Agile Service Development: A Rule-Based Method Engineering Approach

Stijn Hoppenbrouwers[1], Martijn Zoet[2], Johan Versendaal[2,3], and Inge van de Weerd[3]

[1] Radboud University Nijmegen, the Netherlands, {s.hoppenbrouwers@cs.ru.nl}
[2] University of Applied Sciences, Utrecht, the Netherlands, {martijn.zoet, johan.versendaal}@hu.nl
[3] Utrecht University, Utrecht, the Netherlands, {j.versendaal, i.vandeweerd, }@cs.uu.nl

Abstract. Agile software development has evolved into an increasingly mature software development approach and has been applied successfully in many software vendors' development departments. In this position paper, we address the broader agile *service* development. Based on method engineering principles we define a framework that conceptualizes an operational way of working for the development of services, emphatically taking into account agility. As a first level of agility, the framework contains situational project factors that influence the choice of method fragments; secondly, increased agility is proposed by describing and operationalizing these method fragments not as imperative steps or activities, but instead by means of sets of minimally specified, declarative rules that determine the context and constraints within which goals are to be reached. This approach borrows concepts from rules management, organizational patterns, and game design theory.

Keywords: method engineering, agile service development, business rules, business rules management, product management, game design.

1 Introduction

To remain competitive, organizations are increasingly urged to adapt to changes in their business environment. Trends like higher demanding customers, faster changing customers' demands, increased regulation, and offshoring give rise to the re-thinking of business models and processes. In the software development industry, a number of vendors have successfully applied 'agile software development process' principles, decreasing time-to-market and addressing rapidly changing customer demands.

* *This paper results from the Agile Service Development project (*http://www.novay.nl/okb/projects/agile-service-development/7628*), a collaborative research initiative focused on methods, techniques and tools for the agile development of business services. The project consortium consists of BeInformed, BiZZdesign, CRP Henri Tudor, HU University of Applied Sciences Utrecht, IBM, Novay, O&i, PGGM, RuleManagement Group, Radboud University Nijmegen, Twente University, Utrecht University, and Voogd & Voogd. The project is part of the program Service Innovation & ICT of the Dutch Ministry of Economic Affairs.*

Approaching this more generically, any business may likewise apply concepts of agility as a strategy to take up the described challenges in the business environment. Agility is defined as "the ability of a sensitive [organization] that exhibits flexibility to accommodate expected or unexpected changes rapidly, following the shortest time span, using economical, simple and quality instruments in a dynamic environment and applying updated prior knowledge and experience to learn from the internal and external environment" [1]. The aforementioned definition positioned in the context of agile service development asserts that an organization should be able to create or adapt a (business) service efficiently and effectively when changes occur in its environment. A business service is considered an externally visible and accessible unit of functionality offered by an organization to its environment, delivering a meaningful value to that environment. An example of such a service is 'an insurance product tailored towards singles'.

Agile development is not an alien concept in management and information systems research. It plays some role in existing work on *situational method engineering* in software product development literature [2, 3, 4, 5]. These studies acknowledge the need for development methods tuned to the situation of the project at hand. Based on situational factors distilled from the project, meta-methods composed of outlines or more detailed procedures, are selected and integrated into a coherent method appropriate for that specific situation [4].

However, 'situational' is not synonymous to 'agile'. For a method to become truly agile, changing situational factors also have to be linked (if required) to 'run time', changes in the method: quick responses to new situational information, and the installation of short feedback loops applying to the method. Existing studies mainly focus on situational fit of the overall development process while still describing the actual method fragments in terms of 'non-agile', step-by-step, instructions inherent to traditional workflow-like process descriptions.

2 Method Engineering for Agile Service Development

Situationality is the ability of a method to respond and adapt to a specific environment based on defined characteristics [4, 6]. Although scholars approach the concept from different viewpoints, the fundamental basis is the creation of reusable method parts called method fragments [7] or method chunks [8]. The method fragments are stored in a repository called the method base. In addition to the method fragments, also assembly rules and situational factors are stored inside the method base [5].

Utilizing the perspective of situationality, method fragments can be used to provide some degree of agility with respect to the project at hand. Regarding the assembly of method fragments, our approach follows the configuration process for situational method engineering as proposed by Brinkkemper [4]. However, our approach adds a second dimension of agility in operational execution.

Due to changes, predictable or unpredictable, in the environment, the method must be able to quickly adjust to the situation at hand. The method engineering process proposed by Brinkkemper [4] incorporates this by means of a build-in feedback loop. This feedback loop facilitates selecting new process alternatives in terms of method fragments hereby inserting the underlying assumption that changes in the environment will result in replacing complete method fragments. We argue that changes in the environment will not always lead to changes in the executed method but can still influence the operational execution of a specific method fragment.

To realize this, we propose a particular operationalization of the method engineering approach and process in terms of the selection process of method fragments, situational factors and assembly rules. The idea is that participants are given as much freedom as possible within necessary methodical and contextual constraints (minimal specification), and that the ability to respond quickly to desired changes in the method (as indicated by fast feedback) is optimized: increased agility in our approach is supported by defining method fragments in a rule-based, declarative manner. This approach is inspired by principles and practices from (business) rules management, organizational patterns and game design theory. Rule-based specification of methods is vaguely suggested in [9], who argue in favor of using practices instead of processes in software engineering. Our approach is inspired by this line of thinking, but pushes for advanced description, management and operationalization of ‘method rules’ in a specific service development context.

In the following subsections the method engineering meta-model by Brinkkemper [4] will be described in some more detail for 1) situational project factors and characteristics, and 2) method fragment description and identification (see figure 1).

2.1 Situational Project Factors and Characterization

Situational factors can be used to characterize projects, processes, and companies. Bekkers [10] researched the influence of situational factors on the practice of software product management, which resulted in a list of 27 situational factors, divided over the categories (1) business unit characteristics, (2) customer characteristics, (3) market characteristics, (4) product characteristics, and (5) stakeholder involvement. A situational factor influence is, for example: “the amount of requirements that are submitted by the customers has a high impact on how requirements management processes should be carried out”. If this situational factor were to change, the company should also change its processes in order to cope with this change. We intend to apply the 27 situational factors in the context of agile service development.

2.2 Method Fragments Description and Identification

We choose a rule-based, declarative approach to the description of method fragments. Declarative description allows for minimal specification. In an agile environment, ‘just enough’ explicit regulation of the way of working is to be preferred over imperative style, step-by-step instruction inherent to traditional flow-like process description. The declarative approach is mainly what has led us to introduce the ‘game metaphor’ as an image of how we intend to deal with describing agile methods and method fragments.

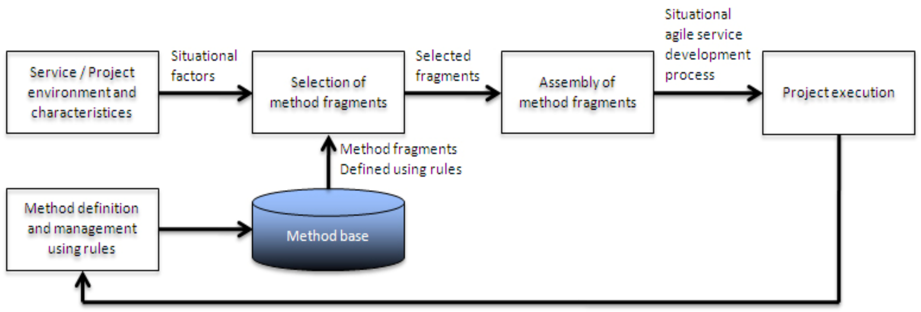
Service / Project environment and characteristices
Situational factors
Selection of method fragments
Selected fragments
Assembly of method fragments
Situational agile service development process
Project execution
Method fragments Defined using rules
Method definition and management using rules
Method base

Fig. 1. Method engineering approach for agile service development.

As suggested in Hoppenbrouwers [11, 12], methods can be fruitfully viewed as games. They have clear objectives and rules the participants are to comply to, or at least choose to be guided by. The driving concepts in this approach are *goals*. These can cover all aspects of what one wants to achieve (deliverable or product goals, like 'create an insurance product for singles, within 2 months') and how one wants to do this (process goals, like 'use SCRUM'; 'comply to HIPAA regulation'; 'actively involve representatives of prospective customers'). Many kinds of goals can be distinguished, and all of them can be represented in the form of rules. Goals are thus covered by *goal rules*.

To realize goals, activities are needed. If goals are logically ordered, so are the activities linked to them (like 'hold SCRUM standup meeting'), which can be planned in space and time, allocating specific people and resources. Activities can be temporally ordered, but do not need to be in principle. This is in line with the principles of declarative workflow [13] and allows for minimal specification: formally planning only what needs to be planned, and leaving the rest to the team's powers of self-organization.

Not only goals can be expressed as rules, but also the temporal ordering (*procedural rules*: *x* before *y*) and even constraints on interaction: *interaction rules* that concern who talks to who ('tester *t* with stakeholder *s*') and by what means ('using think-aloud session using prototype PT2.1'). This links high-level method engineering to more operational method engineering involving communication situations [14]. Additional rules can cover aspects like the format or language (i.e. meta-model notation: 'UML Use Cases, Class Diagrams, Activity Diagrams') of any deliverables strived for. At the operational level of communication situations, the rules have to be specific and readable enough to effectively guide people in their activities –in as far as such guidance is required (minimal specification).

There is a clear parallel between a declarative, rule-based approach, the game metaphor, and the use of patterns; in particular, organizational patterns [15]. Cockburn has advocated game-theoretical use of the game metaphor in studying the software engineering process [16], but not in the applied sense we now propose. Our rules for describing method fragments will cover principles and patterns of agile practice (including many existing ones), and operational reflections thereof.

3 Conclusions

In view of increasing demands for agility in processes for service development, we are in the early stages of applying existing principles and practices from situational method engineering to service development processes and methods, combining these with approaches supporting agile process management and execution. On the method engineering side, this requires some innovation concerning the description, management, and operationalization of methods. Without claiming that the approach put forward in this position paper will guarantee agility of processes for service development, we believe the approach proposed will allow for considerably better agility than existing practices in ME that are more rooted in imperative style specification of methods and method fragments. Our rule-based approach should enable quick adaptation of the method's 'rules of the game' to changing situational factors. 'Games played' will be short cycles or phases in development, in line with widespread agile practices in software engineering. In addition, we pay explicit attention to operationalization of methods by specifying actual 'games to be played' in terms of concrete 'communication situations', and linking these to higher level goals and activities as included in some method and drawn from the method base.

We will test and refine our approach to method engineering in agile service development in close cooperation with a number of partners from industry. We will explore our approach in the re-engineering of past project cases, but will also, even in the early stages of investigation, start applying our framework in real cases of running projects.

Our approach can be seen as complementary to another innovative direction in Method Engineering: that of 'Method as a Serivce' (MaaS) [17]. Method fragments are developed as *method services* which are implemented as web services. To make the method services widely available, a Method-Oriented Architecture (MOA) is proposed. With the concept of MaaS, the authors aim to overcome many drawbacks that exist with existing method fragments, such as lack of interoperability, and lack of interactivity.

4 References

1. Qumer, A., Henderson-Sellers, B., (2007). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and Software Technology*, 50(4), pp 280-295.
2. Olle, T.W., Hagelstein, J., MacDonald, I.G., Rolland, C., Sol, H.G., van Assche, F.J.M. and Verrijn-Stuart, A.A. (1991). *Information Systems Methodologies: a Framework for Understanding (2nd edition).* Addison-Wesley.
3. Kumar, K and Welke, R.J. (1992). Methodology engineering: a proposal for situation-specific methodology construction. In Cotterman, W.W., Senn J.A. (eds), *Challenges and Strategies for Research in Systems Development.*
4. Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4), pp 275-280.

5. Weerd, I. van de, Versendaal, J., & Brinkkemper, S. (2006). A product software knowledge infrastructure for situational capability maturation: Vision and case studies in product management. *Proceedings of the 12th Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'06)*, Luxembourg, pp 97-112.
6. Ralyté, J., Deneckère, R., & Rolland, C. (2003). Towards a generic model for situational method engineering. *Lecture Notes in Computer Science*, 2681, 95.
7. Harmsen, F., Brinkkemper, S and Oei, H. (1994). Situational method engineering for information system project approaches. In Verrijn Stuart, A.A. and Olle T.W. (eds), *Methods and associated tools for the information systems life cycle*. Proceedings of the IFZP WG8.1 Working Conference CRIS'94, Maastricht, September 1994, North-Holland, Amsterdam, pp 169-194.
8. Rolland C., V. Plihon, J. Ralyté, (1998). Specifying the reuse context of scenario method chunks. *Proc. of the 10th Conf. on Advanced Information Systems Engineering*, Pisa Italy.
9. Jacobson, I, Ng, P.W. and Spence, I.: Enough of Processes - Lets do Practices. *Journal of Object Technology*, 6(6), 2007, pp. 41-66. http://www.jot.fm.
10. Bekkers, W., Weerd, I. van de, Brinkkemper, S., & Mahieu, A. (2008). The influence of situational factors in software product management: an empirical study. *Proceedings of the 2nd International Workshop on Software Product Management*, Barcelona, Spain, pp 41-48.
11. Hoppenbrouwers, S.J.B.A., H. Weigand, and E.A.J.A. Rouwette (2009). Setting Rules of Play for Collaborative Modelling. In: N. Kock and P. Rittgen (eds). *International Journal of e-Collaboration (IJeC)*, 5(4), 2009, pp 37-52..
12. Hoppenbrouwers, S.J.B.A., P. van Bommel and A. Järvinen (2008). Method Engineering as Game Design: an Emerging HCI Perspective on Methods and CASE Tools. In: *Proceedings of EMMSAD'08 (Exploring Modelling Methods for System Analysis and Design),* held in conjunction with CAiSE'08. Montpellier, France, June 2008.
13. van der Aalst, W., M. Pesic, and H. Schonenberg (2009). Declarative workflows: Balancing between flexibility and support. *Computer Science-Research and Development* 23(2), pp 99–113.
14. Hoppenbrouwers, S.J.B.A and Wilmont, I. (2010). Focused Conceptualisation: Framing Questioning and Answering in Model-Oriented Dialogue Games. In: Bommel, P. van, Hoppenbrouwers, S.J.B.A., Overbeek, S., Proper, H.A., and Barjis, J. (eds). *The Practice of Enterprise Modeling*. Proceedings of the Third IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling (PoEM 2010).
15. Coplien, James and Neil Harrison (2004). *Patterns of Agile Software Development*. Addison-Wesley.
16. Cockburn, A. (2004). The End of Software Engineering and the Start of Economic-Cooperative Gaming, Retrieved October 23rd 2010 from http://alistair.cockburn.us/The+end+of+software+engineering+and+the+start+of+economic-cooperative+gaming
17. Rolland, C. Method Engineering: Towards Methods as Services (2009). *Software Process: Improvement And Practice* 14(3), pp 143–164.