Round-Trip mit CASE-Werkzeugen: (Alp)traum oder Wirklichkeit?

Jörg Niere, Dirk Ohst
Fachgruppe Praktische Informatik
Universität Siegen

Einleitung. Die Model Driven Architecture (MDA) stellt das Modell eines Softwaresystems in den Mittelpunkt der Entwicklung. Eine Schwierigkeit, die bei der Umstellung von bereits existierenden Systemen auf MDA besteht, ist die Gewinnung der Modelle aus dem Quelltext der vorliegenden Software. Insbesondere der Frage nach Werkzeugen, die eine IT-Abteilung bei einfachen gängigen Aufgaben wie der Erstellung der Modelle unterstützt, sind wir nachgegangen und präsentieren in diesem Artikel unsere Ergebnisse.

Für uns überraschend ist die recht große Werkzeugauswahl, die nach Herstellerangaben eine Unterstützung der Modellerstellung aus Quelltext enthalten. Allerdings differieren sie in ihrem Funktionsumfang doch teilweise erheblich. Wir haben uns daher entschlossen, unsere Auswahl auf CASE-Werkzeuge mit Reverse-Engineering Funktionalität zu beschränken, da die Modelle beim MDA Ansatz erweitert werden sollen.

Der Startpunkt unserer Werkzeugauswahl ist die Liste der CASE-Werkzeuge auf der Webseite von Mario Jeckle†. Die Liste enthält 18 Werkzeuge, die angeben eine Reverse-Engineering Funktionalität zu enthalten. Von diesen 18 Werkzeugen waren allerdings nur 8 Werkzeuge in der Lage Quelltext einzulesen, als UML Klassendiagramm darzustellen und anschließend wiederum Quelltext aus dem Modell zu generieren, was wir als minimalen und unerlässlichen Schritt ansehen. Quelltextgeneratoren sind dabei eine Möglichkeit der Transformatoren, die Modelle in plattformabhängige Modelle zu transformieren. Die getesteten Werkzeuge sind im einzelnen:

- IBM Rational Software Modeler 6.0.0 (Buildid: 20041208_1412)
- Magic Draw UML 9.0 Patch 2
- objectiF 5.0.165
- Rapsody 6.0 in J
- Select Component Architect 5.0 (Build 8)
- Together 7.0 (Build 2159.3)
- UML Studio 7.1 (Build 749)
- Visual UML 4.13 (Build 552)

Testszenario. Reverse-Engineering ist ein kognitiver Prozess und kann somit nur zu einem gewissen Teil automatisiert werden. Wir haben uns daher in unserem Testszenario auf ein automatisierbares Szenario, das von einem Werkzeug ohne manuelle Eingriffe durchgeführt werden kann, beschränkt. Die Werkzeuge sollten dabei als Minimalziel in der Lage sein, Quelltext einzulesen und in einem so genannten rudimentären Klassendiagramm darzustellen. Wir haben uns dabei auf Java Quelltext beschränkt. Neben einer korrekten UML Syntax ① sollten in einem rudimentären Klassendiagramm Klassen im Quelltext als Klassen im Diagramm ② dargestellt werden. Attribute mit Basisdatentyp ③ und Methodensignaturen ④ sollten in ihrer zugehörigen Klasse erscheinen. Vererbungsbeziehungen ⑤ einer Klasse zu anderen Klassen (extends) und Schnittstellen (implements) sollten auf die entsprechenden UML Konstrukte abgebildet werden. Attribute im Quelltext, deren Typen andere Klassen sind, die nicht als Datentyp gekennzeichnet sind, sollten als Referenz ⑥ zur entsprechenden Klasse dargestellt werden. Außerdem sollten keine zusätzlichen, unsinnigen Artefakte ⑦ erscheinen.

Zusätzlich zur Erstellung eines rudimentären Klassendiagramms in korrekter UML Notation haben wir die Werkzeuge unter den folgenden Gesichtspunkten untersucht: ⑧ Übereinstimmung des erneut erzeugten Quelltextes ohne Modellveränderungen mit dem originalen Quelltext, ⑨ Erkennung von Zugriffsmethoden gemäß Java-Beans Standard (get/set), ⑩ Erkennung von zu-n Referenzen, die üblicherweise durch Container in Java implementiert werden und ⑪ die Identifizierung von bidirektionalen Assoziationen.

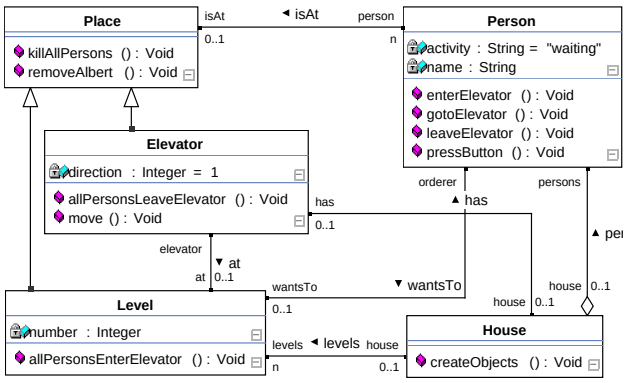

Abbildung 1: Ursprüngliches UML Klassendiagramm

Der Quelltext, den die Werkzeuge als Diagramm darstellen sollen, ist ein Fahrstuhlbeispiel. Der Quelltext wurde dabei von Fujaba, einem UML Entwicklungswerkzeug generiert, was in der Regel ein Reverse Engineering erleichtert. In Abbildung 1 ist das ursprüngliche Klassendiagramm zu sehen. Das Beispiel enthält Klassen, Attribute und Methoden sowie Vererbungen und ohne Ausnahme bidirektionale Beziehungen. Attribute werden bei der Quelltextgenerierung gemäß Java Bean Standard durch get- bzw. set-Methoden gekapselt. Bei der Quelltextgenerierung für zu-n Rollen werden Fujaba eigene Containerklassen verwendet. Bidirektionale Beziehungen zwischen Ob-

jekten werden durch Zugriffsmethoden sichergestellt. Der generierte Quelltext kann von einem Java Compiler unter Berücksichtigung der Bibliotheken kompiliert und in einer Java Virtual Machine ausgeführt werden. Fujaba ist dabei in der Lage den Quelltext als das originale Klassendiagramm darzustellen, weshalb wir es nicht in unsere Auswahl einbezogen haben.

	Minimalziele							**weitere**			
Werkzeug	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪
IBM Soft.Modeler	○	+	+	+	+	-	+	+	-	-	-
Magic Draw UML	○	+	○	+	+	+	+	+	○	-	+
objectiF	○	+	+	+	+	-	+	+	-	-	+
Rapsody	○	+	+	+	+	+	+	-	-	-	-
Sel.Comp.Arch.	○	+	○	○	+	-	-	+	-	-	-
Together	○	+	+	+	+	-	+	+	○	-	-
UML Studio	○	+	+	+	+	-	+	-	-	-	-
Visual UML	○	+	+	+	+	+	+	+	-	-	-
+ erfüllt; - nicht erfüllt; ○ teilweise erfüllt											

Tabelle 1: Werkzeugeigenschaften im Überblick

Ergebnisse. Für uns war es eine Überraschung, dass kein Werkzeug alle Anforderungen erfüllen konnte. Allen gemein ist die Tatsache, dass die Java Datentypen nicht auf die in der UML abgebildet worden sind. Zum Beispiel wird von allen der Datentyp `int` anstatt `Integer` verwendet.

IBM Rational Software Modeler ist in die IDE Eclipse integriert, so dass der Quelltext mit dem Modell und den dargestellten Diagrammen synchron gehalten wird. Bei den Beziehungen im Diagramm handelt es sich um `use`-Beziehungen, unabhängig davon, ob die Klasse ein Attribut besitzt oder der Typ nur als Methoden-Parameter verwendet wird. Ist ein Attribut-Typ nicht bekannt, so fehlt das Attribut insgesamt im Diagramm. Obwohl man manuell weitere Containerklassen dem Werkzeug angeben kann, ignoriert das Werkzeug diese Information bei der Anzeige von Kollektionen, die auf dieser Containerklasse basieren.

Magic Draw UML bietet beim Reverse-Engineering verschiedene Konfigurationsmöglichkeiten. Nicht vorhandene Typen werden in einem Default-Paket angelegt. In den meisten Fällen wurden die Attribute richtig erkannt, lediglich bei der Klasse `Person` fehlten beide Attribute im Diagramm. Das verwundert um so mehr, da die get- und set-Methoden für die Attribute richtig eingefügt wurden und auch wieder im generierten Quelltext vorhanden waren. Die Getter und Setter wurden für Attribute erkannt, nicht jedoch für Kollektionen von Objekten. Alle Getter und Setter werden im Diagramm angezeigt, jedoch hat der Anwender die Möglichkeit einzelne Methoden auszublenden und so die Übersicht zu erhöhen.

objectiF stellt zusätzlich zu den Assoziationen im Diagramm Call-Beziehungen dar, was die Übersichtlichkeit reduziert. Jedoch erkennt es neben Magic Draw als einziges bidirektionale Assoziationen korrekt.

Rapsody bietet bei der Erstellung des Klassendiagramms eine Auswahlmöglichkeit hinsichtlich der hinzuzufügenden Klassen und Beziehungen. Nach der Codegenerierung enthält der erzeugte Quelltext teilweise zusätzliche Attribute, die im ursprünglichen Quelltext nicht vorhanden waren, teilweise fehlen Attribute. Ebenso wird automatisch ein Standard-Konstruktor erzeugt, sofern keiner vorhanden war.

Im von *Select Component Architect* erzeugten Diagramm enthalten die Klassen `Place` und `Person` weder Attribute noch Methoden, obwohl sie im anschließend generierten Quelltext wieder enthalten sind. Des weiteren sind alle Assoziationen als bidirektionale 1-zu-n Beziehungen dargestellt, unabhängig von deren realer Kardinalität. Zusätzlich zu den Assoziationen zwischen den Klassen sind diese Beziehungen auch noch als Attribute im Diagramm visualisiert.

Together ist auch in Eclipse integiert, so dass Modell und Quelltext jederzeit synchron gehalten sind. Together besitzt daher vergleichbare Eigenschaften wie der Software Modeler von IBM. Im Gegensatz hierzu werden in Together Assozisationen nur angelegt, wenn die Klassen entsprechende Referenzen auf andere Klassen besitzen, jedoch fehlen Kardinalitäten und Rollen. Die Getter und Setter werden erkannt, jedoch nicht für Kollektionen von Objekten.

UML Studio stellt alle Assoziationen als Kompositionen dar, an denen die Rollen notiert sind, jedoch fehlen die Kardinalitäten. Bei der Generierung des Quelltextes sortiert UML Studio die Methoden und Attribute um, so dass ein aufwändiger Textvergleich notwendig war. Fälschlicherweise sind einige Attribute im generierten Quelltext doppelt vorhanden.

Das Reverse-Engineering von *Visual UML* ist, genau wie die Quelltextgenerierung, konfigurierbar und das Diagramm besitzt anschließend ein brauchbares Layout. Die Rollen sind an Assoziationen vermerkt, die Kardinalitäten werden jedoch nicht erkannt.

Resümee. Ein Leiter einer IT-Abteilung, der ein Modell seiner Software erzeugen will oder muss, wird durch die vorhandenen Werkzeuge nur mäßig unterstützt. Leider haben die Verfahren und Ansätze, die in den letzten Jahren auf Konferenzen und Workshops vorgestellt worden sind, noch nicht Einzug in (kommerzielle) Werkzeuge gefunden. Einen ersten Schritt für die Unterstützung von MDA Projekten ist dabei sicherlich die Verknüpfung von Quelltext und Modell, wie sie von einigen Werkzeugen angeboten wird.

Danksagung. Unser Dank gilt den Menschen an den Service-Hotlines der einzelnen Produktherstellern, die es uns mit ihren freundlichen Auskünften ermöglicht haben, entsprechende Seriennummern zu erhalten, die Werkzeuge zu installieren oder uns durch die teilweise abenteuerliche Bedienung geführt haben.