Improving Traceability through AOSD

Andreas Rummler
SAP Research CEC Dresden
Chemnitzer Str. 48
01187 Dresden, Germany
andreas.rummler@
sap.com

Birgit Grammel
SAP Research CEC Dresden
Chemnitzer Str. 48
01187 Dresden, Germany
birgit.grammel@
sap.com

Christoph Pohl
SAP Research CEC Karlsruhe
Vincenz-Priessnitz-Str. 1
76131 Karlsruhe, Germany
christoph.pohl@
sap.com

1. INTRODUCTION

In real world business applications traditional software product line engineering and model-driven software development (MDSD) [6] often cannot properly reflect the decomposition of system features. For instance, Governance, Risk and Compliance (GRC) checks or late introduction of security properties often crosscut the architectural design of a system. To overcome these issues Aspect-Oriented Software Development (AOSD) [1] modularizes such crosscutting concerns in independent aspects. Although aspects can already be captured at requirements stage [2, 7], there is no clear mapping to later development stages. MDSD can address this by model transformation. However, AOSD still increases the complexity of traceability (and hence, maintainability) because it adds yet another dimension of variability [3]. This issue is one of the most important arguments against applying AOSD techniques in an industrial context. Future research has to take care of this issue in order to lay the basis for industry acceptance of AOSD. According to an internal audit of customers of SAP, missing traceability during the whole development cycle is the top-rated weakness. In addition missing traceability information was explicitly mentioned as weakness in 2005 in an external ISO certification audit.

2. TRACEABILITY SUPPORT INSIDE SAP

The current state of trace support implemented in SAPs internal development process is outlined in figure 1. Market requirements are linked to software specifications comprising software requirements that can be linked to test cases. Further linkage of requirements to design and development artefacts, or linkage of those artefacts to test cases is possible in general, but not sufficiently supported by tools (indicated by dashed lines). Currently, several interesting questions typically arising during the development process cannot be answered automatically: Have all high-level market requirements really been designed and implemented properly? How can this implementation be tested against these requirements? Have artefacts been developed whose behaviour is not covered by any requirements or which is even unwanted? Why has one artefact been chosen over another, alternative one? These question can be answered on a fine-grained level by experts involved in the development process, but not by people dealing with a coarse-grained view. Therefore tools are needed that are able to give reasonable answers to these questions or at least help people in finding those answers. MDSD has its merits for transforming artefacts into each other based on certain models, but crosscutting functional-
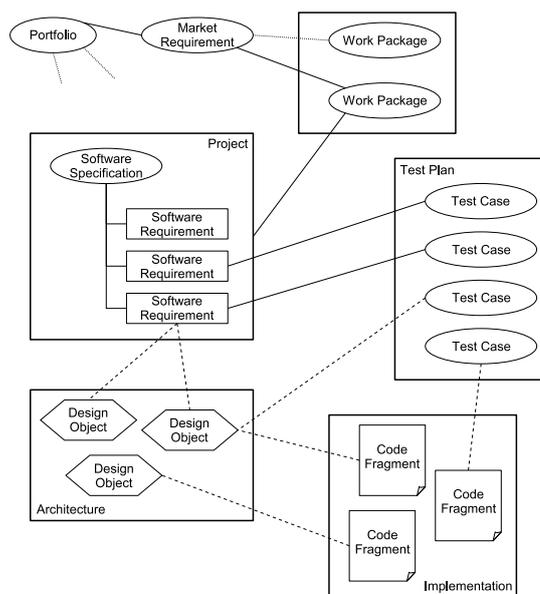

Figure 1: Linking of Artifacts in SAPs Development Process

ity as outlined above tangles not only code but also such models.

3. IMPROVING TRACEABILITY THROUGH AOSD

There are approaches to Aspect-Oriented Requirements Engineering (AORE) [2, 7] that can help address some of these questions. For instance relationships, dependencies and interactions among existing requirements can be identified at early stages of the development lifecycle. However, AORE approaches do not explicitly define mechanisms for mapping information gathered at the requirements level to later development phases. There is a need for defining mapping guidelines, rules, and heuristics for mapping of entities and trace information across the entire development lifecycle. Assets repositories are also required that may collect and maintain product line assets and the mapping rules, guidelines, and heuristics. In addition, there is a need for a traceability meta-model that defines which trace information: assets, concerns, relationships, dependencies, behaviours, compositions, mappings, needs to be captured and managed.

SAP Research is currently involved in an European funded project called AMPLE, which stands for Aspect-Oriented, Model-Driven Product Line Engineering.[1] The focus is on providing a holistic treatment of variability by addressing each stage in the software life cycle. Another point of interest in AMPLE is providing effective forward and backward traceability of variations and their impact.

The approach followed in AMPLE relies on the modularization of cross cutting concerns at model level. Starting already at the stage of requirements engineering will foster traceability. To be able to track dependencies between AO and non-AO artefacts along the development cycle, explicit aspect interfaces need to be defined. To rely on earlier work already made available in [4] and [5] seems to be promising. The intrusive nature of AO techniques is reduced in its intensity by defining aspect interfaces that form some kind of contract between the to-be-extended system and the extending aspects. Other ongoing work concerns a metamodel for variability including the support of AO concepts and appropriate tracing information. Based on this metamodel a tool chain is designed that supports the definition of SPL, product generation and full support for tracing relationships and dependencies among automatically generated or manually created artefacts. A comprehensive case study is currently being implemented, consisting of a complete example from SAPs core application business.

4. CONCLUSIONS

To summarise this position paper, tracing artefacts throughout the whole development process is a key issue in industry, driven by internal *and* external forces. Handling variability and documenting decisions on variations is the core issue of traceability. AOSD approaches introduce interesting concepts to modularise cross-cutting concerns at various development stages but it also complicates traceability. Explicit aspect interfaces are one requirement for easier tracking of dependencies between AO and non-AO artefacts. At the workshop, we would like to share our industry perspective on how AOSD and MDSD could further fertilise each other for improving traceability issues, among other challenges.

5. REFERENCES

[1] R. E. Filman, T. Elrad, S. Clarke, and M. Aksit. *Aspect-Oriented Software Development.* Addison-Wesley, 2005.

[2] J. Grundy. Aspect-oriented requirements engineering for component-based software systems. In *Proceedings of the 4th IEEE Symposium on Requirements Engineering*, 1999.

[3] S. Katz and A. Rashid. From aspectual requirements to proof obligations for aspect-oriented systems. In *Proceedings of the 12th IEEE International Conference on Requirements Engineering*, 2004.

[4] G. Kiczales and M. Mezini. Aspect-oriented programming and modular reasoning. In *Proceedings of the ACM International Conference on Software Engineering*, 2005.

[5] U. Kulesza, V. Alves, A. Garcia, C. Lucena, and P. Borba. Improving extensibility of object-oriented frameworks with aspect-oriented programming. In *Proceedings of the International Conference on Software Reuse ICSR 06*, 2006.

[6] T. Stahl and M. Vlter. *Model-driven Software Development.* John Wiley, 2006.

[7] B. Tekinerdogan, A. Moreira, J. Araujo, and P. Clements. Early aspects: Aspect-oriented requirements engineering and architecture design. In *Workshop Proceedings at AOSD Conference*, 2005.

---

[1]The website for the project can be found at `http://www.ample-project.net`