

Generierung eines Systemabhängigkeitsgraphen mittels Graphtransformation

Tim Heuer, Rafael Weiß

Institute for Computer Science and Business Information Systems
Universität Duisburg-Essen
Universitätsstr. 2
45141 Essen
tim.heuer@stud.uni-essen.de
rafael.weiss@uni-essen.de

Abstract: Elementare Bestandteile des Software Engineerings sind die Analyse und die Darstellung von Programmen. Ein wichtiges Ziel ist die Bewältigung der Komplexität des Programms. Dazu können Methoden des Reengineerings, wie Slicing oder Refaktorisierung, sehr gut eingesetzt werden. Die Meisten dieser Methoden setzen dabei eine Darstellung des Programms in einem Programmabhängigkeitsgraphen (PDG) voraus. Diese Darstellungsform unterstützt sowohl die Darstellung von Kontroll-, wie auch von Datenabhängigkeiten. Eine Erweiterung des PDG stellt der Systemabhängigkeitsgraph (SDG) dar. Mittels SDGs lassen sich auch prozedurale und objektorientierte Systeme darstellen. In den letzten Jahren konzentrierten sich die Entwicklungen vermehrt darauf, objektorientierte Paradigmen in diesen Abhängigkeitsgraphen zu visualisieren, um sie für objektorientierte Programmiersprachen zugänglicher zu machen.

Mit Hilfe der Graphtransformation wird eine systematische und strukturierte Analyse von Graphen, oder wie hier der Programmstruktur, unterstützt. Daher wird die Graphtransformation als Methodik zum Erzeugen eines SDG verwendet.

1 Fragestellung

Diese Bachelorarbeit befasst sich mit der zentralen Frage: „Sind SDGs für die Programmiersprache JAVA mittels Graphtransformation generierbar und wie verhält sich der Aufwand gegenüber dem Nutzen dieses Verfahrens?“. Besonderes Augenmerk liegt auf der Umsetzung „objektorientierter Strukturen“ wie Vererbung und Instanziierungen in eine ansprechende und gut weiterzuverarbeitende Darstellungsform. Eine praktische Umsetzung war gefordert.

2 These

Die grundlegende These dieser Arbeit lautet, dass durch Graphtransformationen eine automatische Erzeugung von Systemabhängigkeitsgraphen möglich ist, wobei der resultierende Graph möglichst exakt den ursprünglichen Java Code widerspiegelt. Zur praktischen Evaluierung dieser These ist es notwendig eine Komponente zu entwickeln, die aus einem Java Programm einen SDG erstellt. Die Technik der Graphtransformation wird hier gewählt, da diese auf einer formal fundierten Grundlage beruht und die konsistente Erstellung und Transformation von Strukturen wie dem SDG unterstützt.

3 Vorgehensweise

Ziel dieser Arbeit ist die automatisierte Erstellung eines Abhängigkeitsgraphen mit Hilfe von Graphtransformationen. Dies beruht darauf, dass uns die geeigneten Mittel zur Verfügung stehen, welche aus JAVA-Quellcode einen attributierten, gerichteten abstrakten Syntaxbaum generieren, der sich im Weiteren mit Graphtransmutationsregeln modifizieren lässt.

Die technische Grundlage für das Projekt wird durch das Werkzeug „Java2GGX“ geschaffen, welches auf der Abschlussarbeit von André Wiesmann beruht, und ein Java-Programm in einen Graphen im GGX-Format übersetzt [Wi04]. Die Erstellung und Ausführung der Graphtransmutationsregeln wird mit dem Werkzeug „AGG“ von der Technischen Universität Berlin, sowie weiteren, an der Universität Duisburg-Essen entwickelten, Werkzeugen durchgeführt.

Die Erstellung des SDG ist angelehnt an die Prinzipien, die Walkinshaw in seiner Arbeit herausgestellt hat [Wa03]. Dabei kommen Konzepte der Objektorientierung wie Vererbung und Schnittstellen zum Tragen.

Die Implementierung der Regeln besteht aus mehreren Schritten. Zuerst müssen Regeln geschaffen werden, die das Zusammensetzen einiger Java-Ausdrücke übernehmen, welche im AST in einer Baumstruktur vorliegen. Dieser Teil des Projektes wurde von Rafael Weiß realisiert. Anschließend folgt die Implementierung weiterer Regeln, die aus den zusammengesetzten Ausdrücken die SDG-Knoten und -Kanten erzeugen. Die Regeln, welche die SDG-Darstellung umsetzen, wurden von Tim Heuer verwirklicht.

4 Ergebnisse

Zu Beginn galt es herauszufinden, ob sich eine SDG-Erstellung mittels Graphtransformation überhaupt realisieren lässt. Diese elementare Frage kann bejaht werden. Durch Graphtransformation lässt sich ein SDG in sehr detaillierter Form erzeugen, auch wenn man dafür teilweise von den theoretischen Grundlagen der SDG-Erstellung durch die Graphtransformation abstrahieren muss. Voraussetzung dafür ist jedoch, dass Werkzeuge vorliegen, die aus Java-Quellcode einen abstrakten Syntaxbaum generieren können, der um alle notwendigen Informationen angereichert ist. Zudem müssen ineinander greifende Werkzeuge zum Bearbeiten des generierten Graphen zur Verfügung stehen.

Bei der Implementierung fällt schnell die hohe Anzahl an Regeln auf, welche nötig sind, um komplexe Sprachen wie Java in ihrem vollen Umfang adäquat umzusetzen. Wie sich herausstellte, können fast alle Konstrukte, welche die Programmiersprache Java aufzuweisen hat, in einen SDG übertragen werden. Dies gilt sowohl für normale Anweisungen, Deklarationen, Schleifen und Bedingungen, als auch für Interfaces, Vererbung oder abstrakte Klassen. Ausnahmen dafür sind bei dem derzeitigen Stand der Werkzeuge noch vorhanden, wie Rekursionen durch den „this“-Operator oder fehlende Berücksichtigung der neuen Konstrukte der Version 1.5 von Java. Dies betrifft jedoch nur wenige Sonderfälle, die auf die technische Realisierung der Werkzeuge und nicht auf den konzeptionellen Ansatz oder die Regelbasis zurückzuführen sind. Somit kann auch die zweite Frage dieser Arbeit, ob Objektorientierung in SDGs gut realisierbar ist, bejaht werden.

Ein anderer Punkt, der betrachtet wurde, ist der Performance-Aspekt. Positiv zu bemerken ist hierbei, dass durch die Möglichkeit der automatisierten Graphtransformation durch die eingesetzten Werkzeuge die Erstellung eines SDGs in annehmbarer Zeit möglich ist. Dies ließe sich an vielen Stellen sogar noch optimieren, wenn die Werkzeuge einige Erweiterungen und Fehlerbehebungen erfahren würden.

Die zuvor genannten Optimierungsvorschläge sollen damit als Anreiz dienen, Graphtransformation und SDGs auch für andere Anwendungsfelder einzusetzen. Zwar ist derzeit die Performance nicht an allen Stellen als optimal anzusehen, aber dafür ist der erzeugte SDG anschließend durch seine detaillierte Strukturierung vielseitig einsetzbar. Zusätzlich stehen durch Graphtransformationen viele Mittel für eine Realisierung in den zuvor genannten Bereichen des Reengineering zur Verfügung. Dies sollte somit zu einer enormen Zeitersparnis gegenüber Einzelverfahren führen, die direkt auf dem Quellcode aufsetzen, was jedoch nicht Teil dieser Studie ist.

Insgesamt kann man resümieren, dass SDG-Erstellung mittels Graphtransformation nicht nur möglich ist, sondern durch die Automatisierung des Verfahrens, den vielseitigen Einsatzzwecken und den inzwischen sehr ausgereiften Werkzeugen für Graphtransformationen eine echte Alternative zu quellcode-basierten Verfahren in Bereichen des Reengineering (sei es des Slicings [HRB90], des Refactorings oder sonstiger Verfahren, die auf Quellcode und seinen Abhängigkeiten sowie Abläufen basieren) bietet.

5 Bisherige Tätigkeiten

Sowohl die theoretischen Vorüberlegungen, als auch der Großteil der konkreten Implementierung sind inzwischen abgeschlossen. Den letzten Teil dieser Bachelorarbeit stellen nun die bisher fehlende Darstellung von abstrakten Klassen, Verbesserungen an der Automatisierbarkeit, sowie der Abschluss der Ausarbeitung selbst dar.

6 Über die Autoren

Tim Heuer und Rafael Weiß sind Studenten an der Universität Duisburg-Essen. Sie studieren „Systems-Engineering“ mit der Vertiefungsrichtung „Software“. Das vorliegende Positionspapier stellt einen Auszug aus ihrer Bachelorarbeit zum Thema „Generierung eines Systemabhängigkeitsgraphen mittels Graphtransformation“ dar. Die Arbeit wird am Lehrstuhl „Spezifikation von Softwaresystemen“ von Prof. Michael Goedicke und Carsten Köllmann betreut.

Literaturverzeichnis

- [HRB90] Horwitz, S; Reps, T.; Binkley, D.: Interprocedural slicing using dependence graphs, ACM Transaction on Programming Languages and Systems 12, 1990; S.26-60
- [Wa03] Walkinshaw, N.: The Java System Dependence Graph, University of Strathclyde, 2003
- [Wi04] Wiesmann, A.: Über die Abbildung von Java Programmen auf Graph-Strukturen, Diplomarbeit, Universität Duisburg-Essen, 2004