European and Mediterranean Conference on Information Systems (EMCIS) 2006,

July 6-7 2006, Costa Blanca, Alicante, Spain

COMPARATIVE EVALUATION OF XP AND SCRUM USING THE 4D ANALYTICAL TOOL (4-DAT)

Asif Qumer, Brian Henderson-Sellers, Faculty of Information Technology, University of Technology, Sydney, NSW, Australia
asif@it.uts.edu.au, brian@it.uts.edu.au

Abstract
The emergence of agile software development methods provides a contribution to contemporary software engineering practices. Agile methods have several benefits over traditional plan-based methods, in particular their ability to handle projects where requirements are not fixed. In the last few years, a number of agile software development methods have been developed but a detailed evaluation (which is essential) of these methods is not available. This paper presents a detailed comparative analysis of two well known agile methods (XP and Scrum), using the previously published 4-Dimensional Analytical Tool (4-DAT), based on four characterization perspectives: those of scope, agility, agile values and software process. A report generated with the help of 4-DAT will assist organizations in making decisions about the selection or adoption of an agile method.

Keywords: Agile methods, Software Development Methodologies

1 INTRODUCTION

Agile software development methods such as Extreme Programming (Beck, 2000) claim to deliver a quick software solution to a client's rapidly changing requirements (Taylor, 2002; Williams and Cockburn, 2003). Despite such claims, industry is often still reluctant to adopt and introduce an agile style of development in their organizations. To assist in their adoption, it is necessary to evaluate the agile methods in detail and determine their suitability for a particular software development project. The focus of this paper is to compare and analyze two agile software development methods (XP and Scrum) in detail using the 4-Dimensional Analytical Tool or 4-DAT (proposed by Qumer and Henderson-Sellers, 2006 and summarized in Section 2 below). 4-DAT is a framework-based assessment tool that may be used to analyze any software development method. The distinguishing feature of 4-DAT is that it specifically provides a mechanism to measure the degree of agility of any method quantitatively at a specific level in a process and using specific practices – not possible with existing analytical tools and frameworks such as those of Kitchenham and Jones (1997), Cuesta *et al.* (2002), Williams *et al.* (2004), Boehm and Turner (2004) or Tran *et al.* (2004). The agility measurement mechanism described in this tool (an agility measurement calculator) has also been prototyped in Java™ and will be fully implemented as the next step in this research project. A report generated with the help of 4-DAT will assist organizations in making decisions about the selection or adoption of an agile method. This paper provides an illustrative application of the 4-DAT assessment tool to two well known agile methods: Extreme Programming (XP) and Scrum. Further methods will be analyzed and compared in future work.

This paper begins with an overview of 4-DAT. It then discusses the two selected agile methods: Extreme Programming (XP) and Scrum. Thirdly, it presents the detailed analysis of the selected agile methods. Finally, it concludes with a discussion of future research work.

2 4-DIMENSIONAL ANALYTICAL TOOL (4-DAT)

4-DAT was developed for researchers and practitioners for the purpose of analysis and comparison of agile methods. 4-DAT has four dimensions that provide evaluation criteria for the detailed assessment of agile software development methods from different perspectives. These dimensions were developed by utilizing previously published concepts regarding agility, agile software development methods,

traditional software development methods, agile values and agile principles. The following sub-sections explain the four dimensions of 4-DAT.

2.1 Method scope characterization

The method scope characterization dimension is a set of key scope items that have been distilled from selected agile methods: Extreme Programming (Beck, 2000), Feature Driven Development (Palmer and Felsing, 2002; Koch, 2005), Adaptive Software Development (Highsmith, 2000, 2002a,b), Dynamic Software Development Method (Stapleton, 1997; DSDM, 2003a,b) and Scrum (Schwaber and Beedle, 2002). This first dimension for the evaluation approach helps to compare the methods at a high level. Table 1 describes the first dimension of 4-DAT.

Scope	**Description**
1. Project Size	Does the method specify support for small, medium or large projects?
2. Team Size	Does the method support for small or large teams (single or multiple teams)?
3. Development Style	Which development style (iterative, rapid) does the method cover?
4. Code Style	Does the method specify code style (simple or complex)?
5. Technology Environment	Which technology environment (tools, compilers) does the method specify?
6. Physical Environmer	Which physical environment (co-located or distributed) does the method specify?
7. Business Culture	What type of business culture (collaborative, cooperative or non-collaborative) does the method specify?
8. Abstraction Mechanism	Does the method specify abstraction mechanism (object-oriented, agent-oriented)?

Table 1. 4-DAT Dimension 1 (after Qumer and Henderson-Sellers, 2006)

2.2 Agility characterization

Agility characterization is the second dimension of 4-DAT, providing a set of agility features derived from the the agility definitions proposed by Qumer and Henderson-Sellers (2006). Dimension two (see Table 2) checks the existence of agility in agile methods at both a process level and a method practices level. This is the only one of the four proposed agility dimensions that is quantitative.

Features	**Description**
1. Flexibility	Does the method accommodate expected or unexpected changes?
2. Speed	Does the method produce results quickly?
3. Leanness	Does the method follow shortest time span, use economical, simple and quality instruments for production?
4. Learning	Does the method apply updated prior knowledge and experience to learn?
5. Responsiveness	Does the method exhibit sensitiveness?

Table 2. 4-DAT Dimension 2 (after Qumer and Henderson-Sellers, 2006)

2.3 Agile values characterization

The characterization of "agile values", dimension three, is a set of six values: four of them are provided by the Agile Manifesto (2001), with the fifth agile value provided by Koch (2005). The sixth value "keeping the process cost effective" was proposed in Qumer and Henderson-Sellers (2006) based on their study of several agile methods. Dimension three examines the support of agile values in different practices of agile methods. Table 3 describes this third dimension of the 4-DAT.

Agile values	Description
1. Individuals and interactions over processes and tools	Which practices value people and interaction over processes and tools?
2. Working software over comprehensive documentation	Which practices value working software over comprehensive documentation?
3. Customer collaboration over contract negotiation	Which practices value customer collaboration over contract negotiation?
4. Responding to change over following a plan	Which practices value responding to change over following a plan?
5. Keeping the process agile	Which practices help in keeping the process agile?
6. Keeping the process cost effective	Which practices help in keeping the process cost effective?

Table 3. 4-DAT Dimension 3 (after Qumer and Henderson-Sellers, 2006)

2.4 Software process characterization

Dimension four has two main components of software process: product engineering process and process management process. The former has a further three categories: development process, project management process and support processes (Jalote, 1997). Dimension four examines the practices that support these four processes in agile methods (Table 4).

Process	Description
1. Development Process	Which practices cover the main life cycle process and testing (Quality Assurance)?
2. Project Management Process	Which practices cover the overall management of the project?
3. Software Configuration Control Process/Support Process	Which practices cover the process that enables configuration management?
4. Process Management Process	Which practices cover the process that is required to manage the process itself?

Table 4. 4-DAT Dimension 4 (after Qumer and Henderson-Sellers, 2006)

3 AGILE SOFTWARE DEVELOPMENT METHODS

Two well-regarded methodologies (XP and Scrum) have been chosen for detailed comparative analysis. Here, we give an overview of these prior to an evaluation of them in Section 4 using 4-DAT.

3.1 Extreme programming (XP)

The XP software development process focuses on iterative and rapid development. XP is characterized by six phases: Exploration, Planning, Iterations to first release, Productionizing, Maintenance and Death (Beck, 2000, p.131). Beck (2000) characterizes the scope of XP as being suitable for small to medium size projects; small and co-located teams; produces simple and clean code in small releases; requires quick feedback from the technological environment; the physical environment of the office should support communication and coordination among the team members at all times; and requires cooperation between the customer, management and development team to form the supportive business culture for the successful implementation of XP.

3.2 Scrum

Schwaber and Beedle (2002) report that Scrum is a flexible, adaptable, empirical and productive method that uses the ideas of industrial process control theory for the development of software systems. According to these authors, Scrum has three phases: Pre-Game, Development and Post-

Game. The Pre-Game phase has a further two sub-phases: planning and high level design (architecture). Scrum supports small and medium level projects but may be scaled up for the development of large projects. Scrum encourages small teams, where each has less than ten members.

4 A COMPARATIVE ANALYSIS OF AGILE METHODS

4.1 Scope of XP and Scrum

In this part of the comparative analysis, the scope of both XP and Scrum is analyzed quantitatively by using the first dimension of 4-DAT (shown in Table 1). Table 5 presents the scope and usability of XP and Scrum. This table shows that XP and Scrum are suitable for small and medium sized projects and that Scrum is also scalable for large projects. XP and Scrum both use an iterative and incremental object-oriented approach to develop software products with a team size of less than 10. XP discusses code style (clean and simple), technology environment (requires quick feedback), physical environment (co-located and distributed teams) and business culture (collaborative and cooperative) whereas Scrum remains silent on these issues.

Criteria	**XP**	**Scrum**
Scope		
Project Size	Small, medium	Small, medium, and scalable for large
Team Size	< 10	< 10 and multiple teams*
Development Style	Iterative, rapid	Iterative, rapid
Code Style	Clean and simple	Not specified
Technology Environment	Quick feedback required	Not specified
Physical Environment	Co-located teams and distributed teams (limited-interaction)	Not specified
Business Culture	Collaborative and cooperative	Not specified
Abstraction Mechanism	Object-oriented	Object-oriented

Table 5. Scope Evaluation of XP and Scrum (indicates that this is a special feature)*

4.2 Degree of agility in XP and Scrum

In this part of the comparative analysis, the degree of agility (DA) is measured in terms of the five variables (features) relating to Dimension 2 (see Table 2) : flexibility (FY), speed (SD), leanness (LS), learning (LG) and responsiveness (RS) that may exist at some specific level or lifecycle phase or as a result of the practices used in the phases of XP and Scrum. If any phase or practice (XP or Scrum) supports a particular agility feature, then 1 point is allocated in that particular cell, otherwise 0; and so on. Tables 6, 7, 8 and Figure 1 demonstrate the results of the agility measurement for both XP and Scrum. The agile methods literature was used as the source of the numerical inputs for the analysis, calculated using the following equation (Qumer and Henderson-Sellers, 2006):

$$DA\ (Object) = (1/m)\sum m\ DA(Object,\ Phase\ or\ Practices)$$

The cell values for XP shown in Table 6 for the five characteristics of Dimension 2 show both the high level (phases) and low level (practices). For example, the planning phase in XP is not fixed and can be changed as we proceed in the software development so it can be said that it is a flexible phase - it is marked as point 1 in the FY cell. Planning is done quickly for each release and iteratively; thus it can be marked as speedy and allotted 1 in the SD cell. The plan may change at any time, bringing an extra cost for fixing it; hence it is not lean and is consequently given a 0 in the LS cell. However, a plan can be considered emergent in XP, with learning from previous releases and is thus marked 1 in the LG cell. Finally, a plan responds whenever required and is thus marked 1 in the RS cell. The rest of the phases and practices in both XP and Scrum have been evaluated in a similar way. In future work, we

will assess other ways of making these evaluations e.g. whether a Delphi approach might be valuable or, indeed, whether a subjective assessment by the project manager might have more local relevance.

Table 7 shows that the degree of agility is high in all categories except leanness (LS) for XP. A similar deficiency is seen for Scrum (Table 7) which has even lower values for the LS characteristic. For XP, the only "poor" phase is that of Death, with Maintenance having the second lowest value. For Scrum, similarly, the later phase of "Post-Game" has a low value (Table 7). Perhaps the most surprising results for XP are seen in Table 6 under Practices where it is clear that both "40 hour week" and "metaphor" score badly – surprising because these are two of the practices most heavily stressed by XP advocates. In contrast, all of Scrum's practices score highly.

	Agility Features					
XP	FY	SD	LS	LG	RS	Total
(i) Phases						
Exploration	1	1	0	1	1	4
Planning	1	1	0	1	1	4
Iteration to release	1	1	0	1	1	4
Productionizing	1	1	1	1	1	5
Maintenance	1	0	0	1	1	3
Death	0	1	0	0	0	1
Total	5	5	1	5	5	21
Degree of Agility	5/6	5/6	1/6	5/6	5/6	21/(6*5)
(ii) Practices						
The Planning Game	1	1	0	1	1	4
Short Release	1	1		1	1	4
Metaphor	0	1	1	0	0	2
Simple Design	1	1	1	1	1	5
Testing	1	1	0	1	1	4
Refactoring	1	1	1	1	1	5
Pair Programming	1	0	0	1	1	3
Collective Ownership	1	0	0	1	1	3
Continuous Integration	1	1	1	1	1	5
40-Hour Week	0	0	0	1	0	1
On-site Customer	1	0	0	1	1	3
Coding Standards	1	1	1	1	1	5
Total	10	8	5	11	10	44
Degree of Agility	10/12	8/12	5/12	11/12	10/12	44/(12*5)

Table 6. Degree of agility in XP – cell values

The cell values of Tables 6 and 7 are summarized in Table 8, which shows the overall degree of agility for both phases and practices of XP and Scrum. These assessments of the degree of agility permit a separate ranking and visual comparison for both a process-based viewpoint and a practice-based viewpoint (Figure 1). There is, of course, no easy and valid mathematical way of combining these two numbers and rankings. Thus the decision-maker needs to include their own, often subjective, weightings to any evaluation of the most appropriate agile method to be adopted by their organization or for a particular project. Thus, this dimension of 4-DAT is intended to provide a project manager with assistance and will never fully automate the process of agile process selection or construction.

	Agility Features					
Scrum	FY	SD	LS	LG	RS	Total
(i) Phases						
Pre-Game	1	1	0	1	1	4
Development	1	1	0	1	1	4

Post-Game	0	1	0	0	0	1
Total	2	3	0	2	2	9
Degree of Agility	2/3	3/3	0/3	2/3	2/3	9/(3*5)
(ii) Practices						
Scrum Master	1	1	0	1	1	4
Scrum Teams	1	1	0	1	1	4
Product Backlog	1	1	0	1	1	4
Sprint	1	1	0	1	1	4
Sprint Planning Meeting	1	1	0	1	1	4
Daily Scrum Meeting	1	1	0	1	1	4
Sprint Review	1	1	0	1	1	4
Total	7	7	0	7	7	28
Degree of Agility	7/7	7/7	0/7	7/7	7/7	28 / (7*5)

Table 7. Degree of agility in Scrum – cell values

Process & Practices	**XP**	**Scrum**
Phases	21/30 = 0.70	9/15 = 0.60
Rank	**1**	**2**
Practices	44/60 =0.73	28/35 =0.80
Rank	**2**	**1**

Table 8. Degree of agility in XP and Scrum – overall comparison

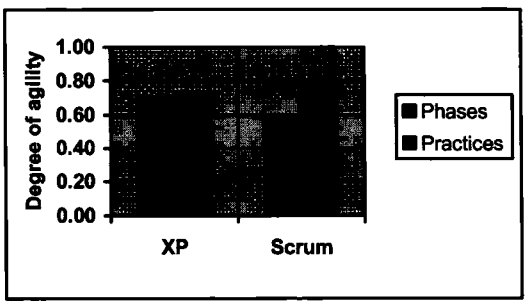

Figure 1 Degree of agility – a comparison of Phases and Practices for XP and Scrum

4.3 Agile values in XP and Scrum

In this part of the evaluation, cell values for Dimension three (see Table 3) are evaluated to examine the support of agile values in the different practices of XP and Scrum. Table 9 presents this comparison. However, in this case (unlike Dimension 2), the third dimension of 4-DAT is only qualitative and presents the agile values that are promulgated by the agile methodology under investigation. Here, we can see that both XP and Scrum offer support for all the initial agile values but the two most recently identified (Qumer and Henderson-Sellers, 2006) are less well supported. XP offers no support for either "Keeping the process agile" (which might suggest a missing link to the SPI (software process improvement) community) or for "Keeping the process cost effective" (a pragmatic agility value for commercial adoptability). This last characteristic is not seen in Scrum either.

Agile Values	**XP**	**Scrum**
Individuals and interactions over processes and tools	1. The planning game 2. Collective ownership 3. On-site customer 4. Pair programming	1. Scrum teams. 2. Sprint planning meeting. 3. Daily scrum meeting
Working software over	1. Short releases	1. Sprint

comprehensive documentation	2. Testing. 3. Continuous integration	2. Sprint review
Customer collaboration over contract negotiation	1. The planning game 2. On-site customer	1. Product backlog 2. Sprint planning meeting.
Responding to change over following a plan	1. Metaphor 2. Simple Design 3. Refactoring 4. Coding standards	1. Sprint review 2. Sprint planning meeting.
Keeping the process agile	-	1. Sprint review 2. Daily scrum meeting.
Keeping the process cost effective	-	-

Table 9. Degree of agility in XP and Scrum for Dimension 3

4.4 Software process in XP and Scrum

In this part of the evaluation, we examine the practices of both XP and Scrum for the support of software processes. Table 10 presents the evaluation report for this fourth dimension (see Table 4). As with Dimension 3, the comparison is purely qualitative and informative (descriptive rather than prescriptive). Both XP and Scrum offer practices for the Development and Project Processes but say nothing about configuration management or process management.

Software Process	**XP**	**Scrum**
Development Process	1. Short releases 2. Metaphor 3. Simple design 4.Testing 5.Refactoring 6. Pair programming 7.Collective Ownership 8.Continuous integration 9. On-site customer 10. Coding standard	1. Scrum teams 2. Product backlog 3. Sprint 4. Sprint review
Project Management Process	1. The Planning game	1. Scrum master 2. Sprint planning meeting 3.Daily scrum meeting
Software Configuration Control Process/Support Process	Not specified	Not specified
Process Management Process	Not specified	Not specified

Table 10. Software process in XP and Scrum (Dimension 4)

5 CONCLUSION

In this paper, we have analyzed and compared two agile software development methods (XP and Scrum) by using 4-DAT, a framework-based assessment tool, described in detail by Qumer and Henderson-Sellers (2006). This analysis used both a qualitative and a quantitative approach to evaluate the agile methods at both the phase level and the practice level. On the basis of this analysis, we may rank the methods – here, XP has more agile phases but less agile practices than Scrum (Figure 1).

Such an evaluation is, of course, not an easy task and the managers and developers may use this tool (4-DAT) and assessment report and tool (and may include their own criteria of evaluation) as a first step to select the most appropriate agile method to be adopted by their organization or for a particular project. In future, we intend to refine and update the proposed 4-DAT and apply it to the assessment of other available agile methods.

REFERENCES

AgileManifesto. 2001. *Manifesto for Agile Software Development.*

Beck K. 2000. *Extreme Programming Explained*, Addison-Wesley Pearson Education, Boston.

Boehm B and Turner R. 2004. 'Balancing agility and discipline: evaluating and integrating agile and plan-driven methods'. *Proceedings of the 26th International Conference on Software Engineering*, IEEE Computer Society, Washington, DC, USA, 718-719.

Cuesta P, Gómez A, González-Moreno J.C and Rodríguez F.J. 2002. 'A framework for evaluation of agent oriented methodologies'. *Workshop on Agent-Oriented Information Systems (AOIS-2000).*

DSDM. 2003a. *DSDM Consortium*, Dynamic Systems Development Method Ltd.

DSDM. 2003b. *Guidelines for Introducing DSDM into an Organization Evolving to a DSDM Culture.* DSDM Consortium.

Highsmith J.A.I. 2000. *Adaptive Software Development: A Collaborative Approach To Managing Complex Systems.* Dorset House Publishing, New York.

Highsmith J. 2002a. *Agile Software Development Ecosystems.* Pearson Education, Inc., Boston.

Highsmith J. 2002b. 'What is agile software development?' *CrossTalk Magazine.*

Jalote P. 1997. *An Integrated Approach to Software Engineering*, 2nd edn, Springer-Verlag, New York.

Kitchenham B.A and Jones L. 1997. 'Evaluating software engineering methods and tool part 5: the influence of human factors'. *ACM SIGSOFT Software Engineering Notes*, 13-15.

Koch A.S. 2005. *Agile Software Development: Evaluating the Methods for Your Organization.* Artech House, Inc, London.

Palmer S.R and Felsing J.M. 2002. *A Practical Guide to Feature-Driven Development.* Prentice-Hall Inc, Upper Saddle River.

Qumer A and Henderson-Sellers B. 2006. 'Measuring agility and adoptability of agile methods: A 4-Dimensional Analytical Tool'. *Procs. IADIS International Conference Applied Computing 2006*, IADIS Press, 503-507.

Schwaber K and Beedle M. 2002. *Agile Software Development with SCRUM.* Prentice Hall.

Stapleton J. 1997. *DSDM: The Method in Practice.* Addison-Wesley, Inc.

Taylor J.L. 2002. 'Lightweight Processes for Changing Environments (Book). Book Review'. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 27.

Tran Q.-N.N, Low G and Williams M.-A. 2004. 'A preliminary comparative feature analysis of multi-agent systems development methodologies'. *6th International Bi-Conference Workshop on Agent-Oriented Information Systems*, Springer-Verlag, New York, USA, 386-398.

Williams L, Krebs W, Layman L, Anton A.I and Abrahamsson P. 2004. 'Toward a framework for evaluating Extreme Programming', *Empirical Assessment in Software Engineering*, Edinburgh, 11-20

Williams L and Cockburn A. 2003. 'Agile software development: it's about feedback and change'. *Computer*, 36