Ingo Stürmer: Systematic Testing of Code Generation Tools – A Test-suite oriented Approach for Safeguarding Model-based Code Generation

Promotion: Technische Universität Berlin, Fakultät für Elektrotechnik und Informatik

Erstgutachter: Prof. Dr. Peter Pepper (Technische Universität Berlin)

Zweitgutachter: Prof. Dr. Andy Schürr (Technische Universität Darmstadt)

Datum der Prüfung: 16.02.2006

Veröffentlichung: Ingo Stürmer, Systematic Testing of Code Generation Tools – A Test-suite oriented Approach for Safeguarding Model-based Code Generation, PRO BUSINESS, Berlin, 2006.

Kurzfassung:

Der Einsatz modellbasierter Entwicklungsmethoden für die Entwicklung eingebetteter Software ist für die Industrie von hoher Bedeutung. Der *Modellbasierten Codegenerierung* kommt dabei eine Schlüsselrolle zu, da sie deutliche Effizienzgewinne gegenüber der traditionellen Programmierung ermöglicht. Voraussetzung hierfür ist aber, dass bei der automatischen Übersetzung bereits getesteter Modelle keine Fehler in die Software eingebracht werden. Häufige Fehlerquellen können hierbei Optimierungstechniken sein.

Es mangelt derzeitig an Testmethoden, die solche Optimierungen umfassend und weitgehend automatisiert prüfen. Diese Lücke soll das hier vorgestellte Testverfahren schließen; es bietet und integriert Lösungsmöglichkeiten für vier zentrale Problemfelder beim Test von Codegeneratoren und deren Optimierungen:

(1) Eine Vorgehensweise die es ermöglicht **Testfälle**, also zu übersetzende Modelle, zu entwerfen, die mit hoher Wahrscheinlichkeit einen Fehler in der Implementierung der Optimierung aufdecken.

(2) Eine Methodik zur Erzeugung von **Testdaten**, die es erlauben die Testmodelle und den daraus generierten Code auf funktionale Gleichheit zu überprüfen.

(3) Ein **Testauswerteverfahren**, das die Testergebnisse auf Gültigkeit überprüft. Dem Vergleich der dynamischen Eigenschaften bei der Modellsimulation und der Codeausführung im Hinblick auf die benötigten Testdaten kommt dabei eine zentrale Rolle zu.

(4) Eine **Testumgebung**, die es ermöglicht, die verschiedenen Testwerkzeuge zu koppeln und den Testablauf zu verwalten, zu steuern und zu dokumentieren. Der Aufbau der Testumgebung sollte so gewählt sein, dass mögliche Fehlerquellen in der Werkzeugkette aufgedeckt werden können.

Die Verwendung einer Spezifikation der Optimierungen, die einerseits den Übersetzungsprozess der Modelle abstrakt und formal beschreibt und andererseits Implementierungsdetails für den Testfallentwurf berücksichtigt, verspricht den Entwurf von *wirkungsvollen* Testfällen. Da der Übersetzungsprozess vom Modell zum Code im Wesentlichen auf der Transformation von Graphen beruht, werden **Graph Transformationsregeln** als Basis für den Testfallentwurf verwendet. Die automatische Erzeugung der Testmodelle erfolgt über einen hierfür entwickelten Modellgenerator *ModeSSa* (Test **Mode**l Generator for **S**imulink and **Sta**teflow). Die benötigten Testdaten werden über einen kombinierten **Strukturtest** auf Modell- und Codeebene gewonnen. Diese Testdaten erlauben eine hinreichende Überdeckung funktionaler Eigenschaften des Modells und Codes. Die Auswertung der Testergebnisse wird über **Signalvergleichsverfahren** ermöglicht, die das dynamische Verhalten

der Modellsimulation und der Codeausführung berücksichtigen. Die Realisierung des Testverfahrens erfolgt über eine **modulare Testsuite** als zentraler Bestandteil einer automatisierten Testumgebung.

Die vorliegende Arbeit zeigt, dass ein erfolgreiches Durchlaufen der Testsuite eine hinreichende Absicherung der Optimierungen des Codegenerators ermöglicht.

Inhalt: