Developer Driven Approach to Situational Method Engineering

Antero Järvi[1], Harri Hakonen[2], and Tuomas Mäkilä[1]

[1] Department of Information Technology, University of Turku, Finland
[2] Aginit Oy, Turku, Finland
antero.jarvi@utu.fi

This position paper reflects SME into software development. We argue that to apply SME in software development projects, construction of method fragments should also take place during the project by the method users. The topic is current due to two key technologies, EPF and SPEM, that enable illustrative and prompt method construction. The paper looks at the relevant background in both SME and software development processes, identifies four levels of method management work, discusses the method reuse strategy, and presents an example of on-the-fly method construction.

1 Introduction

Our background is on software engineering and on pragmatic research with the companies. Currently, we focus on process modeling technologies and their utilization in, for example, reducing the process/project gap. To retain the applicability of the results we work with the processes and process frameworks that are in real use. It has turned out that our work is closely related to Situational Method Engineering (SME) in the Information Systems field, and we see direct applicability of the SME concepts in the software development projects. In what follows, we use 'process' and 'method' as synonyms.

The topic of this paper has become significant due to recent technological advances that have improved our ability to create, organize, reuse, and manage methods. Two key technologies are The Software Process Engineering Metamodel (SPEM) and The Eclipse Process Framework (EPF). SPEM is a standard for defining processes and process components and it is fostered by Object Management Group (OMG). Currently, version 2.0 is at the final stage of standarization [1]. EPF is an open source project that provides tools and content for software process engineering [2]. The EPF Composer supports for all essential SPEM modeling mechanisms although it is not fully SPEM compliant.

Situational Method Engineering (SME) focuses on providing techniques and tools for creating and using project specific methods, instead of having a single generic method. The fundamental goal is to achieve flexibility, as opposed to rigid methods, without sacrificing control over the development project. There are several approaches for pursuing this goal that are reviewed and summarized

Please use the following format when citing this chapter:

Järvi, A., Hakonen, H., Mäkilä, T., 2007, in IFIP International Federation for Information Processing, Volume 244, Situational Method Engineering: Fundamentals and Experiences, eds. Ralyté, J., Brinkkemper, S., Henderson-Sellers B., (Boston Springer), pp. 94-99.

in [3]. The majority of SME methods approach the goal by creating situational method fragments that are selected according to project's situation and then assembled into a project specific method. Another strategy is to start with a full method framework comprising of myriad of method contents capable of supporting a wide range of project situations. A workable method is obtained by configuring the framework with the characteristics of a particular project, or common characteristics of several projects. This approach is widely used in software development industry; a well known example of such commercial method frameworks is Rational Unified Process (RUP) [4].

Distinctive work in flexible processes in the software engineering field includes Boehm's risk-based approach for making methodology decisions that integrate agile and plan-driven practices [5]. In Cockburn's approach, a method is selected according to staffing size and system criticality [6]. Even though this aims at pre-selecting the method, changing the selected method during the project is not uncommon. This indicates the difficulty of seeing the situational forces in advance and the volatility of the project situation.

While these approaches have many differences, they all share a common attribute: separation of method design from its use in terms of time and participation roles. Methods are designed almost solely in advance by method engineers. Also, project specific methods are typically created at the beginning of the project by a method engineer that is external to project's staffing. Recent approaches shift part of the method design into method users' responsibility. Mirbel and Ralyté describe a two step method approach: The first step builds a new method adapted for project situation, while the second step allows the method users to configure further the obtained method for their particular needs [7]. However, the method users do not create new solutions for the situation, but they select what existing method guidance is used in the project. This is very different from a practice-driven approach by Ivar Jacobson et al. [8]. The approach puts a reusable practice in the center of process design; teams will mix and match practices to create efficient ways of working. Practices are used in a framework that allow, for example, to track how value is created and captured in work products.

The hallmarks of recognized SME approaches – separating software process design from its use, externalizing process knowledge and structuring the process modules to form a coherent system – form only one possible strategy for coping with the complexity and uncertainty of current software projects [9]. We argue that this strategy should be complemented with developer driven method design also during the project execution in the real project context.

2 Method management strategy

We identify four levels of method engineering. Firstly, *method library management* takes a facilitating viewpoint to process use. A practical goal is to maintain method content modularized so that method use and reuse in the other

three levels is expedient. This level is the responsibility of method engineers and higher management. Secondly, *project specific method design* describes a method for a particular project situation. The method imposes control onto the development, but leaves choices open where need for adaptation is anticipated. This level implements planned process flexibility. Thirdly, *method fitting* is the responsibility of the process users. Based on the project's real situation, the users select method content that best fits their needs. The fitting is constrained by the project specific method. Fourthly, *on-the-fly method construction* responses to unanticipated situations. A new method fragment is created in the project's process context. The constructed fragment communicates the plan for coping with the situation to all participants, and documents it for further use in process improvement activities.

The balance between these four levels of method engineering should be treated as a strategic choice depending on the company's business and individual project's method needs. One of the main issues is the balance between repeatability and helping the project staff to manage the unanticipated situations. It is evident that one scheme does not fit all needs; some companies operate in a highly dynamic business environment, whereas others operate in a stable business context [10]. The former will not benefit from rigid method repositories. Instead, the strategy should emphasize facilitation of the on-the-fly method construction with method fragments that reflect the teams true capabilities and can be combined flexibly and promptly. The project situation, involving both the business and engineering contexts, resolves on what levels we should put the emphasis.

Business context involves any goals that the project has in addition to producing the deliverables. Requirement of high predictability of cost and time of delivery, need to demonstrate quality or progress during the project, and creating reusable software components highlight the need for the project specific method design. High emphasis on time-to-market and innovative or technically challenging products require maneuverability of the teams. In this kind of surroundings the method is used as a facilitator of team capabilities in unexpected situations. This calls for on-the-fly method construction.

Engineering context involves the predictability and the stability of the method needs in a project. For example, a project affected by many forces not controlled by itself has unpredictable method needs. An unstable project has characteristics that change over time, for example, growing project staff or decision to outsource parts of development. The less predictable and stable the project is, the more we have to rely on on-the-fly method construction.

3 Method reuse strategy

The reusability of a method fragment is determined by its project situation coverage and the engineering scope it impacts, illustrated in Fig. 1. Wide project situation coverage implies high reuse value, whereas a fragment with narrow

coverage describes a solution to an unfrequent situation. The upper levels of method management strategy should concern fragments of high reuse value. Wide engineering scope means that the fragment affects several development disciplines, and thus, should not be tampered with from a local point of view without proper authorization. Fragments with narrow engineering scope are localized and have well-defined and explicit interdependencies in the process.

Every company has a unique mixture of method needs from each of the quadrants, and the challenge is to make the method quadrants work together. Method fragments in the on-the-fly quadrant are solutions to local and possibly unique situations. The challenge is how to construct methods on the fly without impeding software development. The practice quadrant together with the disciplined quadrant is the home ground of SME allowing specific method design for wide range of process types. The challenge is the compatibility and composability of the method fragments so that they can form a seamless method. The disciplined quadrant captures the backbone and dominant assumptions of methods. The challenge is how to retain the process user's ability to modify the method using fragments from practice quadrant [10]. The specialized methods do not involve the reuse aspect, but are highly efficient end-to-end methods for a specific development purpose.

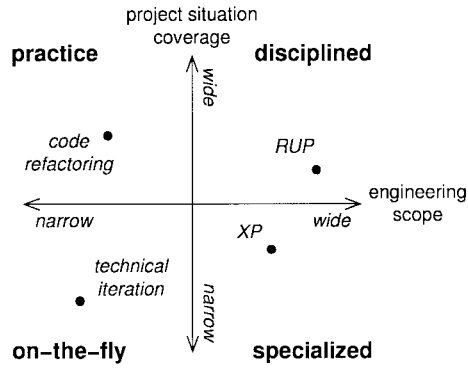

Fig. 1. Project situation coverage and engineering scope characterize the reuse strategy of a method fragment. 'Technical iteration' is an example of the result of on-the-fly construction, 'code refactoring' is a highly reusable fragment having only a local impact, 'RUP' is an example where project dependent practices are intertwined into a process backbone. 'XP' is specialized method for situations including on-site customer, single development team and no architectural risks.

4 Example of the on-the-fly method construction

The following example serves two purposes: Firstly, it shows a typical on-the-fly constructed fragment, and secondly, it illustrates how effortless on-the-fly

construction can be made. The example in Fig. 2 is taken from a real project using an agile development process in *Gaudí Software Factory* [11]. The method modification concerns using a customer requirement driven development iteration as a stating point for creating an iteration where the focus is on solving the technical challenges of the product and new customer requirements are not added. The customer driven acceptance testing is replaced with exploratory testing that is run by the technical expert. 'Write user manual' is removed as unnecessary and 'Refactor' is added to improve the code quality.

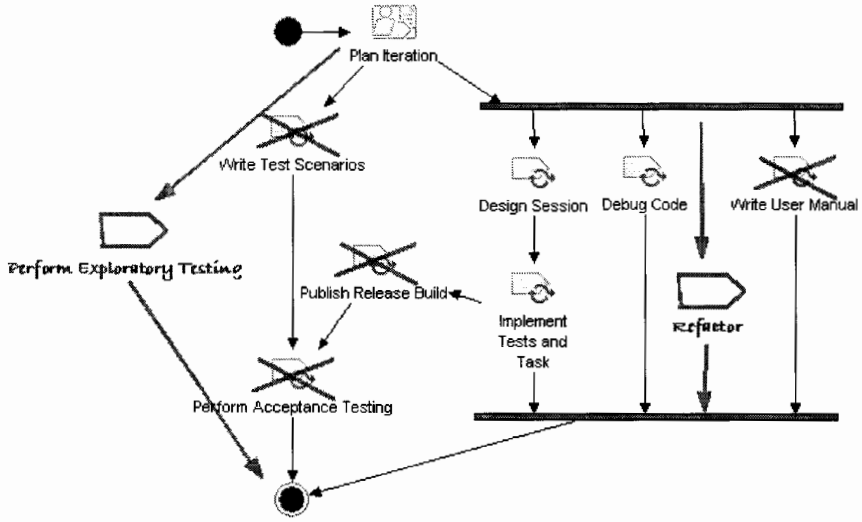

Fig. 2. Example of developing the method fragment 'technical iteration' in on-the-fly construction. The starting and the resulting fragments are combined. The removed activities are crossed out and the additions are shown as free-hand symbols. In practice, the modifications are made with process modeling tools, in this case EPF 1.0. Free-hand graphics is used here for illustrative purposes.

The example demonstrates that on-the-fly construction does not go into details, instead it should focus on devising a plan rather than writing guidance. When this is combined with reusing existing process fragments (e.g. 'refactor' in the example) the construction becomes rapid. The fragment representation is understandable, it communicates the created solution, and shows explicitly the dependencies of the fragment so that they can be taken into account. Finally, the created fragment would probably be useful in other projects and can be analyzed and refined into a reusable practice.

5 Conclusion

The recent development in process standards and tools makes on-the-fly method and method fragment construction feasible in practice. This enables us to allocate part of process management work to development teams: (i) The methods can reach down to operational level development work as it is carried out in the project, narrowing the process/project gap, and (ii) the actual process needs in projects can be captured by on-the-fly construction and they can be communicated to process management to keep processes up to date.

Integrating on-the-fly method construction into existing process management practices is not straightforward. We have presented four levels of process management strategy, and outlined a framework for understanding the reuse strategy and realization of the fragments. However, there are open questions on, for example, structuring of method libraries, composability of method fragments and backbones, roles and responsibilities in process management, and process improvement practices. On-the-fly method construction itself needs further research, in particular the required tool support, the modeling conventions, and sufficient content and level of details in the constructed models.

References

1. Object Management Group. *Software Process Engineering Metamodel Specification, v2.0 Final Adopted Specification ptc/07-03-03*, 2007. http://www.omg.org/cgi-bin/doc?ptc/07-03-03.
2. Eclipse process framework project homepage. http://www.eclipse.org/epf/. Accessed on May 31 2007.
3. Mauri Leppänen. Conceptual evaluation of methods for engineering situational ISD methods. *Softw. Process Improve. Pract.*, 11:539–555, 2006.
4. Philippe Kruchten. *The Rational Unified Process: An Introduction (Second Edition)*. Addison-Wesley Professional, March 14 2000.
5. Barry Boehm and Richard Turner. *Balancing Agility and Discipline, A Guide for the Perplexed*. Addison-Wesley, 2003.
6. Alistair Cockburn. Selecting a projects methodology. *IEEE Software*, pages 64–71, July/August 2000.
7. Isabelle Mirbel and Jolita Ralyté. Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering*, 11:58–78, 2006.
8. Ivar Jacobson, Pan-Wei Ng, and Ian Spence. Enough of processes: Let's do practices part I. *Dr.Dobb's Journal*, April 2007.
9. Ivan Aaen. Software process improvement: Blueprints versus receipes. *IEEE Software*, pages 86–93, October 2003.
10. Antero Järvi, Tuomas Mäkilä, and Harri Hakonen. Changing role of SPI — opportunities and challenges of process modeling. In *The Proceedings of the 13th European Conference, EuroSPI 2006*, LNCS 4257, 2006.
11. Ralph-Johan Back, Luka Milovanov, and Ivan Porres. Software development and experimentation in an academic environment: The Gaudí factory. In *Product Focused Software Process Improvement*, LNCS 3547, 2005.