

Scrums, Stories and Showcases - What the Heck is Agile Development?

If you follow the latest trends in project management, and in particular, in software development, you can't have failed to notice the huge interest in something called Agile Development. Advocates claim that it can dramatically improve the quality and timeliness of software products and, in an industry that has struggled with a poor reputation for both, this is a claim that has caught peoples attention.

But does it stand up to scrutiny or is it just another management fad?

On the surface there is an element of mysticism about Agile that makes digging into it a challenge. Wrapped up in arcane terminology, it's difficult to get to grips with what it is and how it works. So we've attempted to dispel some of the mystery and reduce it down to its basics. How valuable it is and how different from more traditional development project management techniques is a personal judgement and we've formed our own opinion, but as always, we'd advocate keeping an open mind.

What exactly is 'Agile'?

Basically, it's an approach to software development that focuses on the people aspects rather than the process aspects.

It attempts to address perceived limitations of the traditional 'waterfall' approach to development projects, where a very rigid process of specify, design, develop, test and implement could result in:

- Long delivery schedules
- Lack of user involvement
- Inflexible and expensive to introduce changes
- Waste due to rework and poor quality

In 2001 a group of developers, dissatisfied with this approach, came up with the now infamous 'Agile Manifesto' that highlighted key values that defined the Agile approach:¹

*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.

This they expanded into 12 guiding principles that showed how these values could be translated into day-to-day activities and from which the whole agile 'industry' has developed. Now there are a myriad tools, methodologies, training courses and consultants, helping development teams follow these values and principles.

But as stated in the manifesto, these values aren't at the expense of the traditional focus of thorough processes, documentation and separation of roles. They really just attempt to find the optimum balance between these extremes.

¹ <http://agilemanifesto.org>

How new is this?

For us, this is the first key insight. And it's that Agile isn't new at all.

Ever since software developers have been using the waterfall approach, (which is pretty much forever), the limitations and risks have been widely known. For example, not involving users until the end of the process has always been bad practice and likewise, ignoring change control was always going to end in tears. So most experienced developers have built steps into their processes, to involve users and manage changes, without throwing out the positive aspects of the waterfall process, namely a rational, systematic process that focused on a quality deliverable at the end.

It's just they didn't dress it up as a new approach.

What are the practices that make up Agile today?

The key ones are:

- Develop in short cycle times
- Plan short term i.e. for the next cycle
- Develop only what is needed
- Close collaboration with the user
- High visibility and daily progress reporting
- Empower the project team
- Test in parallel with the development

How do these differ from traditional approaches?

Again, not as much as you might think.

Of course, if you have come from a background of two to three year development projects with a single, gargantuan deliverable at the end then this may seem like a radical departure from business as usual, but for the vast majority of developers, who are managing projects with timelines in months not years, the benefits of short cycles and multiple deliveries is well known.

Likewise, the impracticality of a detailed long term plan is also really common sense and most developers would only sketch out a blueprint to allow them to plan ahead for resource requirements and other long lead time concerns. In fact, there seems to be a risk that **not** having a long term plan would result in a loss of alignment with business strategy.

Grouping together the 'soft' practices of empowering teams and regular management communication, we'd suggest well run development teams do this anyway and it's more a characteristic of modern project management than the preserve of Agile. The bad old days of Theory X management have long gone, except in the most unenlightened organisation, and certainly they are compatible with a traditional waterfall approach to software development.

How do agile managers manage?

This is where the 'scrum' comes in. Basically, it's the project management framework and as you would expect from the manifesto, it's very 'people-centric'. The project manager (or 'scrum-master' as it's called in the Agile world) acts as a coach and mentor rather than a traditional manager and it works on the principle that if you give the team the tools and information to do the job, they will do it and do it well.

Setting aside the issues of leadership, teambuilding and motivation required to get your team into this 'performing' state, this sounds fine and dandy. Modern management theory has focused on the manager as a

facilitator and coach for ages and again, the Agile perspective seems to think traditional development teams are run by Victorian factory owners! Certainly our own experience is that these teams are most effectively managed by pointing the team members in the right direction, making clear the constraints they have to work in and then getting out of the way, unless they need your help.

This is not the exclusive territory of Agile teams.

How do Agile teams involve the user?

Now we move on to 'stories' and 'showcases'.

A story is a set of user requirements that a piece of software is developed to meet and so is the lowest level of detail that the user would be exposed to. Typically, the user and developer would agree on the story, the developer would produce the code that meets the requirements, and then they review it together. The work involved to deliver the code is called a 'sprint' and the review is called a 'showcase'. The key is that, the sprint is short in duration and the showcase isn't a protracted user acceptance test, but a highly interactive, intensive 'sign-off' that signals the software can be rolled out for general use.

Anything radically different or new here?

If you have some familiarity with business analysis, the concept of user requirements and the technique of use cases to document and analyse them will be familiar and has the same purpose as the 'user story'. And just as user stories have levels of granularity so the business analyst would produce business requirements (high level) and functional requirements (low level) to create a framework of requirements.

Likewise user testing and approval before implementation isn't a new concept and it's highly unlikely that any experienced development team would release software without a degree of user acceptance and involvement. Protracted parallel running and other time-consuming user testing techniques are generally avoided now in favour of short, intensive test plans, anyway.

So no, not really.

How do Agile teams organise themselves?

The main 'actors' are:

- The **product owner** has overall responsibility in terms of meeting the business case that would have justified the project. They would decide the requirements to be met (the **product backlog**) and the priority of the user stories.
- The **scrum master** ensures the agreed practices are followed and mentors and supports the development team.
- The **team members** do the work, of course.

For a traditional PMI or Prince2 project team, these map neatly onto the project sponsor and project manager roles. It seems the classic organisation structure of director-manager-doer is hard to beat!

What benefits do Agile claim to deliver?

According to Agile proponents, you will see:

- Improved quality
- Reduced waste
- Earlier delivery
- Better response to changes

- Constantly evolving working practices

It's hard to argue with any of these, if you were coming from a 'pure' waterfall style environment. But in reality, if you have assimilated modern project management and business analysis concepts and techniques into your development processes already, you may find further improvement hard to realise.

Our view is that Agile is a prettily packaged set of practices, all based on sound management concepts, which would deliver dramatic improvement - if the last project you did was in 1974.

Unfortunately, most of us are doing this already. It's just we forgot to give it a new name!

SGK Consulting Ltd
Pine Trees, Dark Lane
Tiddington
Stratford upon Avon
Warwickshire CV37 7AD

Phone: 0845 858 8000
Mobile: 07967 559375
E-mail: info@sgkconsulting.co.uk
Web: www.sgkconsulting.co.uk