Method transfer using an open source community

Anders Mattsson[1] and Björn Lundell[2]
1 Combitech AB
P.O.Box 1017, SE-551 11 Jönköping, Sweden
anders.mattsson@combitech.se
2 University of Skövde
P.O. Box 408, SE-541 28 Skövde, Sweden
bjorn.lundell@his.se

Abstract. Method transfer is a problematic task. This paper discusses issues to be addressed when transferring a method and tool to a wide community by forming an open source community contrasted with the experiences from transfer in the context of a company.

1 Introduction

Transfer of software engineering methods and tools has long been [1], and still is [2] considered as a problematic task. In this paper we discuss the transfer of a specific method and tool, from the developer to a gradually wider community, where the final step is to form an open source community around the method and tool to enable global transfer. The purpose with this paper is to raise a number of issues regarding this last step.

The method [3] and tool to be transferred is developed by Combitech AB in collaboration with researchers at University of Skövde and University of Limerick. The purpose of the method and tool is to allow automatic enforcement of architectural design rules in Model-Driven Development (MDD), a task that today is dependent on laborious and error prone manual reviews [4]. The method defines how to formally model the architectural design rules in UML in a meta-model and the tool validates the detailed design in the system model against the modelled rules.

In line with the steps suggested in [5] we see the transfer as a gradual process from the developer to a gradually wider community, but expanded with an additional step where an open source community is formed around the method and tool. This additional step is motivated by the increasing relevance of open source software to the industry [6]. In this particular case Combitech see an opportunity to increase the market for its process improvement services primarily based on the following assumptions:

1. The resistance from companies to adopt the tool, due to lock-in issues, will be reduced since they will have access to the source code. This is particularly interesting to customers who are faced with long term maintenance requirements or build safety critical systems (e.g. Airplane manufacturers).

2. The tool needs to be adapted to different modelling tools, if it is available as open-source, companies do not have to wait for Combitech to do it; they can do it themselves (or pay someone else to do it).

We have previously demonstrated that it is possible for the method developer to formally model a large part of the architectural rules of a representative system. Currently we are transferring it to a practical situation, where it is used in another company (i.e. not Combitech).

Our approach on transfer is based on a set of important factors on transferability of a method defined in [7]. We use the factors of transferability of the framework to structure the following sections. First we describe the conceptual framework of the method and then we discuss each factor of transferability (Learnability, Usability and Acceptability) first related to a current transfer to a specific company context and then contrast this with the issues we see in the transfer to an open-source community.

2 Conceptual framework of the method

The method is documented in a textual document in a set of ten mappings from UML constructs in the architectural model to constraints in the system model. Each mapping is expressed in natural language with an accompanying abstract example. The document also explains why there is a need for the method, provides a couple of simple examples on the use and explains the tooling approach. The method assumes that the user is familiar with UML modelling and that the method is used in a project using MDD using UML models as primary design artefacts. It also assumes that a model validator tool is used for automatic enforcement of the architectural design rules.

3 Company transfer case

Issues on learnabililty:

- Architect and developers are already familiar with UML modelling.
- The method developer and the architect will do the modelling of the initial architectural rules together.
- The tool and the mapping rules are available; they will be modified when necessary. Availability to the tool is essential for learning since the tool will support congruence in interpretations because there will be no room for different interpretations on how follow the design rules.
- There will be a short course (half-day) on the method and tool for the developers before they start the detailed design.

Issues on usability:

The company has a process suited for the Method and are using the Rhapsody modelling tool for which the validation tool is adapted. This means that there is little need for flexibility in the method and tool for it to improve efficiency in the process.

Issues on acceptability - *Management support*:
The method is designed to make it "cheap" to introduce:

- No investments are needed in new tools.
- Very little need for education since we use UML.
- No major impact on the development process, no new activities it only automate old ones (architectural reviews).
- Easy to convince management that it will improve productivity and quality since we automate an earlier error prone, vital and work intensive manual routine.

Issues on acceptability - *Motivation of participants:*
The architect is highly motivated and the strategy is to select developers that are motivated. There is however a risk that we have to use some developers that is a bit antagonistic not only to this method but to model driven development in general. We are very aware of this risk and the strategy is to a) try to avoid using such persons, b) try to convince them of the benefits by concrete examples.

Issues on acceptability - *Value system congruence:*
The method developer has long experience of working as a consultant in the company so he is very knowledgeable on how they do system development. He has worked with several persons involved, including the architect, some developers and most of the management. So there is high value congruence between the method developer and the method users in this case.

4 Open source transfer issues

Issues on learneability:

- The method assumes that it will be deployed in a setting where UML and MDD are used. This limits the potential number of adopters which may make it hard to reach a critical mass to form a successful open-source community around the tool.
- Since there will be no or very limited access to the method developer, the documentation of the method probably needs to be complemented with more tutorial material, after an initial effort by Combitech this will also be driven by the community when it has been formed.
- There is also an obvious need for good documentation on the design of the tool, to make it easy for developers in the community to learn how to adapt and enhance the tool.

Issues on usability:
In an open-source setting flexibility both in the tool and in the method is probably much more important for success. This means for example that we must carefully consider which variation points are important in this setting and possibly redesign the architecture of the tool. Another issue around the tool is that today it is developed

using the quite expensive proprietary modelling tool Rhapsody. In an open-source setting this is not viable. Instead we should move it to an open-source tool chain, e.g. eclipse, to increase our chances to develop a healthy community.

Issues on acceptability - *Management support:*
There may be a need for adopters in investing in an adaptation of the tool to the modelling tool used by the adopter, otherwise the same arguments hold for the open source transfer situation as for the company transfer case. Additional arguments for management support in this case would be the greater flexibility and reduced lock in.

Issues on acceptability - *Motivation for participants and value system congruence:*
In an open source scenario the method and tool developer, Combitech in this case, can not select the adopters. This increases the risk for unsuccessful transfers and the subsequent risk for discrediting of the method. To mitigate this risk it is important that Combitech gets actively engaged in the community to build trust and control. These are also central aspects in establishing a healthy community.

References

[1] C. C. Huff, "Elements of a realistic CASE tool adoption budget", *Communications of the ACM,* vol. 35, pp. 45-54, 1992.

[2] T. Punter, R. L. Krikhaar, and R. J. Bril, "Software engineering technology innovation - Turning research results into industrial success", *Journal of Systems and Software,* vol. In Press, Corrected Proof, 2009.

[3] A. Mattsson, B. Lundell, and B. Lings, "An approach to modelling architectural design rules in UML for model-driven development", in *EMMSAD'08, the Thirteenth International Workshop on Exploring Modeling Methods for Systems Analysis and Design* Montpellier, France.

[4] A. Mattsson, B. Lundell, B. Lings, and B. Fitzgerald, "Linking Model Driven Development and Software Architecture: A Case Study", *IEEE Transactions on Software Engineering,* vol. preprint, October 2008.

[5] B. Lundell, B. Lings, A. Mattsson, and U. Ärlig, "Taking steps to improve working practice: a company experience of method transfer", in *IT Innovation for Adaptability and Competitiveness: IFIP Working Group 8.6*, B. Fitzgerald and E. Wynn, Eds. Boston: Kluwer, 2004, pp. 173-178.

[6] F. van der Linden, B. Lundell, and P. Martiin, "Commodification of industrial software, a case for open source", *IEEE Software,* vol. Accepted for publication, 2009.

[7] B. Lings and B. Lundell, "On Transferring a Method into a Usage Situation", in *Information Systems Research*, B. Kaplan, D. Truex, D. Wastell, T. Wood-Harper, and J. DeGross, Eds. Boston: Springer, 2004, pp. 535-553.