Agile Development

Breaking some myths of the cult of Agile

Chances are that if you work in or around IT you have probably heard of Agile Development. You may have direct experience of working with it on a project, or you may only know the name, but it's hard to deny that it's an increasingly popular and prevalent means for IT and business teams to collaborate to deliver change.

Agile is a delivery framework that promotes collaboration between development and functional teams for the early and continuous delivery of valuable software. The Agile Manifesto, which sets out the values and principles of Agile working, was created in 2001 by a group of software engineers frustrated by the long lead times and process overheads of traditional "waterfall" delivery methods. As an alternative, they proposed a framework that embraces iterative, incremental software development.

An Agile project has some quite different features from that of a waterfall project. Users and developers will co-locate for much if not all of the project. Change will be encouraged, even quite late in the development. The team may produce little formal documentation (relying on interaction instead) and working software will be demonstrated on a regular basis throughout development. These features are all geared to being responsive and adaptive to changing business situations and ensuring technology directly supports rather than hinders business goals.

But Agile is not in itself a panacea; it requires strong and experienced team members working with great discipline to really get the most out of it. Too much focus on tangible software development rather than planning and management means there can be risks of unconstrained scope inflation, budget overruns and delivery delays

Over the last eleven years, Agile has progressed to develop a cult following amongst the developer community; which has driven its adoption in many organisations with varying degrees of success. In this article, we look at some of the myths surrounding Agile and highlight some of the key lessons learned from our experience.

Myth 1: Agile is applicable for all software projects

Undoubtedly some of the principles of Agile can be beneficial across the vast majority of IT projects – start testing and integrating as early as possible; involve business SMEs and senior users throughout the course of the project and don't create reams of bureaucracy for minor changes – but these principles are not exclusive to Agile and are part of good project management on any project.

If you can answer 'Yes' to the following questions, then it may be that Agile would be suitable for your project and organsation:

- Can the requirements be prioritised into a minimum subset for launch?
- Is there a level of novelty to the requirements?
- Can subsets of the code be demonstrated easily during the course of the development?
- Does the development have a fixed timescale?
- If you are using a 3rd-party to develop the solution, are youflexible on either scope or cost? Agile really does not lend itself to fixed scope, fixed price...
- Is there commitment to provide significant senior business user involvement, on-site with the delivery team, every week throughout the project?
- Does the team include significant Agile experience and expertise? If your team only has the checklists and a few Agile training sessions under their belt, then they are not ready to tackle a mission-critical, must-have project... It will be very likely that they fail to deploy Agile correctly and potentially fall back into old habits. Use experienced people - and gain that experience on 'safe' projects.
- Will users and developers be able to define requirements interactively?
- Can the organisation's release management approach accept frequent delivery of incremental steps?

If you answered 'no' to more than a couple of these, then we would recommend that it may be better to continue with a traditional methodology, perhaps cherry-picking agile techniques where appropriate.

Myth 2: Agile means that you don't need a Project Manager... or a plan

Blind adherence to Agile principles can be a license for chaos. Some Agile 'fundamentalists' will argue that you don't need plans and project managers if you have the right developers. We disagree.

It is essential within an Agile delivery to have strong risk management controls. One of the biggest and most fundamental risk management controls is to have a plan.

Within any project the primary variables are scope, quality, cost and time - Agile projects will tend to vary scope to a higher degree than cost or time. A project manager is required to manage these variables, to ensure overall scope control and to define and manage a plan that delivers within the time and budget allocated. This is not withstanding key activities such as managing quality, risks, stakeholder communications and, if applicable, vendors.

The scale and complexity of the project will determine what level of experience the project manager will require and whether they are full or part-time. It is possible that the project manager has dual roles in the project, but this decision should not be taken without assessing and managing the potential risks.

Our view would be, for any Agile project, to ensure that there is a 'fit for purpose' set of risk management controls in place, including a project plan, with a Project Manager appropriately experienced and resourced to co-ordinate and manage the project.

Myth 3: Agile means you don't need to define requirements, common designs and standards until late in a project, or even at all!

A good principle in Agile development is to understand the minimum set of features with which you could go live in your first release – Agile terms this the 'minimum marketable release (MMR)'. As a rough rule of thumb, around 60% of the available development effort should be spent delivering these. The remaining 40% of development capacity should be available as either contingency (worst case) or to include additional functionality.

Central to this principle is that you need to understand the complexity and volume of your requirements. Failure to perform a high-quality detailed requirements exercise can result in your MMR being based on a set of user requirements (or 'user stories') that are either low quality or not fully broken down. This can then cause slower development due to rework or the nightmare scenario where your backlog of requirements is growing faster than your developers are coding. Indeed we've heard of scenarios where the volume of user stories has more than doubled during the development phase; suddenly that 40% contingency doesn't look so large... We'd recommend validating that your requirements are fully decomposed by defining and agreeing both the high level User Interface wireframes and the acceptance criteria before the iterative development 'sprints' begin.

As well as detailed requirements, the application development is dependent on key preparatory work that should be completed before commencing development. This should include the solution and architecture designs and functional and technical standards and guidelines. If appropriate, major integration components (for example web services) should also be prioritised to avoid delays to the application development.

In our experience, completing these preparatory steps is important to ensuring efficient and high-quality coding and reducing the risk of rework or significant expansion in the volume of work.

Jargon Buster

Below are some of the key terms used by Agile methods. This list is by no means comprehensive

Backlog	The 'User Stories' (see definition below) that the team will work on in the future, but that have not yet been prioritised for delivery
Burn up Charts	A means of tracking progress. Typically captured as a graph showing the percentage of User Stories completed
Iteration	A finite period (typically between 2 and 4 weeks) during which a subset of software is created
MMR/MMF Minimum Marketable Requirements/Features	The minimum set of features with which you could go live in your first release
Story points	A metric in Scrum used to size and estimate Agile projects in terms of units of scope rather than hours/days
Scrum	The most widely recognised application development framework within Agile
Sprint	The "Scrum" term for a software development Iteration
User Stories	A way of defining the functionality required of a system, expressed in language that is clearly and simply understandable by the end user of that system
Waterfall	A traditional software delivery methodology that prescribes containment of activities into contiguous and non-overlapping phases that must be completed before the next can begin (e.g. analysis, design, build, test)
Wireframes	"Blueprints" of the pages of a website, illustrating their contents and functions

Myth 4: Agile is lower risk

If deployed correctly, Agile can significantly lower the overall risk of a project and improve stakeholder satisfaction. An Agile project will start to build risky aspects of the solution early – giving more time to react to problems.

It will also flush out missed or misrepresented requirements through regular user demonstrations that might otherwise have remained hidden until a User Acceptance Test phase; but it inevitably introduces new risks.

There is an assumption that Agile improves transparency and communication across stakeholders; but there is also the risk that functional teams operate in silos or agile newcomers don't really understand the new terms either performing inefficiently or reverting to old habits.

The key here is often experience. Our recommendation would be that if you are deploying Agile for the first time then don't put it on a critical and complex delivery; test it on something smaller first and build some lessons learned within your organisation.

Myth 5: Collaboration tools mean everyone doesn't need to be in the same location

Fundamental to Agile is the close interaction between the senior users / subject matter experts and the delivery team. This will always work best when the teams are onsite together. Collaboration tools can be used to complement this face-to-face time, but if they are a substitute for it then there will be an impact on efficiency. This should be factored into the plans and, wherever possible, mitigated. We would always advise to have these teams in the same location, as a minimum 2 days per week.

Summary

In conclusion, Agile can really enable your IT projects to deliver business results more responsively and more quickly, but beware some of the more wild promises of what Agile will enable you to do. It does not negate discipline and rigorous management. If you're thinking about using Agile on your projects, we would recommend the following:

- Be clear whether Agile is right for your project;
- Assign a project manager and ensure they define and monitor the project against a plan
- Complete detailed requirements and common design elements ahead of commencing application development
- Ensure your team has the necessary experience and understanding before using Agile for the first time.
- Don't apply Agile mechanically; tailor it for your project and use on-going risk management controls including self-examination at an individual and group-level.
- And finally, Agile is for software. Don't forget about change management, training, communications and documentation!

To read more about Berkeley's IT work, visit our website

http://www.berkeleypartnership.com/articles/it-outsourcing-consulting