Metriken und Regeln für eine durchgängige und modellbasierte SW-Entwicklung im Automobilbereich

Martin Mutz

TU Braunschweig, Institut für Programmierung und Reaktive Systeme
Mühlenpfordtstr. 23, D-38106 Braunschweig
mmutz@tu-bs.de

Abstract: Im Folgenden wird ein prototypisches Werkzeug zur Überprüfung der Einhaltung von Modellierungsregeln vorgestellt. Der Prototyp wurde im Rahmen eines Automotive-Projekts entwickelt und eingesetzt. Mit Hilfe des Programms können Modellierungsregeln in Java und OCL definiert werden, um Inkonsistenzen, Inkompatibilitäten sowie Designfehler in zustandsbasierten Modellen zu analysieren. Anschließend kann die Qualität der Zustandsautomaten durch Metriken bewertet werden.

1 Einleitung

Die Entwicklung von Software-Funktionen im Automobilbereich wird derzeit mit Hilfe von CASE-Werkzeugen, wie Ascet-SD und Matlab/Simulink/Stateflow vorgenommen. Mit derartigen Werkzeugen kann die Steuerungssoftware auf einer sehr detaillierten Modellebene entwickelt und simuliert werden. Darüber hinaus ist es möglich Code für unterschiedliche Targets zu generieren. In den letzten Jahren wurden die Entwicklungsmethoden im Automobilbereich durch objektorientierte Konzepte ergänzt [vdBBRS01, MHGK03]. Mit Hilfe von objektorientierten Methoden und dazugehörigen Werkzeugen können Spezifikationen bereits in frühen Phasen modelliert, simuliert und getestet werden (ausführbare Lastenhefte). In der Praxis ist der Einsatz unterschiedlicher Werkzeuge häufig problematisch. Durch proprietäre Datenstrukturen, fehlende Schnittstellen und Nichteinhaltung von Standardnotationen kann eine durchgängige Modellentwicklung bzw. ein Modellaustausch in den meisten Fällen nicht gewährleistet werden.
In diesem Papier wird das Werkzeug *Regel Checker*, welches im Rahmen des STEP-X Projektes [HHKM03, MHGK03] entwickelt und eingesetzt wurde, vorgestellt. Mit diesem Programm werden zustandsbasierte Modelle (Statecharts) unterschiedlicher CASE Tools durch Modellierungsregeln analysiert. Auf diese Weise ist es möglich die oben genannten Probleme zu erkennen und zu beheben. Darüberhinaus kann die Softwarequalität durch Metriken bewertet werden. Das Papier ist wie folgt gegliedert. Nach der Einleitung werden im zweiten Kapitel Regeln und Metriken klassifiziert. In Kapitel 3 wird ein Überblick über den Überprüfungsmechanismus des Regel Checkers gegeben. Abschließend erfolgt eine kurze Zusammenfassung.

2 Klassifikation von Regeln und Metriken

In den letzten Jahren wurde die Bedeutung von Modellierungsrichtlinien [FO00] und Metriken [Ra01] im Automobilbereich erkannt. Die Implementierung derartiger Modellierungsvorschriften in Form von Regeln wird jedoch nur von wenigen Modellierungswerkzeugen unterstützt. Häufig sind diese Regeln unvollständig und statisch, so dass eine Selektion und Anpassung der vorgegebenen Regeln nicht möglich ist. Daher wurden in den letzten Jahren spezielle Werkzeuge wie z.B. *mint* [GT04] entwickelt, mit denen eine Modellüberprüfung komfortabel durchgeführt werden kann. Solche Überprüfungswerkzeuge sind generell nur auf bestimmte Werkzeugtypen begrenzt. Eine Aussage über die Durchgängigkeit eines Entwicklungsprozesses oder die Kompatibilität zu weiteren Modellierungswerkzeugen kann somit nicht getroffen werden.
Mit dem Regel Checker kann eine Vielzahl von unterschiedlichen Regeln und Metriken in Form von Java-Code und OCL (Object Constraint Language) werkzeugneutral definiert und überprüft werden. Zur besseren Übersicht wurden die Regeln und Metriken in mehrere Kategorien gegliedert. Die wichtigsten werden im Folgenden vorgestellt:

- **Konsistenzregeln:** Überprüfen, ob gleiche Modellierungskonstrukte in unterschiedlichen Diagrammen (Klassen und Statecharts) konsistent sind.
- **Kompatibilitätsregeln:** Überprüfen, ob die Modellkonstrukte eines Zustandsautomaten zwischen zwei Werkzeugen kompatibel sind.
- **Konformitätsregeln:** Überprüfen, ob das Statechart konform zur UML-Spezifikation [OM03] ist.
- **Modellierungsregeln:** Überprüfen das Modell nach weiteren Regelverletzungen wie z.B. die fehlerhafte Verwendung von Modellelementen.

Im Gegensatz zu Modellierungsregeln, mit denen bestimmte Modellierungskriterien und Bedingungen definiert werden können, werden Metriken zur Modellbewertung herangezogen. Für die Metriken wurde folgende Kategorisierung vorgenommen:

- **Zustandsmetriken:** Bewerten das Modell nach strukturellen Gesichtspunkten. Dazu gehören unter anderem die Anzahl und Verteilung der (Pseudo-)Zustände, die Länge der Beschriftungen und die Hierarchietiefe.
- **Transitionsmetriken:** Bewerten die Komplexität eines Statecharts, indem die Transitionsübergänge zwischen den Zuständen analysiert werden. Dabei wird vorwiegend zwischen ein- und ausgehenden Transitionen unterschieden und Transitionsverläufe (z.B. interlevel- oder self-loop Transitionen) bestimmt.
- **Layoutmetriken:** Bewerten die Übersichtlichkeit und das Verständnis des zustandsbasierten Modells, indem Informationen zu Positionierungen, Skalierungen und der symmetrischen Anordnung von Modellelementen analysiert werden.

3 Analyse und Bewertung der Modelle

Die Hauptaufgaben des Regel Checkers bestehen in der automatischen Überprüfung der Einhaltung von Regeln sowie in der Bewertung des Modells durch Metriken. In Abbildung 1 ist das Analysekonzept des Regel Checkers vereinfacht dargestellt. Das Einlesen eines oder mehrerer Statecharts aus einem Werkzeug ist der erste Schritt der Analyse. Zurzeit können zustandsbasierte Modelle aus den drei Werkzeugen *Rhapsody* von I-Logix, *Stateflow* von MathWorks und *Ascet-SD* von ETAS eingelesen werden. Eine Erweiterung um weitere Werkzeuge ist durch die flexible Datenstruktur des Regel Checkers problemlos möglich. Im zweiten Schritt werden in Abhängigkeit des geladenen Modells die dazugehörigen Regeln automatisch eingelesen. Der Anwender kann anschließend die nicht relevanten Regeln deaktivieren bzw. weitere Regeln für besondere Aufgaben aktivieren. Um die große Anzahl von Regeln und Metriken übersichtlich zu halten, sind diese, je nach Aufgabenbereich, in mehrere Kategorien eingeteilt (vgl. Kapitel 2).

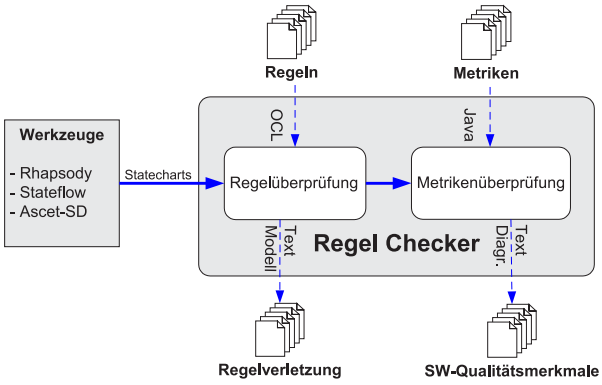

Abbildung 1: Das Analysekonzept des Regel Checkers

Als nächstes werden die Statecharts im Modul *Regelüberprüfung* analysiert. Um die identifizierten Regelbeanstandungen zu einem späteren Zeitpunkt aussagekräftig darstellen zu können, wurde ein Nachrichtensystem entwickelt, das die Aufbereitung der Ergebnisse ermöglicht. Da nicht jede Beanstandung gleichbedeutend ist, muss bei der Auswertung der Ergebnisse differenziert werden. Zu diesem Zweck existiert eine Klassifizierung nach Schweregrad. Hierbei wird eine Beanstandung als Vorschlag, Warnung oder Fehler gekennzeichnet. Vorschläge sind Nachrichtentypen, die einen Hinweis bezüglich einer möglichen Verbesserung enthalten (z.B. darf die Länge der Zustandsnamen einen bestimmten Bereich nicht unter- bzw. überschreiten). Warnungen sind Nachrichten, die auf mögliche Inkonsistenzen hinweisen (z.B. dürfen zwei oder mehrere Zustände mit gleichem Namen nicht existieren). Fehlermeldungen werden generiert, wenn ein Statechart fehlerhafte Eigenschaften aufweist (z.B. muss eine Transition einen Endzustand betreten). In Abbildung 2 sind auf der linken Seite die unterschiedlichen Fehlermeldungen aufgelis-

tet. Die rechte Seite zeigt das Statechart als Baumstruktur, wobei die Quelle der momentan selektierten Fehlernachricht im Baum als Pfeil dargestellt wird. Im unteren Fensterausschnitt werden die Programm- und Benutzeraktivitäten protokolliert.

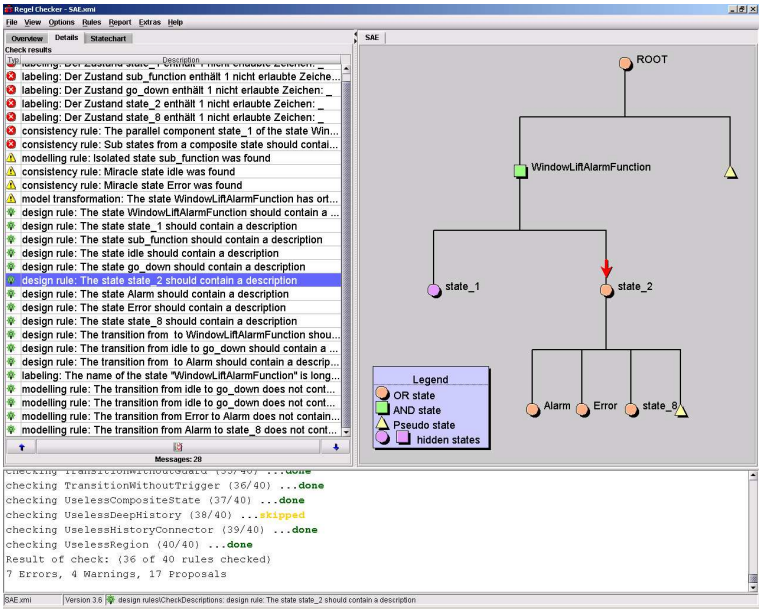

Abbildung 2: Ein Analyseergebnis im Regel Checker

Wird das Statechart nach der Regelüberprüfung so modifiziert, dass keine Beanstandungen vom Regel Checker erzeugt werden, ist das Modell als korrekt einzustufen. Nach der Regelüberprüfung kann im letzten Schritt die *Metrikenüberprüfung* (vgl. Abbildung 1) gestartet werden. Die beiden Überprüfungsarten können grundsätzlich unabhängig voneinander durchgeführt werden. Aus methodischem Gesichtspunkt sollte die Modellbewertung grundsätzlich nach der Regelüberprüfung durchgeführt werden, da eine quantitative Bewertung nur bei korrekten Modellen sinnvoll ist. Wie bereits bei der Regelauswahl können Metriken je nach gewünschtem Detaillierungsgrad ausgewählt werden. Nach der Metrikenüberprüfung wird das Ergebnis entweder in textueller Form oder als Balkendiagramm dargestellt.
Am Beispiel der *Zustandsmetriken* aus Kapitel 2 sollen einige Analyseergebnisse erläutert werden. Zunächst wird die Anzahl aller Zustände im Projekt ermittelt, um den Umfang des Projekts einzuschätzen. Dabei werden Statechart-Modelle bis 500 Zustände als klein, bis 2000 als umfangreich und darüber hinaus als groß klassifiziert. Des Weiteren wird das Minimum, Maximum sowie der Mittelwert der Zustände berechnet, um die ungefähre Verteilung der Funktionskomplexität zu ermitteln. Ein weiteres Merkmal bei der Zustandsanalyse ist die hierarchische Struktur von Statecharts. Dabei wird die Tiefe und Breite aller Statecharts ermittelt und ins Verhältnis gesetzt. Dadurch können unterschiedliche

Aussagen getroffen werden. Beispielsweise sind tiefe (viele Ebenen) und schmale (wenig Zustände pro Ebene) Statecharts genauso unübersichtlich, wie flache aber sehr breite Statecharts. Mit Hilfe des Regel Checkers wurde in einer Fallstudie aus dem Automobilbereich [HHKM03] die Steuerungssoftware für den *elektrischen Fensterheber* analysiert. Aufgrund mehrerer Regelverletzungen und nicht befriedigender Qualitätsmerkmale wurde dieses Teilsystem neu modelliert. Durch zusätzliche Standardisierungsmaßnahmen und klardefinierte Modellierungskonstrukte konnte die Lesbarkeit verbessert und die Durchgängigkeit bei der Modellierung gewährleistet werden.

4 Zusammenfassung

Das vorgestellte Verfahren leistet einen großen Beitrag zur Verbesserung des modellbasierten und durchgängigen Entwicklungsprozesses im Automobilbereich, indem Werkzeugkompatibilitäten, Konsistenzsicherungen und weitere Modellierungsvorschriften eingehalten werden müssen. Des Weiteren können Qualitätsmerkmale der Modelle durch objektorientierte Metriken quantitativ analysiert und bewertet werden. Das Ergebnis der beiden Überprüfungsverfahren wird textuell sowie grafisch in Form von abstrahierten Modellen und Balkendiagrammen repräsentiert. Im STEP-X Projekt wurde das Verfahren anhand einer Fallstudie aus dem Kfz-Komfortbereich erfolgreich erprobt. Die Analyseergebnisse trugen zur generellen Verbesserung des Modellierungsstils bei. Dadurch konnte die Lesbarkeit der Modelle erhöht werden, was zu besserem Verständnis und schnellerer Einarbeitung in das umfangreiche Modell führte. Außerdem konnte die manuelle Umsetzung der UML-Statecharts (Grobentwurf) in Stateflow bzw. Ascet-SD (Feinentwurf) erheblich vereinfacht werden.

Literatur

[FO00] FORD: Structured Analysis & Design Using Matlab/Simulink/Stateflow. *Smart-Vehicle Challenge Problems*. 2000.

[GT04] Gilhead, P. und Tarragon, R. Style Matters. www.ricardo.com/mint. 2004.

[HHKM03] Harms, M., Horstmann, M., Krömke, C., und Mutz, M.: STEP-X: Strukturierter Entwicklungsprozess für eingebettete Systeme im Automobilbereich. *Automotive Electronics 1/03*. S. 46–49. 2003.

[MHGK03] Mutz, M., Huhn, M., Goltz, U., und Krömke, C.: Model Based System Development in Automotive. In: *SAE World Congress 2003*. 2003.

[OM03] OMG. OMG Unified Modeling Language Specification. 2003. Version 1.5.

[Ra01] Rau, A.: Einsatz von Metriken bei der modellbasierten Software-Entwicklung. *15. Symposium in Paderborn*. 2001.

[vdBBRS01] von der Beeck, M., Braun, P., Rappl, M., und Schröder, C.: Modellbasierte Softwareentwicklung für automobilspezifische Steuergerätenetzwerke. *VDI Kongress: Elektrik in KFZ, VDI Berichte 1646*. S. 293 ff. 2001.