cost with the desired features/functionality. But Agile doesn't accept that as better, Agile negotiates over features/functionality and aims for business value.

A report from Cranfield University (Ward, 2006) suggested than much traditional IT work failed to truly capture business value because people focused on: time, budget, features. The same report suggested that the use of formal methodologies gave managers "a false sense of security, and perhaps an excuse for not being sufficiently involved."

Then there is a question of bugs. Traditional development has been very accepting of bugs, Agile isn't. Traditional managers accept bugs and negotiate over them. Agile managers (should) have a very low tolerance of quality issues because they know that rework and technical debt slow teams down.

Maybe asking for evidence about Agile is aiming for too much. Maybe we should look at the practices instead. Here there is some evidence.

The techniques

Over the years there have been various studies on pair programming which have been contradictory. Since most of these studies have been conducted on students you might well question the reliability.

Test Driven Development is clearer. A study from Microsoft Research and North Carolina University which is pretty conclusive on this, TDD leads to vastly fewer bugs (Nagappan et al., 2008). In the case of Visual Studio the number of bugs fell by 91%.

Keith Braithwaite has also done some great work looking at Cyclomatic Complexity of code and there seems to be a correlation between test coverage and better (i.e. lower) cyclomatic complexity. Keith is very careful to point out that correlation does not imply cause although one might

hypothesis that test driven development leads to "better" design. (Keith's work can be found in his Blog http://cumulative-hypotheses.org.)

TDD and source code are relatively easy to measure. I'm not really sure how you would determine whether Planning Meetings or User Stories worked. Maybe you might get somewhere with retrospectives. Measure how long a team spends in retrospectives, see if the following iteration delivers more or less as a result.

For their book Organizational Patterns of Agile Software Development Coplien and Harrison spent over 10 years assessing teams (Coplien and Harrison, 2004). This lead to a set of patterns which describe much of Agile software development. This is qualitative, or grounded, research rather than qualitative but is just as valid. These patterns themselves are useful, whether they are "better" depends on what you are comparing them with.

Anecdotal evidence

If we switch from hard-core research to anecdotal evidence and case studies things become easier. As many readers know I've been working with teams in Cornwall for over 18 months. During my March visit we held a workshop with the leaders of the companies and software teams. Without naming names some comments stood out here:

- "The main benefit [of Agile] was time to market... I don't know how we would have done it without Agile"
- "Agile has changed the way we run the company"
- "It is hard to imagine a world without Agile"

The last company quoted in this list is now finding their source code base is shrinking. As they have added more and more automated tests the design has changed and they don't see the need for vast swaths of code. Is that success? Each line of code is now far more expensive and they have less. (Heaven only knows what it does to function point analysis.)

Commercial evidence

If you want something a little more grounded there was a 2012 Forrester report which said: "Agile enables early detection of issues and mid-course corrections because it delivers artefacts much sooner and more often. Finally, Agile improves IT alignment with business goals and customer satisfaction" (Lo Giudice, 2012).

Similarly, back in 2006 a Gartner report in said: "It's a fact that agile efforts differ substantially from waterfall projects. It's also a fact that agile methods provide substantial benefits for appropriate business processes." (Agile Development: Fact or Fiction, 2006).

But I'm back to that nebulous thing "Agile."

Back to the Waterfall

What I have not done yet is look for, let alone present, any evidence that Waterfall works. Frankly I don't believe Waterfall ever worked. Full stop. Winston Royce who defined "the waterfall" didn't so why should I?

Royce's original paper argued that the Waterfall model described how we conceived software development but didn't actually show what happened in practice (Royce, 1970). If you read to the end of the paper Royce present a different model with significant feedback loop.

OK, sometimes, to save myself form a boring conversation I'm prepared to concede that: Waterfall worked in 1972 for developers using Cobol on OS/360 with a hierarchical IMS database. But this isn't 1972, you aren't working on OS/360 and very few of you are using Cobol with IMS.

What is the question?

Since we cannot define what Agile is, what domains we are interested in, what constitutes better, or what the alternative we are comparing it with asking "Where is the evidence for Agile?" is the wrong question.

Even if I could present you with some research that showed Agile, or Waterfall, did work then it is unlikely that the context for that research would meet you context.

(If you do want to stall the Agile people in your company, first ask "Where is the evidence Agile works?". When they produce it ask "How does this relate to our company? This is not our technology/market/country/etc. etc.")

I think it might come down to one question: *Is software development a defined process activity or an empirical process activity?*

If you believe you can follow a defined process and write down a process to follow and the requirements of the thing you wish to build then waterfall is probably for you. On the other hand, if you believe the requirements, process, technology and a lot else defies definition then Agile is for you.

Do it yourself

Most of the evidence for Agile tends to be anecdotal, or at best qualitative. Very little is available as quantifiable numbers and that which is available is very narrowly focused.

All in all I can't present you any clear-cut evidence that "Agile works". I think there is enough evidence to believe that Agile might be a good thing and deserves a look.

Ultimately the evidence must be in the results. You organization, your challenges, your context, are unique to you. So my suggestion is: *make your own evidence.*

Set up two similar teams. Tell one work Waterfall and one to Agile and let them get on with it. Come back every six months and see how they are doing.

Anyone want to give it a try?

And if anyone knows of any research please please please let me know!

References

BENEFIELD, G. Year. Rolling Out Agile at a Large Enterprise. *In:* Hawaii International Conference on Software Systems, 2008 Big Island, Hawii.

COPLIEN, J. O. & HARRISON, N. B. 2004. *Organizational Patterns of Agile Software Development,* Upper Saddle River, NJ, Pearson Prentice Hall.

LO GIUDICE, D. 2012. Justifying Agile with Shorter, Faster Development. *Forrester.*

NAGAPPAN, N., MAXIMILIEN, E. M., BHAT, T. & WILLIAMS, L. 2008. Realizing quality improvement through test driven development: results and experiences of four industrial teams. *Empirical Softwae Engineering,* 13**,** 289-302.

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

WARD, J. 2006. Deliverying Value from Information System and Technology Investments: Learning from success. Cranfield, Bedford: Cranfiedl School of Managament.