Proceedings of the 25th IASTED International Multi-Conference
SOFTWARE ENGINEERING
February 13–15, 2007, Innsbruck, Austria
ISBN Hardcopy: 978-0-88986-641-6 / CD: 978-0-88986-643-0

IMPROVING AGILE SOFTWARE DEVELOPMENT BY THE APPLICATION OF METHOD ENGINEERING PRACTICES

B. Henderson-Sellers, M.K. Serour, C. Gonzalez-Perez and A. Qumer
University of Technology, Sydney
P.O. Box 123
Broadway
NSW 2007
Australia
{brian, mserour}@it.uts.edu.au. cesargon@verdewek.com, asif@it.uts.edu.au

ABSTRACT
Despite the vast attention and wide acceptance of the newly engineered agile methods for software development, those methods are seldom linked to the goals of software process improvement (SPI), an approach that aims to provide support for significant improvement of both the quality of those methods as well as the resultant software products. In this paper, we propose an extension to agile methods by adding extra characteristics in order for agile methods to better support SPI. We explain how agile methods can gain those extra attributes through the application of a method engineering approach along with our new tool (4-DAT) that assists method engineers and managers in selecting the most appropriate method fragments for their needed agile methods. Finally, we summarize a number of industrial case studies carried out over several years in order to test and improve the efficiency of our theory of adding SPI to an agile methodological approach.

KEY WORDS
Software methodologies, Agility, SPI, Method engineering

1. Introduction

While there are very many influences on the success or failure of a software development project, two factors of high importance are the people involved and the methodological approach they use [1,2]. Since people (and even organizations: [3]) learn, a static methodological approach, as advocated by almost all methodologists, can never support the goals of software process improvement (SPI) [4].
As a reaction to so-called "heavyweight" or plan-based methodologies [5], many practitioners have adopted the recently introduced ideas of agility [6]. Use of an agile method can indeed create a high quality environment and high quality product. Its focus on the people rather than on reporting deliverables is often seen as a welcome shift of balance towards the most important factor in software development: the personnel involved. However, their agility is seen in the way that the people practising this approach are able to react to changing situations, particularly changing user requirements [7,8]. Otherwise, they are fairly didactic in their rigidity. This means that they only work if the knowledge of the people involved is static i.e. the users and developers learn how to apply the agile method as documented (e.g. in a book) but do *not* learn further throughout the development process. This is clearly the antithesis of a learning approach.
A second degree of agility was therefore introduced in [4]. It is this second kind of agility that permits the process itself to evolve and hence to offer the capability and potential to support the ideals of SPI. In this paper, we explain how dual agility can be attained more effectively through the use of method engineering, and we involve a new tool that assists a manager in performing method engineering within an agile environment. This tool, called 4-DAT [9], provides a framework in which an assessment of the four degrees of agility can be made for an existing methodology and, more importantly, for SPI. This is accomplished by assisting engineers in choosing method fragments most appropriate for an agile SPI development environment.

2. Agility and its Evaluation

Concerns about the viability of plan-based and potentially mechanistic software developments around the turn of the century (e.g. [10]) are often identified as the precursor to the rise of so-called agile methods (e.g. [11]). Although Cockburn [12] 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" and the Agile Manifesto [6] provides agile principles and agile values that qualitatively characterize the agile methods, there remains no widely-agreed, precise and comprehensive definition of agility. Based on a survey and assessment of the various contemporary definitions, Qumer and Henderson-Sellers [9,13] offer the following definition:

"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."

In order to benefit from this definition, these authors developed a four dimensional framework (4-DAT) to crystallize the key attributes of agility: flexibility, speed, leanness, learning and responsiveness. Flexibility is the ability to respond to the expected change and leanness accentuates lower cost, reduced timeframe and quality production [14]. A speedy method may help to show the results quickly by following a specific approach; whereas responsiveness refers to life, reaction and sensitivity. Finally, learning refers to knowledge and improvement and is an indispensable ability of an entity, 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.

Consequently, by applying the above definition of agility to the notion of a software development methodology, we derive the definition of an "agile method" as:

"A software development method is said to be agile if it is people focused, 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 without compromising on quality), responsive (reacts appropriately to expected and unexpected changes), and learning (focuses on improvement during and after product development)". (modified from [13]).

We can now apply this definition, in the form of the four dimensional framework of 4-DAT (delineated in Table 1) to an evaluation of existing (and future) software development methods. Such an evaluation can determine not only whether a method's elements can be considered agile or not (binary) but can determine the *degree* of agility exhibited by the method. The evaluation can be applied at various levels of granularity. Since most agile methods favour a discrimination between a high level process or "phase" level and a lower level "best practices" level, we select these two granularity levels as those to be assessed separately in this study also.

Dimension 2 is the only one of the dimensions that can be assessed quantitatively. Details of the algorithms proposed are found in [9] and their application to two exemplar agile methods (XP and Scrum) in [15]. They found that, while XP was evaluated as being more agile at the phase level, Scrum showed more agility at the practices level (Figure 1).

Table 1: 4-DAT dimensions (derived from [9])

DIMENSION 1	
Scope	**Description**
1. Project Size	Does the method specify support for small, medium or large projects (business or other)?
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 Environment	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)?
DIMENSION 2	
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?
DIMENSION 3	
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 helps in keeping the process agile?
6. Keeping the process cost effective	Which practices helps in keeping the process cost effective?
DIMENSION 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?

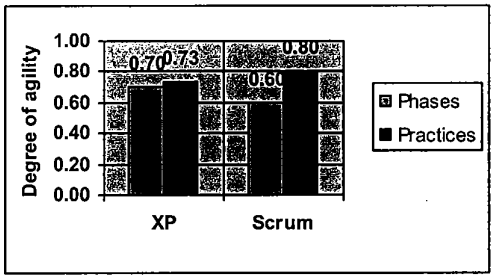

Figure 1 Comparison of the degree of agility for XP and Scrum as measured for the phases level and the practices level (after [15])

3. Increasing Agility through SPI

As noted earlier, the foci of many agile methods are for them to support flexibility at the level of a single enactment i.e. within a single project, where they support changes well, particularly those engendered by the client. However, they only support *method* flexibility (as opposed to the flexibility of method enactment for which they were solely designed) very informally and in an ad hoc manner (e.g. [16]). Addition of a second kind of agility, method agility, was proposed by [4]. The resulting method is said to exhibit "dual agility" and can now support SPI, particularly if linked with a method engineering mindset.

3.1 Dual Agility

The first agility/flexibility dimension makes the method more able to adapt to changes not only for requirements but also for design, technology and people.
The second agility dimension makes the method more flexible so that it can be changed or reengineered if and when a need arises, in response to organizational evolution and maturation. Although this can be accomplished informally, a more repeatable approach is seen to be beneficial. This is more easily facilitated when using a rich repository of method fragments (see Section 3.2) such as those found in the OPEN Process Framework (OPF) [17] or in recent versions of the Rational Unified Process (RUP) [18].

3.2 Situational Method Engineering and the use of 4-DAT

In a situational method engineering (SME) approach [19-22], small pieces of a method are identified and stored as method fragments or method chunks [23] in a repository or methodbase [24,25]. For each project, the method engineer selects appropriate method fragments from the methodbase, perhaps with the help of an outside consultant or a software tool such as a Computer Assisted Method Engineering (CAME) tool [25,26]. The method is thus "constructed" or engineered from its component parts in such a way that only relevant components, as represented by the method fragments, are incorporated into the constructed method and those not useful can be safely ignored (e.g. [21]). Construction rules are required for this assembly process [27,28]. This is arguably one of the most challenging parts of SME. In the construction of an agile method (as discussed in this paper), it is important to be able to identify those fragments that may have appropriate agility (and the extent of that agility characteristic). One potential problem in the context of agile methodologies is that the agile culture perceives a method as being emergent rather than constructed. Notwithstanding, it is interesting to evaluate whether the addition of SME practices to an agile approach is both valuable and can be accepted by that community's culture.
In traditional SME, it is left to the judgement of the method engineer to ascertain whether each method fragment selected from the methodbase is appropriate or not. For the construction of an agile methodology, we can use the 4-DAT to add an "agility value" to any such fragments. This value will assist method engineers in their decision-making process.
4-DAT's dimension 2 (Table 1) is the only quantitative measure and can be applied at the individual practice level e.g. one specific technique. For this purpose a table can be constructed (Table 2) in which cell values of 0 or 1 are entered for each phase (for a high level assessment) or for each individual practice (technique in the OPF example) [15]. The five agility features of flexibility (FY), speed (SD), leanness (LS), learning (LG) and responsiveness (RS) for each fragment are considered and then the overall method total (and hence average degree of agility) can be calculated. Based on our quantitative evaluations of pre-existing (and known) agile methods such as XP and Scrum, we can offer as a ballpark figure a threshold value of around 0.5-0.6 for any constructed agile method to

Table 2 Schematic table for the calculation of the agility of x individual phases and y practices (OPF techniques) and the overall agility of the constructed method

	Agility Features					
	FY	SD	LS	LG	RS	Total
(i) Phases						
Phase 1	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0-5
Phase 2	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0-5
Phase 3	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0-5
etc.	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0-5
Total	(0-x)	(0-x)	(0-x)	(0-x)	(0-x)	0-5*x
Degree of Agility (high level)	(0-x)/ x	(0-x)/ x	(0-x)/ x	(0-x)/ x	(0-x)/ x	Total divided by number of cells in table
(ii) Practices						
Practice 1	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0-5
Practice 2	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0-5
etc.	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0-5
Total	(0-y)	(0-y)	(0-y)	(0-y)	(0-y)	0-5*y
Degree of Agility (low level)	(0-y)/ y	(0-y)/ y	(0-y)/ y	(0-y)/ y	(0-y)/ y	Total divided by number of cells in table

have sufficient measured agility to qualify for consideration as an agile method.

The other three dimensions of 4-DAT, as shown in Table 1, are qualitative and can be used in an SME context once the proposed method has been constructed. These three dimensions relate to an evaluation of various higher level aspects of the methodology. If they are found to be missing or poorly represented, then the 4-DAT has been used as an indicator of poor quality. Hence, before release to the development team, the method engineer has the chance of improving the proposed method.

In summary, the use of situational method engineering has three direct advantages to the software development organization and to the software development team:

i) The method that is constructed for the current project is *ultimately* what is required both in terms of processes activities, tasks, techniques, guidelines (depending on your choice of terminology) as well as in terms of the people involved (actually roles), the extent of bureaucratic reporting and project management and the lifecycle model itself.

ii) For subsequent projects, the team will develop confidence in self-tuning (tailoring) their own method to fit these projects and also in responding to any change in the development environment - perhaps the scope of the projects they undertake will change; as they grow in sophistication and software development capability so must the process (see also [29]). Consequently, elements in the method repository previously eschewed may now prove useful. Perhaps there is a technique for finding classes and agents that the project team felt they didn't have the skills to use previously; perhaps there is a training role for team expansion.

iii) Building trust in the team members' ability to make use of their method. The development team will be highly motivated and enthusiastic to fully utilize their method as a result of gaining method ownership.

4. Brief Summary of Case Study Results

The application of dual agility in order to support method maturation and hence software process improvement has been studied using Action Research [30] by [31-33] in a number of projects in two major organizations in Sydney, Australia. To illustrate the efficacy of this approach, we briefly summarize some of these empirical studies here using an SME approach. The first project [31] took more than two years and successfully assisted an IT department within a legal publisher company in Sydney to construct an agile method for their new web development projects; whereas the second project [32] was conducted for more than a year in a governmental department within the New South Wales government during their transformation process to e-government.

The main objective of these empirical evaluations was therefore to evaluate the introduction of an SME approach using the OPF in order to test our construction of a dual-agile methodology for these two organizations, a construction done in such a way that the methodology can be instantiated and then fully customized to suit individual projects.

In one of these experiments, a minimal subset of method fragments was chosen purposefully from the OPF methodbase as a precursor to future SPI by the maturation of the approach, typically by the addition of more method

fragments traditionally associated with higher SPICE/CMM levels. The web development team within the study organization was facing an increased pressure to develop a set of new web-based applications under evolving requirements and changing technologies within a very tight timeframe. The method engineering team realized the urgent need for engineering a process that was more adaptive than predictive and more people-oriented than process-oriented - the newly constructed "agile" process was thus in accordance with the agile manifesto, as exemplified by the four key values of the "Manifesto for Agile Software Development". This method was only successful for the very first project [34]. For the second and subsequent projects, some of the existing method fragments were identified as being unnecessary and other method fragments were seen to be missing. As a result, the company's previously engineered method was modified by the addition of the second agility feature, by supplying an ability for the developer to self-tune or reengineer their method to accommodate different projects. For nearly eight months, both the researchers and practitioners teams worked in close collaboration with everyone involved, including customers, to produce a flexible method to satisfy the process needs of organization. The resulting methodology was well accepted and is still cherished by both management and technical people.

5. Conclusion and Future Work

Extant agile methods can be seen as rigid in their application over a number of projects. While supporting agility within the project they are poor at supporting agility across several projects, agility that leads to support for SPI. Consequently, a second degree of agility was proposed [4]. In this paper, we extend these ideas to link in with, firstly, the concepts of method engineering and, secondly, with a new assessment framework, the 4-DAT tool [9]. To exemplify the synergy of these components, we relate the theory to our empirical evidence from several Sydney-based IT groups who have adopted an agile approach that also takes into account SPI as an objective and context.

The next stage of this work is to evaluate companies's use of agile methods and SPI outside of our local environment in Australia and to commence some inter-regional evaluations. At the same time we anticipate some possible feedback to improve the 4-DAT evaluative framework discussed here.

Acknowledgement

We acknowledge financial support from the Australian Research Council for this project. This is contribution number 06/06 of the Centre for Object Technology Applications and Research.

References

[1] D. Phillips, *The software project manager's handbook: principles that work at work* (Hoboken, NJ, USA: John Wiley & Sons, 2004).

[2] H. Kerzner, *Project management: a systems approach to planning, scheduling, and controlling* (9th ed.) (Chichester, Sussex, UK: Wiley, 2005).

[3] P. Senge, *The fifth discipline: the art and practice of the learning organization* (New York, NY, USA: DoubleDay, 1990)

[4] B. Henderson-Sellers & M.K. Serour, Creating a dual agility method - the value of method engineering, *J. Database Management, 16*(4), 2005, 1-24.

[5] T. Chau, F. Maurer & G. Melnik, Knowledge sharing: agile methods vs. Tayloristic methods, *Procs. 12th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03)*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2003, 302-307

[6] AgileManifesto, *Manifesto for agile software development.* http://www.agilemanifesto.org/. Accessed 14 March 2005.

[7] J. Newkirk, Introduction to agile processes and extreme programming, *Procs. ICSE '02*, ACM Press, 2002, 695-696.

[8] J. Bettin, Practicalities of implementing component-based development and model-driven architecture, *Process Engineering for Object-Oriented and Component-Based Development. Procs. OOPSLA 2003 Workshop*, Centre for Object Technology Applications and Research, Sydney, Australia, 2003, 19-30.

[9] A. Qumer & B. Henderson-Sellers, 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, 2006, 503-507.

[10] J. Nandhakumar & D.E. Avison, The fiction of methodological development: a field study of information systems development. *Information Technology & People, 12,* IEEE Computer Society, Washington DC USA, 1999, 176-191.

[11] K. Beck, 2000, *Extreme Programming Explained* (Boston, MA, USA: Addison-Wesley, 2000).

[12] A. Cockburn, *Agile Software Development* (Boston, MA, USA: Addison-Wesley, 2002).

[13] A. Qumer & B. Henderson-Sellers, 2006, Crystallization of agility: back to basics, *Procs. ICSOFT*, Setubal, Portugal, 2006.

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

[15] A. Qumer & B. Henderson-Sellers, 2006, 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, UK.

[16] S. Henninger, A. Ivaturi, K. Nuli & A. Thirunavukkaras, Supporting adaptable methodologies to meet evolving project needs, *Procs. 1st ICSE Workshop on Iterative, Adaptive, and Agile Processes,* Orlando, Florida, USA, May 25, 2002.

[17] D.G. Firesmith & B. Henderson-Sellers, *The OPEN process framework. an introduction* (London, UK: Addison-Wesley, 2002).

[18] B. MacIsaac, An overview of the RUP as a process engineering platform, *Process Engineering for Object-Oriented and Component-Based Development. Procs. OOPSLA 2003 Workshop*, Centre for Object Technology Applications and Research, Sydney, Australia, 2003, 43-52.

[19] K. Kumar & R.J. Welke, Methodology engineering: a proposal for situation-specific methodology construction, in *Challenges and Strategies for Research in Systems Development* (eds. W.W. Cotterman and J.A. Senn) (Chichester, Sussex, UK: J. Wiley & Sons, 1992), 257-269.

[20] S. Brinkkemper, Method engineering: engineering of information systems development methods and tools, *Inf. Software Technol., 38*(4), 1996, 275-280.

[21] S. Brinkkemper, M. Saeki & F. Harmsen, Meta-modelling based assembly techniques for situational method engineering, *Information Systems*, *24*(3), 1999, 209-228

[22] B. Henderson-Sellers, Method engineering for OO system development, *Comm. ACM, 46*(10), 2003, 73-78

[23] A.H.M. Ter Hofstede & T.F. Verhoef, On the feasibility of situational method engineering, *Information Systems, 22,* 1997, 401-422.

[24] J. Ralyté, Reusing scenario based approaches in requirement engineering methods: CREWS method base, *Proceedings of the 10th International Workshop on Database and Expert Systems Applications (DEXA'99), 1st International Workshop on the Requirements Engineering Process – Innovative Techniques, Models, Tools to support the RE Process (REP'99),* Florence, Italy, 1-3 September 1999, IEEE Computer Society, Los Alamitos, CA, USA, 1999, 305-309

[25] M. Saeki, CAME: the first step to automated software engineering, *Process Engineering for Object-Oriented and Component-Based Development. Procs. OOPSLA 2003 Workshop*, Centre for Object Technology Applications and Research, Sydney, Australia, 2003, 7-18

[26] J.-P. Tolvanen, M. Rossi & H. Liu, Method engineering: current research directions and implications for future research, in *Method Engineering, Principles of Method Construction and Support* (eds. S. Brinkkemper, K. Lyytinen and R. Welke), Chapman-Hall, London, UK, 1996, 296-317

[27] J.-P. Tolvanen, *Incremental method engineering with modeling tools: theoretical principles and empirical evidence*, Ph.D. Thesis, University of Jyväskylä, Finland, 1998.

[28] J. Ralyté & C. Rolland, An assembly process model for method engineering, *Procs. CAiSE 2001*, LNCS 2068, Springer-Verlag, Berlin, 2001, 267-283.

[29] M. Bajec, M. Krisper & R. Rupnik, The scenario for constructing flexible, people-focused systems development methodologies, *Procs. ECIS 2004*, 2004.

[30] D.E. Avison, F. Lau, M. Myers & P.A. Nielsen, Making academic research more relevant, *Communications of the ACM, 42*(1), 1999, 94-97.

[31] M.K. Serour & B. Henderson-Sellers, Introducing agility: a case study of situational method engineering using the OPEN Process Framework, *Procs. 28th Annual International Computer Software and Applications Conference. COMPSAC 2004*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2004, 50-57.

[32] M.K. Serour & B. Henderson-Sellers, Empowering a software development team with a new methodology: a case study of e-government in Australia, *Information Technology and Organizations in the 21st Century: Challenges & Solutions* (ed. K.S. Soliman), International Business Information Management Association, 2004, 214-223

[33] M.K. Serour & B. Henderson-Sellers, OPEN for agility: an action research study of introducing method engineering into a government sector, *Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education* (eds. O. Vasilecas, A. Caplinskas, W. Wojtkowski, W.G. Wojtkowski, J. Zupancic and S. Wrycza), Vilnius Gediminas Technical University, Vilnius, Lithuania, 2004, 105-116.

[34] M. Serour, B. Henderson-Sellers, J. Hughes, D. Winder & L. Chow, Organizational transition to object technology: theory and practice, *Object-Oriented Information Systems* (eds. Z. Bellahsène, D. Patel and C. Rolland), LNCS 2425, Springer-Verlag, Berlin, 2002, 229-241.