

Thoughts on Agile

What is Agile:

Agile is a way of developing software and other 'soft products' focused on flexibility and adapting to changing user or customer requirements to maximise value. In many circumstances the end user / customer either doesn't have the knowledge to specify its requirements fully before development starts, or circumstances change, or new insights during the course of the project suggest better ways of achieving value. Agile can adapt to maximise the value created in these circumstances.

Traditional project management focuses on up-front planning and estimating to define the project and then efficiently delivering the defined requirements 'to plan'. The traditional approach is optimal in a wide range of circumstances including:

- When the outcomes or deliverables are well understood; eg, a typical construction project.
- When knowing the precise form of the outcome or deliverable is critical to other aspects of the overall program of works; eg, the software that is integrated with an aircraft's avionic control systems.
- When there are long lead time items to manufacture and then fit into other aspects of the project's overall work; eg, typical engineering and infrastructure projects.

These types of project are known as 'closed' or 'semi-closed' projects: the objective is clear¹.

For 'semi-open' and 'open' projects the challenge is altogether different. The final objective is not clear and the ways of achieving the objective may or may not be known. This is typical for most business software projects and business change projects. There may be a clear aspiration or strategy, discovering how to get there is a journey. This is the space where Agile methodologies offer significant advantages over more traditional software development methods provided the performing organisation can properly manage an agile development environment². Agile is not a synonym for anarchy.

The Agile Manifesto outlines the philosophy for the 'movement'; see: <http://agilemanifesto.org/>. From this starting point a range of methodologies have developed including Scrum and XP.

Understanding Agile:

The first key point of understanding is that Agile is not of itself a project management methodology. Agile is a soft product development methodology primarily used in software development but with significant potential in a wide range of application areas. The methodology can be used for routine operational maintenance of software within an organisation as effectively as for the development of a new software system within a project³.

The difference between operations and projects can be summarised as follows:

- Project management is the process of defining scope, deciding on methodologies, creating teams, and all of the other project management processes defined in the *PMBOK® Guide*. If Agile is chosen as the product development methodology for an IT project it will certainly influence the way the project is planned, resourced and controlled but of itself, Agile is not 'project management'. Projects are delivered by temporary teams assembled to work on the unique project deliverable (as described in the Project Charter) and then reassigned to other work as the project closes down.

¹ For more on project typology, see: **Projects aren't projects – Typology:**
<http://mosaicprojects.wordpress.com/2009/04/09/projects-arent-projects2>

² For more on selecting the right projects for Agile see:
<http://mosaicprojects.wordpress.com/2010/06/13/selecting-the-right-projects-for-agile/>

³ For more on this see: **Agile is NOT a Project Management Methodology:**
<http://mosaicprojects.wordpress.com/2009/03/05/agile-is-not-pm>

Thoughts on Agile

- Operational work in IT tends to be characterised by stable teams working on dozens of minor objectives selected on the basis of an organisation wide prioritisation. The work still needs to be planned, managed, budgeted and resourced but so does all operational work. Unquestionably, Agile can be a very effective methodology for the management of IT maintenance work⁴.

The major difference between operations and projects is permanent teams -v- temporary teams and the overall objective of the activity.

- Projects are about implementing a changed state for the organisation or community; eg, a new building. Creating a new capability.
- Operations are about maintaining and improving the current status quo; eg, typical plant and software maintenance. Incrementally creating an improved or enhanced capability⁵.

Scrum and XP are Agile product development methodologies that can be chosen for many IT applications. They can usefully be deployed in both operations and projects (unlike 'waterfall' which is almost exclusively a project based methodology). Agile would probably also be extremely useful in other situations such as developing training materials and many business change projects where most of the deliverables are relatively intangible and subject to change based on new learning as the project progresses. However these advantages cannot turn them into an IT project management methodology any more than deciding to use a particular construction technique such as precast concrete can make 'pre-casting' a construction project management methodology.

Agile Project Management:

There are two aspects to consider. One is applying Agile principles to the management of any project. The other is the need to adapt traditional project management processes to facilitate the effective management of an IT project using Agile as the product delivery mechanism.

The adaptations to traditional project management processes needed to allow Agile to work best are briefly discussed in *Managing Agile Projects*⁶.

In the opposite direction, the Agile community has some good ideas to pass on to conventional project managers, including:

Customer Engagement

While it may not be possible to iterate the building of a piece of machinery, engaging and explaining to the customer in their language -no jargon- what's happening will highlight issues early. If the customer doesn't like something, the sooner you know the better.

One of the key tenets of Agile is to engage effectively with your customer and end-users, understand their needs and problems, and then deliver an effective solution. This requires regular and effective communication, openness and accountability, and a good measure of trust to support robust relationships between the project team and their key stakeholders.

Going Light and Lean

Those are hardly new ideas, but they've been embraced by the Agile philosophy for a good reason, they work!

- Lean⁷ was developed by Toyota as a manufacturing philosophy and has been adapted to many other areas. Some of its key principles, such as minimising unnecessary movement,

⁴ For more on this see: **De-Projectising IT Maintenance:**

<http://mosaicprojects.wordpress.com/2009/03/06/de-projectising-it-maintenance>

⁵ For more on the definition of a project see: **Developing a concise definition of a project:**

http://www.mosaicprojects.com.au/Resources_Papers_007.html

⁶ See **Managing Agile Projects:** <http://mosaicprojects.wordpress.com/2009/03/07/managing-agile-projects>

⁷ For more on Lean see: http://en.wikipedia.org/wiki/Lean_manufacturing

Thoughts on Agile

simplifying process and continuous improvement, have huge potential in project management. The principles of 'Lean' are:

- specify value from the perspective of the end user or customer
 - review all of the steps in the value stream for each product – eliminate those steps that do not create value
 - make the value creating steps occur in a tight sequence so the product flows smoothly towards the customer
 - as flow is introduced let customers pull value from the preceding step (upstream activity)
 - once the full system has been introduced, continually improve the process to eliminate all waste.
- Light is focused on the minimising unnecessary overhead. Complex plans and processes should be simplified, but only to remove excess complication, not to remove core requirements.

Slimming down the project management overhead to its optimal level is probably the easiest way to free up the resources needed to engage your stakeholders more effectively and is definitely supported by A Guide to the Project Management Body of Knowledge (*PMBOK® Guide*) 4th Edition.

The Three basic Agile Methodologies:

These are probably the three most popular agile methodologies:

1. Scrum

Scrum has found its way into a variety of projectised organisations, including law firms and universities. The non-profit Scrum Alliance⁸ defines the key elements of Scrum as:

- A prioritised wish list called a product backlog is created.
- During the planning phase, the team selects a small chunk from the top of that wish list, called a sprint backlog, and decides how to implement those pieces.
- The team is given a certain amount of time, called a sprint, to complete its work and meets each day to assess its progress.
- At the end of the sprint—usually two to four weeks—the work should be ready to hand to a customer.
- The sprint ends with a sprint review and a retrospective.
- The next sprint then begins.

For Scrum to cross over into other industries, you need to be able to break down the requirements into a discrete set of items that could be worked across a set of iterations with usable (or defined) deliverables at the end of each sprint.

2. Kanban

Kanban originated in the motor vehicle industry and is adaptable to non-software development projects, including human resources and legal because its principles are not associated with any specific industry. Its key principles are:

- **Visualize the workflow.** This can be done by using a card wall, with the columns on the wall representing the states or steps in the workflow and the cards representing the work items.
- **Limit the works in progress.** Select the most important and valuable work items and keep the number small to ensure the team is making good progress.

⁸ For more on Scrum see: http://www.scrumalliance.org/pages/what_is_scrum

Thoughts on Agile

- **Manage flow.** The flow of work through each state or step should be actively monitored, measured and reported in order to evaluate positive or negative effects of incremental and evolutionary changes.
- **Make process policies explicit.** Ensure an explicit understanding of the mechanism of a process to achieve a rational, objective discussion of issues—and facilitate consensus around improvement suggestions.
- **Improve collaboratively.** To truly leverage Kanban, teams must collaborate. As with any other agile method, the team should meet as a team to plan, meet daily for a stand-up, and can choose to do retrospectives to inspect and adapt their process.

To adapt to a human resources project, for example, visualize the hiring process through a Kanban board. Categories on the board would include a column for the candidates who submitted résumés, a column for candidates who are qualified for the position and a column for candidates who have moved past the phone interview process. Support that workflow with a document that outlines who is responsible for these different roles and kickoff the process with a short meeting attended by all stakeholders. (For more on Kanban see separate heading below).

3. Extreme Programming (XP)

XP focuses on test-driven development, small releases and a team structure that includes the customer. Many of the rules for this agile methodology are designed specifically to address coding, designing and testing. XP recommends planning the release at a high level, then planning each iteration at its start (or every two weeks).

Key Agile Processes:

Some of the key differences between an 'Agile' approach to software development and the more traditional 'waterfall' approach include:

Agile Requirements Gathering with User Stories

Knowing the goals of your end users and project stakeholders is the necessary first step to any successful project, so just as in traditional projects, Agile projects start with basic requirements gathering. However, instead of trying to nail down all of the details up front, Agile seeks to capture just enough information to have a detailed conversation with the customer at a later date. This information is gathered as 'user stories'; short descriptions of the functionality in customer terms.

A user story is a placeholder and a promise to have a future conversation about the needed functionality. A useful structure for brainstorming user stories (although certainly not the only one) is as follows: *As a [Type of User], I want to [Function to Perform] so that [Business Value]*. This format focuses the story descriptions so that they stay customer and business value oriented instead of technical in nature.

For example, in a sales management system, some typical user stories might be:

- As a sales person, I want to add a new contact so that I can follow up later with prospects.
- As a sales manager, I want to view new contacts added by salesperson, so I can track leads
- As a system administrator, I want to add a new sales person so they can access the system

Delaying the detailed conversation until shortly before development avoids some of the waste that is typical of projects where detailed requirements are gathered early, but become invalid before the work begins. Additionally, some requested features may no longer be needed by the customer after a few months, saving the cost and effort of a detailed analysis and design.

To develop the most comprehensive list of requirements, it is important to define all of the key stakeholders first, with a particular focus on the business users of the system, using a tool such as the

Thoughts on Agile

Stakeholder Circle⁹. Once the appropriate stakeholders are identified, many options exist for capturing the requirements including focused interviews and group sessions using techniques such as brain storming and 'Six Thinking Hats'¹⁰.

A key strength of Agile development approaches is the ability to incorporate new requests for functionality as they are discovered. If some of the user stories aren't uncovered in the first planning sessions they can easily be added later; particularly if the business climate changes or if the stakeholders find they have forgotten something important.

Prioritising the Work:

After the initial requirements gathering, the user stories are prioritised with your customers:

- Prioritise the full list based on what's important to your customer.
- Select a subset of these for your first release.
- Then choose an even smaller subset for your first iteration or sprint (a fixed length time period during which development will take place, usually 1 to 4 weeks).

It is during the iteration planning process that the more detailed conversations with the customers occur to define the details of the 'user story' and clarify any issues. This close contact with the customer continues during development to enable the asking questions as needed.

Iteration / Release Planning:

Planning a release for an Agile software development is a collaboration between the development team and customer team. Once a list of candidate user stories have been identified for the release, the development team estimates the relative effort for each of the software features, as an input for prioritising within the release.

One technique for this high level estimate is to use 'points', so that the estimates are based on the relative size of each function (this one is twice as big as that one).

For example, for five stories around an online book store, the estimates might look like this:

- As a customer I want to browse books by category 4 points
- As a customer I want to search books by title 2 points
- As a customer I want to search books by author 2 points
- As a customer I want to buy a book with a credit card 8 points
- As an administrator, I want to add books to the store 4 points

This point allocation suggests the team believes that the search-related stories are about the same size (2 points), the 'browse' story is twice as difficult as those (4 points), the 'buy' story is significantly harder (8 points), and the administrative story is about as hard as the browse story (4 points).

To plan the iteration you start with the expected capacity of the team. If a similar team completed 10 points during the last release, then that could be the starting point for planning this one. However, for the initial iteration, it's best to be conservative - you can always add more functionality if the first few iterations show that you can get more done than you thought.

Based on the prioritised list of user stories and their estimated size a balanced set of work is determined for the first iteration. You prioritise based on the business value of the story, as well as the developer estimate. In more complex developments, the software architecture may require development in a particular sequence; eg, until some data tables are developed, a screen input module may not be possible to develop and test.

⁹ For more on the **Stakeholder Circle** see: <http://www.stakeholder-management.com>

¹⁰ For more on **Six Thinking Hats** see: http://www.mosaicprojects.com.au/PMP_Sup/PMP_Mod09_HR.html#6hats

Thoughts on Agile

Different combinations of stories should be considered to fill the release capacity in the most sensible way. For example, it might make more sense to schedule 2 or 3 smaller stories of less importance instead of a single larger, slightly more important story. Tools such as ExtremePlanner¹¹ let's you try out these scenarios until you've reached a stable point.

In the scenario above, the first iteration may build the administrative and browse functions with one search. This would allow the book store to be populated with books and the basic system road tested whilst the purchasing function is built together with the second search as a second iteration. Once both iterations are 100% complete, the first release of the bookstore to the buying public can occur.

Another key Agile tenet is that before each iteration can be completed; the functionality must be fully tested and signed off by the client.

Day-to-Day Agile Management

One of the key attributes of an Agile approach to software development is the emphasis on close communication within the team as well as with the customer. Team communication usually takes the form of a brief daily meeting where the team members can share progress and identify obstacles. In the Scrum method, this meeting is called a *Daily Scrum*, while the XP method refers to this as a *Daily Standup Meeting*.

The purpose of the daily meeting is for each team member to communicate three critical items:

- What they've accomplished since the last meeting
- What they are planning to work on next
- Obstacles or challenges they have encountered

It's important that the team is safe to honestly communicate status and issues in the daily meeting because the overall project controls are likely to be at the iteration level, not the detailed activity level inside each iteration [see: Managing Agile Projects¹²].

Project managers should take care not to stifle the flow of information by challenging honest reporting or trying to control the meeting. The successful Agile PM focuses on supporting the team and removing obstacles to their success [see: Servant Leader¹³].

The use of a visual aid in the meeting room to show the stories and tasks that they are being worked in the iteration on is valuable. This can take the form of a taskboard with index cards that can be moved around, or a computer and a projector that can show an electronic representation of the iteration. The taskboard should be updated either during or before the daily meeting. This usually involves identifying any completed tasks from the previous day, updating in-progress tasks with new estimates (if necessary) and team members selecting new tasks to work on today.

Managing the Iteration and Release Date

Agile methods use iterations, or short, time-boxed development cycles¹⁴. During an iteration, developers work on the highest priority features. They plan to complete whatever they committed to by the end of the iteration. However, due to the unpredictability of software development, it is likely the team will have too much to do (and less frequently, not enough to do) if they are estimating honestly.

When there is too much to do, the best option is to cut scope, since changing the iteration date undermines the primary advantage of time-boxed iterations - quick technical and business feedback; and adding more resources part way through rarely works.

¹¹ For more on **ExtremePlanner** see: <http://www.extremeplanner.com>

¹² **Managing Agile Projects**, see: <http://mosaicprojects.wordpress.com/2009/03/07/managing-agile-projects>

¹³ **Servant Leader**, see: http://www.mosaicprojects.com.au/PMP_Sup/PMP_Mod13_Professional.html#PM

¹⁴ For more on **timeboxing** see: http://www.mosaicprojects.com.au/PMP_Sup/PMP_Mod06_Time.html#Timeboxing

Thoughts on Agile

There are three effective ways to cut scope while preserving high quality, each of which has different tradeoffs:

- **Simplify Over-engineered Designs:** Make sure that you are doing the simplest thing that meets the requirements for today's functionality, testability, and ease of maintenance – don't over-engineer for the possible future.
- **Simplify Features:** Most specific feature requests can be solved by alternative means that still address the business problem. An automatic notification whenever any order is placed may be solved by a complex real-time change to the on-line order system, or by an hourly email with an order summary report. The latter might be a matter of minutes to implement with a simple database report, while the former might mean intrusive, risky changes to the order processing logic in the system.
- **Cut Low Priority Features:** As a last resort, you may need to eliminate the lowest priority features to make a deadline.

You build trust and confidence by involving the stakeholders in tough decisions while there's still time for them to react; so communicate with your customer about the proposed cuts as soon as possible to allow them to re-negotiate priority if necessary and maintain value.

Kanban development:

Kanban is a lean manufacturing process designed to eliminate unnecessary work in progress¹⁵. Kanban thinking applied to Agile development results in sweeping changes including:

- time-boxed development is out¹⁶
- stories are larger and fewer
- estimation is optional or out completely
- velocity is replaced by cycle time

Kanban development revolves around a visual board used for managing work in progress.



¹⁵ For more on Kanban in manufacturing see: <http://en.wikipedia.org/wiki/Kanban>

¹⁶ For more in time-boxing see: http://www.mosaicprojects.com.au/WhitePapers/WP1020_Time_Boxing.pdf

Thoughts on Agile

The basic idea is stories start on the left side of the board and move quickly through the phases of development necessary for them to be considered 'done'. Finished stories, ready to release into production pile up at the end.

Each process step column is divided into two parts: The top is used for stories currently in progress in that phase. The bottom is the buffer. When work for that phase of the story is completed, it moves from 'in progress' in the current phase to the 'buffer' where it will wait to be pulled into the next phase.

Because this is a Kanban board the amount of work in progress is limited, as is the number of stories allowed on the board. The numbers written on the bottom limit the number of stories allowed at each station. The stations themselves are not fixed; they are optimised for any particular software environment. The number of stories allowed overall and at each station are a factor of the people capable of doing the work. The team needs to be balanced and flexible to minimise throughput time and avoid bottlenecks at individual stations.

Kanban stories are larger than the minimal stories used in other Agile techniques. Each story represents a 'minimal marketable feature' (MMF). To be marketable the feature needs to be large enough to be useful. A MMF may take weeks to build. But the important thing isn't how long it takes to build, but that it will be understandable and valuable to those who'll receive it.

Conclusion:

Agile is an effective tool for use in delivering projects where the outcome is uncertain. Whilst created as a software development methodology, many of the ideas can translate to mainstream project management.

P109 - This paper is a 'work in progress' – it will be updated on a regular basis as we find more interesting ideas.