Service Engineering: The Sensoria Model Driven Approach

Martin Wirsing, Matthias Hölzl, Nora Koch, Philip Mayer, Andreas Schroeder
Institut für Informatik, Ludwig-Maximilians-Universität München
Oettingenstr. 67, 80538 München, Germany

[wirsing, hoelzl, kochn, mayer, schroeda]@pst.ifi.lmu.de

Abstract

Service engineering and Service-Oriented Architectures (SOAs) have recently been embraced by both industry and research, as they promise high reusability and maintainability, and a flexible environment for future changes in business requirements and workflows. In this paper, we present a model-driven service engineering approach called the SENSORIA Development Approach (SDA). The SENSORIA project has developed a number of formal techniques that support service engineering, e.g., quantitative or qualitative analyses of service artifacts. The main contribution of the SDA is the integration of these techniques in a process for engineering service-oriented systems.

1. Introduction

The aim of the IST-FET project SENSORIA [1] is the development of a novel comprehensive approach to the engineering of service-oriented software systems where foundational theories, techniques and methods are fully integrated into pragmatic software engineering processes.

Within SENSORIA, a development approach for service engineering has been developed which we call the SENSORIA Development Approach (SDA). The SDA is intended to support model-driven engineering by including formal methods and tools at the appropriate steps, thus building a formally underpinned development approach. The SDA is discussed in section 2.

The SENSORIA Development Approach is tool-supported: All tools are integrated into a common integration platform, the SENSORIA Development Environment (SDE), which is described in section 3.

We conclude in section 4.

2. The Sensoria Development Approach

The SENSORIA Development Approach (SDA) follows the patterns described in the OMG's Model Driven Architecture [2], which is "*an approach to using models in software development*" [3]. The SDA, like MDA, builds computational independent models (CIMs), platform independent models (PIMs) and platform specific models (PSMs). In addition, the SDA contributes formal methods, notations, and tools to the engineering process. Figure 1 gives an overview of the SDA approach.

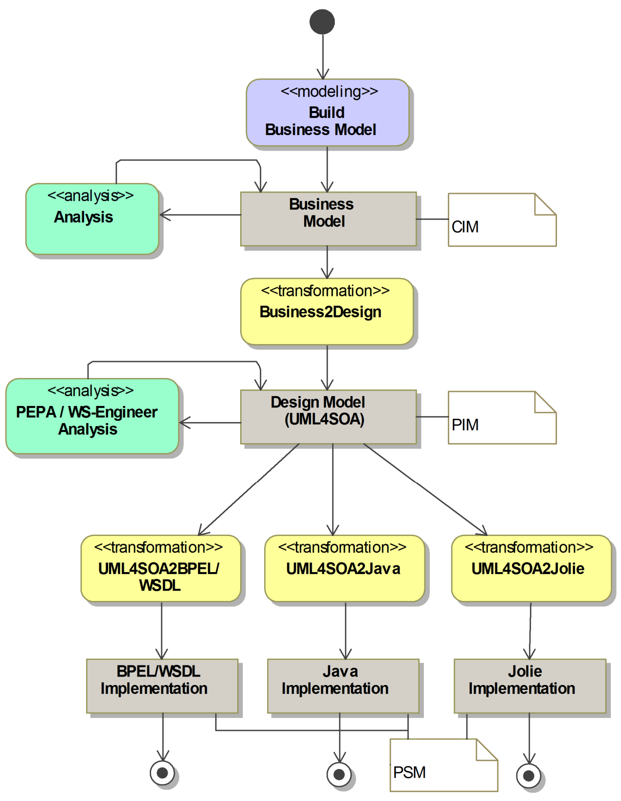

Figure 1: The SDA Approach

As can be seen in the figure, the first point of interest is building the business model. The requirements engineering approach Tropos [4] has been selected for building the CIM business model. Tropos is well suited for requirements analysis of services as it allows for a uniform treatment not only of the system functional requirements, but also of its non-functional requirements. It can easily be integrated in the proposed approach, as Tropos is a model-based and transformational approach.

A Tropos characteristic is to distinguish between early and late requirements engineering. The early requirements analysis is concerned with the understanding of a problem evaluating organizational aspects. The result achieved in this first stage is an organizational model including actors and dependencies among them. The goal of the late requirements analysis is the description of the operational environment, i.e. the functionality of the system.

In the next step, we deal with service artifacts on a platform independent level, defining, through UML models, both the static and dynamic structure of the service-oriented system. We employ the UML4SOA [5] notation for modeling service artifacts, which has been developed as part of the SENSORIA project. The UML4SOA profile defines several stereotypes and constraints for working with SOA artifacts. For example, we extend UML activity diagrams with stereotypes for modeling service orchestrations, in particular for service-to-service communication (send, receive, among others) for long running transactions (scopes, compensation handlers, and primitives for invoking compensation), and event handling.

Besides the usual benefits of graphical models – i.e., they are easy to create, understand, and can be used for communicating with non-technical people – the UML4SOA models also enjoy formal support by means of analysis tools. WS-Engineer [6] is a tool for analyzing the problem of engineering software components as services and the interaction behavior between them in a qualitative fashion. In particular, WS-Engineer focuses on service process analysis for service interaction verification and validation, service composition and choreography obligations, and scenario-based synthesis of service components. The techniques used are verification on the one-hand, i.e. the question whether a model is built correctly, analysis of properties of interest (deadlock, liveness, fluent, etc.), and validation and simulation on the other hand, i.e., the question of whether the system is the right solution for the problem by animating models back to designers.

The PEPA tool [7], along with the SENSORIA Reference Markovian Calculus (SRMC), has the goal of capturing non-functional properties, in particular performance aspects, early in the life-cycle. Non-functional properties are specified on UML models and transformed into the stochastic process algebra PEPA for analysis. Results of the analysis are back-annotated to the models, or displayed in the form of graphs.

One of the benefits of model-driven approaches is the fact that platform-dependent models, or at least parts of them, can be generated from the platform-independent models. In our case, transformers are available which convert the platform-independent UML models into platform-specific models of the target languages, which can be serialized down to code.

The SENSORIA project includes various tools and methods for transforming models. The VIATRA2 project [8], for example, comes with a rule-based language for specifying transformation between EMF [9] models. We also have a Java-based transformation infrastructure available [10] which has been developed specifically for behavioral specifications (in particular, for service orchestrations) and converts platform-independent input models (in this case, UML) to an intermediate model (called Intermediate Orchestration Model, or IOM), and from there to Platform-Specific Models (PSMs).

The transformers currently handle the transformation from UML to BPEL and WSDL files, to Java, and to the formal language Jolie which has the backing of the SOCK calculus [11].

3. The Sensoria Development Environment

In the previous section, we have reported on methodologies and tools supporting the SENSORIA Development Approach. In order to enable developers to use those tools in combination in a coherent environment, we have developed a specialized development environment for service artifacts which offers service modeling, analysis, and code generation functionality, the SENSORIA Development Environment (SDE) [6]. The SDE itself is based on a Service-Oriented Architecture, allowing easy integration of tools and querying the platform for available functionality. The tools hosted in the SDE are presented as discoverable, technology-independent services. As development of services is a highly individual process and may require several steps and iterations, the SDE offers a composition infrastructure which allows developers to automate commonly used workflows as an orchestration of other tools.

The SDE is based on Eclipse and its underlying OSGi platform. As many tools in the greater area of service orientation are available for Eclipse as well, the SDE-enhanced Eclipse platform offers a pragmatic way of adding formal methods to the development process.

4. Conclusion

With the number of systems built with Service-Oriented Architectures (SOAs) on the rise, a rigorous development process for such systems becomes imperative. In this paper, we have presented the results of the SENSORIA project on a development approach for services, which we call the SENSORIA Development Approach (SDA). The SDA supports a model-driven approach to service engineering.

The main contribution of the SDA is the provision of formal tools and methodologies to the engineering of service-oriented software. In particular, we provide tools with the backing of formal languages for both quantitative and qualitative analyses of service artifacts.

It is our hope that the practices described in the SENSORIA approach will find their way into mainstream development processes such that all developers may profit from the results in the more formal research areas of computer science.

10. References

[1] M. Wirsing, A. Clark, S. Gilmore *et al.*, "Semantic-Based Development of Service-Oriented Systems". pp. 24–45, *FORTE06, Paris, France.* 2006.

[2] OMG, "The Model Driven Architecture (MDA)", http://www.omg.org/mda/, [2003]. Last visited: 16.05.2008.

[3] OMG, "MDA Guide Version 1.0.1", ftp://ftp.omg.org/pub/docs/omg/03-06-01.pdf, [2003]. Last visited: 16.05.2008.

[4] L. Diana, and M. John, "Designing Web Services with Tropos", in Proceedings of the IEEE International Conference on Web Services, 2004.

[5] P. Mayer, A. Schroeder, and N. Koch, "A Model-Driven Approach to Service Orchestration". *Intl. Conference on Services Computing (Short Paper, 4 pages), Honolulu, USA.* 2008.

[6] H. Foster, and P. Mayer, "Leveraging Integrated Tools for Model-Based Analysis of Service Compositions". *Third International Conference on Internet and Web Applications and Services (ICIW 2008) Athens, Greece.* 2008.

[7] M. Tribastone, "The PEPA Plug-in Project". pp. 53–54, *4th International Conference on the Quantitative Evaluation of SysTems (QEST), Edinburgh, Scotland.* 2007.

[8] D. Varró, and A. Balogh, "The model transformation language of the VIATRA2 framework", *Sci. Comput. Program.,* vol. 68, no. 3, pp. 214-234, 2007.

[9] Eclipse Foundation, "EMF: The Eclipse Modeling Framework", http://www.eclipse.org/emf, [2008]. Last visited: 28.04.2008.

[10] P. Mayer, A. Schroeder, and N. Koch, "The UML4SOA Profile and Model Transformations", http://www.uml4soa.eu/, [2008]. Last visited: 21.05.2008.

[11] F. Montesi, C. Guidi, and G. Zavattaro, "Composing Services with Jolie". *ECOWS'07, Halle, Germany.* 2007.

About the Author

Prof. Dr. Martin Wirsing
Institut für Informatik
Ludwig-Maximilians-Universität München

Martin Wirsing is Full Professor and Chair for Computer Science at Ludwig-Maximilians-University of Munich, where he is also vice-chairman of the Senat. His current research interests comprise software engineering for service-oriented systems and for hypermedia applications and software development based on formal methods. Currently he is the scientific co-ordinator of the Integrated EC Project SENSORIA on semantic-based software development for service-oriented computing. He is president of the scientific committee of INRIA France and member of several other international scientific committees including University of Nancy (France) and the John von Neumann Minerva Center for the Development of Reactive Systems (Israel).