Tool- und Schnittstellenarchitektur für eine SOA-Migration

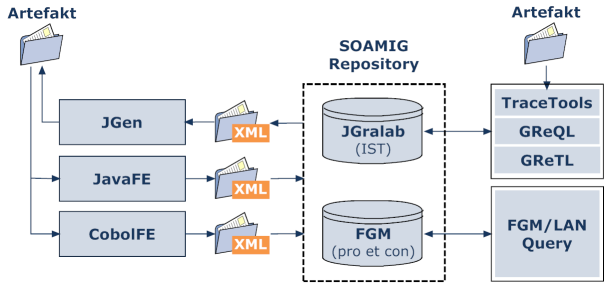
Yvonne Zimmermann, Denis Uhlig, Uwe Kaiser
pro et con Innovative Informatikanwendungen GmbH, Dittesstraße 15, 09126 Chemnitz
yvonne.zimmermann@proetcon.de,denis.uhlig@proetcon.de, uwe.kaiser@proetcon.de

Abstract

In dem vom BMBF geförderten Projekt SOAMIG (Migration von Legacy-Software in serviceorientierte Architekturen) arbeiten die pro et con Innovative Informatikanwendungen GmbH, Amadeus Germany, das Institut für Softwaretechnik der Universität Koblenz-Landau (IST) und der OFFIS e. V. als Partner im Verbund. Projektziel ist es, ein allgemeingültiges Prozessmodell für eine SOA-Migration zu entwickeln, prototypische Werkzeuge zu realisieren, welche die Migration automatisieren, und das entwickelte Prozessmodell an kommerziellen Referenzsystemen für COBOL und Java zu evaluieren. Im Projekt kommen verschiedene, schon existierende Werkzeuge der Projektpartner zum Einsatz bzw. sie werden im Rahmen des Projektes neu- und weiterentwickelt. Der vorliegende Beitrag zeigt einen exemplarischen Überblick über einige Werkzeuge, für deren Entwicklung pro et con verantwortlich zeichnet.

1 Tool-Architektur und Schnittstellen

Da im Projekt SOAMIG mehrere Partner kooperieren, kommen auch verschiedene Werkzeug-Portfolios zum Einsatz. Eine Herausforderung bestand darin, diese miteinander zu synchronisieren, um von der Synergie der Werkzeuge zu profitieren. Aus diesem Grund wurde eine zentrale Schnittstelle zwischen den Werkzeugen der Partner geschaffen. Diese bedient sich der Extensible Markup Language (XML), um Informationen zwischen den Werkzeugen auszutauschen [4]. Die folgende Abbildung gibt einen Überblick zu den Werkzeugen und deren Schnittstellen:

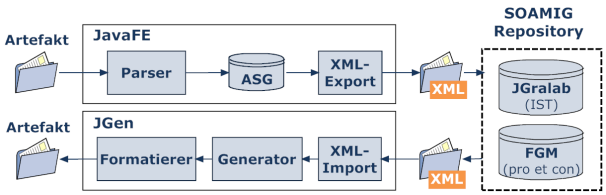


Den Start der Verarbeitung eines Artefakts (COBOL- bzw. Java-Programm) bilden von pro et con entwickelte Frontends für Java (JavaFE) und COBOL (CobolFE). Die hier gewonnenen Informationen werden im SOAMIG-Repository ablegt, welches von dem Reverse-Engineering-Werkzeug Flow Graph Manipulator (FGM) [3] von pro et con und dem Java-Graphenlabor (JGraLab) von IST verwaltet wird. Neben den reinen Programminformationen auf Basis eines Code-Modells werden im Repository auch Geschäftsprozessmodelle und Traceability-Informationen verwaltet. Nach Transformationen über den Repository-internen Darstellungen liefert das SOAMIG-Repository XML-Input für das Unparser-Werkzeug Java-Generator (JGen) von pro et con. JGen generiert Java-Sourcen, welche der für die SOA-Migration definierten Zielarchitektur einerseits und kundenspezifischen Anforderungen wie Wartbarkeit des Zielcodes andererseits entsprechen.

Der Informationsaustausch zwischen den Tools erfolgt über XML-Interfaces. Für diese existieren zwei definierte Schemata. Jedes Schema entspricht dem Aufbau des abstrakten Syntaxgraphen (ASG) des jeweiligen Frontends. Jede XML-Datei entspricht genau einem analysierten Artefakt (COBOL- bzw. Java-Programm). Für die Erstellung der Schemata gab es unterschiedliche Herangehensweisen. Während bei Java das Schema manuell auf Basis des abstrakten Syntaxbaumes (ASG) des JavaFE aufgebaut wurde, kam beim COBOL-Schema ein weiteres Werkzeug von pro et con zum Einsatz. Dieses generiert das Schema sowie die Methoden zur Generierung der XML-Dateien aus einer deskriptiven Beschreibung des ASG des COBOL-Frontends. Diese Technologie wurde in abgewandelter Form auch bei JGen eingesetzt und wird im Abschnitt 3 erläutert.

2 Analyse/Migration/Generierung von Java-Quellen

Eine wesentliche Aufgabe des SOAMIG-Projektes ist die Migration von Java-Legacy-Systemen in eine SOA. Dazu werden unter anderem die Werkzeuge JavaFE und JGen eingesetzt. Diese bilden die Schnittstellen des zentralen Repository zu den importierten und generierten Java-Sourcen (siehe folgende Abbildung).



Der Verarbeitungsprozess eines Java-Programms beginnt mit JavaFE. Dieses ist in der Lage, alle gängigen Formate (java, class, zip, jar) bis zur Sprachversion 1.6 und Unicode feingranular zu verarbeiten. Die Parser-Komponente von JavaFE wurde mit dem firmeneigenen Parsergenerator BTRACC2 (Backtracking Compiler Compiler, Version 2) implementiert [2]. JavaFE verfügt über eine Reihe von Exportschnittstellen, u.a. für Rochade und FGM. Für das Projekt SOAMIG wurde eine zusätzliche XML-Exportschnittstelle implementiert. Die über die Schnittstelle für ein Java-Programm generierten XML-Informationen repräsentieren dessen internen ASG und entsprechen dem Eingabeformat für das zentrale

SOAMIG-Repository.

Die Generierung der Java-Programme nach durchgeführter SOA-Transformation erfolgt mit JGen. JGen besteht aus drei lose gekoppelten Komponenten. Ein eigens für SOAMIG entwickelter Importer überführt die im XML-Format bereitgestellten Informationen in eine interne Repräsentation (XML-Import). Diese wurde mittels einer deskriptiven Beschreibung definiert, aus der automatisch die zur Nutzung notwendige Klassenhierarchie generiert wird. Die zentrale Quelltext-Generierungskomponente von JGen (Generator) erzeugt aus der internen Repräsentation unter Nutzung dieser Klassenhierarchie minimal formatierten Java-Quelltext. Zur nutzerspezifischen Formatierung des generierten Quelltextes kommt nachfolgend ein Werkzeug mit komplexen Einstellungsmöglichkeiten zum Einsatz, welches auf den in der Eclipse-IDE zur Java-Quelltextformatierung verwendeten Klassen aufsetzt (Formatierer).

Damit existiert eine komplette Werkzeugkette vom Java-Quelltext in das zentrale SOAMIG-Repository und wieder zurück.

3 Modellierung der Objekthierarchie und Generierung der Schnittstelle

Objekthierarchien werden aktuell häufig mittels UML (Unified Modeling Language) modelliert. Die konkrete Transformation in eine Implementierungssprache (z.B. C++) ist zumeist an vorgegebene Paradigmen gebunden, welche sich nur bedingt mit bereits bestehenden Werkzeugen vereinbaren lassen. Aus diesem Grund wurde von pro et con ein alternativer Ansatz entwickelt, welcher voll auf die Anforderungen der hauseigenen Werkzeuge angepasst wurde. Dieser soll im Folgenden vorgestellt werden:

Ausgangspunkt ist eine deskriptive, vollständige Beschreibung aller benötigten Objekte. Diese genügt einer definierten Syntax und gestattet die Angabe von Attributen und Verbindungen (Assoziationen) zwischen Objekten. Zum Beispiel definiert

```
/*----------------------------------------
 * Bezeichner mit Qualifikation (A OF B)
 *----------------------------------------*/
CLASS IN_OF_NODE EXTENDS L_VALUE_NODE
   CHILD left TYPE IDENT_NODE
   CHILD right TYPE {IN_OF_NODE,IDENT_NODE}
   ATTRIBUTE symbol sym=0;
END-CLASS
```

eine Klasse für die Qualifikation von Datenobjekten in COBOL. Die vollständige Beschreibung der Objekte für COBOL umfasst insgesamt 234 Klassen. Ein von pro et con entwickeltes Programm `mkobj` generiert daraus eine im COBOL-Frontend benötigte Klassenhierarchie in C++ inkl. der notwendigen Methoden. Dabei werden auch Methoden generiert, welche die Ausgabe der Objekte in XML realisiert. Das erzeugte XML muss einem definierten Schema genügen. Auch dieses Schema wird nicht manuell vorgegeben, sondern ebenfalls aus der deskriptiven Beschreibung generiert. Es wurde so entworfen, dass Werkzeuge, welche diese Informationen verarbeiten, diese unkompliziert importieren können. Wenn diese Weiterverarbeitung in einem C++-Programm geschehen soll, so ist es auch möglich, Methoden zu generieren, die das dem Schema entsprechende XML importieren und die interne Darstellung aufbauen.

Dieser generative Ansatz hat mehrere Vorteile: Die manuelle Arbeit wird reduziert, wodurch die Werkzeugentwicklung wesentlich beschleunigt und weniger fehleranfällig wird (der aus der COBOL-Klassenbeschreibung generierte Code umfasst ca. 55.000 LOC). Änderungen sind einfach zu realisieren, da lediglich die Beschreibung geändert werden muss. Auch Erweiterungen, wie z.B. die Generierung von Objekten und Methoden in anderen Implementierungssprachen (z.B. Perl, Java) sind möglich. Der beschriebene Ansatz wurde bei CobolFE und JGen im SOAMIG-Projekt, aber auch in anderen Werkzeugen von pro et con verwendet und hat sich bewährt.

Literaturverzeichnis

[1] Erdmenger, U.; Kaiser,U.; Loos, A.; Uhlig, D.:
Methoden u. Werkzeuge für die Software Migration.
In: Gimnich, R.; Kaiser, U.; Quante, J.; Winter, A. (Eds.): 10th Workshop Software-Reengineering (WSR 2008), 5.-7. May 2008, Bad Honnef.
Lecture Notes in Informatics, (LNI)-Proceedings, Volume P-126, S. 83-97

[2] Erdmenger, U.: Der Parsergenerator BTRACC2.
11. Workshop Software-Reengineering (WSR 2009), Bad Honnef 4.-6. Mai 2009
In: GI-Softwaretechnik-Trends, Band 29, Heft 2, ISSN 0720-8928, S. 34 und 35

[3] Beier, A.:
Flow Graph Manipulator (FGM) 3.0 - Reverse-Engineering-Tool für komplexe Softwaresysteme.
11. Workshop Software-Reengineering (WSR 2009), Bad Honnef 4.-6. Mai 2009
In: GI-Softwaretechnik-Trends, Band 29, Heft 2, ISSN 0720-8928, S. 39 und 40

[4] Großmann, E.; Strauß, S.; Horn, T.; Riediger, V.:
Abbildung von grUML nach XSD SOAMIG.
Institut für Softwaretechnik, Universität Koblenz-Landau. Arbeitsberichte aus dem Fachbereich Informatik, Nr. 15/2009.