Contents

1 Agile Projects

This chapter provides the background to the need for an agile approach to projects. It covers the benefits of using an agile approach and the typical components of an agile project.

This book is not a basic introduction to project management. That is covered in Effective Project Management in easy steps.

Introduction

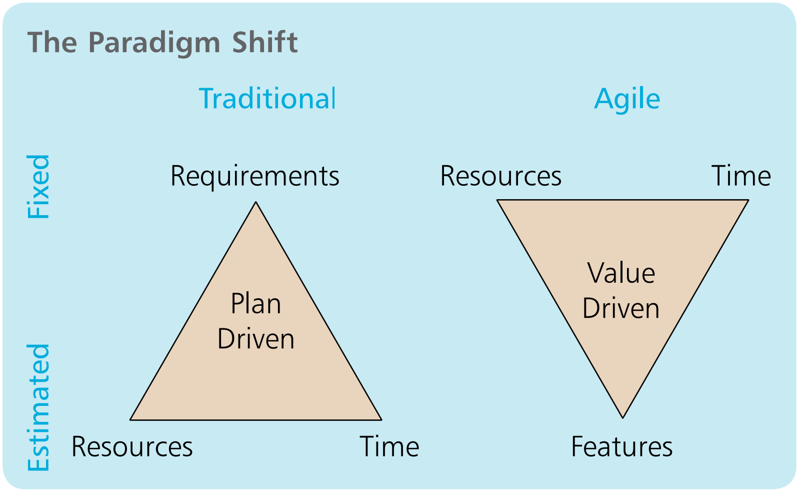
This book is primarily intended for project managers who are moving into the project management of agile projects. It will also be of interest to agile developers who wish to know more about project management. Finally, it will also be of interest to anyone else who wishes to know more about the management of agile projects.

Traditional Projects

The traditional approach to projects and project management started by defining exactly what the project was expected to produce. This was termed the requirements or specification and was agreed and signed off between the project team and the business or customer. The project team then went away and built a product or system that they thought met those requirements and, some time later, presented the finished product to the customer. The problems with this approach are set out later in this chapter but the end result was all too often that it was not what the customer needed.

Agile Projects

The agile approach to projects starts out with the expectation that the requirements (or features) will evolve and change during the course of the project. What is fixed and agreed between the project team and the customer is the resources that will be used and the time that will be taken by the project team to deliver as much as possible of the prioritized features the customer wants. The difference between the two approaches is illustrated below:



About the Book

This first chapter provides an introduction to agile projects, beginning with some of the challenges of using the traditional approach and how the agile approach deals with them. It then examines the typical components of an agile project.

Agile Project Management

Chapter 2 describes Agile Project Management in detail; compares it with traditional approaches; and introduces a framework for all aspects of project managing an agile project.

Other Agile Methods

There is a broad range of methods that could be described as agile, and Chapters 3-7 cover the more popular ones: Scrum, Dynamic Systems Development Method, Extreme Programming, Lean Development, and Feature-Driven Development.

Getting Started

Chapter 8 deals with the agile approach to starting a project, including pre-project activities, such as: assessing the feasibility, producing the Terms of Reference, and outlining the project plan.

Project Phases

Chapter 9 introduces the Foundation Phase, including establishing the Business Foundations, developing the Requirements List (Backlog), Systems Architecture, Development Approach, Solution Prototype and Delivery Plan.

Chapter 10 covers the development of the solution including the exploration and engineering phases. It also covers user experience, product assurance, testing, deployment planning, and review.

Chapter 11 deals with the deployment of the solution together with the project and increment reviews. It looks at benefits enablement and ends with the end-project assessment.

Project Closure

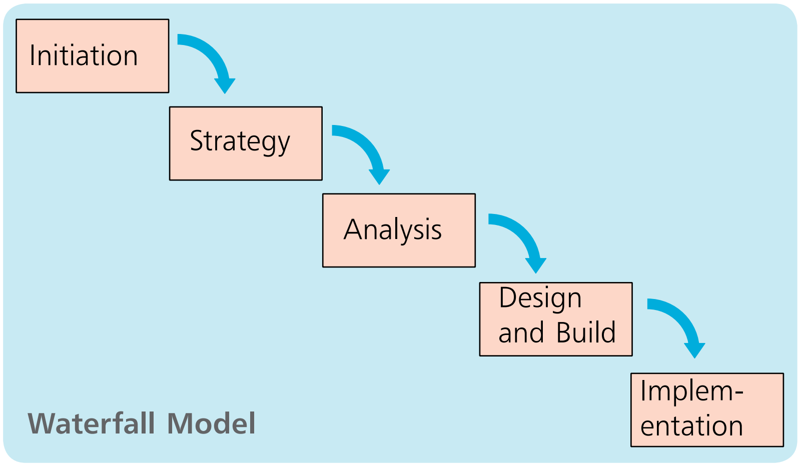
Chapter 12 covers closing the project down, planning and conducting a post-project benefits assessment. It concludes with a set of guiding principles for the success of an agile project.

Agile Projects at Scale

Chapter 13 describes a number of frameworks for managing agile projects at scale, with hundreds of developers, across multiple locations, within portfolios of many programmes of change.

The Problems

Traditional project methodology originally evolved from the construction and manufacturing industries, where the prohibitive cost of making late changes meant requirements were frozen as early as possible. It was based on a sequential approach, with progress flowing steadily downward through the stages of the project, each stage building on the work of the preceding stage. This led to its description as the Waterfall Model.



Fixed Requirements

This methodology was mandated by government, but it did not always suit the software development process, as requirements typically change throughout the course of a project. Various attempts were made to deal with this, such as 'upstream propagation' in which changes made later in the project should be fed back upstream to the earlier stages of the project so that requirements could be updated to reflect the changes.

While this had some success, a more radical approach was needed and methodologies such as Rapid Application Development (RAD), Spiral Development, and Iterative and Incremental Development were developed. In these, the requirements emerge throughout the course of the project, are delivered in a series of releases, and gradually build up the complete functionality.

In addition to the problems arising from rigid methodology and project life-cycle, there were a couple of further problems stemming from the use of the traditional approach.

Business Involvement

In the traditional approach the business was kept well away from the Development Team. They were consulted in the early stages to define the requirements and they were involved at the end to test the finished product. But if they saw what was being done during the project, they might well decide it was wrong and ask the developers to change it. This would be disastrous for the project, as re-working, and the consequent need for re-testing, would delay the project. Delivering on time was the prime aim of the developers, although it was rarely achieved.

The problem was, that delivering something on time that was not what the business needed was an even worse outcome. It gradually dawned on people that it was better to involve the business actively in the project. By having them involved right through the project, it would at least ensure that what was delivered was what the business needed, even if it was a little late.

Project Management

Traditionally, the Project Manager operated in what has now been referred to as a Command and Control style. In this, the Project Manager developed the detailed Project Plan, identifying all the tasks that required completion. They then allocated tasks to each member of the Project Team telling them exactly what to do. While this might, originally, have been fine for the construction industry, it did not get the best or most creative output from software or engineering developers.

The solution was to empower the Project Team more, and actively engage them in developing the detailed plans for the project. That way, the plans were not only likely to be better but the developers would also be more committed to them. Secondly, by allowing the developers to decide for themselves what needed to be done to deliver the requirements, they could make best use of the skills and knowledge of each team member and probably get much better commitment from the team members.

These types of changes to the methods being used on software development projects were collectively referred to as "Lightweight Methods", to differentiate them from the old heavyweight approach. These included DSDM, Scrum, Crystal Clear, Extreme Programming, Feature-Driven Development, Lean Development, and Agile Testing, amongst others.

The Agile Manifesto

In February 2001, representatives and users of the more popular lightweight development methods met at the Snowbird ski resort in Utah to discuss the need for an alternative to the existing heavyweight, documentation-dependant, software development processes. At the end of the conference, they published their thoughts as a manifesto to define a common approach to what they described as agile software development:

The Agile Manifesto

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.

What it Means

The meanings of the manifesto items on the left within the agile software development context are as described below:

- **Individuals and Interactions**: in an agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

- **Working Software**: working software will be much more useful and welcome than just presenting detailed documents to clients in meetings.

- **Customer Collaboration**: requirements cannot be fully defined at the beginning of the software development cycle; therefore continuous customer involvement is very important.

- **Responding to Change**: agile development is focused on quick responses to change and continuous development, harnessing change for the customer's competitive advantage.

Twelve Principles

These 12 principles underlie this agile manifesto:

1. **Customer Satisfaction**: by the early and the continuing delivery of useful software
2. **Changing Requirements**: welcome changing requirements, even late in the development process
3. **Frequent Delivery**: of working software, from every couple of weeks to every couple of months
4. **Measure of Progress**: delivery of working software is the principle measurement of progress
5. **Sustainable Development**: so the sponsors, developers and users can maintain a constant pace indefinitely
6. **Close Cooperation**: business people and developers must work together daily throughout the project
7. **Motivated Individuals**: by giving them the support they need and trusting them to get the job done
8. **Face-to-face Conversation**: the most efficient and effective method of conveying information in a Development Team
9. **Technical Excellence**: through continuous attention to technical excellence and good design
10. **Simplicity**: by keeping things simple the amount of work that has to be done is minimized
11. **Self-organizing teams**: the best architectures, requirements and designs emerge from self-organizing teams
12. **Regular Adaptation**: the team reflects on how to become more effective and adjusts its behavior accordingly

Agile Approach

As we have already seen, traditionally, project management and software development were largely based on a sequential design process referred to as the 'waterfall' model. However this did not suit the software development process, where requirements could and often did change during the course of a project and more agile methods were evolved.

Project Management Methodology

Although traditional project management methodology can be applied to all types of projects, there are some special constraints that apply to agile projects. The features (requirements) are not only allowed, but expected to change and evolve through the course of the project, while the resources and time are frozen. So the project will deliver as much of the prioritized requirements as can be delivered in the available time and within the cost budget.

Within each phase of an agile project, the developers collaborate closely with representatives of the business or customer so they understand the detail of the next step and can create an evolving solution. Before the product, process or software goes into production the business can decide if they want to continue on the same path or make alterations.

Agile Project Management

Because of the radical nature of these methods, the traditional (waterfall based) approach to project management, with requirements being defined and fixed early in the project, did not fit comfortably with this new approach. So a new form of Agile Project Management began to develop.

In 2010 the DSDM Consortium published a definition of Agile Project Management, based on the DSDM method and interfacing with other agile methods such as Scrum and XP. This Agile Project Management differs from traditional project management in a number of key respects:

Management Style

On a traditional project, the Project Manager may be actively involved in directing the work of the team and telling them what to do. This is sometimes referred to as Command and Control. In Agile Project Management the Project Manager is more of a facilitator and his/her role is to ensure that the collaboration between the business and the developers is effective.

...cont'd

Features

As the required features are expected to develop and change during the project, the traditional approach of fixing requirements and allowing time and resources to flex to meet them is reversed. In an agile project, time and resources are fixed (through time-boxing and small teams) and features are allowed to change at the start of each new iteration of the product.

Planning

In a traditional project, the Project Manager would develop and own the project plan. In an agile project, the features are constantly changing, so planning for each phase, release or iteration is carried out as late as possible. Further, although the outline project plan is produced by the Project Manager, the detailed plans are produced by the Development Team.

Project Phases

In place of the traditional (waterfall) project stages, agile projects use a number of phases, containing several iterations, leading to a number of product releases and, therefore, a series of implementations.

Change Control

The traditional project concept of change control is replaced by the Features Backlog. This is a list of prioritized business requirements, which is controlled by the business.

Risk Management

In place of the traditional approach to risk management and concerns about scope creep, a broader approach to risk is taken in an agile project. The Developers own the development risks and the business takes a more proactive role as the Product Owner.

Organization

In a traditional project, the Project Manager hands out work packets to the team. In an agile project, this is managed by the Development Team, and the Project Manager takes on more of a supportive role to the team.

Monitoring Progress

In a traditional project, the Project Manager has a detailed Gantt chart against which to monitor progress. In an agile project, his/her role is to record the effort used on a Burn-down chart.

Hot tip

In an agile project, the Project Manager's role is to facilitate and support the team, not to tell them what to do.

The Benefits

The traditional (or waterfall) project management methodology has been criticized for not being able to cope with the uncertainty that is typical for software development projects, for working in silos, for delivering nothing until the very end, and for compromising on quality to meet deadlines. These are probably the most significant benefits of using agile methodology.

Changing Requirements

Not only does the agile approach have the benefit of dealing with changing requirements, it also overcomes the difficulty that customers have in adequately specifying their requirements in the first place, before they have even seen some sort of prototype. Specifying requirements before starting the actual development places a huge and unnecessary overhead on the project and is likely to cause long delays to the project starting to produce anything. The iterative nature of the agile approach makes it an excellent choice when it comes to managing development projects.

Customer Involvement

Failure to involve the business or external customer fully in the project is likely to lead to dissatisfaction with the final product; whereas, close involvement means they can share in making decisions, setting priorities, and resolving problems. They will be committed to the final product as it will meet their requirements.

Quick Results

The use of agile practices will ensure that the project delivers a quality product much earlier than on a traditional project. The product will not yet have all of the required features but, on the Pareto Principle, the delivery of the most important 20% of the features should deliver around 80% of the benefits.

This also means that you have the option, should the customer decide they have received sufficient functionality and benefit, of bringing the project to a close much earlier, and thus freeing up resources for other much needed change and improvement.

Progress Measurement

Using the delivery of a series of working products as the main form of progress measurement is one that everyone involved in the project can understand. The Project Manager can measure and report on it. The developers can work to delivering it, and the customers can touch and feel it.

Team Motivation

By empowering the Development Team, allowing them to organize themselves and having active customer involvement, the team will be much more highly motivated and produce better results. Close cooperation on a daily basis with the customer will also add to their motivation and the delivery of a better product.

Product Quality

The focus on technical excellence and good design in an agile project, coupled with continuous testing, will ensure that a product of excellent quality is delivered. The close involvement of the customer will ensure their feedback to the process so that the product is not only excellent but it is what the customer needs.

Ideal Projects for Agile Methods

We can now begin to see the type of project that would most benefit from the agile approach. Any development project, particularly one that has unclear requirements, will benefit from the agile approach, especially if it has a fairly short timeline – ideally less than one year. The business or customer has to be able to make use of iterative product delivery with gradually enhanced features. Finally, the team should not be too large; because of the close cooperation and face-to-face working methods, a large team would place a heavy communications burden on the project.

Hot tip

If your project fits this profile you should seriously consider using agile methodology.

Unsuitable Projects for Agile Methods

Having established the ideal type of project to benefit from the agile approach, we can now see where it may not be so beneficial or even necessary. If this were not the case then all projects would use agile methods.

Some projects (such as major construction projects) start with requirements that are clearly defined, with a low likelihood of change over the course of the project, and a clear or mandated technical solution. For such a project, agile methods would not seem to offer any advantage, and many project managers feel that it would probably best be managed using the traditional methods.

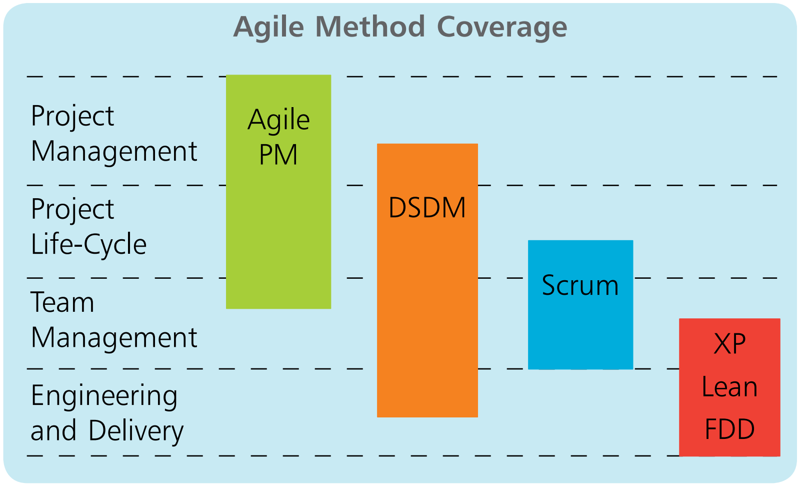
Beware

If your project fits this profile you may consider traditional methods more appropriate.

Many project management practitioners believe that agile methods do not scale well, and so choose to manage large projects using traditional methods. However, over the last few years, a number of frameworks and patterns have emerged that enable the management of agile projects at scale (see Chapter 13).

Project Components

In addition to Agile Project Management, the other typical agile methods that may be used on an agile project include: DSDM, Scrum, Extreme Programming, Lean Software Development, and Feature-Driven Development. The relationship between these elements is illustrated in the following method coverage diagram:



As can be seen from the diagram above, there is a significant amount of cross-over between the various methods. All of these agile methods are compatible and complementary, and many project managers adopt elements from each in parallel.

Agile Project Management

Agile Project Management was a new initiative launched in 2010 by the DSDM Consortium (**www.dsdm.org**). It took the project management and project life-cycle elements of DSDM and enhanced them by the addition of advice and guidance based on existing good practice. This methodology enables project managers to adopt an agile approach within their organizations and to interface well with agile development teams.

DSDM

The Dynamic Systems Development Method (DSDM) provides a framework for agile project delivery, with good practice guidance for the delivery of quality products, on time and within budget. It is designed to be tailored and used in conjunction with other agile methodology and traditional configuration and quality management systems.

...cont'd

Scrum

Scrum was originally formalized for software development projects, but works well for any complex, innovative project. Starting with the prioritized Product Backlog (requirements), the team has an agreed amount of time (a Sprint) to complete its work (typically two weeks), and pulls sufficient items from the top of that Backlog and decides how to implement them.

At the end of the Sprint, the work is reviewed and potentially deployed, then the next Sprint begins. This continues until all the Product Backlog has been completed or the project ends.

Extreme Programming

Extreme Programming (XP) is another popular agile software development process, which stresses customer satisfaction. Instead of delivering everything on a date in the future, this process aims to deliver the software that is needed as and when it is required.

Extreme Programming allows the software developers to respond confidently to changing customer requirements, even late in the life-cycle. Extreme Programming emphasizes teamwork with management, customers and developers all being equal partners in a collaborative team. The team is self-organized around the problem to solve it as efficiently as possible.

Lean Development

Lean Software Development was developed from Toyota's manufacturing process, and can best be summarized as seven major principles, which are derived from lean manufacturing: Eliminate Waste (including unnecessary code and functionality); Amplify Learning; Decide as Late as Possible; Deliver as Fast as Possible; Empower the Team; Build Integrity In (so components work well together) and See The Whole (software systems are the product of their interactions).

Beware

XP, Lean, and FDD only apply to software projects, whereas DSDM and Scrum can be used on all types of development projects.

Feature-Driven Development

Feature-Driven Development (FDD) is another iterative and incremental software development approach that aims to deliver, in a timely manner, quality software that clients value. FDD combines a number of technical practices, including: object modeling, class ownership, feature teams, code inspections, regular builds, and strongly promotes the benefit of continuous integration, automated testing, and continuous deployment.

Summary

- This book provides an introduction to agile projects, a definition and framework of Agile Project Management, a set of typical agile methods and a description of each phase of an agile project with their deliverables.
- Traditional projects start by specifying the requirements; agile projects start by defining the resources and time frame, then allowing the requirements to emerge during the project.
- Traditional methodology was based on the Waterfall Model which is not well-suited to software development projects.
- In addition to fixing the requirements too early, traditional methodology also suffered from a lack of customer involvement and an authoritarian style to managing the team.
- As a result of these problems software developers started to define lightweight methods, including: Scrum, DSDM, Extreme Programming, Lean Development, and FDD.
- These methods were codified in the Agile Manifesto, which called for a focus on individuals and interactions; working software; customer collaboration; and responding to change.
- Trying to manage an agile project with traditional project management methodology proved problematic so the concept of Agile Project Management was born.
- A definition of Agile Project Management was published by the DSDM Consortium and this focused on managing a project following the principles of the agile manifesto.
- The ideal agile project includes any small software development project, with a short time frame, a small team, and full involvement of the business or customer.
- Conversely, projects with fixed requirements and solutions may be better managed using traditional methodology.
- A number of frameworks have emerged that mean larger projects and programs can also follow with agile methods.
- The typical components of an agile project are therefore likely to include some or all of: Agile Project Management, DSDM, Scrum, Extreme Programming, Lean Development, and FDD.