A Process for Explicitly Integrated Software Architecture

Marco Konersmann
marco.konersmann@paluno.uni-due.de
paluno, Universität Duisburg-Essen, Germany

Abstract

This paper presents the Explicitly Integrated Architecture Process for integrating architecture models with program code. The process allows to edit program code using architecture model views with automatic bidirectional translations.

1 Motivation

When long-living software evolves, developers need to understand the software and the program code structures that build the software. It is easier to understand abstract models of the code than the code itself. During the evolution of long-living software, models tend to become outdated. When such inconsistencies between the model and the code exist, faults might be introduced during maintenance and evolution when developers rely on outdated information.

Our approach to this challenge is to integrate architecture model specifications into program code. The models can be embedded into and reliably extracted from the code. This paper briefly presents the Explicitly Integrated Architecture Process for integrating architecture models described in an Architecture Description Language (ADL) with program code that complies to existing component models (CM).

2 Related Work

For synchronizing models with code several approaches exist: Methods for code-generation mainly include model-driven development (MDD) (e.g. [2]) and round trip engineering (RTE) (e.g. [5]). Regarding the motivation of our approach, these approaches either allow only one-way synchronization (MDD) or do not bridge the gap between abstraction levels (RTE). Model reconstruction (e.g. [6]) aims to construct models from code. These approaches create persistent models and require considerable manual effort. Model execution (e.g. [3]) handles models as data for model execution engines. These approaches can bridge the gap between abstraction levels, but only refer to behaviour models.

3 Process

Currently no approach exists to edit code in an architecture model view, that bridges abstraction levels, with an automatic two-way translation. To overcome this gap, we now describe a process for integrating architecture model information with program code complying to component models. The code will reuse structures required by the CM framework in use, and contain additional structures, which represent architecture model elements that cannot be expressed in the CM language. The process makes explicit, what traditionally is only implicit in the code. It is therefore called *Explicitly Integrated Architecture Process*.

3.1 Process Concepts and Elements

The process is based on three main concepts. Figure 1 gives an overview of the elements used within the process and their interrelationships, and shows where the stated concepts are used.

Model Integration Concept: The model integration concept describes how model information is integrated with program code. In figure 1 the concept provides vertical integration. It is used to integrate and extract architecture model information from a CM and the intermediate language in the program code (e.g. [4]). The concept is based on the ideas of Moritz Balz [1]. The code is statically analyzed for program code structures that identify CM elements. E.g. Java EE enterprise beans are identified via their annotations on Java types. A model is built from this analysis that represents the code in terms of the CM meta model. Additional architecture information from the intermediate language can be applied using various techniques, such as marker interfaces or annotations. The model can be edited in its model form, or by editing the corresponding code structures.

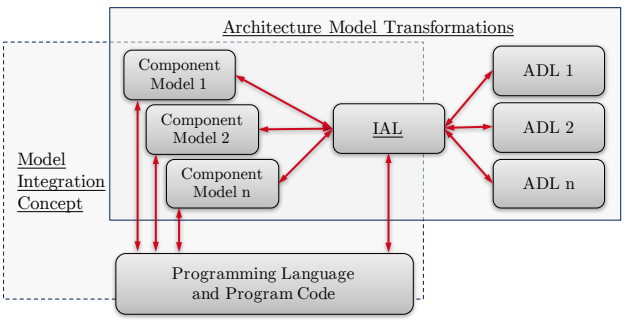

Figure 1: The elements of the process and where the three main concepts (underlined) are located.

Intermediate Architecture Description Language (IAL): The IAL is used to represent architecture information independently from the ADL and the CM that are used to specify and implement the architecture. It has the role to increase the process' interoperability with different ADLs and CMs, and to increase the evolvability of the approach. ADLs and CMs have different kinds of information they are able to describe. E.g. in contrast to ADLs, CMs often cannot describe a deep component hierarchy. The IAL handles these differences using a profile concept similar to UML profiles.
Architecture Model Transformations: In figure 1 the transformations provide the horizontal integration. Two kinds of transformations are used within the process: (a) ADL models are synchronized with IAL and CM representations. (b) Translations between interrelated profiles of the IAL are defined.

3.2 Process Steps

In the process, architecture model information is extracted from the code for editing, and integrated with the code after editing or creating the model. The process is visualized in figure 2. It can be started either from code that complies to a CM – including code that has not been developed using the process yet – or from either a new, or an extracted ADL model. The process defines three main activities:

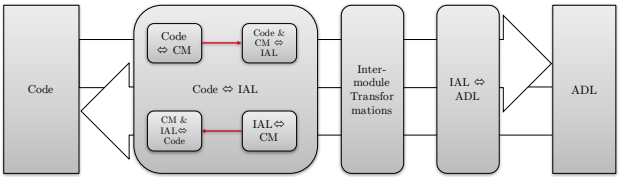

Figure 2: Overview of the Explicitly Integrated Architecture Process

(1) **Code ⇔ IAL:** An IAL model is created from the code using the model integration concept. The activity comprises two subactivities: In the subactivity *Code to CM* the code is translated into a model of the CM meta model. In the subactivity *Code & CM to IAL* this CM representation is translated into an IAL model. The IAL model is enriched from the code with information, that cannot be expressed in the given CM, using the model integration concept.
(2) **Intermodule Transformations:** The IAL representation is translated according to the features of the involved languages. E.g flat architectures can be represented hierarchically with one hierarchical level. The transformations do not delete original information. Therefore no information is lost.
(3) **IAL ⇔ ADL:** The IAL representation is translated into an ADL representation. The ADL model can now be edited with its original editor. The changed model can then be integrated with the existing program code by executing the steps in reverse direction.

4 Conclusion

We are currently implementing *Codeling*, a tool for the process based on the Eclipse IDE. The source code is publicly available[1]. We evaluate the approach using the CoCoME[2] project and during the development of Java EE programs in our working group. The current implementation works with an EJB and CoCoME codebase and allows for the extraction and integration of Palladio and UML models. The approach allows for translations between many CMs and ADLs, whereas the tool is currently focused on Java-based CMs and Ecore-based ADLs. However the tool could be extended to address further languages.

This paper gave an overview about the Explicitly Integrated Architecture Process. It is used to edit architecturally relevant code in architecture model views. After a motivation, we addressed related work. Then we stated the concepts and the steps within the process. At last we described the evaluation.

References

[1] Moritz Balz. *Embedding Model Specifications in Object-Oriented Program Code: A Bottom-up Approach for Model-based Software Development*. PhD thesis, Universität Duisburg-Essen, 2011.

[2] Alan Brown, Jim Conallen, and Dave Tropeano. Introduction: Models, Modeling, and Model-Driven Architecture (MDA) Model-Driven Software Development. In *Model-Driven Software Development*. Springer, Berlin/Heidelberg, 2005.

[3] A. Hen-Tov, D. H. Lorenz, and L. Schachter. ModelTalk: A Framework for Developing Domain Specific Executable Models. In *Proceedings of the 8th OOPSLA Workshop on Domain-Specific Modeling*, 2008.

[4] Marco Konersmann and Michael Goedicke. A Conceptual Framework and Experimental Workbench for Architectures. In *Software Service and Application Engineering*, volume 7365 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012.

[5] U. A. Nickel, J. Niere, J. P. Wadsack, and A. Zündorf. Roundtrip Engineering with FUJABA. In *Proceedings of the 2nd Workshop on Software-Reengineering (WSR)*, 2000.

[6] A. van Deursen and C. Riva. Software architecture reconstruction. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 745–746, May 2004.

[1] https://s3gitlab.paluno.uni-due.de/ADVERT/codeling
[2] http://cocome.org