Thoughts on the Evolution Towards Model-Integrating Software

Mahdi Derakhshanmanesh[1], Marvin Grieger[2], Jürgen Ebert[1], Gregor Engels[2]
[1]University of Koblenz-Landau, Germany, Institute for Software Technology
[2]s-lab – Software Quality Lab, Germany, Paderborn University
Email: {manesh|ebert}@uni-koblenz.de, {mgrieger|engels}s-lab.uni-paderborn.de

1 Background and Motivation

Developing software that can be modified and evolved easily is a challenging task. Yet, the fast-paced market requires quick adaptation of products in reaction to emerging requirements.

As a basis for flexible software, we proposed to develop software based on *Model-Integrating Components* (MoCos) in previous work [1]. A MoCo is a non-redundant, reusable and executable combination of logically related models and code in an integrated form where both parts are stored together in one component. The outcome of two comprehensive feasibility studies on building software with MoCos let us conclude that combining the strengths of modeling languages (e.g., abstraction, separation of concerns) and programming languages (e.g., performance) within components yields flexible and well-performing software, indeed.

Even though we provided the MoCo concept as a basis for designing a flexible target architecture, we neglected to introduce process descriptions for engineering *Model-Integrating Software* (MIS), so far.

As a first step towards closing this gap, we introduce the current state of an engineering process for the development of MIS. In addition, we describe a set of evolution scenarios to evolve existing software towards MIS and discuss related evolution processes.

2 Model-Integrating Development

The central phases of *Model-Integrating Development* (MID) are aligned with the Model-Driven Architecture (MDA), a well-accepted development style for complex software. The core activities and artifacts of MID are shown in Figure 1.

The main phases of MID are: (i) *Platform-Independent Design* where *Platform-Independent Models* (PIMs) express the central software parts without relations to a concrete target platform, (ii) *Platform-Specific Design* where *Platform-Specific Models* (PSMs) express the mapping to a platform-specific solution and (iii) *Implementation* where the final software is derived in its deployable form.

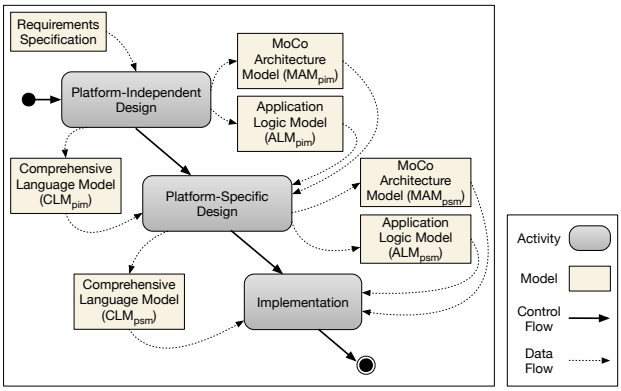

Figure 1: Core Activities and Artifacts of the Model-Integrating Development Process

Platform-Independent Design Phase. First, using the *MoCo Template* [1], the *MoCo Architecture Model* ($\mathrm{MAM_{pim}}$) is specified in terms of components and connectors. Second, the same model is used to specify the parts of each MoCo to be represented by models and by code. The *Application Logic Model* ($\mathrm{ALM_{pim}}$) describes the domain-specific data and behavior encapsulated by a MoCo including the internal interfaces between models and code. Third, based on the desired language capabilities for the $\mathrm{ALM_{pim}}$, different modeling language are specified with a *Comprehensive Language Model* ($\mathrm{CLM_{pim}}$) that conforms to the *CLM Template* (specification of syntax, semantics and pragmatics).

Platform-Specific Design Phase. First, a component technology such as *OSGi* is chosen and the $\mathrm{MAM_{pim}}$ is transformed accordingly. This step yields the *MoCo Architecture Model* ($\mathrm{MAM_{psm}}$). Second, the $\mathrm{ALM_{pim}}$ is refined to match a concrete programming language, its suitable execution environment as well as a concrete technological space for model-

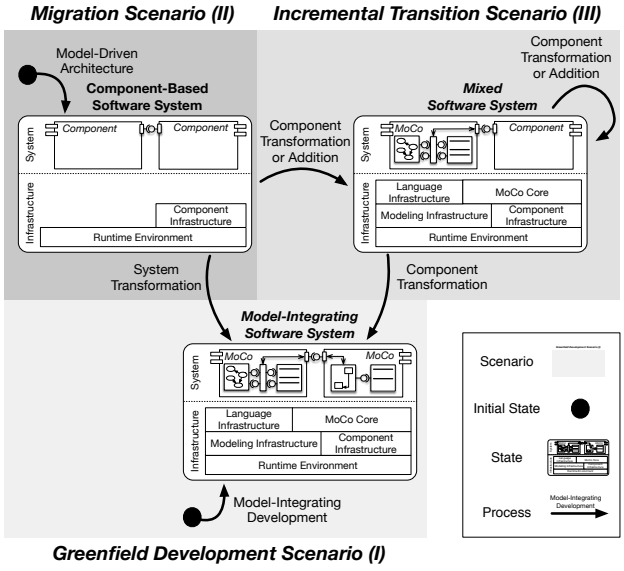

Figure 2: Evolution Scenarios Towards Model-Integrating Software

ing. This step yields the *Application Logic Model* (ALM_{psm}). Third, the specified modeling languages are mapped and optimized for the chosen target platform, which yields a specific *Comprehensive Language Model* (CLM_{psm}) for each modeling language.

Implementation Phase. First, all technical dependencies are derived from the PSMs and the target environment is set up. Second, the MAM_{psm} is used to generate a project structure that reflects the software architecture. Third, the application logic code part of each MoCo is generated using the ALM_{psm} and integrated with the remaining *Application Logic Model* (ALM_{rt}) to be used at runtime.

3 Evolution Towards MIS

To describe *Model-Integrating Evolution* (MIE) processes, i.e., processes to evolve existing systems towards model-integrating systems, we introduce the envisioned *evolution scenarios*. These scenarios can be seen in Figure 2.

For the sake of completeness, we include the *Greenfield Development Scenario* (I). Thereby, an MIS is developed from scratch. In this scenario, the MID process that we introduced in the previous section can be applied to guide the endeavor.

The *Migration Scenario* (II) describes the situation that an existing component-based system is turned into an MIS. Thereby, all components are migrated to MoCos at once. In this scenario, we envision to apply a model-driven transformation process [2] to guide the *System Transformation*. This process is aligned with the Architecture-Driven Modernization (ADM) and uses the same abstraction levels as the MDA. We assume that the use of this specific process is beneficial for two reasons. On the one hand, the overall process is model-driven, i.e., models are central artifacts to realize automatic transformations. Some of these models can be directly used within the resulting MoCos. On the other hand, tools like metamodels that are required to automatically transform the system can later on be reused in the *MoCo Infrastructure*.

The first two scenarios represent situations in which the MIS is created at once. In contrast, the *Incremental Transition Scenario* (III) describes the situation that an existing component-based system is incrementally turned into an MIS. In this scenario, new components that are added to the system are MoCos (*Component Addition*). In addition, existing components are transformed to MoCos over time, i.e., they are replaced or migrated (*Component Transformation*). We envision to guide the replacement as well as the addition by a variant of the MID process, while the model-driven transformation process guides the migration of single components. In this scenario, a specific challenge is to incrementally evolve the already existing infrastructure to a complete MoCo Infrastructure.

4 Concluding Remarks

MoCos support the development of flexible software that can adapt itself at runtime and can also be evolved easily. In this short paper, we presented an overview of the MID process as a systematic way of engineering MoCo-based software. We also sketched the opportunities of evolving existing systems towards model-integrating systems. This research direction will be deepened in future work.

Acknowledgements. This work is supported by the Deutsche Forschungsgemeinschaft (DFG) under grants EB 119/11-1 and EN 184/6-1.

References

[1] M. Derakhshanmanesh, J. Ebert, T. Iguchi, and G. Engels. Model-Integrating Software Components. In J. Dingel and W. Schulte, editors, *Model Driven Engineering Languages and Systems, 17th International Conference, MODELS 2014, Valencia, Spain, September 28 - October 3, 2014*, Valencia, Spain, 2014. Springer.

[2] M. Grieger and M. Fazal-Baqaie. Towards a Framework for the Modular Construction of Situation-Specific Software Transformation Methods. *Softwaretechnik-Trends*, 35(2):41–42, 2015.