An Integrated Tool Suite for Model-Driven Software Migration towards Service-Oriented Architectures

Andreas Fuhr, Tassilo Horn, Volker Riediger

University of Koblenz-Landau, Germany

{ afuhr | horn | riediger }@uni-koblenz.de

Abstract

Model-driven approaches as well as migration projects rely on a strong tool support. As part of the SOAMIG project, a tool suite has been developed, supporting the model-driven migration of legacy Java and COBOL systems towards Service-oriented Architectures (SOAs). The tool suite integrates a global repository (representing business processes, code and architecture) and capabilities for (i) parsing legacy artifacts (code and project specific architecture constructs), (ii) analyzing legacy code (static and dynamic analysis) and (iii) transforming artifacts.

1 Introduction

Model-driven approaches heavily rely on tools. Without a strong tool support for modeling as well as for querying and transforming models, model-driven techniques are not applicable on real-life projects. Considering the tools, software migration projects have similar dependencies. Without powerful tools for analyzing and migrating legacy systems, these projects are doomed to failure. As a consequence, a combination of both disciplines – called model-driven software migration approaches – requires the development of a strong tool suite for model-driven analysis and migration of legacy systems.

As part of the SOAMIG project[1], a tool suite has been developed, supporting the model-driven software migration of legacy systems towards Service-Oriented Architectures (SOAs). The suite integrates tools for the following tasks:

1. Metamodeling and generation of repository structure; model persistence, querying and transformation
2. Parsing various legacy artifacts (e.g., Java/Cobol code, business processes or project-specific architecture constructs)
3. Analyzing legacy artifacts (static and dynamic analyses)
4. Transforming legacy artifacts
5. Generating code (Java, isolated service code)

In this paper, we describe the SOAMIG tool suite covering the tasks mentioned above. In addition, we present two industrial case studies the tool suit was applied to.

The remainder of this paper is structured as follows. Section 2 introduces the integrated tool suite. Section 3 briefly describes the two case studies, the tool suite was applied to. Finally, Section 4 concludes the paper.

2 The SOAMIG Tool Suite

This section describes the SOAMIG tool suite as shown in Figure 1.

2.1 Repository Technology – The TGraph Approach

One of the core elements of the tool suite is the integrated repository. Artifacts that are used during migration are stored as models in this repository.

The main part of the repository is implemented by TGraphs. TGraphs are typed, attributed, directed and ordered graphs. They are specified by grUML (graph UML) and due to their generic nature used to represent the artifacts of the migration.

With the *Graph Repository Querying Language (GReQL)*, a general-purpose querying language for TGraphs is provided. GReQL is used in static and dynamic analyses to retrieve information from the repository.

With the Graph Repository Transformation Language (GReTL), a generic transformation language for TGraphs is provided. GReTL is used for graph/model transformations.

For accessing and manipulating TGraphs, the Java framework JGraLab (Java Graph Laboratory)[2] can be used.

2.2 Extractor Tools

Various artifacts have been stored as model during the SOAMIG project. For parsing legacy source code,

[1]This work is partially funded by the German Ministry of Education and Research (BMBF) grant 01IS09017C/D. See http://www.soamig.de for further information.

[2]TGraphs, grUML, GReQL, GReTL and JGraLab are developed by the Institute for Software Technology

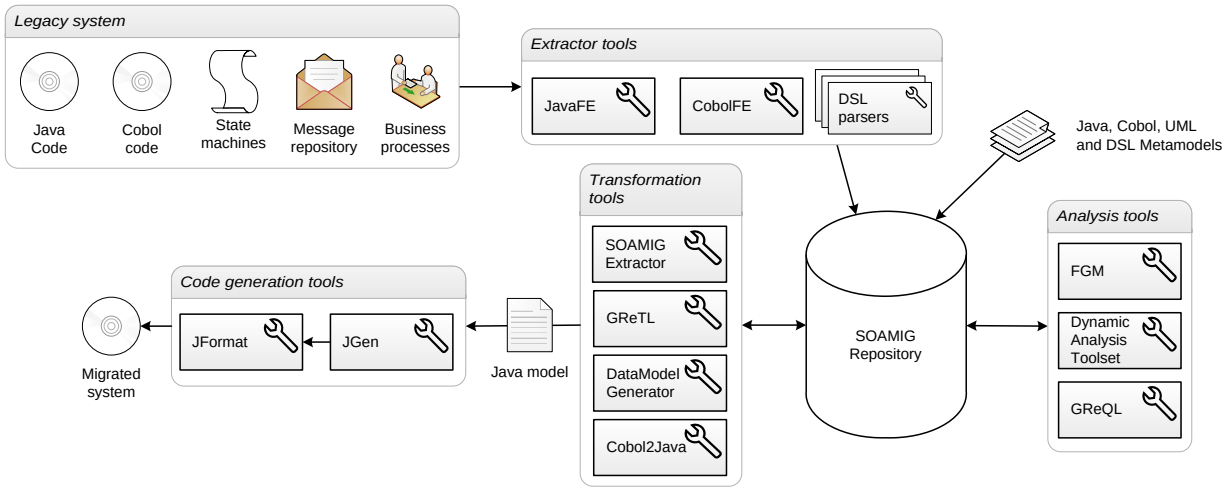

Figure 1: The SOAMIG tool suite.

the *Java Frontend (JavaFE)* and the *Cobol Frontend (CobolFE)*[3] have been used. Both parsers generate fine-grained syntax-graphs which are stored in the repository.

Redocumented business processes (modeled as UML 2.0 activity diagrams), state machines controlling GUI behavior (modeled as UML 2.0 state machines) and an XML message repository (project-specific model) have been parsed using specialized DSL parsers.

2.3 Analysis Tools

For understanding and redocumenting legacy code, the *Flow Graph Manipulator (FGM)*[4] was used. Further static analyses were supported by GReQL.

In addition, a dynamic analysis tool-set-up was developed, tracing which legacy code was executed during a business process. This information was used to integrate the business and code model and to identify legacy code for service implementations.

2.4 Transformation Tools

For Cobol to Java language migration, the tool Cobol2Java[5] was used. Based on sophisticated transformation rules, Cobol2Java transforms Cobol models into Java models.

To enable message exchange in SOA environments, the *Data Model Generator (DMG)* tool was used to generate service-specific data structures. The DMG combines dynamic traces and message repository information.

The *SoamigExtractor*[6] tool provides a graphical user interface for Java model transformations, e.g., pruning generalization hierarchies, slicing of multi-class Java models based on execution traces or establishing traceability links between source and target models. Model transformations may be supported by GReTL.

2.5 Code Generation Tools

Two tools are used for Java code generation: JGen and JFormat[7]. *JGen* is used to unparse a Java model to Java source code, supporting only basic formatting. *JFormat* is an adaptable Java source formatter based on the Eclipse JDT formatter.

3 Case Studies

The tool suite was developed during and tested on two case studies: LCOBOL and RailClient.

LCOBOL was a language migration from Cobol to Java. The tool suite was used to parse various Cobol dialects, transform the Cobol models to Java models (semantics-preserving) and to unparse the Java models to formatted Java source code.

RailClient was a architecture migration aiming at migrating a legacy monolithic Java client to a Service-Oriented Architecture. The tool suite was used for program understanding and for service identification and service migration.

4 Conclusion

As part of the SOAMIG project, a comprehensive tool suite was developed and integrated, supporting model-driven migration towards Service-Oriented Architectures. The tool suite was tested on two case studies. Experiences of these two case studies indicate, that the integrated tool suite is able to provide the required tool support for model-driven software migration projects.

[3] JavaFE and CobolFE are developed by pro et con
[4] FGM is developed by pro et con
[5] Cobol2Java is developed by pro et con
[6] The Data Model Generator and the SoamigExtractor are developed by the Institute for Software Technology
[7] JGen and JFormat are developed by pro et con