

Automatisierte Überprüfung von Modellierungsrichtlinien für Matlab/Simulink Modelle

Kathrin Bröker

kathyb@upb.de

Organisatorischer Rahmen

Die hier vorgestellte Arbeit entsteht im Rahmen meiner 3 Monate dauernden Studienarbeit an der Universität Paderborn. Die Arbeit wird in Kooperation zwischen dem Software Quality Lab (s-Lab) der Universität Paderborn und der Firma dSPACE erstellt.

Auf das Thema meiner Studienarbeit stieß ich im Zusammenhang mit einer Seminararbeit im Bereich Softwaretechnik. Dort habe ich eine Ausarbeitung zum Thema „Modellierungsrichtlinien in automotiven Softwaresystemen“ erstellt und bin auf die Problematik fehlender Guidelinechecker in diesem Bereich gestoßen.

Professor Wilhelm Schäfer und mein Betreuer haben mir dann das Projekt zur „Automatisierten Überprüfung von Modellierungsrichtlinien für Matlab/Simulink Modelle“ vorgestellt, welches mich sehr interessiert. So entschloss ich mich zu diesem Thema für meine Studienarbeit, dessen thematische Grundlagen ich zuvor schon in meiner Seminararbeit erfasst hatte. So fiel mir der Einstieg ins Thema nicht besonders schwer.

Thema der Arbeit

Die Anzahl an Softwaresystemen in Automobilen hat in den letzten Jahren enorm zugenommen. Die zunehmende Vernetzung der Systeme führt zu einer höheren Komplexität und Fehleranfälligkeit. Die Form der Modellbasierten Softwareentwicklung unterstützt die Entwickler automotiver Softwaresysteme, so dass diese die Software schneller und kostengünstiger entwickeln können.

Zur Modellierung und zur Modellanalyse werden für automotive Softwaresysteme sogenannte Blockschaltdiagramme verwendet. Modelle haben einen deutlich höheren Abstraktionsgrad, im Gegensatz zum implementierten Sourcecode und tragen dadurch erheblich zum besseren Verständnis bei. Nach der Modellierung lässt sich dann automatisch Sourcecode generieren.

Eines der am häufigsten verwendeten Werkzeuge zur Modellierung von Blockschaltdiagrammen ist Matlab/Simulink der Firma Mathworks. Aus diesen Matlab/Simulink Diagrammen wird dann mittels Codegeneratoren der Sourcecode automatisch erzeugt. Da der Code in der automotiven Softwareentwicklung aber vielen Restriktionen unterliegt, wie zum Beispiel geringe Speicherkapazität, muss der generierte Code sehr effizient sein.

Die Firma dSPACE hat hier mit TargetLink einen Codegenerator auf Basis von Mat-

lab/Simulink entwickelt. Dieser ist in der Lage sehr effizienten Sourcecode gezielt für bestimmte Plattformen zu erzeugen.

MATLAB/Simulink bietet einige Modellierungsmöglichkeiten für Blockschaltdiagramme, die für die Codegenerierung, z.B. mit TargetLink, nicht geeignet sind. Diese nicht geeigneten Modellierungsmöglichkeiten würden die Effizienz des generierten Codes stark beeinflussen, in einigen Fällen wäre keine Code-Generierung möglich. Aus diesem Grund gibt es für TargetLink einen so genannten Guidelinekatalog. In diesem Katalog sind Richtlinien spezifiziert, gegen die nicht verstoßen werden sollte, da sonst eine saubere Codegenerierung nicht garantiert werden kann.

Dieser Guidelinekatalog umfasst allerdings inzwischen über hunderdfünfzig Seiten und kaum ein Entwickler will all diese Regeln lernen bzw. ständig seine Modelle von Hand auf die Einhaltung dieser Regeln überprüfen. Deshalb wünscht man sich eine automatische Überprüfung der Regeln.

Ziel der Arbeit ist es, zu überprüfen, in wie weit das Prinzip des Pattern Matching aus dem Java Reengineering Umfeld, für die Entwicklung eines Guidelinecheckers für Matlab/Simulink Diagramme geeignet ist. Verwendet wird hierzu das Prinzip des Pattern Matching, wie es in der Dissertation von Jörg Niere vorgestellt wird¹. Dieser Ansatz wurde entwickelt, um große Softwaresysteme besser Reverse engineerieren zu können. Im Gegensatz zu anderen Ansätzen gibt es hier kein Problem mit unterschiedlichen Implementierungsvarianten von Design Patterns. Der verwendete Algorithmus nutzt Domänen- und Kontextwissen des Reverse Engineerings und einer speziellen Datenstruktur. Genauer gesagt eine spezielle Form eines annotierten abstrakten Syntaxgraphen. Dieser Algorithmus wurde im UML-CaseTool Fujaba der Universität Paderborn (Fachgebiet Softwaretechnik) implementiert und soll hier zur Analyse von Matlab/Simulink Diagrammen verwendet werden.

Vorgehen

Um die Diagramme zu überprüfen erfolgt zunächst die Spezifikation der Richtlinienverletzungen in einer abstrakten Syntax. Diese basiert auf einem Metamodell für Matlab/Simulink Diagramme. Auf Grundlage dieses, zurzeit noch eingeschränkten Metamodells wurde bereits ein Patternkatalog mit einigen wenigen Regelverletzungen aus den dSPACE Styleguides spezifiziert.

In der Arbeit werden weitere Regelverletzungen aus den dSPACE Styleguides auf ihre Umsetzbarkeit als Pattern überprüft und wenn möglich ebenfalls in der abstrakten Syntax als Pattern spezifiziert.

Hierzu muss das Metamodell für die Matlab/Simulink Modelle um weitere notwendige Konstrukte erweitert werden.

Die Spezifikation der Regeln erfolgt zunächst auf der formalen Ebene und domainenspezifisch mit einem Metamodell für Matlab/Simulink. Anschließend erfolgt eine Integration in Fujaba. Die zur Spezifikation der Regeln notwendigen Erweiterungen des Metamodells werden implementiert, um anschließend Matlab/Simulink Modelle auf die Einhaltung der Styleguides überprüfen zu können.

¹Niere, Jörg. Inkrementelle Entwurfsmustererkennung, Paderborn, Univ., Fakultät für Elektrotechnik, Informatik und Mathematik, 2003

Da es über hundert Regeln in den dSPACE Styleguides gibt, die nicht alle im Rahmen dieser Studienarbeit untersucht werden können, werden einige typische Regeln exemplarisch untersucht, um herauszufinden in wie weit die vorgestellte Methode in der Praxis umsetzbar und sinnvoll ist.

In einem weiteren Teil der Arbeit soll die Effizienz des verwendeten Algorithmus untersucht und dieser gegebenenfalls verbessert werden. Da der Prototyp einen Algorithmus zur Patternerkennung verwendet, der auch, wie schon erwähnt, in anderen Zusammenhängen eingesetzt wird.

Fazit

Da meine Arbeit zur Zeit noch nicht abgeschlossen ist, kann ich kein Fazit über die gesamte Arbeit ziehen. Dennoch kann ich jetzt schon sagen, dass ich die Kombination aus einem theoretischen und einem praktischen Teil in meiner Studienarbeit sehr gut finde. So ist die Arbeit für mich selbst sehr abwechslungsreich. Des weiteren kann man im praktischen Teil viele Erfahrungen für den späteren Beruf sammeln, die man sonst in einem doch eher theoretischen Studium nicht sammeln würde.