# SOAMIG Project: Model-Driven Software Migration towards Service-Oriented Architectures

**A. Winter, C. Zillmann**
OFFIS Oldenburg
Institute for Information Technology
{zillmann, winter}@offis.de

**A. Fuhr, T. Horn, V. Riediger**
Institute for Software Technology (IST)
University of Koblenz-Landau
{afuhr, horn, riediger}@uni-koblenz.de

**A. Herget, W. Teppe, M. Theurer**
Amadeus Germany
{aherget, wteppe, mtheurer}@de.amadeus.com

**U. Erdmenger, U. Kaiser, D. Uhlig, Y. Zimmermann**
pro et con GmbH
{uwe.erdmenger, uwe.kaiser, denis.uhlig, yvonne.zimmermann}@proetcon.de

***Abstract*—The SOAMIG project aims at developing a general migration process for model-driven migrations towards Service-Oriented Architectures. This paper highlights the model-driven tools developed during the SOAMIG project for two case studies: A language migration from a COBOL transactional server to Java web services, and a second study on an architecture migration from a monolithic Java fat client to a SOA-based JavaEE web application.**

## I. Motivation

Today, companies are facing a growing competition in their markets. Competitors are forced to achieve higher flexibility and faster time-to-market in order to survive. Often, so-called legacy software developed in the companies can not keep up with this highly dynamic environment and therefore slows down innovation.

For this reason, companies are looking for flexible software concepts supporting fast adaptability to business changes. A promising approach to achieve the required flexibility are Service-Oriented Architectures (SOAs). SOAs encapsulate functionality in coarse-grained, loosely-coupled and reusable units, called services.

Adopting SOAs, companies do not want to throw away their existing systems because much money and knowledge has been put into them. Instead of reimplementing the service functionality from scratch, companies are striving to reuse their legacy software as much as possible. Transferring existing code into a new technology without changing functionality is called software migration.

The SOAMIG project, partially funded by the German Ministry of Education and Research (BMBF)[1], brings together both: transition into SOA by migrating the legacy code. The overall goals are i) to define a reference process [1], ii) to achieve a high degree of automatic code migration, and iii) to support the migration process by analysis and transformation tools.

In this project, two universities and two companies have been involved: the *Universities of Oldenburg* (OFFIS) and *Koblenz-Landau* (IST) supplying reengineering knowledge and model-driven tools, *pro et con*, supplying long-time expertise in industrial migration projects, language analysis, and migration tools development, and *Amadeus Germany*, providing one of the industrial legacy systems and know-how in migration of large-scaled systems.

[1]Grant no. 01IS09017A-D. See http://www.soamig.de for further information.

During the SOAMIG project, two industrial case studies were selected: the *LCOBOL* case study deals with a language migration from a transaction driven COBOL system to Java Web Services, while the *ASPL* case study is about an architecture migration from a Java fat client into a Java SOA. This short-paper presents the tools developed during the SOAMIG project.

## II. LCOBOL: Language Migration

The LCOBOL case study is conducted by pro et con, one of the industrial partners. The main challenge in this case study is to yield a very high degree of automation for a language migration from COBOL to Java. Also, the resulting Java code has to be understandable and maintainable. Figure 1 shows the set-up of the tool chain.

Every COBOL source file is parsed into a fine-grained abstract syntax graph by the COBOL front-end `CobolFE`. `CobolFE` can handle various COBOL dialects. The main translation to Java is done by the model-to-model transformation `Cobol2Java`. This tool takes the COBOL model as input. The actual transformation is defined by many sophisticated rules defining a semantics-preserving transformation into a Java model. Project specific rules, e.g., on how to transform specific transaction monitor calls, amend the language translation. The transformations are implemented in C++. Finally, Java source code is generated by `JGen` and `JFormat`. `JGen` takes a model of a Java translation unit as input and creates syntactically correct, but only roughly formatted output. `JFormat` is a stand-alone, scriptable Java source code formatter based on the Eclipse JDT formatter and is individually configurable to various formatting conventions.

## III. RailClient: Architecture Migration

In the RailClient case study, an architecture migration from a monolithic Java system into a SOA-based web application is investigated by all four project partners. Figure 2 outlines the tools developed in this part of the SOAMIG project. Amadeus Germany provided the business case and the subject system, an order management and booking system for train tickets. OFFIS contributed to the definition and realization of the target architecture.

The *SOAMIG repository* forms a common core of the tool-chain. In this repository, artifacts used during migration are stored as models. The main part of the repository

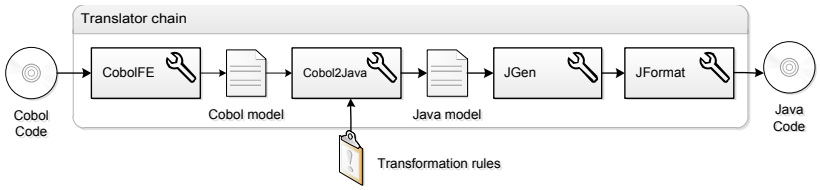


Figure 1. Tool set-up for the LCOBOL case study

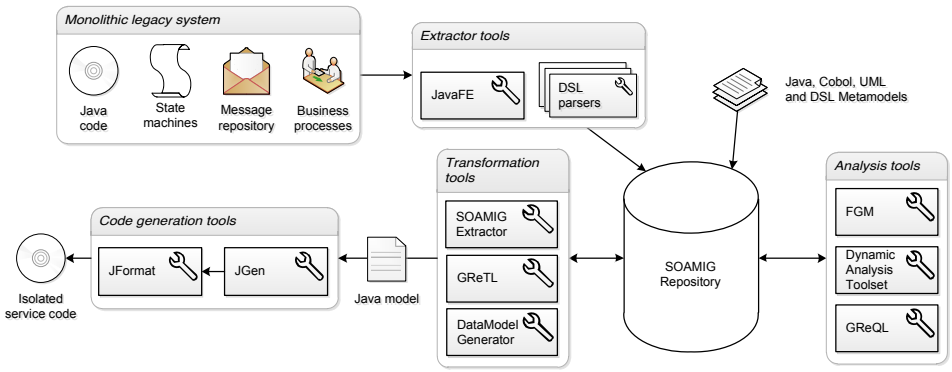


Figure 2. Tool set-up for the RailClient case study

is based on the TGraph technology developed by IST. TGraphs are a versatile data structure formally defined by grUML (graph UML). The TGraph technology is generic and can represent arbitrary artifacts. In SOAMIG, the tools are integrated by an XML-based exchange format for metamodels and models.

To ease initial program understanding and redocumentation, the explorative tool `FGM` (Flow Graph Manipulator) by pro et con was used. This partner also provided `JavaFE`, an extractor from Java source to fine-grained abstract syntax graphs stored in the repository. Not only source code, but also various other parts of the legacy system, such as automatons controlling GUI behavior, message descriptions, and redocumented business processes are combined into one comprehensive model. Links between these parts are established by static and dynamic analysis. Static analysis is covered by `JavaFE` and `GReQL` (Graph Repository Query Language). Dynamic analysis of certain test cases covering the selected business processes is used to detect relevant portions of the source code, and to mark service candidates [2].

Among the transformation tools in this case study is `GReTL` (Graph Repository Transformation Language), a general-purpose transformation language that allows to define and execute arbitrary graph transformations. A system specific `DataModelGenerator` combines message descriptions and dynamic traces to compile service specific data structures for the target architecture.

The `SoamigExtractor` tool provides a graphical interactive user interface to enable model transformations. Examples are incorporating dynamic traces, pruning generalization hierarchies, selection and completion (slicing) of multi-class Java models based on execution traces, establishment of traceability links between source and target models, and export of translation unit models to the above mentioned `JGen` and `JFormat` tools. The generated Java code requires manual rework. It contains human readable as well as machine processable annotations to link to the relevant legacy sources.

## IV. Conclusion

Summarizing, a set of powerful model-driven tools and technologies has been developed to support various tasks during the migration process. Most of the tools are independent of the concrete legacy system and are reusable as-is, others have to be configured or have even been built from scratch. The SOAMIG repository technology is largely generic, enabling integration of additional metamodels and traceability to the already existing parts.

Every migration project requires adaption and specialization of process, repository, and tools to the specific needs of the legacy and target systems, the organizational requirements, and other factors. While the complete migration of the case study systems is out of scope of the project, the model-driven tools have proven to be applicable in real-world scenarios. Transfer to other business cases and different migration tasks is an opportunity to further evolution of the SOAMIG process and technology.

## References

[1] U. Erdmenger, A. Fuhr, A. Herget, T. Horn, U. Kaiser, V. Riediger, W. Teppe, M. Theurer, D. Uhlig, A. Winter, C. Zillmann, and Y. Zimmermann, "The SOAMIG Process Model in Industrial Applications," in *Proceedings of the 15th European Conference on Software Maintenance and Reengineering*, T. Mens, Y. Kanellopoulos, and A. Winter, Eds. Los Alamitos: IEEE Computer, 2011, pp. 339–342.

[2] A. Fuhr, T. Horn, and V. Riediger, "Dynamic Analysis for Model Integration (Extended Abstract)," *Softwaretechnik-Trends*, vol. 30, no. 2, pp. 70–71, 2010.