# Agile methods: a comparative analysis

**Diane Strode**
Faculty of Business and Information Technology
Whitireia Community Polytechnic
Porirua, New Zealand
d.strode@whitireia.ac.nz

## Abstract

The agile methods are systems development methodologies currently used in the software development industry both internationally and in New Zealand. This article provides an overview of the agile methods, including the key publication of each method, the major influences on the agile methods, and describes comparative studies where analysis and comparison of methodologies has been carried out. Then a comparative study is described which was carried out on five agile methods in order to address the question 'what is an agile method'? A comparative analytical framework suitable for this purpose is described along with the results of applying the framework to five agile methods: DSDM, XP, Scrum, ASD, and Crystal Methods. The results provide an analysis of the properties common to agile methods, the differences between the methods, the unique properties peculiar to agile methods, and provide an understanding of appropriate method combinations.

*Keywords*: Agile methods, XP, systems development methodologies.

## 1 Introduction

There are many systems development methodologies (Jayaratna, 1994) and many different types of systems development methodologies; structured (Eva, 1994; J. Martin & Finkelstein, 1981; Royce, 1987), object-oriented (Graham, Henderson-Sellers, & Younessi, 1997; Jacobson, Booch, & Rumbaugh, 1999) and RAD methodologies (J. Martin, 1991) are three common types. The agile methods are another group of methodologies published in the late 1990s to 2002. The agile methods are currently used in the software development industry both internationally (R. Charette, 2001) and in New Zealand (Strode, 2005). This article presents the results of applying an analytical framework to five of the agile methods to determine what they consist of, how they differ from one another, and how they can meaningfully be combined. The research question guiding the study was 'what is an agile method'.

This quality assured paper appeared at the *19th Annual Conference of the National Advisory Committee on Computing Qualifications* (NACCQ 2006), Wellington, New Zealand. Samual Mann and Noel Bridgeman (Eds.). 

This article first defines 'systems development methodology' then presents a history of the agile methodologies, a description of the research question 'what is an agile method?' and its sub questions, a description of the use of frameworks in methodology selection and comparison, the framework itself, how the framework was applied to five of the agile methods, and the results of this application.

## 2 Systems development methodology

A number of definitions of 'systems development methodology' have been published over the last 20 years. This definition is both comprehensive and explicit:

*An information systems development methodology can be defined as a collection of procedures, techniques, tools, and documentation aids which will help the systems developers in their efforts to implement a new information system. A methodology will consist of phases, themselves consisting of sub-phases, which will guide the systems developers in their choice of the techniques that might be appropriate at each stage of the project and also help them plan, manage, control and evaluate information systems projects* (D.E. Avison & Fitzgerald, 1995a, p. 10).

The authors added the proviso that a methodology is based upon a philosophical view and a set of stated objectives and is not just a recipe to be followed without adaptation to the particular situation in which it is used.

Note that in this article the terms 'method' and 'methodology' are used interchangeably.

## 3 The agile methods

The agile methods share common characteristics and were first defined as a group in 2001 with the publication of the agile manifesto (AgileAlliance, 2001). The authors of the manifesto were people who had published individual software development methods with similar characteristics. Each of these methods is based on practitioner experience and evolutionary development practices with a focus on early delivery of quality software. The methods and methodologists that contributed to defining the agile methods were (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003):

- Extreme programming (Beck, 1999, 2000),
- Crystal Methods (Cockburn, 2002),

- Adaptive Systems Development method (J. A. Highsmith, 2000),
- Dynamic Systems Development Method (Stapleton, 1997),
- Pragmatic Programming (Hunt & Thomas, 2000),
- Scrum (Schwaber & Beedle, 2002),
- Other practitioners and methodologists

Based on an analysis of the agile methods investigated by Abrahamsson, Salo, Ronkainen, & Warsta (2002), Cockburn (2002), Highsmith (2002), and Abrahamsson, Warsta, Siponen, & Ronkainen (2003) I identified twelve agile methods published from 1997 to 2003 (see Table 1).

**Table 1: Agile methods by earliest publication date**

| | Agile Method | | Primary Source | |
|---|---|---|---|---|
| | | | **Journal Article** | **Book** |
| 1 | Dynamic Systems Development method | DSDM | DSDM_ Consortium (1995) | Stapleton (1997) |
| 2 | Crystal methods | Crystal | Cockburn (1998) | Cockburn (2002) |
| 3 | RUP (configured) | dX | Martin (1998) | |
| 4 | Extreme Programming | XP | Beck (1999) | Beck (2000) |
| 5 | Adaptive Software Development | ASD | | Highsmith (2000) |
| 6 | Scrum | Scrum | Beedle, Devos, Sharon, Schwaber, & Sutherland (1999) | Schwaber & Beedle (2002) |
| 7 | Pragmatic Programming | PP | | Hunt and Thomas (2000) |
| 8 | Internet Speed Development | ISD | Cusumano & Yoffie (1999) Baskerville & Pries-Heje (2001) | |
| 9 | Agile Modeling | AM | | Ambler (2002) |
| 10 | Feature Driven Development | FDD | | Palmer & Felsing (2002) |
| 11 | Open Source Software Development | OSS | Sharma, Sugumaran, & Rajagopalan (2002) | |
| 12 | Lean Development | LD | Charette (2002) | Poppendiek & Poppendiek (2003) |

## 4 The evolution of agile methods

The precursor methods and major influences on each of the agile methods is available in an evolutionary map developed by Abrahamsson et al. (2003). Their study indicated four main influences on agile software development: object-orientation, evolutionary development, internet technologies and methodology engineering. The object-oriented influences were from UML (Rumbaugh, Jacobson, & Booch, 1999) and RUP (Kruchten, 2000). Evolutionary approaches include the influence of Boehm's spiral model (1988), prototyping and RAD. The internet technologies include ideas from Microsoft's sync and stabilise method (M. Cusumano & Selby, 1997) and Internet speed development (Baskerville, Levine, Pries-Heje, Ramash, & Slaughter, 2001; Baskerville & Pries-Heje, 2001; M. A. Cusumano & Yoffie, 1999). The methodology engineering ideas came from Kumar and Welke (1992) and amethodological development (Baskerville, Travis, & Truex, 1992; Truex, Baskerville, & Klein, 1999; Truex, Baskerville, & Travis, 2001). This evolutionary path shows that agile methods are an amalgamation of previous methodologies, ideas, and practices from software engineering.

Influences from information systems on the agile methods include the Participatory Design movement (Kuhn & Muller, 1993) which focused on stakeholder participation in development, and soft systems methodology (Checkland, 1999) which provided ideas of human activity systems being holistic and evincing emergent properties.

## 5 Research question

The research question that guided this study was; what is an agile method? This question is important for a number of reasons. A common theme or set of characteristics of agile methods has not been substantially identified. Reifer (2002, p. 18) stated that it is necessary to 'clearly identify what "agile methods"' means because, after questioning software engineers, he found that there is no common meaning of the term. A definition is needed so that the method users and those who want to study, evaluate, and compare methods have a common understanding of what an agile method is and which methods legitimately belong to the set of agile methods. Currently when new system development methods claiming to belong to the set of agile methods appear there is no way to show that the claim is legitimate. One method for checking the claim would be to benchmark the new method against the distinct properties of agile methods. However these properties must be determined before making this type of classification. Another outcome of having a definition for 'agile method' is to enable agile methods as a whole to be compared with other classes of methodology (Iivari, Hirschheim, & Klein, 2001). Currently in order to compare methods and select the most suitable for use in a project it is necessary to read a book on each method. This is a challenge for any project manager who needs to select an appropriate agile method from those currently available (Nerur, Mahapatra, & Mangalaraj, 2005). Those who want to select a suitable method for their project or who want to study the techniques (e.g. pair programming) of the methods have very few unbiased studies to choose from that define, describe and compare the published agile methods. This article provides some of this information by defining and comparing five of the earliest published

agile methods in order to answer the question; what is an agile method? The question was broken down into sub questions (see Table 2) and answering these sub questions provides an explicit definition of the properties, common characteristics and differences between each agile method, and possible combinations of agile methods. Published problems or weaknesses are included as it is important to know in what environments agile methods are not suitable as specified by the method author.

**Table 2: Research question; what is an agile method?**

| | Secondary questions | Purpose |
|---|---|---|
| 1.1 | What are the properties of individual agile methods? | To define each individual agile method by describing the philosophy, principles, practices and other stated properties of the method.<br>To provide an understanding of which agile method is most appropriate for a particular set of circumstances, when those circumstances are stated by the method author.<br>To enable comparisons of methods.<br>To provide technique checklists for use in survey instruments |
| 1.2 | What are the properties common to all agile methods? | To provide an understanding of the philosophies, principles, concepts and practices, common to all agile methods leading to a definition of 'agile method'.<br>To enable accurate classification of new 'agile' methods in the future. |
| 1.3 | What are the differences between agile methods? | To enable appropriate selection of an agile method based on different needs. |
| 1.4 | What properties are unique to particular agile methods within the set of agile methods? | To determine if agile methods are different to one another or are all a version of one basic method. |
| 1.5 | What combinations of agile methods are possible? | To determine how agile methods can be combined to provide a more useful or encompassing method capable of solving a wider range of problems more effectively. |
| 1.6 | What are the published problems with agile methods? | To know in what environments agile methods are not suitable according to the method authors.<br>To clarify the strengths and weaknesses of each agile method and of all agile methods within the criteria of the framework. |

## 6 Comparative analytical frameworks

A comparative analytical framework (framework) is a set of criteria against which a methodology is assessed. A framework provides a systematic way of understanding, comparing and evaluating methodologies. A framework is analytical because it breaks a methodology into parts, and comparative because it organises the detail of the methodology into a format that makes it easy to compare different methodologies. It can be used to characterise the philosophy and properties of an individual methodology, to discover differences and similarities between methodologies, to isolate properties unique to a particular methodology, and to determine the strengths and weaknesses of a methodology. A framework can be used to select a suitable methodology for a project from a set of possible methodologies and can show how methodologies with particular weaknesses can be coupled with other methodologies to improve their coverage or efficacy. The purpose of the framework I used was to clarify the body of knowledge about agile methods based on the method author's publication.

A large number of very different frameworks have been used to define, describe and compare system development methodologies (D.E. Avison & Fitzgerald, 1995a; Jayaratna, 1994; Olle, Sol, & Tully, 1983). A framework is a well-used method for clarifying the properties of, and comparing, methodologies.

In his introduction to the CRIS-2 conference Sol et al. (1983, p. 4) described five approaches to the assessment of methodologies. The most commonly used approach is to "*define a meta-language as a vehicle for communication and as a frame of reference in which various methodologies can be described. The attractiveness of this approach is that implicit, contextual features as well as the process aspects of a methodology can be made explicit. However, a meta-language may have a limited expressive power. It also may blind us for specific features of some methodologies.*"

The problem of 'limited expressive power' is a fundamental and insolvable problem with using frameworks that, I believe, can be mitigated to some extent by selecting criteria used in previously proven frameworks, each with a different perspective.

There is little comparative analysis of agile methods; the main work is carried out by Abrahamsson's group (2002, 2003). Abrahamsson, Salo, Ronkainen, and Warsta (2002) carried out a systematic review of the existing literature on agile software development methods and used a comparative framework to organise the results. They integrated the knowledge and terminology in this area by comparing ten agile methods (see Table 3) and concluded the following: agile methods take a people-centric, developer's view of software development, they are for use in small teams of 10 or less developers, and they are created from existing software development methods used in novel ways, and are not likely to be suitable for all types of project. Scalability problems are likely to constrain the adoption of agile methods into large organisations. Finally, they called for selection models for practitioners so that they could apply the most suitable method for their needs. They then addressed the question of; 'What makes a development method an agile one' (ibid, p. 98) in their systematic study of ten agile methods. They included in their comparison, any methods that met the values published in the agile manifesto. From an analysis of the existing literature (descriptive rather than empirical) they concluded that a development method is agile when software development is incremental, cooperative, straightforward and adaptive. Incremental development consists of rapid cycles with

small software releases. Cooperative development is when the customer and developers work constantly together with close communication. Straightforward development occurs when the method is well documented, easy to learn and to modify, and development is adaptive when customers are able to make last moment changes with ease. I believe there is a problem with this characterisation because it is not clear how the authors drew this conclusion from their review of the literature. The second contribution of this study was to describe the main features of each agile method. They used the criteria of process, roles and responsibilities, practices, adoption and experiences, scope of use and current research. Abrahamsson, et al. (2002) give no reasons for selecting their particular framework criteria, and its coverage is limited. Areas such as the application domain, models, tools and deliverables are not covered. The third contribution of the study was an analysis of each method against the criteria of; the method's key points, any special features, identified shortcomings, maturity (against stated criteria), adoption, and coverage of software development lifecycle activities. The problem with this analysis is that it is subjective in its judgements, a fault of all frameworks. To strengthen the conclusions this analysis could have been more specific in its sources by including more comprehensive referencing to the authors' stated principle or practices in the comparative tables. This study has two main strengths. As a summary of the literature of each of the methods and a comparison of methods this paper is the only article available which is not produced by the authors of an agile method or a member of the agile alliance (AgileAlliance, 2001). In addition it covers methods that are not usually included in the set of agile methods such as Pragmatic Programming, RUP, and Open Source Software development.

Building on the previous study Abrahamsson, Warsta, Siponen and Ronkainen (2003) investigated and described nine agile methods (see Table 3) with respect to software development life-cycle coverage, support for project management, whether the method offers abstract principles or concrete guidance, whether the method is universally applicable or situation appropriate, and they tallied the empirical evidence supporting each of the methods.

Frameworks for methodology assessment do have problems. A framework may not uncover properties of a methodology that are not considered in the framework's 'terms of reference' and the assessment is subject to the bias of the assessor. However they are still the favoured method for assessing and comparing methodologies. The next section describes the agile framework for this study.

**Table 3: Comparative studies of agile methods**

| Short name | Proposed Agile Method | A | B | C |
|---|---|---|---|---|
| AM | Agile Modeling | | + | + |
| ASD | Adaptive Software Development | + | + | + |
| Crystal | Crystal methods | + | + | + |
| DSDM | Dynamic Systems Development Method | + | + | + |
| dX (RUP) | Rational Unified Process | | + | |
| FDD | Feature Driven Development | + | + | + |
| ISD | Internet Speed Development | | | + |
| LD | Lean Development | + | | |
| OSS | Open Source Software Development | | + | |
| PP | Pragmatic Programming | | + | + |
| | Scrum | + | + | + |
| XP | Extreme Programming | + | + | + |
| **Key**<br>A Reviewed by Highsmith 2002<br>B Reviewed by Abrahamsson *et al*. 2002<br>C Reviewed by Abrahamsson *et al*. 2003 | | | | |

## 7 A framework for agile methods

The agile methods framework used in this study is based on that of Avison and Fitzgerald (1995a, 2003). I selected their framework for a number of reasons. Firstly, it is based on a comprehensive analysis of the academic literature on methodology comparison and the design of comparative frameworks for evaluation of methodologies published from 1982 to1998. Secondly, their framework is designed to provide guidance both when examining an individual methodology and when comparing a number of methodologies. Although it is based on a study of academic publications it is also designed for use by practitioners. A third reason for selecting their framework was because it includes all of the criteria used by Jayaratna (1994) with the exclusion of the evaluation step. Jayaratna's evaluation is designed to assess a methodology before it is used, during its adoption and after its introduction to an organisation. My agile methods framework is designed only to assess the published characteristics of methods so Jayaratna's evaluation criterion is inappropriate here. Lastly, the framework criteria developed by Avison and Fitzgerald include all of the criteria used by Abrahamsson et al. (2002) in their framework, with the exception of the 'current research'.

Another possible way of structuring a comparative framework for agile methods would be to analyse each method against the themes described in the agile manifesto (AgileAlliance, 2001) following Sol's (1983) inductive approach. The problem with this is that agile manifesto themes are narrow in scope, and exclude many useful criteria that could help to elucidate the content of the agile methods.

My framework for analysis of agile methods is based upon a large number of past methodology frameworks and therefore provides a good coverage of possible comparison criteria. A broad range of criteria means that the framework highlights both the characteristics of agile methods, and the characteristics which are not part of the agile methods, and allows for a direct comparison of one agile method with another. The framework was not used to determine if one method is better than another. The

framework I have developed is appropriate for addressing the research question, what is an agile method.

The agile method analytical framework includes the criteria of Avison and Fitzgerald (2003), with product and output amalgamated, and I have added the criteria of identifier, assumptions, values, perspective, metrics, and tailorability. The full set of criteria is: identifier, philosophy, model, techniques, tools, scope, outputs, practice, product and tailorability (see Table 4).

**Table 4: The analytical framework for agile methods**

| | Criteria | Sub criteria |
|---|---|---|
| 1 | Identifier | Method name<br>Author<br>Date of first publication<br>Major publication<br>Country of origin |
| 2 | Philosophy | Paradigm<br>Assumptions and values<br>Perspective<br>Objectives<br>Domain<br>Target:<br>Size of project<br>Type of problem<br>Type of organisation<br>Size of organisation<br>Type of development<br>Type of application<br>Technology environment |
| 3 | Modelling technique | |
| 4 | Techniques | Practices<br>Metrics |
| 5 | Tools | |
| 6 | Scope | |
| 7 | Outputs | |
| 8 | Practice | Background<br>Roles and responsibilities<br>Difficulties with the methodology<br>Skill levels |
| 9 | Tailorability | |

## 8 Reasons for selecting early agile methods

Five of the earliest agile methods published between 1995 and 2002 were selected for analysis and comparison: DSDM, XP, Scrum, ASD, and Crystal methods. There are three reasons for this. A study of more than five methods is too large a task to achieve in the time frame available for this study, the analysis of five methods provides enough data for a meaningful comparison of the similarities and differences between each of the methods, and lastly they are likely to contain most of the techniques used in subsequently published agile methods.

The source of data for each method was a first or early edition of a publication in book form which fully described the method (see Table 1).

## 9 Results

The results for the question: *What is an agile method?* were provided by application of the analytical framework. The analysis is presented in Strode (2005) where the results were used to formulate a survey instrument for further in-situ analysis of the use of agile methods in New Zealand. The findings for each of the sub questions are now presented.

Research question 1.1: *What are the properties of individual agile methods?* A table of findings from application of the analytical framework provides a complete summary of each method including the identifier, philosophy, modelling technique, techniques, tools, scope, outputs, practice, and tailor-ability of each method (Strode 2005).

Research question 1.2: *What are the properties common to all agile methods?* The following common properties were found. All of the methods were published in the period 1995 to 2002 by authors based in the USA or UK. The methods are primarily objectivist because they address how to provide a technological solution to a given business problem (Avison and Fitzgerald (2003) provide a discussion of the objectivist and subjectivist approaches). All use incremental development with iterations of one week to four months with one month iterations recommended by all of the methods. Active user involvement, feedback and learning, teamwork, and empowering teams to make decisions, are all important factors in the methods. There is also an emphasis on communication between all of the stakeholders in the project; managers, project leaders, developers, and customers. This is facilitated by frequent meetings. Each of the methods addresses the need for effective control of incremental development in small teams of between 2 and 40 programmers, with the optimal team size of 3 to 10 people. The methodologies are all created by practitioners and are based on their experiences as developers and they all have a project manager and/or developer perspective. The methods are designed for solving business problems where changes in both requirements and technologies affect the project throughout its life cycle. The main product of development is working software. None of the methods specifies any particular modelling technique and minimising documentation is a goal in all of the methods. XP and Crystal place more emphasis on document minimisation. The properties are summarised in table 5.

**Table 5: Common properties of agile methods**

- Published between 1995 – 2002 in the USA and UK
- Objectivist methods which provide technical solutions
- Address business problems
- Practitioner based
- Project manager and developer perspective
- Incremental development
- Iterative development with 1 month iterations optimal
- Projects undergoing constant change
- Active user involvement
- Feedback and learning
- Teamwork
- Empowered teams
- Communication between all stakeholders is critical
- Small teams of 3-10 programmers is optimal
- Frequent meetings, daily is optimal
- Working software is the main product of development
- Modelling techniques are not mandated
- Minimise documentation

Some characteristics are common to subsets of the methods studied. They are as follows:

- DSDM, ASD and Crystal support tailoring of the techniques to fit the project, whereas XP is most effective if all of the techniques are used as specified. Scrum states that no tailoring is needed.
- DSDM, XP, Scrum and Crystal recommend testing throughout the lifecycle. However there are no testing techniques recommend by all of the three methods.
- DSDM, XP, Scrum and Crystal recommend that the customer is located with the developers or on-site and available at all times.
- Time pressure is a problem addressed by ASD and DSDM, it is a project variable to be controlled in Scrum and XP, and is not mentioned in the Crystal methods.
- XP, Scrum and ASD all support the idea of emergent properties of effective teams and state that their techniques will support this. Collocation of teams is another technique shared by these methodologies along with Crystal. Emergence may be related to collocation but this is not clearly explored in the method descriptions. DSDM is not concerned with how teams are located.
- Object technology and Internet technology, including the assumption that knowledge of UML will be used as needed during development, is common to XP, Scrum, ASD and Crystal. DSDM assumes either structured or object-oriented development with appropriate models created if necessary.

Research question 1.3: *What are the differences between agile methods?* The difference between the methods is that they each serve a different purpose. They share some goals and some techniques as described above but it is the purpose of each methodology which distinguishes it from the others. The purpose of each of the methodologies is as follows:

- DSDM is a framework for RAD development.
- Scrum is a methodology for project management of iterative development.
- XP is a methodology for software development in high change environments using small teams and standard software engineering techniques to satisfy customer needs and maintain effective teams.
- ASD is a framework for managing software development projects which are under intense time pressure and have rapidly changing requirements. The method is based on complex adaptive systems theory and uses RAD techniques.
- Crystal methods are for designing a methodology to suit a specific project.

Each of the methods belongs to the family of agile methods but they are designed to achieve different, sometimes overlapping, purposes.

Research question 1.4: *What properties are unique to agile methods within the set of agile methods?* The analysis shows that there are three techniques not previously used by other system development methodologies or any other agile method. These are techniques described in XP; test-first development, pair programming, and the use of a system metaphor.

Research question 1.5: *What combinations of agile methods are possible?* ASD and DSDM are frameworks therefore they do not specify how to carry out each technique described in the method or even what techniques must be used. This means that they can be used with techniques from other methods. ASD and DSDM could both be used with techniques from XP and Scrum. Scrum and XP can be effectively used together with Scrum providing the management practices and XP providing the software development techniques. Crystal method can also be used with XP. XP acts as the base method which is tailored for each project using Crystal principles. Scrum, ASD and Crystal can use any software engineering techniques to fulfil their goals as long as those techniques achieve the goals of the methodology.

Research question 1.6: *What are the published problems with agile methods?* There is no consistent pattern or problems with all of the methods. Scrum reports no problems, ASD reports no problems except that it is not compatible with CMM initiatives. DSDM and XP report problems will occur when their base environmental conditions are not met. Crystal reports problems when projects become large and distributed.

Research question 1: *What is an agile method?* Based on the answers to research questions 1.1 to 1.6, I have defined an agile method in this way: *An agile method is a software development methodology designed for the management and support of iterative and incremental development of business systems in environments where change is constant. Agile methods use software development techniques that enhance teamwork in small empowered teams and support active customer involvement. An agile method is designed to produce working software early, using communication, feedback, learning and frequent meetings in preference to than modelling and documentation. Agile methods adapt existing software development techniques to achieve these goals*

## 10 Conclusions

Application of the framework has shown that each agile method is a methodology in its own right, consisting of most of the parts of a methodology as defined in section 2. Each method has a stated philosophy, collection of procedures, techniques, and tools. The agile methods use phased development and provide techniques to use during the phases. Agile methods enable the planning, management, evaluation, and control of iterative and incremental software development. The agile methods omit documentation aids and modelling techniques and in this respect they do not meet the definition provided by Avison and Fitzgerald (1995b) earlier in this article.

I developed the definition of agile methods by applying an analytical framework to each of the five earliest published agile methods leading to an explicit definition of what makes up each method. This showed the common characteristics and activities shared by these agile methods and the activities that they unanimously minimise. The activities that are minimised are the

documentation and systematic production of modelling artefacts during development. These activities and artefacts are not central to the methods as they were with many of the structured design methodologies of the 1980s (Olle et al., 1983) and object-oriented methodologies of the 1990s (Graham, 1991).

Another outcome of applying the framework was to show that while sharing some characteristics the agile methods each serve a different purpose. Application of the framework also clarified which combinations of agile methods are likely to be most useful. This is because the framework makes clear the goal of the method and the techniques that make up the method. All involved XP with other methods. It is clear why this is so after seeing the number and explicit nature of the techniques belonging to each method. When a method is not explicit as to how a particular activity should be carried out XP provides many suitable techniques.

This research provides a foundation upon which empirical studies can be based as it shows exactly what an agile method consists of based on a detailed analysis of five of the earliest published methods. Technique lists for each of the agile methods studied are available in Strode (2005). I have also provided an analysis of agile methods which provides a suitable foundation for comparison of the agile methods with other systems development methodologies, those from the past, and those yet to be invented.

## 11 References


Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods: review and analysis. *VTT Publications 748* Retrieved 1 January, 2004, from http:/www.inf.vtt.fi/pdf

Abrahamsson, P., Warsta, J., Siponen, M. K., & Ronkainen, J. (2003, 3-10 May). *New directions on Agile methods: a comparative analysis.* Paper presented at the 25th International Conference on Software Engineering.

AgileAlliance. (2001). Manifesto for agile software development. Retrieved 17 February, 2003, from http://www.agilemanifesto.org

Ambler, S. (2002). *Agile modeling: effective practices for Extreme Programming and the Unified Process*. New York: John Wiley & Sons, Inc. New York.

Avison, D. E., & Fitzgerald, G. (1995a). Chapter 7 Methodologies: issues and frameworks. In *Information systems development: methodologies, techniques and tools* (2nd ed.). London: The McGraw-Hill Companies.

Avison, D. E., & Fitzgerald, G. (1995b). *Information systems development: methodologies, techniques and tools* (2 ed.). London: The McGraw-Hill Companies.

Avison, D. E., & Fitzgerald, G. (2003). Chapter 27: Methodology comparisons. In *Information systems development: methodologies, tools and techniques* (3rd ed.). London: McGraw-Hill Publishing Company.

Baskerville, R. L., Levine, L., Pries-Heje, J., Ramash, B., & Slaughter, S. (2001). How internet companies negotiate quality. *IEEE Computer, 34*, 51-57.

Baskerville, R. L., & Pries-Heje, J. (2001). Racing the E-bomb: how the Internet is redefining information systems development methodology. In B. Fitzgerald, N. Russo & J. DeGross (Eds.), *Realigning research and practice in IS development* (pp. 49-68). New York: Kluwer.

Baskerville, R. L., Travis, J., & Truex, D. P. (1992). Systems without method: the impact of new technologies on information systems development projects. In K. E. Kendall, K. Lyytinen & J. DeGross (Eds.), *Transactions on the impact of computer supported technologies in information systems development* (pp. 241-260). Amsterdam: Elsevier Science Publications.

Beck, K. (1999). Embracing change with Extreme Programming. *Computer, 32*(10), 70-77.

Beck, K. (2000). *Extreme programming explained: embrace change*. Boston: Addison-Wesley.

Beedle, M., Devos, M., Sharon, Y., Schwaber, K., & Sutherland, J. (1999). Scrum: a pattern language for hyperproductive software development. In N. Harrison, B. Foote & H. Rohnert (Eds.), *Pattern languages of program design* (pp. 637-651). Addison-Wesley.

Boehm, B. (1988). A spiral model of software development and enhancement. *Computer*, 61-72.

Charette, R. (2001). The decision is in: agile versus heavy methodologies. *e-Project Management Advisory Service, Cutter Consortium, 2*.

Charette, R. N. (2002). *Foundations of Lean Development: The Lean Development manager's guide* (Vol. 2). Spotsylvania, Va.: ITABHI Corporation.

Checkland, P. (1999). *Systems Thinking, Systems Practice. Soft systems methodology: a 30-year retrospective*. Chichester: John Wiley & Sons, Ltd.

Cockburn, A. (1998). *Surviving object-oriented projects: a manager's guide*. USA: Addison Wesley Longman.

Cockburn, A. (2002). *Agile software development*. Boston: Addison-Wesley.

Cusumano, M., & Selby, R. W. (1997). How Microsoft builds software. *Communications of the ACM, 40*, 53-61.

Cusumano, M. A., & Yoffie, D. B. (1999). Software development on Internet time. *Computer*, 60-69.

DSDM_Consortium. (1995). *Dynamic Systems Development Method* (2nd ed.). Ashford: Tesseract Publishing.

Eva, M. (1994). *SSADM Version 4: a users guide* (2nd ed.). London: McGraw-Hill Book Company.

Graham, I. (1991). *Object oriented methods*. Harlow, UK: Addison-Wesley.

Graham, I., Henderson-Sellers, B., & Younessi, H. (1997). *The OPEN process specification*. Harlow, England: Addison-Wesley.

Highsmith, J. (2002). *Agile software development ecosystems*. Boston: Addison-Wesley.

Highsmith, J. A. (2000). *Adaptive software development: a collaborative approach to managing complex systems*. New York, NY: Dorset House Publishing.

Hunt, A., & Thomas, D. (2000). *The pragmatic programmer*. USA: Addison Wesley.

Iivari, J., Hirschheim, R., & Klein, H. K. (2001). A dynamic framework for classifying information systems development methodologies and approaches. *Journal of Management Information Systems, 17*(3), 179-218.

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The unified software development process*. Reading, Massachusetts: Addison-Wesley.

Jayaratna, N. (1994). *Understanding and evaluating methodologies NIMSAD: a systematic framework.* London: McGraw-Hill Book Company.

Kruchten, P. (2000). *The Rational Unified Process: an introduction* (2nd ed.). USA: Addison Wesley Longman.

Kuhn, S., & Muller, M. J. (1993). Participatory design. *Communications of the ACM, 36*(4), 25-28.

Kumar, K., & Welke, R. J. (1992). Methodology engineering: a proposal for situation-specific methodology construction. In W. W. Cotterman & J. A. Senn (Eds.), *Challenges and strategies for research in systems development* (pp. 257-269). New York: John Wiley & Sons.

Martin, J. (1991). *Rapid Application Development*: Macmillan.

Martin, J., & Finkelstein, C. (1981). *Information Engineering. Vol 1 and 2*. Englewood Cliffs, New Jersey: Prentice Hall.

Martin, R. C. (1998). *The process: object oriented analysis and design with applications*. USA: Addison Wesley.

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM, 48*(5), 73-78.

Olle, T. W., Sol, H. C., & Tully, C. (Eds.). (1983). *Information systems design methodologies: a feature analysis. Proceedings of the IFIP WG8.1 Working Conference on Feature Analysis of Information Systems Design Methodologies*. York, United Kingdom: Elsevier Science Publishers B.V.

Palmer, S. R., & Felsing, J. M. (2002). *A practical guide to Feature-Driven Development*. Upper Saddle River, NJ 07458: Prentice Hall PTR.

Poppendiek, M., & Poppendiek, T. (2003). *Lean software development an agile toolkit*. Boston: Addison-Wesley.

Reifer, D. J. (2002, July/August). How good are agile methods? *IEEE Software,* 16-18.

Royce, W. W. (1987). *Managing the development of large software systems.* Paper presented at the 9th Annual conference on Software Engineering, Monterey, California, United States.

Rumbaugh, J., Jacobson, I., & Booch, G. (1999). *The Unified Modeling Language reference manual*. Massachusetts: Addison-Wesley.

Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum*. Upper Saddle River, New Jersey: Prentice Hall.

Sharma, R., Sugumaran, V., & Rajagopalan, B. (2002). A framework for creating hybrid-open source software communities. *Information Systems Journal, 12*(1), 7-25.

Sol, H. C. (1983). A feature analysis of information systems design methodologies: methodological considerations. In W. Olle, H. G. Sol & C. Tully (Eds.), *Information systems design methodologies: a feature analysis*. Amsterdam: North-Holland.

Stapleton, J. (1997). *DSDM Dynamic systems development method*. Harlow, England: Addison-Wesley.

Strode, D. E. (2005). *The agile methods: an analytical comparison of five agile methods and an investigation of their target environment.* Unpublished Master of Information Sciences (Information Systems), Massey University, Palmerston North.

Truex, D. P., Baskerville, R. L., & Klein, H. K. (1999). Growing systems in emergent organisations. *Communications of the ACM, 42*(117-123).

Truex, D. P., Baskerville, R. L., & Travis, J. (2001). Amethodological systems development: the deferred meaning of systems development methods. *Accounting, Management and Information Technology, 10*, 53-79.