CRYSTALLIZATION OF AGILITY
Back to Basics

Asif Qumer
Facutly of Information Technology, University of Technology, Broadway, Sydney, Australia
asif@it.uts.edu.au

Brian Henderson-Sellers
Facutly of Information Technology, University of Technology, Broadway, Sydney, Australia
brian@it.uts.edu.au

Keywords: Agility, Traditional software development, Agile software development.

Abstract: There are a number of agile and traditional methodologies for software development. Agilists provide agile principles and agile values to characterize the agile methods but there is no clear and inclusive definition of agile methods; subsequently it is not feasible to draw a clear distinction between traditional and agile software development methods in practice. The purpose of this paper is to explain the concept of agility in detail; and then to suggest a definition of agile methods that would help in the ranking or differentiation of agile methods from other available methods.

1 INTRODUCTION

Traditional plan-based software development methods work well if the project requirements are fixed; but are often considered slow and insensitive when the project requirements are being changed frequently (Paetsch *et al.*, 2003). The concept of agile software development evolved when people argued that the traditional software development approach often fails to produce valuable software products in certain situations; consequently, it was argued, new software development methods were needed (Beck, 2000). Furthermore Nandhakumar and Avison (1999) point out that traditional software development methods are too mechanistic to be used in detail (Abrahamsson *et al.*, 2002). Such concerns have encouraged software engineering practitioners to develop new approaches to software development, including lightweight and agile methods.

Cockburn (2002) defines the core of agile methods as "the use of light-but-sufficient rules of project behaviour and the use of human-and communication-oriented rules" (Abrahamsson *et al.*, 2002). The Agile Manifesto (2005) provides agile principles and agile values that qualitatively characterize the agile methods, but there is no precise and comprehensive definition of agile methods, *per se*. Conboy and Fitzgerald (2004) point out that the agile methods are significantly dependent on the principles embodied in the Agile Manifesto (2005) and in agile values; and there is no inclusive definition of agile methods.

The objective of this paper is to describe in detail the concepts of agility (basic elements of agility) that have been identified in our current research; and then to propose a more formal definition of agile software development methods in terms of the underlying concepts of agility, existing agile software development methods and agile principles (AgileManifesto, 2005). This paper has three sections. Firstly, it describes the concept of agility and its application. Secondly, it gives an overview of agile software development methods: Extreme Programming (XP) (Beck, 2000), Feature Driven Development (FDD) (Palmer & Felsing, 2002), Adaptive Software Development (ASD) (Highsmith, 2000), Dynamic Software Development Method (DSDM) (DSDM, 2003) and Scrum (Schwaber & Beedle, 2002) and building on both the definitions of agility and their manifestations in current agile methods, it proposes a definition of what should be the essence of an agile software development method. Finally, the paper concludes with a discussion of options for future research.

2 WHAT IS AGILITY?

The concept of "agility" conceals many facets such as nimbleness, suppleness, alertness, responsiveness, swiftness and activeness – yet it is difficult to give a rigorous or complete definition of agility. Agility may be taken as the demonstrable ability of anything that is capable of adapting to changes quickly and allowing anything to occur whenever it is required – and to do so with flexibility. According to Dove (1997), agility is a very seductive word, evidencing confusion for many with immediate and personal definitions. Hence, a clear, technical understanding of this concept needs to be crystallized.

2.1 Attributes of Agility

The concept of agility is not new; to understand the concept of agility, we first need to study the underlying concepts of flexibility, speed, leanness, learning and responsiveness. Flexibility is the ability to respond to the expected change whereas leanness accentuates lower cost, reduced timeframe and quality production (Dove, 1997). According to dictionary definitions, flexibility is the ability to adapt to change; speed characterizes rapid and quick behaviour; leanness refers to compactness and tidiness; responsiveness refers to life, reaction and sensitivity; and learning refers to knowledge and improvement.

2.2 Existing Definitions of Agility

Wong and Whitman (1999) argue that agility refers to the effective response to rapid and unexpected change with flexibility, which is a characteristic of agility (Table 1). This implies adaptability and versatility in the domain in order to respond to unexpected changes. Conboy and Fitzgerald (2004) propose the definition of agility as: "the continual readiness of an entity to rapidly or inherently, proactively or reactively, embrace change, through high quality, simplistic, economical components and relationship with its environment". These two definitions seem to overlook two important factors of agility: the learning factor and factors external to the domain and the environment. The learning factor demonstrates the capability of an agile entity that improves over a period of time as it gains in experience and acquires knowledge from its internal and external environment (Henderson-Sellers & Serour, 2005). Boehm and Turner (2004a; 2004b) assert that "agility applies memory and history to adjust to new environments, react and adapt, take advantage of unexpected opportunities, and update the experience base for future". This definition incorporates the learning factor but seems to have only a vague concept of environment since it overlooks the external environment factor. An agile entity should consider such external factors that may affect its working.

Table 1: Attributes of agility.

Features	Wong & Whitman (1999)	Conboy & Fitzgerald (2004)	Boehm & Turner (2004a; 2004b)
Flexibility	X	X	X
Speed	X	X	
Leanness		X	
Learning			X
Responsiveness	X	X	X

2.3 Comprehensive Definition of Agility

We suggest a definition of agility in the light of agility concepts and existing definitions proposed by different researchers (Section 2.1). Indeed, we have already applied and tested this definition (which is a basic requirement of a science) to measure the degree of agility of two well-known agile methods (Qumer & Henderson-Sellers, 2006b and Section 4.3). This definition may be used to measure the degree of agility of any method or technique, not only so-called agile methods. For example, Software Development Life Cycle (SDLC) (a.k.a. "waterfall": Royce, 1970) or Rapid Application Development (RAD) may be assessed for agility since we believe it is possible for any method to encompass some degree of agility, which may be ranked from weak to strong. This is an independent definition of agility (yet in the light of above definitions) that defines the concept of agility in terms of flexibility, speed, leanness, learning and responsiveness; and covers the inadequacy of existing definitions.

Our proposed definition is as follows: "Agility is a persistent behaviour or ability of a sensitive entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, uses economical, simple and quality instruments in a dynamic environment and applies updated prior knowledge and experience to learn from the internal and external environment."

Here, we justify and expand upon this definitional statement, which has five facets. In future we will evaluate this definition more extensively as we proceed further in our research.

Flexibility (FY)
Flexibility is the ability or behaviour (flexible) of an entity that allows adapting to changes whenever it is required. A method or phase in a method may demonstrate flexibility by accommodating expected or unexpected changes.

Speed (SD)
Speed of an entity characterizes rapid and quick behaviour to get to the desired destination or to achieve goals. A speedy method may help to show the results quickly by following a specific approach.

Leanness (LS)
Leanness refers to compactness and tidiness. A lean method gives the desired quality output, economically, in the shortest possible time frame by applying simple and quality means of development.

Learning (LG)
Learning refers to knowledge and improvement and is an indispensable ability of an entity, which is achieved primarily by using up-to-date knowledge and experience, gained from previous practices. A learning method shows continuous improvement over the period of time.

Responsiveness (RS)
Responsiveness refers to life, reaction and sensitivity. A responsive method is method that does not remain silent when response is required in different situations

3 ANALYSIS OF PROPOSED DEFINITION OF AGILITY

This section discusses and analyzes the proposed definition of agility in the context of a software development method. We will discuss to see which elements may affect the five attributes of agility and help to classify a software development method as flexible, speedy, lean, learner and responsive. A method will be classified as flexible if we can change, delete or add new practices (software development) in the method during software development; speedy, if it produces workable code in the form of small increments; lean, if it takes minimal possible timeframe and resources to produce such increments; learning, if it improves primarily by using up-to-date knowledge and experience gained from previous practices and feedback mechanisms (feedback loops in an iterative development) in a dynamic environment (where things are not fixed and are handled as we progress towards our targets); responsive, if it responds to the questions (asked) by responding to its internal and external entities. A question may be asked of a method: "When and how to code the design?" The method should have a phase in which different practices may be available to produce the code from design. The following sections present the analysis of the agility in more detail.

3.1 Agility Priority Patterns (APP)

All the five attributes of agility are equally important but we may add weights to show the priority among the agility attributes according to the specific situation (project). There are five attributes; therefore we can use the priority weights value from 1 (minimum priority) to 5 (maximum priority). The 'null' value weight will be used if the specific agile attribute is not present in the agile entity and, as a result, there is no point of assigning weight to that attribute. Two or more agility attributes may have the same priority in some specific pattern. Table 2 presents the different agility priority patterns (APP) as an example. We will discuss APP in detail in a future paper.

Table 2: Agility Priority Patterns.

Attributes	FY	SD	LS	LG	RS
Priority Pattern 1	4	3	3	null	5
Priority Pattern 2	1	3	5	2	4
Priority Pattern 3	1	5	5	4	2

3.2 Application of Agility (Testing the Definition)

Agility demonstrates attributes that may be applied to any object (e.g. Organizations, Systems, Methods, Processes, Software and Documents) to make them agile. Agility, measured in terms of the five variables described above (flexibility, rapidness, leanness, responsiveness and learning) may exist to varying degrees in an object at some specific level or lifecycle phase. For example, a software development method may encompass agility at the design phase level, planning phase level or at the requirements engineering phase level (labeled LA) – but not necessarily all three. We characterize the degree of agility (DA) for each of these phases/levels as the fraction of the five agility variables that are encompassed and supported. If all five variables are supported, then we categorize the level and the object has possessing full agility (FA).

If 1-4 variables are supported, we label it partial agility (PA). We have applied attributes of agility to measure the degree of agility of both XP and Scrum at process level and practices level (see Qumer and Henderson-Sellers, 2006b and results for the degree of agility of both XP and Scrum summarized here in Section 4.3 for reference). In future, we will apply definition of agility to measure the degree of agility of other available agile and non-agile methods.

The following equations may describe the application of agility to different entities.

```
Object (OB) = {Organization, System,
Method, Process, Software,
Documentation, Activities, Techniques,
Metamodel, Method Engineering,
Development……..}

Agility (AG) = {flexibility, rapidness,
leanness, responsiveness, learning}.
```

We may apply agility (AG) to any of the objects (OB) and that object may have some degree of agility at a specific level. For example, we could write the fact that for OB = Method, LA = Requirements Engineering, the value of DA was classed as partial (PA), i.e.

```
DA (Object=Method, LA=Requirements
Engineering) = PA
```

Of especial interest is the degree of agility at the phase and the practice level, i.e.

```
DA (Object, Phase or Practices) =
{PA,FA}
```

4 AGILE SOFTWARE DEVELOPMENT

According to the concepts that have been outlined in different agile methods, agile manifesto values and agile principles (2005), agile software development is characterized by: incremental development, cooperative development, a simple and adaptive development (Stapleton, 1997l Abrahamsson *et al.*, 2002, 2003). We can say in a broad spectrum that agile software development methods mainly develop the software product iteratively and in small releases; where project stakeholders cooperate and collaborate (people focused and communication oriented) to follow simple development steps in an adaptive manner.

This section first explains the concept of software development method and methodology; and then proposes a definition of an agile software development method in the light of tested and applied definition of pure agility (Qumer and Henderson-Sellers, 2006a), existing agile software development methods and agile principles (AgileManifesto, 2003).

4.1 What is a Software Development Method and Methodology?

Software Engineering is the practice of using processes, tools, techniques and guidelines to produce high quality and defect-free software. A method (a.k.a. methodology: Jayaratna, 1994) in software engineering guides the process of developing a software product. Brinkkemper (1996) describes a software development method as being a systematic approach that encompasses directions, rules and specific way of thinking in order to perform development activities with corresponding development products. According to Brinkkemper (1996), a software development methodology for information system is the systematic description and evaluation of all aspects of methodical information systems development. There is another definition that describes the software development method. "The documented collection of policies, processes and procedures used by a development team or organization to practice [sic] software engineering is called its software development methodology" (Chapman, 1997).

4.2 What is an Agile Software Development Method?

An agile software development method may be described by the attributes of agility: flexibility, speed, leanness, responsiveness and learning. We may apply agility to any method to make that method an agile method. According to the above agility definition, any method (entity) that expresses agility is called an agile method. An agile method may be partially or fully agile according to the level and degree of agility encompassed in that method. We may propose a definition of an agile software development method in the light of above study that will help us to differentiate between agile and non-agile software development methods.

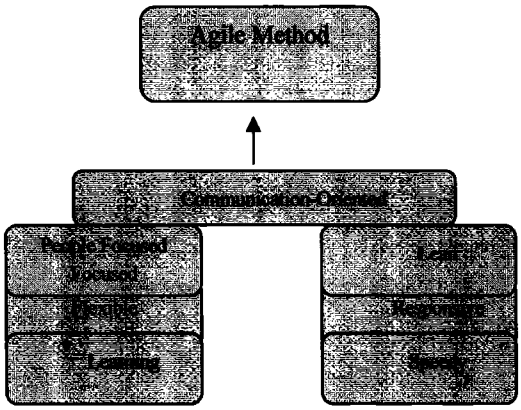

Figure 1: The elements of an agile method.

Our proposed definition for an agile method (Figure 1), paralleling the definition of agility given in Section 2.3, is as follows: "A software development method is said to be an agile software development method when a method is mainly people focused and communication-oriented, flexible (ready to adapt to expected or unexpected change at any time), speedy (encourages rapid and iterative development of the product in small releases), lean (focuses on shortening timeframe and cost and on improved quality), responsive (reacts appropriately to expected and unexpected changes), and learning (focuses on improvement during and after product development)".

4.3 Calculated Degree of Agility in XP and Scrum

As two examples of the application of our definition of agility to software development methods, Table 3 and Figure 2 summarize the results for the degree of agility (DA) for both XP and Scrum (at practices and phases level) (see Appendix for a brief description of XP and Scrum), measured in terms of the five variables (features) in the agility definition: flexibility (FY), speed (SD), leanness (LS), learning (LG) and responsiveness (RS) (Qumer and Henderson-Sellers, 2006b). The degree of agility (DA) and related agility priority pattern (APP) will be used to select a particular agile method for a specific project in-hand. Generally, we may say that the methods with a greater degree of agility are suitable for small and medium size projects; but for large and complex projects, methods with a less degree of agility (more formal) would be better.

Table 3: Degree of agility (Qumer and Henderson-Sellers, 2006b).

Phases & 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

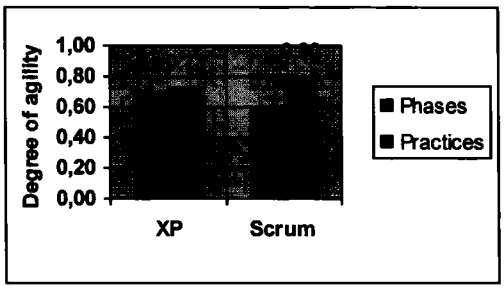

Figure 2: Degree of agility (Qumer and Henderson-Sellers, 2006b).

5 CONCLUSION

We have explained here the concept of agility in detail and, by applying this concept to software development methods, proposed a definition of agile methods. We intend to ratify and, if necessary, extend the definition of agility and agile methods as we proceed further in our research. The definition of agility and agile methods will help us to measure the degree of agility of any method and then to rank software development methods from weakly agile method to strongly agile methods. Such ranking of methods will aid in the differentiation of agile methods from other traditional methods or partially agile methods. In the future, we will also apply these updated concepts of agility and agile methods to develop an agility measurement model.

ACKNOWLEDGEMENTS

We wish to thank the Australian Research Council and Eagle Datamation International for financial support under the Linkage Grants Scheme. This is contribution number 06/04 of the Centre for Object Technology Applications and Research.

REFERENCES

Abrahamsson, P., Wasta, J., Siponen, M.T. & Ronkainen, J., 2003. New Direction on agile Methods: a Comparative Analysis. *25th International Conference on Software Engineering.* IEEE Computer Society, Portland Oregon.

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J., 2002. Agile software development methods. Review and analysis Report. VTT Publications 478.

AgileManifesto., 2005. Manifesto for Agile Software Development.

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

Boehm, B. & Turner, R., 2004a. *Balancing Agility and Discipline: A Guide for the Perplexed,* Pearson Education, Inc. Boston.

Boehm, B. & Turner, R., 2004b. 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.

Brinkkemper, S., 1996. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology, 38.*

Chapman, J.R., 1997. Software Development Methodology. http://www.hyperthot.com/pm_sdm.htm.

Cockburn, A., 2002. *Agile Software Development,* Addison-Wesley. Boston.

Conboy, K. & Fitzgerald, B., 2004. Toward a conceptual framework of agile methods: a study of agility in different disciplines. *Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research.* ACM Press, New York USA.

Dove, R., 1997. The Meaning of Life and the Meaning of Agility. Paradigm Shift International www.parshift.com/library.htm.

DSDM., 2003. DSDM Consortium. Dynamic Systems Development Method Ltd., http://www.dsdm.org..

Henderson-Sellers, B. and Serour, M.K., 2005. Creating a dual agility method - the value of method engineering. *J. Database Management,* 16.

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

Jayaratna, N., 1994. *Understanding and Evaluating Methodologies, NIMSAD: A Systemic Approach,* McGraw-Hill.

Nandhakumar, J. & Avison, D.E., 1999. The fiction of methodological development: a field study of information systems development. *Information Technology & People, 12.* IEEE Computer Society, Washington DC USA.

Paetsch, F., Eberlein, A. & Maurer, F., 2003. Requirements Engineering and Agile Software Development. *Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises.* Linz, Austria.

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

Qumer, A. and Henderson-Sellers, B., 2006a. Measuring agility and adoptability of agile methods: A 4-Dimensional Analytical Tool. *Procs. IADIS International Conference Applied Computing 2006* (eds. N. Guimarães, P. Isaias and A. Goikoetxea), IADIS Press, 503-507

Qumer, A. & Henderson-Sellers, B., 2006b. Comparative evaluation of XP and Scrum using the 4D Analytical Tool (4-DAT), *Proceedings of the European and Mediterranean Conference on Information Systems 2006 (EMCIS2006)* (eds. Z. Irani, O.D. Sarikas, J. Llopis, R. Gonzalez and J. Gasco), CD, Brunel University, West London

Royce, W.W., 1970. Managing the development of large software systems. *Procs. IEEE WESCON.*

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

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

Wolak, R.G., 2001. System Development: Research Paper 1, SDLC on a Diet. http://www.itstudyguide.com/papers/rwDISS725researchpaper1.htm.

Wong, S.-P. & Whitman, L., 1999. Attaining Agility At The Enterprise Level. *Proceedings of the 4th Annual International Conference on Industrial Engineering Theory, Applications and Practice.* San Antonio.

APPENDIX – OVERVIEW OF XP AND SCRUM

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; Wolak, 2001). XP stresses communication and coordination among the team members all the time; and requires cooperation between the customer, management and development team to form the supportive business culture for the successful implementation of XP.

Scrum

Schwaber and Beedle (Schwaber & Beedle, 2002) report that Scrum is a flexible, adaptable, empirical, productive and iterative method that uses the ideas of industrial process control theory for the development of software systems. According to Schwaber and Beedle (2002), Scrum has threes phases: Pre-Game, Development and Post-Game. The Pre-Game phase has a further two sub-phases: planning and high level design (architecture).

ICSOFT 2006

Proceedings of the
First International Conference on
Software and Data Technologies

Volume 2

Setúbal, Portugal

September 11 – 14, 2006

Organized by
INSTICC – Institute for Systems and Technologies of Information, Control and Communication

Sponsored by
Enterprise Ireland
Polytechnic Institute of Setúbal

In Cooperation with
Object Management Group (OMG)

Hosted by
School of Business of the Polytechnic Institute of Setubal

CONTENTS

INVITED SPEAKERS

KEYNOTE LECTURES