Enabling Technologies for Longevity of Software

Mahdi Derakhshanmanesh
Institute for Software Technology
University of Koblenz-Landau
manesh@uni-koblenz.de

Marvin Grieger
s-lab – Software Quality Lab
University of Paderborn
mgrieger@s-lab.uni-paderborn.de

Abstract

In ongoing work, we advocate the integration of models and code as two equal constituents of software components. This approach can be followed to create flexible and self-adaptive software. We claim to extend this direction as a conceptual and technological basis for combining adaptation and evolution to achieve longevity. As a stimulus for discussions at the workshop, we present our work on model-integrating software components as one enabling technology for supporting a seamless software evolution process. In addition, we propose further enabling technologies and provide an initial set of related challenges.

1 Introduction

Our ongoing work on model-integrating software [DEIE14] is motivated by the observation that models can facilitate the engineering of flexible software. The use of *Model-Integrating Components* (MoCos)[1] is our current approach to realize *self-adaptation* [ST09] as a means to react to foreseen changes in a software system's context. However, some changes cannot be foreseen and require further (manual) modification effort. The process of changing a software system after its initial development and deployment is called *software evolution* [RB00].

Hasselbring et al. [HHJ$^+$13] highlight, that it is important to consider an integration of the *adaptation cycle* and the *evolution cycle* as they influence each other. Combining (i) the adaptation cycle, (ii) the evolution cycle and (iii) the (potentially distributed) knowledge base for storing meta-data, best practices and more related to both cycles yields a *Seamless Software Evolution* (SSE) process. We adapt this view in our work as we assume that synergies between these cycles exist that can be utilized to support not only the development of flexible but also long-living software systems.

The contribution of this short paper is a proposal of *enabling technologies* (ETs), for which we assume that they support the SSE process, and corresponding challenges. An enabling technology provides the capabilities and means (e.g., tools, techniques, processes, guidelines) to tackle challenges that arise in the context of the SSE process. Three ETs are exemplarily visualized by a slightly extended version of the iObserve figure that was originally proposed in [HHJ$^+$13]. It is depicted in Figure 1.

[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.

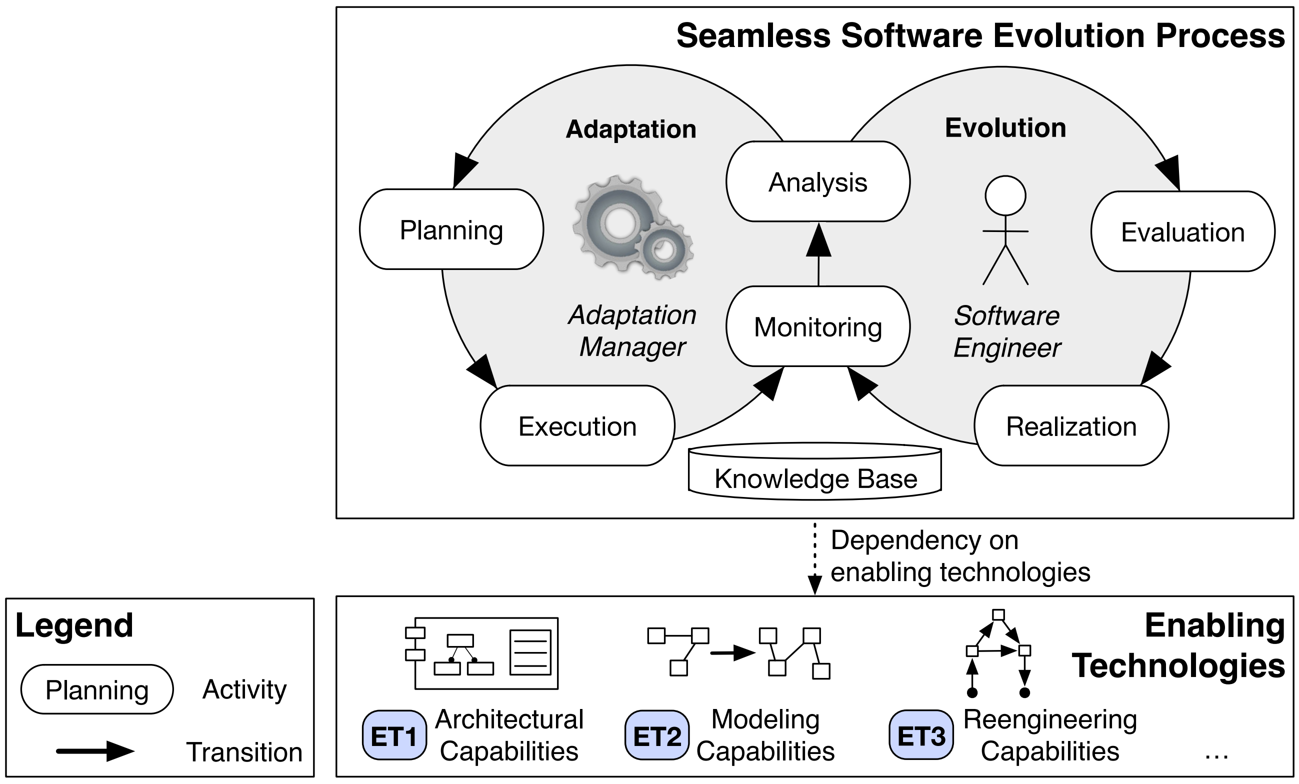

Figure 1: Interwoven cycles of software adaptation and evolution (based on [HHJ$^+$13]) as an integrated process for seamless software evolution that is supported by enabling technologies.

2 Enabling Technologies and Related Challenges

Certainly, there are open challenges related to each cycle and especially to their single activities. For example, the monitoring and analysis activities are crucial when assuming that they need to keep both cycles "in sync". Hence, one may ask how to automatically decide when to switch from one cycle to the other, e.g., in cases where the adaptation manager cannot control an emerged situation.

While very relevant, such questions are out of the scope of this short paper, though. Instead, we like to point out that there is a need for *enabling technologies* to support the SSE process. We believe that they must at least facilitate *architectural capabilities* (ET1) and provide *modeling capabilities* (ET2) as well as *reengineering capabilities* (ET3). Next, we discuss these three enabling technologies in more detail and identify an initial set of related challenges.

We have already discussed some of the benefits of combining models and code within model-integrating software components to enhance flexibility and to offer software engineers a spectrum of possibilities to choose from when designing and implementing their systems [DEIE14]. As knowledge shall be shared between activities in the adaptation and evolution cycles, strategies and guidelines will be needed. Hence, we are convinced that providing the right *architectural capabilities* for software design and development is critical, but still requires some challenges to be tackled. Amongst others, questions related (i) to the distribution of models onto software components as well as (ii) to the role-based accessibility of these models within components must be answered.

Considering that models are used in both cycles – these may be even the very same models – we assume that an elementary set of *modeling capabilities* is required by both cycles. For example, activities in both cycles might attempt to modify the same model, e.g., during execution and realization. This yields the challenge to provide transaction concepts for models and the sketched knowledge base. Another example for capabilities required by both cycles are the specification and the realization of execution environments for models, since executable models can be used to achieve adaptivity and the evolution cycle must be capable of introducing and changing them. Whenever code of a software system is transformed to a model, either by adaptation or evolution, a corresponding execution environment needs to be instantiated. A related challenge is to find ways for efficiently performing this task.

Furthermore, unforeseen events need to be tackled in the evolution cycle. There exists a large body of knowledge in (model-driven) software reengineering that can be applied here. We assume that a solid set of *reengineering capabilities* to support this task will also be beneficial for adaptation. For instance, evolution may require to reverse engineer and restructure a software system, or to generate source code. The same capabilities can be beneficial for adaptation purposes at runtime. Challenges include how to specify and implement reengineering methods that are reusable under varying situational context, namely adaptation and evolution.

Indeed, we are convinced that there are many more challenges to be discussed and we hope to identify a more concrete list of them during the workshop.

3 Concluding Remarks

A decade ago, challenges for software evolution were discussed by Mens et al. [MWD$^+$05]. The topic is still very relevant, today. Existing architectural approaches to develop adaptive software, like MoCos, can be beneficial to achieve longevity. Models (at runtime) – e.g., such as those proposed by Blair et al. [BBF09] – can play a key role at the intersection between adaptation and evolution cycles. However, we believe that these directions alone are not sufficient and claim that a set of enabling technologies that facilitate at least (i) architectural capabilities, (ii) modeling capabilities and (iii) reengineering capabilities is a fundamental prerequisite to achieving the greater vision of long-living software. This short paper intends to provide a starting point and context for discussion.

Acknowledgements

This ongoing work is a result of the MoSAiC project, supported by the Deutsche Forschungsgemeinschaft (DFG) under grants EB 119/11-1 and EN 184/6-1.

References

[BBF09] Gordon Blair, Nelly Bencomo, and Robert B. France. Models@run.time. *Computer*, 42(10):22–27, 2009.

[DEIE14] Mahdi Derakhshanmanesh, Jürgen Ebert, Thomas Iguchi, and Gregor Engels. Model-integrating software components. In Juergen Dingel, Wolfram Schulte, Isidro Ramos, Silvia Abraho, and Emilio Insfran, editors, *Model-Driven Engineering Languages and Systems*, volume 8767 of *Lecture Notes in Computer Science*, pages 386–402. Springer International Publishing, 2014.

[HHJ$^+$13] Wilhelm Hasselbring, Robert Heinrich, Reiner Jung, Andreas Metzger, Klaus Pohl, Ralf Reussner, and Eric Schmieders. iobserve: Integrated observation and modeling techniques to support adaptation and evolution of software systems. *Technical Report*, 1309, 2013.

[MWD$^+$05] T. Mens, M. Wermelinger, S. Ducasse, S. Demeyer, R. Hirschfeld, and M. Jazayeri. Challenges in Software Evolution. In *Principles of Software Evolution, Eighth International Workshop on*, pages 13–22, Sept 2005.

[RB00] Václav T Rajlich and Keith H Bennett. A Staged Model for the Software Life Cycle. *Computer*, 33(7):66–71, 2000.

[ST09] Mazeiar Salehie and Ladan Tahvildari. Self-Adaptive Software: Landscape and Research Challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):14:1—-14:42, 2009.