Tool-Driven Technology Transfer to Support Software Architecture Decisions

Heiko Koziolek, Thomas Goldschmidt

Industrial Software Systems
ABB Corporate Research
Wallstadter Str. 59
68526 Ladenburg, Germany
heiko.koziolek@de.abb.com
thomas.goldschmidt@de.abb.com

Abstract: Software architecture design decisions are key drivers for the success of software systems. Despite awareness for their criticality, software architects often rationalize and document their decisions poorly. On this behalf, ABB Corporate Research initiated a technology transfer project to integrate an architecture decision framework from the University of Groningen into ABB software development processes. The project involved close communication between university researchers, industry researchers, and ABB software architects and resulted in the implementation of a plug-in for the UML tool Enterprise Architect. This paper summarizes success factors for the technology transfer, such as strong buy-in from the stakeholders, short feedback cycles, and seamless integration into existing tool-chains.

1 Introduction

ABB Corporate Research faces the issue of transferring software engineering technologies into a large number of diverse ABB business units, which develop software for power systems and industrial automation. In recent years, we have adopted a tool-driven strategy for several software engineering technology transfer projects joint with University partners. This involves for example the creation of Visual Studio plug-ins or Team Foundation Server extensions for ABB's often Microsoft-centric development environments [SDRF12]. Packaging research results into seamlessly integrated development tools can help to make advanced research concepts available to regular developers. This results in a bottom-up, developer-driven transfer and improves the acceptance of the results in the development units.

This paper reports from a tool-driven technology transfer project in the area of software architecture. Architecture design decision research has gained substantial academic interest in the last ten years [BDLV09] (cf. Section 2). Our tooling originates from a published conceptual framework [vHAH12] and was developed in close collaboration with ABB software architects (cf. Section 3). The article summarizes lessons learned while conducting the technology transfer (cf. Section 4).

2 Research on Architecture Decision Modeling

Despite the growing complexity of modern software systems and the higher awareness for software architecture concerns, there is still limited, systematic architecture decision documentation in practice [TBGH06]. Decision documentation is seen as time-consuming and not immediately rewarding, as its true value may be realized only in the maintenance phase of a system when decisions need to be changed or augmented. If software architects use UML diagrams to document structural or behavioral aspects of their systems, it is usually cumbersome to annotate these diagrams with rationale for the decisions made. Common UML tools only allow informal textual annotations, which are difficult to manage and maintain if the documentation gets more complex.

A recent stream of research advocates treating architectural decisions as first class entities of the architectural documentation on the same level as components and interfaces [JB05]. Tang et al. [TBGH06] surveyed how architecture design rationale is treated in practice. Kruchten et al. [KLvV06] argued for a repository of architecture knowledge explicitly documenting decisions and their rationale. A couple of knowledge management tools spanning from Wikis, UML profiles, and programming language extensions have been proposed [TAJ+10]. Zimmermann et al. created reusable decision models for SOA systems [ZGK+07]. The international standard for architecture description (ISO/IEC/IEEE 42010) added architecture decision documentation to its framework in its 2011 revision.

Our technology transfer project originated from one of the recent approaches to architecture decision modeling. van Heesch et al. [vHAH12] proposed a documentation framework for architectural decisions spanning five viewpoints using the conventions of ISO/IEC/IEEE 42010. The relationship viewpoint shows dependencies between decisions, the chronology viewpoint shows different states of decision over time, the stakeholder viewpoint connects stakeholders and decisions, and the detailed viewpoint contains a detailed description and rationalization for a single decision. Finally, the forces viewpoint published in a separate paper shows the decision forces affecting architecture decisions.

As an open problem for the technology transfer, tool support for architecture decision is still limited. Existing tools [TAJ+10] have been developed in academic, non-commercial contexts, and are often not well embedded into architecture design processes or existing tool chains. Although there is the ISO/IEC/IEEE 42010 standard, it provides only coarse-grained guidance what attributes of a decision could be documented, but gives no concrete guidelines. For architecture documentation, usually informal diagrams or sometimes UML models are used in practice. The lack of specific tool support for documenting architecture design decisions makes some software architects reluctant to document their decision rationale at all. For the conceptual decision framework from van Heesch et al. [vHAH12], no tool support existed at all.

3 From Research to Practice

The goal of our technology transfer project for architecture decision modeling was twofold. First, we wanted to better understand the current practices of software architecture documentation and decision documentation. This included not only finding out what

information was important for software architects to document, but also under what organizational and technical constraints such documentation was created. Second, we wanted to improve the current practice of explicit decision documentation. Therefore, the goal was to create a documentation tool that seamlessly integrated into existing ABB software development processes in order to lower the barrier to document decisions in the future. Besides the concrete tool implementation, involving ABB software architects into the requirements engineering process for the tooling was intended to disseminate the idea of explicit decision modeling from research into practice.

The overall project team consisted of one PhD student and two Master students from the partnering University as well as two researchers from ABB Corporate Research. The project duration was one year. We involved five practicing ABB software architects in our study, which had 10 to 24 years of experience in software development and were all working in projects where they made architecture design decisions. The architects worked in the domain of industrial process automation systems and each designed different products.

To gather requirements for the tooling and better understand the constraints under which the software architects worked, we first conducted a series of semi-structured phone interviews. We asked each architect for the current practice of architecture decision documentation, gave a brief preview of the planned tooling, and asked for specific functional and non-functional requirements. Additionally, we analyzed existing architecture decision documentation from two projects. This for example involved spreadsheet templates, slide sets, and architecture design documents. There was also a document giving architecture modeling guidelines, which we analyzed to find a good way of integrating the decision documentation.

We found that all architects were familiar with the concepts of architecture decision documentation, but that there was no standard way of doing this. Some architects documented decisions in presentation slides or spreadsheets. Others provided decision rationale in informal texts accompanying UML diagrams. Decision rationale was also sometimes only kept in e-mails between different stakeholders or meeting minutes without being integrated into the standard architecture documentation at all.

Most architects used the tool Enterprise Architect from Sparx Systems to model in UML. Some of them had created tool chains (e.g., to generate documents), trainings, and modeling guidelines. Introducing a Wiki- or web-based tooling for documenting architectural decisions was not desirable as it would have added another tool, which required administration as well as synchronizing with the UML models. Therefore, we decided to implement the architecture decision tooling as an add-in to Enterprise Architect. Besides the software architects' familiarity with Enterprise Architect, one reason was to be able to seamlessly connect existing UML models with the decision documentation, which was seen as a great benefit by the software architects.

The software architects added a number of requirements for the tooling, which for example included generation of presentation slides from the model, automatic creating of the chronological decision viewpoint, or the ability to create trace-links between decision alternatives and complete diagrams. The architects also agreed that the tooling should

be flexible and not force the user to document each decision with the same detail. As a major constraint, the architects work under time pressure and are usually not immediately rewarded for complete and high quality documentation. Therefore, complex decision documentation templates were seen critical.

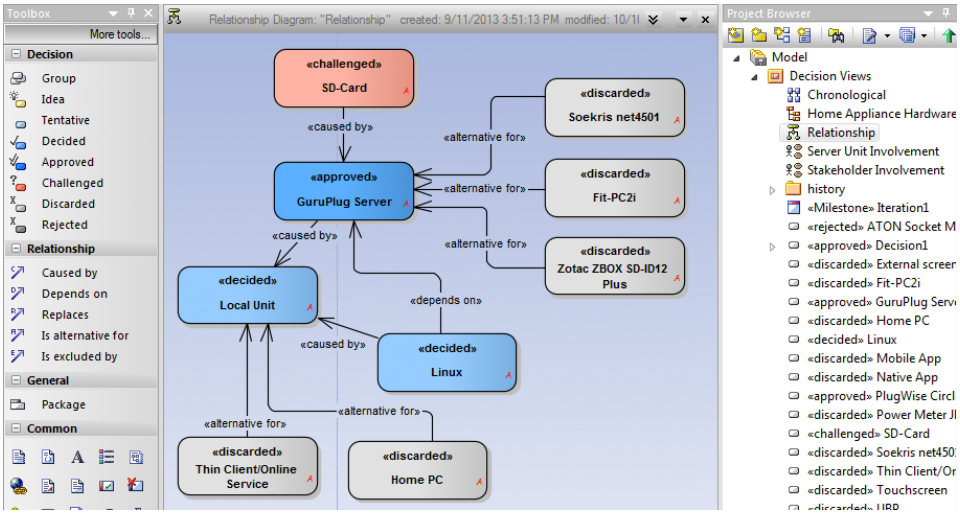

Figure 1: Decision Viewpoint Add-in for Enterprise Architect

Once we had implemented a prototype of the Enterprise Architect add-in, we sent it to the architects and asked them to test it. We then arranged five interview and tool demonstration sessions with each architect each lasting half a working day. We presented the current version of the tooling and asked the architects to document a few decisions from a current project. We observed the participant in using the tool to uncover usability issues. Afterwards, we interviewed the architects for their feedback.

Each architect was generally positive about the tooling, but had different emphasis regarding the different viewpoints. Some favored the forces viewpoint, which allowed a table-like comparison of multiple decision alternatives. Others saw the stakeholder viewpoint as interesting, as it allowed them to trace decisions to particular stakeholders. After using the tooling, the architects requested a change of the decision meta-model, so that the issue, alternatives, and outcome of a decision were captured more explicitly. Besides, they came up with a number of usability improvements.

The final version of the tooling incorporating the software architects' requirements is still under development. After its release, the software architects will use the tooling in their project to retroactively document a number of important decisions. We plan a third interview session with the architects in order to improve the tooling further. Once the Enterprise Architect add-in is a mature state, it is planned to be released as open source software, so that it can be used by other Enterprise Architect users and be extended for additional features by an interested community.

4 Lessons Learned

From our technology transfer project, we learned a number of generic lessons that are helpful for similar situations.

Beneficial to center technology transfer around tooling: In our case it was very useful to drive the technology transfer through the implementation of a software tool. It forces researchers to make former conceptual work applicable for a larger audience. The tooling pre-packages knowledge on how to structure architecture decision documentation and can provide immediate value to a software architect in a given project. It makes the concepts more accessible than through a research paper. The tooling is still generic and can be used for many projects inside and outside of ABB. ABB is following this tool-driven approach to technology transfer for other concepts as well (e.g., for code search [SDRF12]).

Different emphasis in academic vs. technology transfer tool development: The emphasis of tool development in academic contexts is often on creating a proof-of-concept solution that suffices to carry out an empirical validation in a well-protected setting. Instead, the emphasis of tool development in technology transfer projects is rather on process integration and robustness. Existing artifacts (e.g., models), workflows, and tool-chains need to be respected in order to get a buy-in for the tooling. Many of the implemented features provide no academic value (e.g., document generators), but are of high interest for the practitioners. The reliability and usability of the tooling is more important than a comprehensive list of features.

Short feedback cycles important to get buy-in from users: It proved very valuable to closely collaborate and communicate with the eventual users of the approach, i.e., the software architects. We presented some of the most promising concept to the software architects early to get their buy-in. The architects valued that we considered their documentation guidelines and templates. We included the architects already in early requirement gathering phases and made sure that their inputs were addressed. The architects were motivated when they saw that the tooling respected their particular issues. We not only presented the tooling to the architects, but also had them experimenting with early prototypes so that they became more familiar with the overall idea.

Technology transfer as a source for research problems: While technology transfer per se does not provide novel publishable results besides potential empirical validations, its role in finding new research problems may be underestimated. Through the process of collaboration between researchers and software architects, pointers for future research can be identified. For example, we found that architecture decision documentation is still focused mainly to single products and that a knowledge transfer about decision rationale seldom happens between products. Therefore, it is desirable to enable cross-product decision documentation in the future. Understanding better the issues in practice and the constraints (e.g., time, budget, skills) under which a software engineering approach is applied creates a bi-directional knowledge transfer between research and practice.

Tool support requires long-term commitment: If a technology transfer project creates tooling, it must ensure its proper maintenance and evolution after the project has ended. This is often a problem in academic settings, where research projects typically last only 1-3 years and there is little interest in maintenance after the project has finished. The

university collaborations at ABB are usually funded only for a single year, so that the same problem applies. We are thus aiming at making the developed software open source and creating a developer community around it to ensure its evolution. The tooling can also serve as a platform for future tool development for technology as it allows extensions with new concepts. Nevertheless, we deem long-term tool support after technology transfer projects have ended still a hard problem that needs attention from funding committees.

5 Conclusions

We have presented the tool-driven technology transfer process ABB Corporate Research applies in selected software engineering University collaborations. As an example, we have created an add-in to a popular UML tool and developed the tooling in close interaction with the target users. Centering the technology transfer around tool implementations brings many benefits such as the need to make conceptual contributions applicable and the ability to quickly benefit from the new concepts. A challenge to this form of technology transfer is the long-term commitment to the maintenance of the tooling, which we try to address by creating an open developer community. In the future we will carry out more such tool-driven technology transfer projects, which have proven to be valuable instrument of bringing advanced software engineering technologies into our organization.

References

[BDLV09] Muhammad Ali Babar, Torgeir Dingsyr, Patricia Lago, and Hans Vliet, editors. *Software Architecture Knowledge Management: Theory and Practice*. Springer, 2009.

[JB05] Anton Jansen and Jan Bosch. Software Architecture as a Set of Architectural Design Decisions. In *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture*, WICSA '05, pages 109–120, Washington, DC, USA, 2005. IEEE Computer Society.

[KLvV06] Philippe Kruchten, Patricia Lago, and Hans van Vliet. Building up and reasoning about architectural knowledge. In *Proc. 2nd Int. Conf. on the Quality of Software Architectures (QoSA'06)*, QoSA'06, pages 43–58, Berlin, Heidelberg, 2006. Springer-Verlag.

[SDRF12] David Shepherd, Kostadin Damevski, Bartosz Ropski, and Thomas Fritz. Sando: an extensible local code search framework. In *Proc. 20th Int. Symp. on the Foundations of Softw. Eng.*, FSE '12, pages 15:1–15:2, New York, NY, USA, 2012. ACM.

[TAJ+10] Antony Tang, Paris Avgeriou, Anton Jansen, Rafael Capilla, and Muhammad Ali Babar. A comparative study of architecture knowledge management tools. *J. Syst. Softw.*, 83(3):352–370, March 2010.

[TBGH06] Antony Tang, Muhammad Ali Babar, Ian Gorton, and Jun Han. A survey of architecture design rationale. *J. Syst. Softw.*, 79(12):1792–1804, December 2006.

[vHAH12] Uwe van Heesch, Paris. Avgeriou, and Rich. Hilliard. A documentation framework for architecture decisions. *J. Syst. Softw.*, 85(4):795–820, April 2012.

[ZGK+07] Olaf Zimmermann, Thomas Gschwind, Jochen Küster, Frank Leymann, and Nelly Schuster. Reusable architectural decision models for enterprise application development. In *Proc. 3rd Int. Conf. on the Quality of Softw. Architectures (QoSA'07)*, QoSA'07, pages 15–32, Berlin, Heidelberg, 2007. Springer-Verlag.