1 Introduction

1.1 Why This Book?

Lets start of with a basic question "Why should you read this book?" The answer, as I hope you will see, is because it brings together a range of the most popular *Agile Methods* and presents them back to back allowing you, the reader, to gain an insight into what it means to be agile, what agile methods are (and are not), what Agile Modelling is and what XP (Extreme Programming) is. However, it goes further than this and considers how some of the approaches can be used together, how you can plan larger agile project (using a feature-driven approach) and how you can introduce agile methods into your organisation. All of this is done in a practical, no-nonsense manner that cuts through the hype and tells it to you straight!

That is, you should read this book because it tells you how to actually plan, organise and approach software systems in an Agile Manner. It does not try to sell you an evangelical, purist or (pardon the pun) extreme view. Instead it introduces the core concepts and methods in a concise and easy digested form. It also evaluates how successful the core techniques, such as Extreme Programming (often referred to as XP) and Agile Modelling can be, as well as what problems may be encountered. It then shows how some of these problems have been overcome on real-world projects by combining XP, Agile Modelling and Feature-Driven Development.

Without a book like this you can be left wondering what to do with Extreme Programming on a large software project. You might find yourself asking "must I always pair program if I wish to be agile?" or "should I do any design at all if I want to be agile?" or "how can I plan a project to be delivered to a client who uses a PRINCE 2 project management method when I am developing using an agile approach?" With this book you will know!

1.2 A Bit of History

I first encountered Extreme Programming early in 2000 when I was running a software development project, to create a data import and export utility for a large logistics company. The tool to be created need to be interactive, supporting data preview facilities, XML, Excel and database formats and to have excellent error

reporting tools. Two of the developers on the project started talking about working on some of the more difficult problems "Extreme" style. It was somewhat to my surprise that this seemed to mean that they worked on the problem together, with the two of them working on a single machine. Intrigued, I let them continue – partly to see what happened and partly to understand what they were up to. The result was that they dealt with the problem quickly and effectively. They continued to work in this way for the remainder of the project. The delivered system not only worked, but also was well constructed and very maintainable (as many subsequent changes were required as the clients software requirements kept changing).

As my interest in this "Extreme" style grew, I found that many of the aims and objectives of the agile movement fitted well with my own. I also found that I could immediately identify with many of the issues being addressed and began to incorporate them more and more into the projects I was involved in running. This did not mean that we immediately adopted a 100% pure Extreme Programming approach. Rather we gradually became more and more agile on a project-by-project, and client-by-client, basis. One of the more obvious aspects of this to me was that people became more comfortable with working together and in asking for help, input or feedback.

Are the projects I work on agile? Yes, I believe they are. Do they use Agile Modelling techniques – yes, I believe they do. Do we always use an Extreme Programming approach – actually no, it depends on the project and the developers involved, but then to me that is the point. We use each technique where and when we feel it is appropriate – we are pragmatists rather than evangelical purists!

1.3 What Is Agile Software Development?

Agile software development is an attempt to put the software being developed first and to acknowledge that the user requirements change. It is *agile* because it can respond quickly to the users' changing needs. Agile software development, in general, advocates frequent and regular, software releases. This allows new versions of the software to be released to users quickly and frequently. In turn, users can respond frequently and quickly to these releases with feedback, changing requirements and general comments (such as "that's not what we meant!"). As the releases are frequent, the time taken for the users to see the results of their comments is short and they are then able to respond to the new version quickly. This is particularly useful in a volatile environment, such as the environment of the web during the late 1990s and early 21st century.

In turn, agile software development puts the software first, because almost any activity undertaken must be to the benefit of the software that will be delivered. For example, if I create a model of some problem domain, then that model is not the actual delivered software, although it may help me to understand the problem better, in order that I can write the executable software. However, if the only use I make of the model is to help me understand the problem, then I shouldn't worry about updating it after the implementation varies from the design – at least not until I need the model again. That way, the effort is focussed on developing the software, rather than spending time on other supporting (but ultimately less productive) activities.

1.4 Why Be Agile?

You may be wondering at this point "Why would I even want to be agile?," "What difference can agile software development make?" or "What has Extreme Programming and Agile Modelling got to do with me?" The answer to these questions is in this book, but the essence of the whole of the agile movement is the production of working software systems, on time and within budget.

So why should you consider adopting agile development methods – the answer is simple – because they can help you produce software within budget and agreed timescales, that actually does what the users want! And we all want that right!

Of course, you might well argue that you have "seen it all before," and that this is what software engineering has been promising for decades. And you would be right, but things change. Today we live in a fast-moving, uncertain world, in which users' requirements seem to be as changeable as the weather (here in the UK) and software needs to have been delivered yesterday!

Traditional, single iteration, waterfall development approaches often can't cope. Other approaches have been proposed and new technologies created. Yet, still software is late, or over budget, or both. Even when delivered it often has numerous requirements removed or contains undesirable features/bugs.

Being agile doesn't guarantee the removal of all bugs, nor does it guarantee an easy ride, but it does help produce useful software within a project framework that can adapt to changing requirements.

In the current climate, everyone needs to be Agile!

1.5 What This Book Is About?

This book is about exploiting as many features of the agile movement as possible to enhance our software development processes. It is about selling you the concept of agile software development. It is about how to make your projects agile. It is about what tools you should use to become agile.

This book is about not rejecting anything out of hand just because it seems radical, but also of not just accepting all of a concept just because the literature says you should. It is about being realistic about what you can achieve in your organisation/team/situation.

It is not a detailed in-depth presentation of any of the agile methods (there are already very good books available that focus on each approach in depth – so why write another one?), it is not a diatribe for, or against, any particular method, nor is it a prescription for all of software development's ills.

This book is thus a *guide to the agile software development* methods currently available!

1.6 Implementation Languages

For the most part, this book is agnostic when it comes to a particular programming language, as we are dealing with issues relating to programming and software

development rather than the actual software development itself. However, there are times when the book does refer to a particular programming language or to a particular technology or tool used with one programming language or another. In these situations, the book tends to use the programming language Java for its examples. This is intentional – it is the environment that I am most familiar with and within which all my training; mentoring and development work takes place. This does not mean that if you are a hardened C++ developer, or a newly trained C# developer, this book is not of interest to you – merely that the few code examples will be less relevant. However, even here the issue may be blurred as some of the UML diagrams happen to be generated from java code, but would essentially be the same if they were generated from C#, for example.

1.7 The Structure of the Book

The remainder of the book has the following structure.

Chapter 2 : Agile Methods and the Agile Manifesto
This chapter covers the basics of the ideas and the philosophy behind the agile movement. It explains what it is to be agile and presents a summary of the agile manifesto. The manifesto, produced by the *Agile Software Development Alliance*, expresses all the common philosophies that underlie the agile movement. As such it is the heart of what it is to be agile! The chapter then describes the different methods that are loosely grouped under the Agile banner including Agile Modelling, Extreme Programming (XP), The Dynamic Systems Development Method (or DSDM), SCRUM and Feature-Driven Development (FDD).

Chapter 3: Agile Modelling
Modelling has a great number of misconceptions associated with it. This chapter first attempts to dispel these myths. It then introduces Agile Modelling. Agile Modelling can be seen as the design-side equivalent of the coding-oriented Extreme Programming. It tries to bring agile principles to the act of application modelling.

Chapter 4: How to Become an Agile Modeller
In this chapter, we will look at how agile modelling can be implemented, what it means to be an agile modeller and how agile modelling practices can be put into practice.

Chapter 5: Extreme Programming (XP)
Extreme Programming or XP is part of the agile movement that focuses on the writing of the software that will implement the system required by the end users. This chapter introduces XP by presenting the four key values of XP. It then describes the 12 practices that essentially form XP. Note that XP is comprised of a set of practices and is thus a very lightweight process and is actually more of a set of guidelines than a methodology!

Chapter 6: XP in Practice
Chapter 5 describes in more detail the practices that make up XP; however, these practices on their own provide very little in the way of guidance on how to run an actual XP project. This chapter therefore describes how to implement XP on a software project. It takes the XP practices and gives guidance on how to actually make them work.

Chapter 7: Agile Modelling and XP
This chapter considers how to use Agile Modelling within an XP project. It also considers how agile modelling and XP relate to one another.

Chapter 8: Agile Modelling and XP Reviewed
This chapter reviews agile modelling and XP in light of various experiences including the authors' own experiences. It also considers applying agile modelling and XP to larger scale projects. It concludes by considering the best situations within which to apply these techniques.

Chapter 9: Feature-Driven Development
This chapter discusses an agile development process based on Features. A *Feature* is a schedulable piece of functionality, something that delivers value to the user. It considers the role of FDD in planning agile development projects.

Chapter 10: An FDD Example Project
The aim of this chapter is to present an example of a project that employs agile principles but that has been planned using an FDD approach.

Chapter 11: Agile Methods with RUP and PRINCE 2
This chapter considers how Agile Modelling and the Rational Unified Process can fit together. It concludes by briefly discussing how agile methods and PRINCE 2 can work in tandem.

Chapter 12: Introducing Agile Methods into Your Organisation
In this chapter, we consider strategies for introducing Agile Development methods into an organisation.

Chapter 13: Tools to Help with Agile Development
Life can be made a great deal easier if you choose the appropriate toolset to use. In this chapter, we discuss some of the tools that can really help to simplify life on an agile development project. In particular, we discuss the make-like tool ANT, the use of version management software such as CVS (Continuous Versioning System), unit testing software such as JUnit and IDEs such as Eclipse.

Chapter 14: Obstacles to Agile Software Development
This chapter discusses what obstacles may be encountered within an organisation, what problems may occur and the difficulties relating to agile development methods. It also considers some strategies for mediating these issues.

1.8 Where to Get More Information?

There are now many books available on various subjects within the agile movement (there are even books, technologies and organisations that are trying to jump on the agile bandwagon and make themselves appear agile). So which books might you consider? In this section, I list the six or seven books that provide the best coverage of the core aspects of Extreme Programming, Agile Model and Feature-Driven Development (the main subjects covered in this book).

The definitive book on Extreme Programming is

- *Extreme Programming Explained: Embrace Change*. K. Beck, Addison-Wesley, 1999.

Although there are many other books on Extreme Programming, two standout from a practical perspective, these are:

- *Extreme Programming Applied: Playing to Win* (The XP Series), Ken Auer, Roy Miller, Addison Wesley - New York, 2002.
- *Extreme Programming Installed*, Ron Jeffries, Ann Anderson, and Chet Hendrikson, Addison-Wesley, ISBN: 0201708426, 2000.

A very good book that helps to introduce pair programming into an organisation is:

- *Pair Programming Illuminated*, Laurie Williams and Robert Kessler, Addison-Wesley Professional, 0-201-74576-3, 2003.

The definitive Agile Modelling book is by Scott Ambler:

- *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*, Scott W. Ambler, Wiley and Son, Inc., 0 471 20282 7, 2002.

Books relating to Feature-Driven Development include the following books. These books focus on the use of the Together toolset, but include valuable information on helping to plan agile projects by basing project planning around desired features.

- *Better Software Faster,* A. Carmichael, D. Haywood, Prentice-Hall NJ, 0-13-008752-1, 2002.
- *Java Modeling in Color*, P. Coad, E. Lefebvre and J. De Luca, Prentice-Hall, Englewood Cliffs, NJ, 1999.

1.9 Where to Go Online?

There is also a wealth of information available online. There are websites out there that deal with the agile movement in general, with Extreme Programming, with Agile Modelling, etc. Some of these are listed below.

Agile Movement in General
www.agilealliance.org Agile Software Development Alliance
Agile Modelling
www.agilemodeling.com Agile Modelling mailing list
Extreme Programming
http://extremeprogramming.org/
http://www.xpuniverse.com
http://www.pairprogramming.com/