Why Agile?

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

There are many reasons why individuals, teams and companies may want to become "Agile". For some Agile is simply "better" for others there is a specific objective - say, short time to market. While for others there may be nothing more than fashion and the desire to have the latest and greatest method on the resume or company website.

Answering "Why Agile?" should be an important step for any organization embarking on Agile. Knowing "Why?" will help guide the selection of Agile tools, techniques and approaches. Perhaps more importantly, defining the objective also defines what Agile means to an organization.

"Why" is also important for individuals: Why should they change? Why should they make the effort? Why should they step into the unknown?

Relating changes to individuals, illustrating why changes will benefit people personally helps motivate people and involve them. Too often I hear it said that "People resist change", I don't accept this. I agree people accept change which is done to them, change in which they see no benefit to themselves. But I believe people who are involved with change and can see benefit for themselves are much more likely to embrace and support change.

To a careful reader this list may well appear to overlap and duplicate the "What is Agile?" list given earlier. At times it seems Agile has become a bit of a movable feast capable of satisfying any desire. Partly this is because of the Marketing Perspective: the Agile brand is used to sell a solution to whatever your problem seems to be. It is also because of the Toolkit Perspective: the same tools may be used to achieve multiple ends.

Please therefore, before you start down the Agile road ask: Why does this organization want to be Agile? Or: What benefit can Agile bring this organization? And ask the people who do the work: What benefit would you like Agile to bring to you?

For many organizations Agile simply means better. Specifically: Better IT delivery. Anything which promises to reduce the number of IT project failures is attractive to organizations. While I would except this reason - or exasperation - as the motivation for an Agile adoption I would like to think there are better reasons.

Another way of asking "Why Agile" is to ask: "What are the benefits of Agile type working?". What follows are a list of reasons and benefits to explain why you may want to work Agile. There is merit in all these but you, and your organization, need to decide which are your reason, or reasons. Your driving rationale may not even be on this list.

Improved ROI

Simply reducing IT failure will improve the returns from IT investments. Not wasting time and money on developments that never deliver, or pulling the plug on failing developments earlier; are, alone good enough to improve the rate of return on IT investments. However they are not the only way to improve ROI.

Embracing incremental releases to customers is another guaranteed way to improve return on investment. Traditional projects would spend months or even years developing software before any benefit was shown. As a result the cashflow and benefits looked like this:

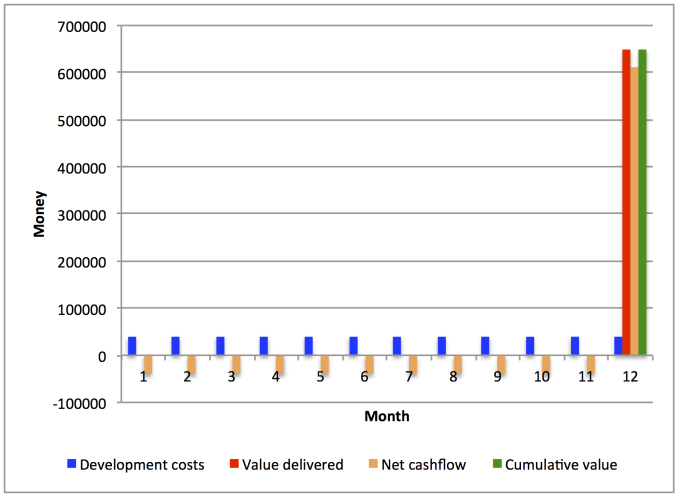

Figure 1: Traditional big bang delivery

This graph shows a perfect project, developers beaver away for 12 months and in the final month deliver the product. Let us assume there is no need for a subsequent test-fix or maintenance phase needed to pick up problems.

The project spends £40,000 each month for 12 months to develop a deliverable product. As a result cash flow is negative, by £40,000, in each of the first 11 months. In the final month £650,000 of value is delivered in one hit. Consequently the net cash flow in the final month is £610,000 and cumulative value £650,000.

Let us assume this is the projection for a proposed project. This scenario would generate an net present value (NPV) of £158,523 - assuming an annual risk free interest rate of 3%, 0.25% per month. Using the same figures for an internal rate of return (IRR) calculation results in a 5% return on investment. (Only 2% above the risk free rate.)

Note: both NPV and IRR are common mechanisms used by businesses for calculating the return on investment. Because of differences in the assumptions and mathematics of both models they do not necessarily agree.

Now consider the incremental model. Delivering some software earlier, allowing some (but not all) benefits to be released earlier produces benefits and cash flow sooner:

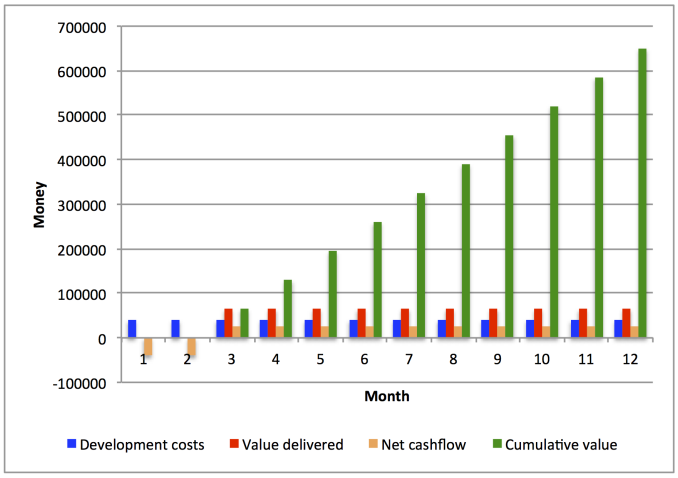

Figure 2: Incremental delivery, a little and often

In this model the team doesn't produce anything for the first two months. The team spend the same amount of money each month but after month two deliver one tenth of the value every month. As a result the cumulative value grows and grows over 10 monthly deliveries.

Now it is reasonable to argue that the team might deliver less than one tenth of the value in the first few months and more than one tenth in the later months. It may also be true that some teams are capable of delivering in some value even earlier. These are both value points but for the purposes of this calculation these simplifying assumptions are made.

Repeating the NPV calculation now produces a value of £637,957 - approximately four times higher than the traditional model. However, the IRR calculation now produces a return of 25% - five times greater than before and 22% over the risk free rate.

Given the size of these increases even if many of the simplifying assumptions are relaxed there is, in all likelihood, still more value in doing the work incrementally. Thus, even if a company rejects the argument that Agile reduces risk, even if it refuses to believe that higher quality will benefit customers and shorten schedules and reduce costs, even if they choose to reject the ability to flex requirements and to decline the options offered to change, then, in raw cash terms Agile Incremental delivery can still improve return on investment from software development.

Reduced risk

Delivering in an incremental model will reduce risk for several reasons. Firstly, rather than tail loading risk into the final phases (test and deployment) risk is spread throughout the work because test and deployment are spread throughout the work. (The Royce paper often cited - rightly or wrongly - as the origin of the "Waterfall" method itself warns of tail loading risk.)

In order to be incremental a work effort must produce deliverable products early in the work. This all release activities will have been exercised early. Software will have been packaged and delivered to users - into the data centre, over the web, or onto a CD and physically mailed out. Doing this will flush out unforeseen problems.

Second releasing software to users allows for feedback. Listening and acting on this feedback will further reduce risk. This lessons the chances of delivering "the wrong thing" or something which is out-of-date before it reaches customers.

Finally, because problems will be seen earlier should they prove insurmountable the sunk costs will be reduced. Therefore there is less at risk.

In general the Agile approach to risk can be characterised as: Face it head on and act. This is not to ignore other approaches to risk, which may still be used, but as a general rule risks should be tackled as they are identified.

In the short run I suspect companies adopting Agile working will experience reduced risk, fewer IT project failures and more IT success. In the longer term I expect that this reduction in delivery risk will allow, even embolden, companies to take risks with IT elsewhere. I expect to see even more ambitious IT projects attempted using fewer resources and more less time.

While I am convinced Agile software development reduces risk compared to traditional (i.e. waterfall style) development I am sceptical about the ability of Agile to reduce IT failure rates in the long run. The history of IT in general is one of taking risks, pushing technology, pushing change.

However many of these risks have been taken despite the waterfall model not because of it. Although in the popular press Agile competes with waterfall the reality is that Agile competes with chaos. It is not uncommon to find managers in a company have never previously managed software development. To borrow a term from Tom Davenport the approach might best be described as HSPALTA - "hire smart people and leave well alone" (Davenport 2005).

Companies complete IT projects without even attempting the waterfall model - they might not even have heard about the model! The waterfall model might be what people have been taught in college but a) many people in IT didn't study IT in college and b) the model is such a bad fit that even those who knew of it often didn't try it.

A third group of organizations knew of the model and even attempted to impose it on their workers but in doing so create unworkable processes and even destroyed

successful working practices. One research study even went as far as suggest these paradoxes created "Janus developers" having to present two meet incompatible demands (Howcroft 2003).

Case study: CMMI destroys Reuters

I worked as a developer at Reuters around the turn of the millenium. At the time the company was grappling with the Y2K bug and the Euro introduction, dot-com boom and plenty of other usual stuff.

In London I worked at a team handling data feed from exchanges which fed the Reuters terminals found in every major financial company. We had to respond to changing data formats, new APIs, bug fixes and a lot more. My small team - 3 of us within a bigger team (about 20 people) - did something which in retrospect looked quite Agile.

Processes were not particularly well documented and when they did exist people - in all teams - treated them flexibly. Many teams just did their own thing, as long as it worked they were generally left alone.

However Reuters wanted to do better and decided to become CMMI level 3. For those who don't know, CMMI, and its previous incarnation CMM, are models for measuring software process maturity: 1 is poor, 5 is excellent. The models are best considered rulers for measuring maturity rather than processes in their own right. A ruler allows measurement but cannot enforce a given size.

Management hired some consultants to make the CMMI level 2 and take them to level 3. These consultants wrote processes which were imposed on teams all through Reuters. From the fast moving financial markets were I worked to the stayed product development teams. Someone compared the new processes to "pouring concrete on the rails of the train of progress."

I left Reuters as the changes enveloped my team, I could see what was about to happen. From the reports I heard from others it was disaster. I do not think it is too much of an exaggeration to say that Reuters destroyed a large part of their own software development capability.

Several years later I met one of the consultants who was hired by Reuters. My blood pressure rose as I recounted my experience. Then

he told me that the consultants had foreseen exactly that scenario and warned Reuters. However Reuters management chose to press on regardless, replacing the original consultants with more compliant ones where necessary.

Realisation of IT benefits

Moore's Law states processor power doubles every 18 months, successful IT implementations allow organizations to benefit from some of that change. However coupling an organization to a rapidly changing domain also brings change.

Almost since it was invented in the 1940s and 1950s, certainly since the 1960s and 1970s, IT has raced ahead: processor power increasing, memory sizes increase, physical sizes shrinking. The potential benefits of these improvements have sometimes been difficult to pin down. This lead the economist Robert Solow state ""You can see the computer age everywhere but in the productivity statistics" [@Solow1987]

Part of the problem has been that IT came to be seen as a block to change rather than an enabler. IT departments which gain a reputation for working slowly, resisting change, delivering late and with poor quality. Some of these problems were rooted in mind set that saw extra time as the solution to every problem.

If all Agile does is allow corporate IT groups stop being the blocks to change then Agile itself doesn't need to do much else. Allowing corporate IT to be an enabler will allow more of the true potential of IT to be realised.

Better Quality

Finally, under better one should note quality. Quality is important because delivering quality they can be proud of is a big motivator for technical engineering staff. Too often in the past quality has been compromised in the mistaken belief that it can be traded for speed.

Reversing this position has an immediate benefit: higher quality leads to shorter schedules as already discussed. It also leads to better quality code which has benefits of its own but perhaps more importantly motivates staff. Rather than feeling "under the thumb" engineers can do quality work which makes them happy and proud.

Better is more predicable?

When I ask development teams, particularly corporate IT teams, "Why Agile?" one of the reasons I am usually given is: "To be more predictable in delivering what we said we would deliver." I accept

this, I smile, I add it to the list but inside I'm thinking "Sorry, Agile doesn't do that."

Many managers have been trained to believe that "On time, on budget, on requirements" is not only the key to success but the measure of success. Agile may well allow a team to deliver something on time, it can allow a team to stay within budget but it often does so by being flexible the "what is being delivered." (See the Iron Triangle discussion earlier.)

True, if I team want to stick to the original requirements statement then Agile would allow that too. But in that case the time and/or budget would need to change.

Agile as Fashion

In the name of honesty one should recognise Agile as a fashion statement. While (male) software developers might not like to admit fashion has a role in their (rational) decision making process it does. The desire to have the latest technology on the CV/resume is actually a sensible employment strategy.

But it is not just developers who are fashion conscious. Some managers, and even the organizations they lead, are guilty of talking Agile to be in with the in crowd. In some cases such talk might even extend to actually doing something!

Within fashion I will also include Badge Collecting. A bit like Boy Scouts there are those in the IT profession who collect Certifications. Developers and Testers are not immune but Project Managers seem to be the most prone to this phenomenon: Prince 2 and PMI are the traditional badges. Agile provides Scrum Master Certification and Agile Project Manager badges to add to the collection.

There may well be rational arguments behind an apparent fashion driven adoption of Agile. For example, in a world where more and more teams are Agile, and where the best developers want to work Agile companies which are openly not-Agile may find it hard to recruit new staff.

While I do not regard fashion as the best reason for a company to adopt Agile one should at least acknowledge it sometimes happens.

Flexibility and Opportunities

Agile, as the name implies, is responsive. Agile teams renegotiate what they are building all through the development process. This allows them to respond to changing needs a seize opportunities that arise. Although it should be noted that some see this as a disadvantage of Agile.

Flexibility itself has a number of benefits. Firstly it reflects the world we live in: the world changes, businesses change, new opportunities arise and so on. Things don't stop just because someone is writing software. Being flexible keep the process close to the real world.

Flexibility is also valuable to the process itself. By being flexible throughout the development process the need to settle everything up front in detail is removed. Without flexibility businesses engage in crystal-ball-gazing trying to make every decision before any coding is done. This is somewhat self defeating as the longer it takes to make these decisions - and capture then in documents - the more time there is for change to happen.

This flexibility, particularly when coupled with incremental delivery, spills out of IT and into the business as a whole. Rather than waiting for the final delivery businesses can start to integrate the partially completed products into their existing working practices. As a result big-bang change programmes are less applicable. While this might initially create complications for a business such change programmes have a poor track record. A series of small-bangs may create more to manage but also reduce risk.

On a personal level flexibility is good because it should lead to a reduction in the disputes between people over what is "in" and what is "out." All possible work is acceptable no matter when it was proposed, it is simply a question of priorities and value at any point.

This flexibility comes at the cost of predictability. Or rather, if the flexibility option is used then predictability suffers. If flexibility is not exercise then a good team should be fairly predictable.

Unfortunately for software developers businesses crave predictability and also crave flexibility and responsibility. Agile working gives the choice to the business customers.

Higher productivity

Higher productivity is one of the most regularly cited advantages of Agile. While I am confident that Agile working can improve team productivity it should be recognised that measuring productivity in software development is a incredibly difficult.

For example, one naive way of measuring productivity is lines of code. However the same algorithm can be expressed in different languages and even in different ways within one language. Correlating lines of code with delivered functionality is pretty much impossible.

Even if one could find a way of measuring coding productivity the problem would not be solved. Good teams might well find a way of eliminating requested features, or re-using existing code. (One of the principles behind the Agile

Manifesto actually encourages teams to do just this: "Simplicity–the art of maximizing the amount of work not done–is essential.")

Measuring features delivered is not a good measure either because more features do not necessarily make for a better product. Indeed they may over complicate a product.

Measuring bugs fixed is no good either since, according to Capers Jones (Jones, Bonsignour, and Subramanyam 2011) teams with a greater number of bugs will have disproportionally more "easy to fix" bugs. When systems have few bugs a higher percentage of those bugs will be difficult to fix.

Consequently one should examine any claims of improved productivity with a large pinch of salt.

It is worth noting that measuring productivity for IT use in general is itself difficult. For some years statisticians and academics have struggled with the so-called "Productivity Paradox" and characterised by Robert Solow's comment "we see the computer age everywhere except in the productivity statistics." (New York Times Book Review, July 12, 1987).

Still, I believe there are several reason why one would expect Agile teams to exhibit greater productivity:

- Agile teams generally adhere to "self-organization" or "self-managing" ideas; such teams are believed to result in more productive teams. (A later chapter will look at such teams in more depth and discuss the productivity question.)
- The Agile approach to quality should result in fewer bugs and thus fewer fixes. Capers Jones (Jones 2008) reports that projects with low defact potentials and high defect removal rates have shorter schedules, i.e. the same people produce the same products in less time
- Agile teams regularly review the work they are asked to do to prioritise the highest value. Thus one could expect the work delivered to have more business value than a similar team which simply worked from the first to last item.

I'm sure this list could be longer but these three points should serve to demonstrate the point.

Competitive advantage

At the time of writing I believe Agile working still confers competitive advantage for many organizations. Working Agile companies are able to bring working software into us sooner, refine what to build in the light of feedback, capitalise

on better quality to reduce schedules and cost as well as reducing maintenance costs.

However I do not expect this situation to remain so for very long. As more and more companies switch to Agile working there will be less competitive advantage in working Agile. Instead Agile will be essential to even attempt to compete.

To illustrate this point think of Toyota: when only Toyota worked lean Toyota had an advantage over GM, Ford and other car companies. Today any car company which does not embrace some form of Lean is at a competitive disadvantage.

(Toyota hasn't stood still, while it has been open in letting GM, Ford and others copy what it is doing it has moved on. While Toyota isn't perfect and has problems of its own it does demonstrate how a learning organization uses the ability to learn, and act on the learning, to advance.)

Unfortunately many companies are scared of Agile when I describe it. I am frequently asked "How can you expect our customers to agree to an open ended development without the time, cost and features all being fixed?" - or some variation on this question.

I would wager that for every customer who thinks like this there are more potential customers who don't know what they want and can't enter into negotiations because of it. Further, anyone who has worked with traditional IT groups should know that they are not very good at delivering fixed requirements to fixed time on a fixed budget.

Since moving to Agile working at least one of my clients has found that their market has increased. They have more customers because they do not demand everything is fixed. Further this company has on-going relationships with customers resulting in bigger deals overall. In fact the Managing Director will not bid on such contacts and has actively moved to get rid of customers who demand a triple fix.

There are more than enough potential clients who want to work this way and by doing so he has a competitive advantage.

Governance

Agile working, specifically regular releasable deliveries should lead to more effective governance of IT work. Governance of traditional work was often based on the delivery of proxy artefacts at stage gates: requirements documents, project plans, staffing plans, test reports and such. It was delivery of these documents, and sometimes the content, that was examined by the governance process.

Should a work effort be deemed to be in trouble corrective action was difficult. The effort could be stopped entirely, staffing adjusted or specific changes requested. However this was governance at arms length because the reports might not tell the whole, or true, story.

In the worst cases cancelling an effort would mean writing off the money spent on it with nothing to show. This was a drastic course of action which governance groups would be loathed to take. Even failing work efforts took on their own momentum and could be difficult to stop.

Governance of Agile efforts can, and should, be based on the assessment of working software. Documentation and reports might supplement this but the guts of the assessment should be based on:

- Does the product delivered so far indicate the spend has been worthwhile? Or, to put it another way: From what we can see do we have confidence the ability of the team to continue delivering? And has the money been well spent?
- Does the opportunity the team have identify justify spending some more money?

Therefore governance is based on more honest data not proxies that can be fudged.

Further should review deem it not worth spending any more money they company still has a working product of some value from the work so far, i.e. not all the spend needs to be written off. (Since each iteration should end with a deliverable product.)

Taken together these benefits increase the options actually available to a review. Traditional reviews had limited options because some of the options were distasteful.

Observant readers may notice that the governance process sketched looks a lot like portfolio management. Indeed Agile Governance might be best described as "Governance through Portfolio Management" - and will be covered in a later chapter.

Evidence?

One not unreasonable question asked of Agile is "Where is the evidence that Agile works? And that Agile is better than X ?". The short answer to this question is: There isn't any.

There are case studie of Agile working well. However most of these are written by those who have an interest promoting Agile, and perhaps their own services. It is also true that there are case studies - or at least reports - of Agile failing.

As regards hard laboratory trials of Agile there are no studies I know of. Nor are there any statistical - or quantitative - studies of Agile.

Software development isn't the kind of thing you can run laboratory trials on. You either have to use different people or have the same people repeat a task.

Both of these conditions will lead to questionable results. Even if you could create a laboratory style experiment it would be expensive. The cost of multiple development professional for several months if more money than most researchers can muster.

There are those who will refuse to believe case studies along, they look for hard laboratory data. Academics, particularly in social sciences, long ago came to accept qualitative research (e.g. case studies) as a form of research alongside quantitative research (e.g. numerically back studies.) Unfortunately most of the case studies cited in Agile texts would fail to meet the standards set by academics.

Part of the problem in finding evidence for Agile is simply defining what is Agile and what is not. Therefore, in looking for evidence, it is better to focus on tools and practices and ask: is there evidence practice X is beneficial?

Here there is some evidence that some of the Agile tools are effective. Take for example Test Driven Development. One study found this lead to a reduction in defects between 39% and 91% [Nagappan2008]. (As this is an evolving field I do not intend to perform a comprehensive review of the literature and evidence. If this is an important issue to you then I suggest you perform your own study of the practices you are considering.)

However the lack of hard evidence for Agile needs to be seen context: there isn't much hard evidence that Waterfall or any other method works either. Certainly I've never seen much evidence and I have looked - although perhaps not as rigorously as I should, or as often and as recently as I should.

What are you trying to optimise?

Ultimately you, and your organization, needs to determine what it is trying to optimise for. Optimising for low cost production would lead an emphasis on quality but also an acceptance of large backlogs and long response time. (Think of the UK National Health Service prior to 1997, very efficient in terms of money spent but with long waiting lists for many procedures and negligible patient choice.)

Conversely optimising for rapid response to requests would lead again to quality but an acceptance that backlogs need to be small - or non-existent - with a low utilisation rate and higher costs. (Think of the health providers in the USA, wide choice, short (if any) waiting times but the most expensive system in the world. One study I saw in the early 2000s suggested one in three beds in America was empty at any given time.)

Optimising for growth might well lead one to emphasis the need to recruit new staff and customers which would, in turn, lead to adoption of the fashionable tools which enticed interest in these groups. This approach requires an acceptance that costs are likely to race ahead of revenues.

Unfortunately, in my experience, few companies explicitly state what their primary goal is. Instead staff are left to make their own assumption about what the company wants and divine meaning in the tea-leaves of management statements. Naturally this leads conflict as different staff aim to optimise different things.

Conclusion

If nothing else one benefit of Agile is to offer a model which fits modern software development better than the 1970's inspired Waterfall model. But the benefits can be much more.

Before embarking on your own Agile initiative seek to understand what your organization is optimising for and why Agile is a good idea. Even if you can't understand your organizations motivations at least understand your own.

References

Davenport, T. H. 2005. *Thinking for a Living*. Boston: Harvard Business School Press.

Howcroft, and Wilson, M., D. 2003. "Paradoxes of participatory practices: the Janus role of the system developer." *Information and Organization* 13 (1): 1–24.

Jones, C. 2008. *Applied Software Measurement*. McGraw Hill.

Jones, C., B. Bonsignour, and J. Subramanyam. 2011. *The Economics of Software Quality*. Addison-Wesley.

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).