Werkzeugübergreifende Konsistenzsicherung von Artefakten bei der Entwicklung softwarebasierter Systeme im Automobil

Tibor Farkas[1], Andreas Leicher[2], Harald Röbig[2]
Marc Born[1], Torsten Klein[2], Justyna Zander-Nowicka[1]

[1] Fraunhofer FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin

[2] Carmeq GmbH
Carnotstr. 4
10587 Berlin

{tibor.farkas, marc.born, zander-nowicka}@fokus.fraunhofer.de
{andreas.leicher, harald.roebig, torsten.klein}@carmeq.com

Abstract: Die Entwicklung von eingebetteten Systemen im Automobilbereich zeichnet sich zunehmend durch steigende Komplexität der zu entwickelnden Fahrzeugfunktionen aus. Die Handhabung dieser Komplexität wird über entsprechend mächtige Werkzeuge, welche die verschiedenen Aktivitäten des Softwareentwicklungsprozesses unterstützen, angestrebt. Als eine Schwachstelle erweist sich dabei, dass die eingesetzten Werkzeugketten im Automobilbereich die Konsistenz der jeweils erzeugten Artefakte nicht garantieren können. Infolgedessen sind in der Entwicklungspraxis häufig Schnittstelleninkonsistenzen oder fehlerhaft bzw. nicht nachvollziehbar umgesetzte Anforderungen zu beobachten. Das MESA[1] Forschungsprojekt adressiert dieses Problem, indem es Verfahren zur automatischen Konsistenzsicherung von Entwicklungsartefakten über Werkzeuggrenzen hinweg anbietet. Das vorliegende Papier beschreibt die in MESA angestrebte werkzeugübergreifende Konsistenzprüfung und stellt die bereits erzielten Ergebnisse an einem Beispiel vor.

1 Motivation und Einleitung

Die Beherrschung der durch die zunehmend komplexeren Funktionen bedingten Systemkomplexität wird für Fahrzeughersteller und deren Zulieferer zu einem zentralen, wettbewerbsentscheidenden Faktor. Von größter Wichtigkeit sind dabei sorgfältig ausgewählte Entwicklungsmethoden und -werkzeuge, da nur effiziente und

[1] MESA — Metamodellierung zur Automatisierung von Analyse- und Entwicklungsmethoden für Software im Automobil

Dieses Vorhaben wird von der europäischen Union und vom Land Berlin kofinanziert (Europäischer Fonds für Regionale Entwicklung).

fehlervermeidende Prozesse diesen Wettbewerbsvorteil generieren. Problematisch bezüglich der Fehlervermeidung erweist sich, dass die zur Verfügung stehenden Werkzeugketten im Automobilbereich keine Konsistenz der jeweils erzeugten Artefakte garantieren können. Als Folge dieses Sachverhalts ist häufig mit Problemen, wie Schnittstelleninkonsistenz oder fehlerhaft bzw. nicht nachvollziehbar umgesetzten Anforderungen, zu rechnen. Ziel des Projektes MESA ist es, eine werkzeugübergreifende und automatisierte Konsistenzsicherung von Entwicklungsartefakten zur Unterstützung aktueller Entwicklungsprozesse zur Verfügung zu stellen [MSA06].

Beispielhaft könnte ein realer Entwicklungsprozess für eine Systementwicklung im Automobil mit folgenden Werkzeugen realisiert werden: Anforderungsanalyse mittels *Telelogic* DOORS [TLG06], die Spezifikation der Anforderungen mit Verhaltensmodellen in MATLAB/Simulink/Stateflow [MAT06] sowie der Test des Designs durch *dSpace* MTest inklusive CTE/ES [WEW05]. Diese Werkzeuge stellen die Werkzeugkette dar, welche im Rahmen des Projektes MESA untersucht wird.

2 Automatisierung der Konsistenzsicherung

2.1 Inkonsistenz durch fehlende Automatisierung der Konsistenzprüfung

Bisher sind Systementwickler, die sich um Einhaltung der Konsistenz von Entwicklungsartefakten (z.B. Dateien, Dokumenten) bemühen, auf sich alleine gestellt. Das Erfüllen von Konsistenzkriterien kann von ihnen oftmals nur durch manuelles Vergleichen einer großen Menge von Daten erreicht werden. Diese Tätigkeiten geschehen manuell, da sie typischerweise einen Übergang zwischen Artefakten betreffen, die oft mit verschiedenen Werkzeugen erstellt werden.

Ein Beispiel für einen Konsistenzcheck, den MESA künftig vollständig automatisieren wird, betrifft die Werkzeuge *Telelogic* DOORS und MATLAB/Simulink/Stateflow (ML/SL/SF) von *The Mathworks*. In DOORS beschriebene funktionale Anforderungen werden aus Gründen der Qualitätssicherung und mit Blick auf eine frühzeitige Funktionsvalidierung in ML/SL/SF modelliert. Die resultierenden Verhaltensmodelle erlauben eine Simulation der gewünschten Funktionalität. Durch die Simulation, aber vor allen Dingen auch schon während der Modellierung, werden Unklarheiten oder Lücken in den textuellen Anforderungen aufgedeckt. Wenn das fertige Funktionsmodell konsistent zu den Anforderungen ist, lässt es sich beispielsweise zum Testen des Steuergeräts verwenden. Des Weiteren besteht die Möglichkeit, durch Anreicherung des Verhaltensmodells um Implementierungsinformationen, vollautomatisch Code zu generieren, der sich für eine Zielplattform (Rapid Prototyping oder Seriensteuergerät) kompilieren lässt.

Die Konsistenzbeziehungen zwischen den funktionalen Anforderungen und dem Verhaltensmodell betreffen sowohl die verwendeten Signale, als auch die umgesetzten Funktionen. Überprüfungen jeglicher Art, zwischen den Anforderungen in DOORS und den Modellen in ML/SL/SF, geschehen bisher manuell.

2.2 Verfahren zur automatisierten Konsistenzsicherung

Die Model Driven Architecture [OMG2] der OMG [OMG1] definiert und standardisiert zur Problemlösung einen modellbasierten Ansatz mit dem Fokus auf die Softwareentwicklung von verteilten Systemen. Sie stellt Modelle in das Zentrum jeder Entwicklungsphase. In MESA kommen Modellierungswerkzeuge für die Erstellung von Metamodellen [SPX06] nach dem MOF Standard [OMG3] sowie Repositories mit entsprechenden operationalen Schnittstellen zum Einsatz. Für eine durchgängige Verarbeitung der Entwicklungsartefakte wird eine Modellierungsumgebung und Infrastruktur implementiert, welche auf diesen Repositories basiert [HAN06; IKV06].

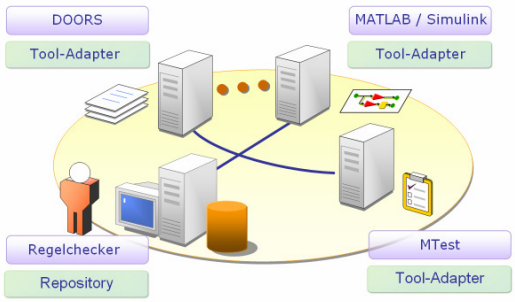

Abbildung 1: Übergreifende Konsistenzsicherung von verteilten Entwicklungsartefakten

Auf dem in MESA entstandenen ASD[2]-Metamodell lassen sich dann Konsistenzregeln in OCL [OMG4] definieren. Die formalen OCL-Ausdrücke können danach durch einen Regelprüfer automatisiert und werkzeugunabhängig ausgewertet werden. Dieser arbeitet dazu auf den Modell-Repositories, die den unternehmensweiten Zugriff auf verteilte Daten (Modelle), leistungsfähige Multiuser- und Zugriffssteuerung, eine hohe Integrität sowie die konsistente Speicherung der Informationen ermöglichen.

Abbildung 1 skizziert schematisch eine automatisierte Konsistenzsicherung am Beispiel unterschiedlicher Werkzeuge in der Fahrzeugfunktionsentwicklung. Durch einen werkzeugintegrierten Tool-Adapter wird ein beliebiges Simulink-Modell [DAB04] aus dem MATLAB-Arbeitsspeicher [HAL05] über die COM-Schnittstelle [MCT06] ausgelesen. Anschließend kann dieses Modell durch einen CORBA-Zugriff [OMG5] über das Netzwerk als eine Modellinstanz im Repository gespeichert werden. In diesem Szenario entspricht das Instanzmodell exakt der Struktur des ASD-Metamodells. Eine auf dem .NET Framework 2.0 [MNT06] basierte Applikation (ASD[2]-Regelchecker) [MSA06] für die Richtlinienprüfung kann nun eine netzwerkfähige OCL-Engine [IKV06] ansprechen und die in OCL definierten Richtlinien auf einem oder mehreren Instanzmodellen ausführen. Die Anbindung weiterer Entwicklungswerkzeuge, wie die bereits eingeführten Werkzeuge DOORS, Stateflow oder MTest, funktioniert analog.

---

[2] ASD – Automotive System Development: Name des Metamodells

3 Metamodellbasierte Regelbeschreibung und Überprüfung

3.1 Beispiel einer werkzeugübergreifenden Konsistenzprüfung

Im diesem Abschnitt wird anhand einer Fahrzeugfunktion ein Beispielszenario des in MESA entwickelten Regel-Checkers erläutert.

3.1.1 Basisfunktion SCHEIBENWISCHER_AKTIVIEREN

3.1.1.1 Beschreibung

Diese Funktion aktiviert den Scheibenwischer, nach Zündung ein.

3.1.1.2 Eingaben

s_zuendung_ein
s_benutzerw_scheibenw_aktivieren

3.1.1.3 Ausgaben

s_scheibenw_aktivieren

3.1.1.4 Parameter

3.1.1.5 Verarbeitung

Bedingung:
- s_zuendung_ein UND
- s_benutzerw_scheibenw_aktivieren

Aktion:
- s_scheibenw_aktivieren

3.1.1.6 Fehlerbehandlung und Diagnose

Abbildung 2:
Funktionale Anforderung in DOORS

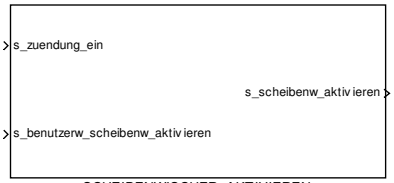

Abbildung 3:
Subsystem in MATLAB/Simulink

Abbildung 2 zeigt exemplarisch einen Ausschnitt einer funktionalen Anforderung an eine Steuerung eines Scheibenwischers. Abbildung 3 zeigt das entsprechende Subsystem in MATLAB/Simulink, das die dargestellte Anforderung in ein Verhaltensmodell umsetzt. Es gilt nun beispielsweise zu überprüfen, ob die Bezeichnung der Basisfunktion in DOORS mit der Bezeichnung des Subsystems in Simulink übereinstimmt. Diese Anforderung an die Entwicklungsmethodik lässt sich, basierend auf dem von uns entwickelten ASD-Metamodell, folgendermaßen beschreiben:

```
context ASDMetamodel.Activities.AM.Basisfunktion
inv: self.allInstances()->
select(ASDMetamodel.Tools.Simulink.SLSubsystem.allInstances()->
select(self.Bezeichner = identifier)->isEmpty())
```

Dieser OCL-Ausdruck besagt Folgendes: „*Aus allen Instanzen der Klasse ‚Basisfunktion' wähle diejenige aus, für die gilt, dass es* ***keine*** *Instanz der Klasse ‚SLSubsystem' mit dem gleichen Bezeichner gibt.*". Die Rückgabe ist somit eine Menge von Basisfunktionen, für die es noch keine entsprechenden Subsysteme in Simulink gibt. Ist die Menge leer, so existieren für alle Basisfunktionen entsprechende Subsysteme mit dem gleichen Bezeichner.

3.2 Prototypische Umsetzung der Richtlinien- und Konsistenzprüfung

Wie bereits erwähnt, wird in den laufenden Arbeiten des Forschungsprojekts MESA am Fraunhofer Institut für offene Kommunikationssysteme (FOKUS) derzeit ein vollständig funktionsfähiger Prototyp (ASD-Regelchecker) für die werkzeugübergreifende Richtlinienüberprüfung von Entwicklungsartefakten implementiert. Durch integrierte Menüs kann der Regel-Checker aus den jeweiligen Werkzeugen kontextbasiert aufgerufen werden. Es können nach Bedarf einzelne Elemente des gerade verwendeten Werkzeugs selektiv geprüft werden. Der Regel-Checker kann zudem auch als eine separate Einzelapplikation gestartet werden, wenn Richtlinien über mehrere Artefakte hinweg evaluiert werden sollen. Vor dem Prüfdurchlauf lädt der Regel-Checker die in OCL formulierten Richtliniendefinitionen und erlaubt die Selektion von kategorisierten Richtlinienprofilen. Abbildung 4 zeigt einen Ausschnitt des Regel-Checkers für die Analyse von Prüfergebnissen anhand eines Simulink-Modells nach einem erfolgten Prüfdurchlauf.

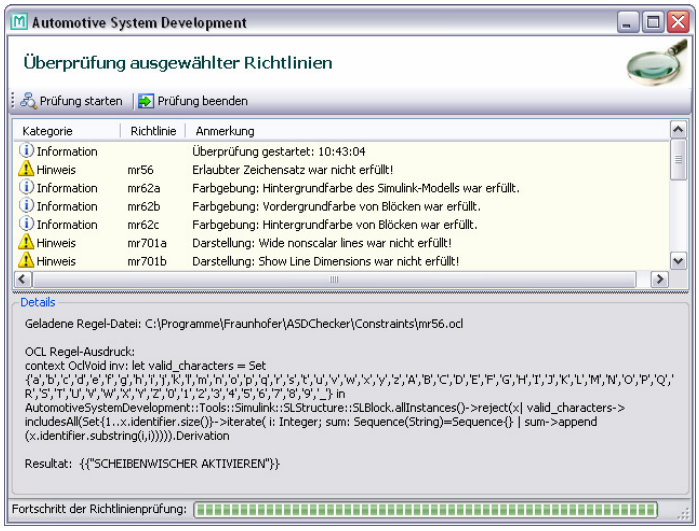

Abbildung 4: Richtlinien- und Konsistenzchecks
mittels Regel-Checker in MATLAB/Simulink

Ein Ergebnisprotokoll in Listenform gestattet es dem Entwickler herauszufinden, welche Richtlinie gegebenenfalls verletzt wurde und welche konkreten Elemente für die Regelverletzung verantwortlich sind. Nach dem Prüfdurchlauf werden die im Ergebnisprotokoll selektierten Informationen, Hinweise oder Fehler in einem speziellen Ausgabefeld („Details“) ausführlich beschrieben. In Abbildung 4 ist exemplarisch dargestellt, dass die Bezeichnung „SCHEIBENWISCHER AKTIVIEREN“ nicht der geltenden Namenskonvention für den erlaubten Zeichensatz entspricht, da diese Bezeichnung ein unerlaubtes Leerzeichen enthält.

4 Zusammenfassung und Ausblick

Dieses Papier stellt den Ansatz des Projektes MESA zur werkzeugübergreifenden Konsistenzsicherung von Entwicklungsartefakten dar. Die betrachteten Werkzeuge sind DOORS, MATLAB/Simulink/Stateflow und MTest. Der Lösungsansatz beruht auf der Abbildung werkzeugspezifischer Artefakte in ein MOF konformes Metamodell. Die Konsistenzprüfung wird auf dem Metamodell anhand der OCL vorgenommen. Dieser Ansatz erlaubt gleichermaßen die Überprüfung werkzeugspezifischer Richtlinien, z.B. der Modellierungsrichtlinien für MATLAB/Simulink/Stateflow. Die Praxistauglichkeit des Ansatzes wird durch ein prototypisch entwickeltes Softwarewerkzeug demonstriert.

MESA stellt derzeit eine konzeptionell fundierte und prototypisch umgesetzte Lösung zur Konsistenzsicherung von Artefakten verschiedener Aktivitäten der Entwicklung softwarebasierter Systeme zur Verfügung, die mit unterschiedlichen Werkzeugen erstellt wurden. Die skizzierte technische Lösung ist ein weiterer Schritt zur Unterstützung einer wettbewerbsfähigen Entwicklung softwarebasierter Systeme im Automobil, der durch die steigende Komplexität vernetzter und zunehmend sicherheitsrelevanter Fahrzeugfunktionen im Automobil notwendig wird.

5 Literaturverzeichnis

[DAB04] Dabney, J.; Harman, T.: *Mastering Simulink*, Pearson Education, Prentice Hall, 2004

[HAL05] Hanselman, D.; Littlefield, B.: *Mastering Matlab 7*, Pearson Education, Prentice Hall, 2005

[HAN06] Publikation in hanser automotive electronics + systems: *Schnellere Softwareentwicklung — Optimierter Entwicklungsprozess*, Ausgabe: 1-2/2006, Carl Hanser Verlag, München, 2006

[IKV06] IKV++ Technologies AG: *Medini Meta Modeler*, URL: www.ikv.biz

[MAT06] The MathWorks Inc., *MATLAB/Simulink/Stateflow*, URL: www.mathworks.com

[MSA06] Forschungsprojekt des FhI FOKUS und der Carmeq GmbH: *Metamodellierung zur Automatisierung von Analyse- und Entwicklungsmethoden für Software im Automobil,* URL: www.fokus.fraunhofer.de/bereichsseiten/projekte/MESA

[MCT06] Microsoft: *Component Object Model Technologies*, URL: www.microsoft.com/com

[MNT06] Microsoft: *.NET Framework*, URL: www.microsoft.com/germany/msdn/netframework

[OMG1] Object Management Group (OMG), URL: www.omg.org

[OMG2] Object Management Group (OMG*): Model Driven Architecture (MDA)*, URL: www.omg.org/mda

[OMG3] Object Management Group (OMG): Meta Object Facility (MOF), Version 1.4, URL: www.omg.org/technology/documents/formal/mof.htm

[OMG4] Object Management Group (OMG): *Object Constraint Language 2.0 Specification*, URL: www.omg.org/docs/ptc/05-06-06.pdf

[OMG5] Object Management Group: *Common Object Request Broker Architecture (CORBA)*, OMG-Document formal/2004-03-12

[SPX06] Sparx Systems Ltd.: *Enterprise Architect*, Version 6.1, URL: www.sparxsystems.com

[TLG06] Telelogic Deutschland GmbH: *DOORS*, Version 8.0, URL: www.telelogic.com

[WEW05] Wewetzer, C.: *MTest – eine offene Testumgebung für die modellbasierte Entwicklung, Abteilung Produktmanagement*, dSPACE GmbH, Paderborn, Design und Elektronik Entwicklerforum, 2005