

Gesellschaft für Informatik (GI)

publishes this series in order to make available to a broad public recent findings in informatics (i.e. computer science and information systems), to document conferences that are organized in cooperation with GI and to publish the annual GI Award dissertation.

Broken down into the fields of

- Seminar,
- Proceedings
- Dissertations
- Thematics

current topics are dealt with from the fields of research and development, teaching and further training in theory and practice. The Editorial Committee uses an intensive review process in order to ensure the high level of the contributions.

The volumes are published in German or English

Information: <http://www.gi-ev.de/service/publikationen/lni/>

ISSN 1617-5468  
ISBN 978-3-88579-206-1

The volumes P109 and P110 contain the Proceedings of INFORMATIK 2007 – the 37<sup>th</sup> Annual Conference of the Gesellschaft für Informatik e.V. (GI) which took place at the Universität Bremen from September 24 to 27, 2007.

The conference featured invited keynote talks of leading experts from industry and academia and selected workshops from different fields of computer science, mostly covering the general conference topic „Computer Science meets Logistics“ (“Informatik trifft Logistik”).



Herzog, Koschke, Rödiger, Ronthaler (Hrsg.):  
INFORMATIK 2007 – Band 2

110



# GI-Edition

## Lecture Notes in Informatics

**Rainer Koschke, Otthein Herzog,  
Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.)**

## INFORMATIK 2007 Informatik trifft Logistik Band 2

**Beiträge der 37. Jahrestagung  
der Gesellschaft für Informatik e.V. (GI)  
24. – 27. September 2007 in Bremen**

# Proceedings

Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger,  
Marc Ronthaler (Hrsg.)

**INFORMATIK 2007**  
**Informatik trifft Logistik**  
**Band 2**

**Beiträge der 37. Jahrestagung  
der Gesellschaft für Informatik e.V. (GI)**

**24.-27. September 2007  
in Bremen**

Gesellschaft für Informatik e.V. (GI)

**Lecture Notes in Informatics (LNI) - Proceedings**  
Series of the Gesellschaft für Informatik (GI)

Volume P-110

ISBN 978-3-88579-204-8

ISSN 1617-5468

**Volume Editors**

Prof. Dr. Rainer Koschke

Universität Bremen, Fachbereich 03 Mathematik/Informatik,  
28359 Bremen, Germany

Email: koschke@tzi.de

Prof. Dr. Otthein Herzog

Universität Bremen, Fachbereich 03 Mathematik/Informatik,  
28359 Bremen, Germany

Email: herzog@tzi.de

Prof. Dr. Karl-Heinz Rödiger

Universität Bremen, Fachbereich 03 Mathematik/Informatik,  
28359 Bremen, Germany

Email: roediger@informatik.uni-bremen.de

Dr. Marc Ronthaler

Universität Bremen, Fachbereich 03 Mathematik/Informatik,  
28359 Bremen, Germany

Email: ronthaler@tzi.de

**Series Editorial Board**

Heinrich C. Mayr, Universität Klagenfurt, Austria (Chairman, mayr@ifit.uni-klu.ac.at)

Jörg Becker, Universität Münster, Germany

Ulrich Furbach, Universität Koblenz, Germany

Axel Lehmann, Universität der Bundeswehr München, Germany

Peter Liggesmeyer, TU Kaiserslautern und Fraunhofer IESE, Germany

Ernst W. Mayr, Technische Universität München, Germany

Heinrich Müller, Universität Dortmund, Germany

Heinrich Reinermann, Hochschule für Verwaltungswissenschaften Speyer, Germany

Karl-Heinz Rödiger, Universität Bremen, Germany

Sigrid Schubert, Universität Siegen, Germany

**Dissertations**

Dorothea Wagner, Universität Karlsruhe, Germany

**Seminars**

Reinhard Wilhelm, Universität des Saarlandes, Germany

© Gesellschaft für Informatik, Bonn 2007

printed by Köllen Druck+Verlag GmbH, Bonn

## Programme committee

### Chairs

Rainer Koschke, Universität Bremen  
Karl-Heinz Rödiger, Universität Bremen

### Members of the programme committee

Otthein Herzog, Universität Bremen  
Christian Hochberger, TU Dresden  
Frank Kirchner, Universität Bremen  
Alois Knoll, TU München  
Stefan Messerknecht, CargoSoft GmbH, Bremen  
Heidi Schelhove, Universität Bremen  
Gerhard Schrott, TU München  
Walter Tichy, Universität Karlsruhe (TH)  
Markus Voss, sd&m AG  
Alfred Winter, Universität Leipzig

### Organizers

Raimar Falke, Universität Bremen  
Pierre Frenzel, Universität Bremen  
Frank Große, T-Systems  
Wiebke Henrici, Hochschule Bremen  
Marc Ronthaler, Universität Bremen  
Jochen Quante, Universität Bremen  
Cornelia Winter, GI e.V., Bonn

### Sponsors

**Microsoft®**

**BLG**  **LOGISTICS**

**sd&m**  
A Company of  Capgemini

**big bremen**  
Die Wirtschaftsförderer.

IVU Traffic Technologies AG 

 **Springer**



<b>Automotive Software Engineering</b>	<b>521</b>
<i>H. Lucke, D. Schaper, P. Siepen, M. Uelschen, M. Wollborn</i> The Innovation Cycle Dilemma	526
<i>M. Hagner, M. Huhn</i> Modellierung und Analyse von Zeitanforderungen basierend auf der UML	531
<i>A. Dold, M. Trapp</i> Herausforderungen und Erfahrungen eines OEM bei der Gestaltung sicherheitsgerechter Prozesse	536
<i>A. Hermann</i> Fahrsituationsspezifische Datenverteilung im Verteilten Umgebungsmodell für Fahrzeugsoftware	541
<i>O. Hartkopp</i> Fahrzeuganbindungen durch Standard-IT-Verfahren	546
<i>F. Höwing</i> Effiziente Entwicklung von AUTOSAR-Komponenten mit domänenspezifischen Programmiersprachen	551
<i>I. Fey, H. Kleinwechter, A. Leicher, J. Müller</i> Lessons Learned beim Übergang von Funktionsmodellierung mit Verhaltensmodellen zu modellbasierter Software-Entwicklung mit Implementierungsmodellen	557
<i>S. Prochnow, R. von Hanxleden</i> The Use of Complex Stateflow-Charts with KIEL — An Automotive Case Study	564

# Automotive Software Engineering

Stefan Kowalewski

Lehrstuhl für Informatik 11  
RWTH Aachen  
Ahornstr. 55  
52074 Aachen  
kowalewski@cs.rwth-aachen.de

Michael Reinfrank

SV P ED GS  
Siemens VDO Automotive AG  
Siemensstrasse 12  
93055 Regensburg  
michael.reinfrank@siemens.com

## Vorwort

Um ein Fahrzeug auf die Straße zu bringen, braucht die Automobilindustrie heute hunderttausende von Programmzeilen in den elektronischen Steuergeräten. Andererseits kann eine fehlerhafte Programmzeile hunderttausend Fahrzeuge im Feld lahm legen. Dies zeigt sehr plastisch die immense Bedeutung von Automotive Software einerseits für Funktion und insbesondere Innovation im Fahrzeug, andererseits aber auch für die Einhaltung der hohen Qualitätsstandards.

Um dieser Bedeutung gerecht zu werden, ist es zwingend erforderlich, in der Automobilindustrie moderne Softwaretechnologie einzusetzen, das ist die Holschuld der Fahrzeughersteller sowie ihrer Elektronik- und Softwarezulieferer. Andererseits sehen wir eine Bringschuld seitens der Forschung und Lehre, Softwaretechniken bereit zu stellen, die den spezifischen Anforderungen im Automobil und den entsprechenden Entwicklungsprozessen gerecht werden, sowie die Studentinnen und Studenten durch gezielte Ausbildung auf dieses Arbeitsumfeld vorzubereiten.

Dies kann nur in einem engen Schulterschluss zwischen Industrie und Hochschule geschehen. Der mittlerweile fünfte Workshop "Automotive Software Engineering" soll zu diesem Dialog zwischen der weltweiten Software Community und dem Anwenderkreis aus der Automobilindustrie beitragen.

Im Jahr 2003 haben Manfred Broy, Klaus Grimm, Michael Reinfrank und Alexandre Saad auf der ICSE – International Conference on Software Engineering – in Portland, Oregon, eine erste Session zu Automotive Software organisiert. Die kurz darauf folgende Gründung der GI-Fachgruppe Automotive Software Engineering unter Leitung von Dr. Grimm und Prof. Broy war ein wesentlicher nächster Schritt, Automotive Software als integralen Bestandteil der Informatik zu etablieren. Wenige Jahre später können wir heute feststellen, dass Automotive Software bei der GI eine Heimat gefunden hat.

Es ist uns erfreulicherweise gelungen, weit über 20 Mitglieder für das Programmkomitee zu gewinnen, mit einem gesunden Mix aus Vertretern der Hochschule und Forschungsinstitute, der Fahrzeughersteller, der Elektronikzulieferer und kleinerer spezialisierter Firmen aus den Bereichen Software und Tools.

Das Programmkomitee konnte aus ca. 30 eingereichten Beiträgen die nachfolgenden 8 Papiere auswählen, die ebenfalls das ganze Spektrum von wissenschaftlichen Grundlagen bis hin zur industriellen Praxis abdecken. Der Beitrag „The Innovation Cycle Dilemma“ von H. Lucke et al. behandelt das Problem der unterschiedlichen Innovationsfrequenz in der Automobilelektronik und der Unterhaltungselektronik und die damit verbundenen Auswirkungen. M. Hagner und M. Huhn stellen in „Modellierung und Analyse von Zeitanforderungen basierend auf UML“ einen Ansatz zur frühzeitigen Spezifikation und Analyse von Echtzeiteigenschaften vor. In dem Beitrag „Herausforderungen und Erfahrungen eines OEM bei der Gestaltung sicherheitsgerechter Prozesse“ von A. Dold und M. Trapp geht es um die praktische Umsetzung der kommenden Norm ISO 26262 zur funktionalen Sicherheit. A. Hermann befasst sich unter dem Thema „Fahrsituationsspezifische Datenverteilung im verteilten Umgebungsmodell für Fahrzeugsoftware“ mit einem effizienten Datenmanagement für Fahrerassistenzsysteme. Das Papier „Fahrzeuganbindung durch Standard-IT-Verfahren“ von O. Hartkopp beschreibt die Möglichkeit, Netzwerkprotokolle aus dem PC-Bereich in die Automobilvernetzung zu integrieren. Das aktuelle Thema AUTOSAR und seine Umsetzung durch geeignete Beschreibungsmittel wird in dem Beitrag „Effiziente Entwicklung von AUTOSAR-Komponenten mit domänenspezifischen Programmiersprachen“ von F. Höwing aufgegriffen. Die letzten beiden Beiträge beleuchten unterschiedliche Aspekte der modellbasierten Software-Entwicklung: I. Fey et al. berichten über praktische Erfahrungen mit dem Umstieg auf eine modellbasierte Funktionsentwicklung, und S. Prochnow und R. von Hanxleden stellen ein Werkzeug zur besonderen Unterstützung der Modellierung mit Statecharts vor.

Abgerundet wird das Programm durch einen eingeladenen Vortrag von Dr. Kai Storjohann und eine Podiumsdiskussion mit Vertretern aus Hochschule und Industrie.

Was sind denn aber die "spezifischen" Herausforderungen in der Automotive Software?

Es gibt weitaus komplexere Softwaresysteme, z.B. in der Luft- und Raumfahrt, es gibt Massenprodukte mit höherem Kostendruck, z.B. bei Handys oder bei der weißen Ware, es gibt Anwendungen mit höheren Echtzeitanforderungen. Für jeden einzelnen Aspekt gibt es also Bereiche, die mindestens genau so anspruchsvoll sind. Das Besondere beim Auto ist es, dass all diese Herausforderungen gleichzeitig auftreten: Echtzeit, Komplexität, Zuverlässigkeit, und das alles in einem extrem kostenoptimierten Umfeld bezüglich Kosten der Hardware und der Entwicklung. Ein Euro mehr oder weniger für die Prozessorhardware kann über die Profitabilität eines Produktes entscheiden, weshalb viele Methoden wie z.B. objekt-orientiertes Programmieren nicht einfach eins-zu-eins zum Einsatz kommen können. Automotive Software ist nach wie vor hardware-naher C-Code. Die Herausforderung ist also, moderne Softwaretechnologien dahingehend zu verfeinern, dass sie in einem solchen Umfeld im industriellen Maße einsetzbar sind.

Eine weitere Hürde stellt der Entwicklungsprozess dar: Automotive Software Engineering ist immer hochgradig vernetzt mit der Entwicklung des Fahrzeuges und seiner Aggregate. Entwickler der Fahrzeughersteller und der Zulieferer arbeiten weltweit verteilt gemeinsam in den Projekten. Die Anzahl der Änderungen steigt stetig bis zum Produktionsstart an und erreicht in der Regel ihren Höhepunkt beim Serienanlauf. Entsprechende Entwicklungsmethoden und das zugehörige Projektmanagement müssen dieser Tatsache gerecht werden.

Und nicht zuletzt muss die Industrie ein geeignetes Geschäftsmodell für Software etablieren. Nach wie vor ist Tom de Marcos Frage "why is software so expensive" eine der meist gestellten Fragen in Management Meetings der Autoindustrie. Software wird in der Regel nach Aufwand verrechnet, was eine Software letztendlich für den Kunden wert ist, wird noch viel zu selten diskutiert und in den Geschäftsmodellen abgebildet.

Das führt uns wieder zu unserem Motto des Workshops: Automotive Software befindet sich in einem Spannungsfeld zwischen Markt und technisch-wissenschaftlicher Herausforderung. Wir wünschen uns, dass die Vorträge sowie die Diskussionen auf und am Rande dieses Workshops einen Beitrag leisten, sich dieser Herausforderung zu stellen.

Unser Dank gilt den Mitgliedern des Programmkomitees für ihre engagierte Mitarbeit und der Gesellschaft für Informatik für die organisatorische Unterstützung. Besonders danken wir Dr. Bernhard Hohlfeld, Jacob Palczynski und Gerhard Wirrer für die vielfältige und wertvolle Hilfe.

*Stefan Kowalewski*

*Michael Reinfrank*

### **Programmkomitee des 5. GI-Workshops Automotive Software Engineering**

Hans-Jürgen Belz,  
Hella KGaA Hueck AG  
Jürgen Bortolazzi, Porsche AG  
Manfred Broy, TU München  
Mirko Conrad, The Mathworks GmbH  
Michael Daginnus; Volkswagen AG  
Werner Damm, Universität Oldenburg  
und OFFIS  
Bernd Frielingsdorf,  
Ford Werke GmbH  
Bernhard Hohlfeld,  
DaimlerChrysler AG  
Michaela Huhn, TU Braunschweig  
Stefan Jähnichen,  
TU Berlin und FhG FIRST  
Hubert Keller, FZ Karlsruhe  
Thorsten Kölzow, Audi AG

Rainer Koschke, Universität Bremen  
Stefan Kowalewski, RWTH Aachen  
Thomas Kropf, Robert Bosch GmbH  
Hans-Jürgen Kugler, Kugler-Maag und  
Lero, Univ. Limerick  
Stefan Ortmann, Carmeq GmbH  
Helmuth Partsch, Universität Ulm  
Klaus Pohl, Universität Duisburg-Essen  
und Lero, Univ. Limerick  
Wolfgang Pree, Universität Salzburg  
Michael Reinfrank, SiemensVDO  
Dieter Rombach, TU Kaiserslautern  
und FhG IESE  
Alexandre Saad, BMW Group  
Christian Salzmann, BMW Group  
Jörn Schneider, Robert Bosch GmbH  
Gerhard Wirrer, SiemensVDO

# Effiziente Entwicklung von AUTOSAR-Komponenten mit domänenspezifischen Programmiersprachen

Dr. Frank Höwing

LINEAS Automotive GmbH  
Theodor-Heuss-Str. 2  
D-38122 Braunschweig  
frank.hoewing@lineas.de

**Abstract:** Der AUTOSAR-Standard erfordert in der Entwicklung automobiler Steuergeräte einen weitgehenden Einsatz von Werkzeugen, u.a. weil viele Informationen die bisher implizit im Programmcode steckten, jetzt explizit konfiguriert und auf einer höheren Abstraktionsebene betrachtet werden müssen. Für viele Embedded-Entwickler entsteht hier in ihrer täglichen Arbeit ein Bruch im Umgang mit Tools und Abstraktionsebenen. Dieser Beitrag beschreibt einen Ansatz eine „AUTOSAR-Programmiersprache“ zu entwickeln, die der gewohnten Sprache C ähnelt und dennoch die neuen Möglichkeiten von AUTOSAR einfach und im selben Werkzeug nutzbar macht. Hauptvorteil wäre neben der gesteigerten Akzeptanz eine effizientere Komponenten-Entwicklung.

## 1 Der AUTOSAR-Prozess und seine Tools

Innerhalb des AUTOSAR-Prozesses (siehe [www.autosar.org](http://www.autosar.org)) gibt es drei primäre Stellen, an denen spezifische Tools eingesetzt werden:

- High-level Modellierung des Gesamtsystems aus der Software-Komponenten-Struktur, den Steuergeräte-Ressourcen sowie des Fahrzeug-(Netz-)Systems. Hier kommen üblicherweise grafische Werkzeuge zum Einsatz.
- Low-level Konfiguration der Steuergeräte-Software, d.h. des Betriebssystems, der Basis-Software sowie ggf. der RTE. Dieser Bereich erfordert spezifische Software aus der Steuergeräte-Entwicklung, wie sie z.T. auch außerhalb von AUTOSAR zum Einsatz kommt.
- Implementierung der Software-Komponenten. Das Entwickeln der Logik einer Komponente erfolgt modellbasiert mit entsprechenden Werkzeugen oder wird mit Hilfe eines Editors bzw. einer integrierten Entwicklungsumgebung (IDE) traditionell programmiert.

Das zentrale Datenformat für die Beschreibung der Informationen (im Wesentlichen aus der High-level Modellierung) ist durch ein XML-Schema (autosar.xsd) festgelegt.

Dieses XML-Schema wird von den AUTOSAR-Arbeitsgruppen aus einer UML-Darstellung des AUTOSAR Meta-Modells generiert und spiegelt die in den AUTOSAR-Spezifikationen dokumentierten Modellelemente formal wider. Werkzeuge speichern ihre Daten in AUTOSAR-XML-Dateien, um sie an andere Werkzeuge in anderen Prozessschritten oder bei Entwicklungspartnern weiterzugeben.

## 2 Die Sicht des Komponenten-Entwicklers

Aus den vielfältigen Tätigkeiten innerhalb der Steuergeräte-Entwicklung stellt die Entwicklung funktionaler Software-Komponenten, d.h. ihrer Architektur und Logik, einen Hauptanwendungsfall dar. Trotz sich verbreitender modellbasierter Ansätze verwenden Entwickler dazu häufig noch traditionelle Programmierwerkzeuge und die Programmiersprache C. Für diese Entwickler stellt sich ein wesentlicher Teil ihrer Arbeit mit AUTOSAR grob wie folgt dar:

1. Modellierung der Software-Komponente mit verwendeten Interfaces, Ports, Runnables sowie der zugehörigen Kommunikationsmodi. Modellierung der Kommunikationsbeziehungen der Komponente mit anderen Komponenten.
2. Umsetzung der Kommunikation auf reale Bussysteme, Zuordnung von Runnables auf Tasks und ggf. weitere Konfigurationen.
3. Generieren der RTE-Header-Datei <Komponente>.h mit RTE-API und Runnable-Prototypen für diese Komponente.
4. Implementierung der Komponentenlogik in <Komponente>.c unter Verwendung des RTE-Headers.

Von diesen stufenweise konkreter werden Schritten erfordert insbes. Schritt 1 Kenntnis der z.T. neuen AUTOSAR-Konzepte und ihrer Anwendung in den entsprechenden Werkzeugen.

```
void DoorReceiverRightRear (Rte_Instance self)
{
    DoorStatusType statusRightRear;
    Rte_Read_RDoorRightRear_status (self, &statusRightRear);
    if(statusRightRear != DoorClosed) {
        Rte_IWrite_DoorReceiverRightRear_PHorn_cmd (self, HornOn);
    }
    else {
        Rte_IWrite_DoorReceiverRightRear_PHorn_cmd (self, HornOff);
    }
}
```

Abbildung 1: AUTOSAR-Implementierung einer sehr einfachen Beispiel-Komponente (Auszug)

Zur Implementierung der Runnables sowie der Anwendung der RTE-API muss der Entwickler sowohl die abstrakten AUTOSAR-Konzepte als auch deren Entsprechung im C-Code laut AUTOSAR-RTE-Spezifikation kennen (vgl. hervorgehobenen Code in Abbildung 1).

### 3 Prinzip der domänenspezifischen Sprachen

Domänenspezifische Sprachen (engl. *domain-specific language*, DSL) stellen (Programmier-)Sprachen dar, die spezielle Notationen und Konstrukte für eine Anwendungsdomäne bereitstellen. Dadurch bieten DSLs eine höhere Ausdruckskraft und einfachere Anwendbarkeit in ihrem Anwendungsbereich als allgemeine Programmiersprachen, wie C oder Java. Dies führt zu einer effizienteren Entwicklung und verringerten Wartungskosten.

Bereits Programmierbibliotheken oder die in der Embedded-Entwicklung verbreiteten #define-Sammlungen können als DSLs aufgefasst werden. Der aus dem Bereich der Modellgetriebenen Softwareentwicklung [SV05] stammende Begriff geht jedoch weiter, indem nicht nur vorhandene Programmiersprachen erweitert, sondern gezielt neue erstellt werden. Die neue Sprache ist unabhängig von eventuellen Restriktionen einer zugrundeliegenden Basissprache. Durch die formale Definition einer solchen praxisnahen Sprache ist es außerdem z.B. möglich für diese Sprache Werkzeuge wie Code-Editoren, Parser und Konverter zu generieren.

### 4 Entwurf einer textuellen AUTOSAR-DSL

Das Beispiel in Abbildung 2 veranschaulicht, wie eine textuelle DSL für AUTOSAR aussehen kann. Modellelemente wie z.B. Interfaces, die üblicherweise in grafischen Tools modelliert werden, können im Programmcode der AUTOSAR-DSL (ADSL) textuell, d.h. für den Entwickler „wie gewohnt“ beschrieben werden. Ein Wechsel zwischen verschiedenen Tools ist nicht erforderlich.

Ebenfalls ohne Bruch in der Anwendung unterschiedlicher Tools oder Abstrahierungsgrade, erlaubt die ADSL in einem Zug sowohl die Modellierung als auch die Implementierung von Komponenten. Zur Modellierung von Runnables wurde z.B. das Schlüsselwort `runnable` eingeführt. Der Programmierer muss keine C-Funktion zur Implementierung dieses Runnables mehr erstellen, da aus der ADSL-Datei im Buildprozess automatisch eine AUTOSAR-konforme C-Datei generiert wird. Die Programmierung der vollständigen Komponentenlogik in ADSL ist möglich, da in die AUTOSAR-DSL die Sprachelemente der Programmiersprache C eingebettet wurden.

Ein Beispiel für effiziente Sprachkonstrukte sind die eingeführten Sende- und Empfangsoperatoren `->` und `<-`, die die teils unhandliche RTE-API verbergen. Aus der C-Implementierung

```
rte_IWrite_DoorReceiverRightRear_PHorn_cmd (self,HornOn);
```

wird somit die einfachere und wartbarere ADSL-Zeile

```
af_cmd_horn.HornOn -> PHorn.cmd;
```

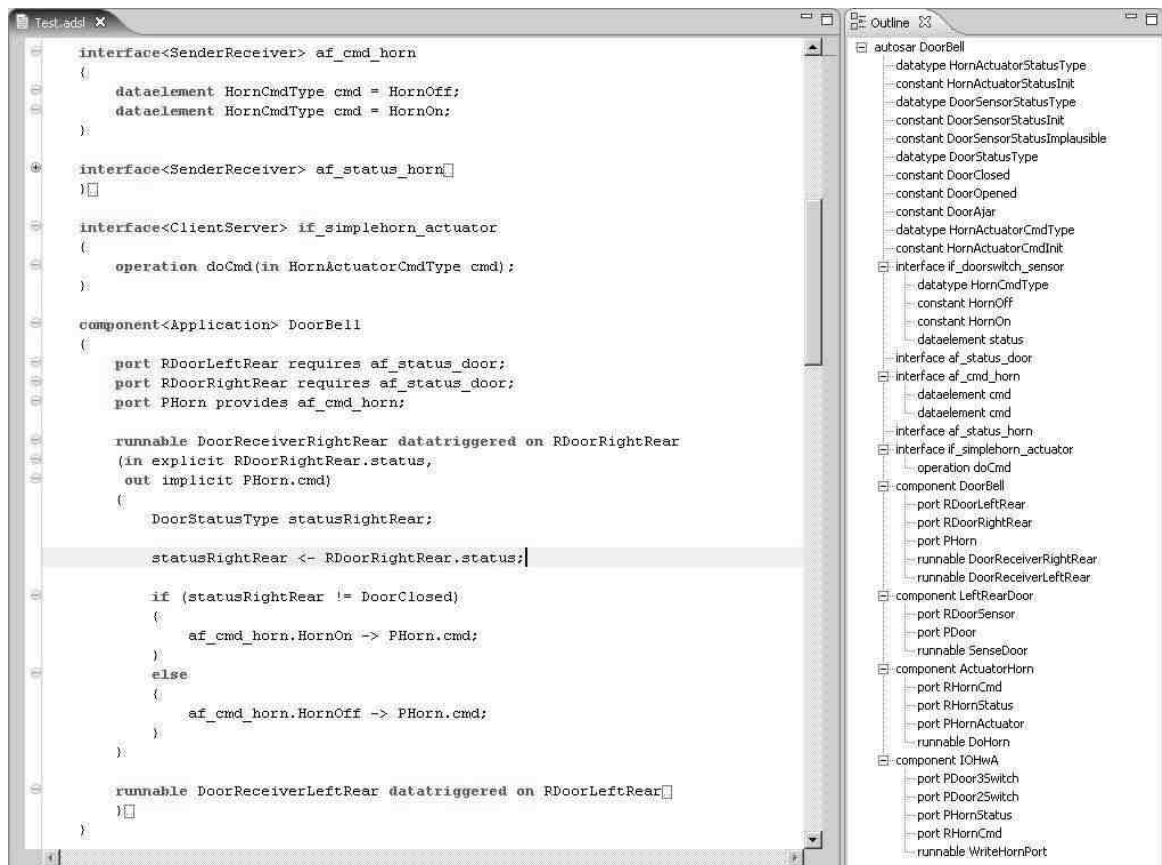


Abbildung 2: Generierter AUTOSAR-DSL-Editor mit Syntaxhervorhebung und Strukturbaum

Trotz dieser Vorteile stellt die Entwicklung der DSL selbst einen nicht unerheblichen Aufwand dar, was wiederum Werkzeuge erforderlich macht. In der vorliegenden Arbeit kam das Sprachentwicklungs-Framework MontiCore [Gr06] des Instituts für Software Systems Engineering (SSE) der Technischen Universität Braunschweig zum Einsatz. Als Eingabe erhält das Framework eine Sprachbeschreibung in Form einer kontextfreien Grammatik. Daraus kann es z.B. den in Abbildung 2 gezeigten Editor als Java Eclipse-Plugin automatisch generieren. Um aus den mit dem Editor erstellten ADSL-Dateien C-Code oder AUTOSAR-XML erzeugen zu können, wurden die Mechanismen von MontiCore um spezifische Transformationen erweitert.

Die Grammatik der AUTOSAR-DSL wurde in vier Schritten aus der standardisierten autosar.xsd abgeleitet:

1. Reduktion der autosar.xsd auf die in der ADSL benötigten Elemente.
2. Generische Vortransformation der reduzierten autosar.xsd in eine grammatiknahe, weiterverarbeitbare Form.
3. Hinzufügen der Elemente der Programmiersprache C.
4. Anpassen der automatisch gewonnenen AUTOSAR-Bezeichner und ihrer Syntax an die gewünschte DSL-Form.

## 5 Entwicklungsprozess mit AUTOSAR-DSL

Abbildung 3 veranschaulicht die Einbettung der DSL in den AUTOSAR-Prozess. Zentrales Werkzeug ist der vom DSL-Tool generierte Editor, der die Anwendung der DSL z.B. durch Syntaxhervorhebung unterstützt. Damit erzeugt der Entwickler AUTOSAR-DSL-Dateien (.adsl), die neben AUTOSAR-Elementen wie z.B. Runnables auch deren Implementierung und ggf. Konfiguration beinhalten.

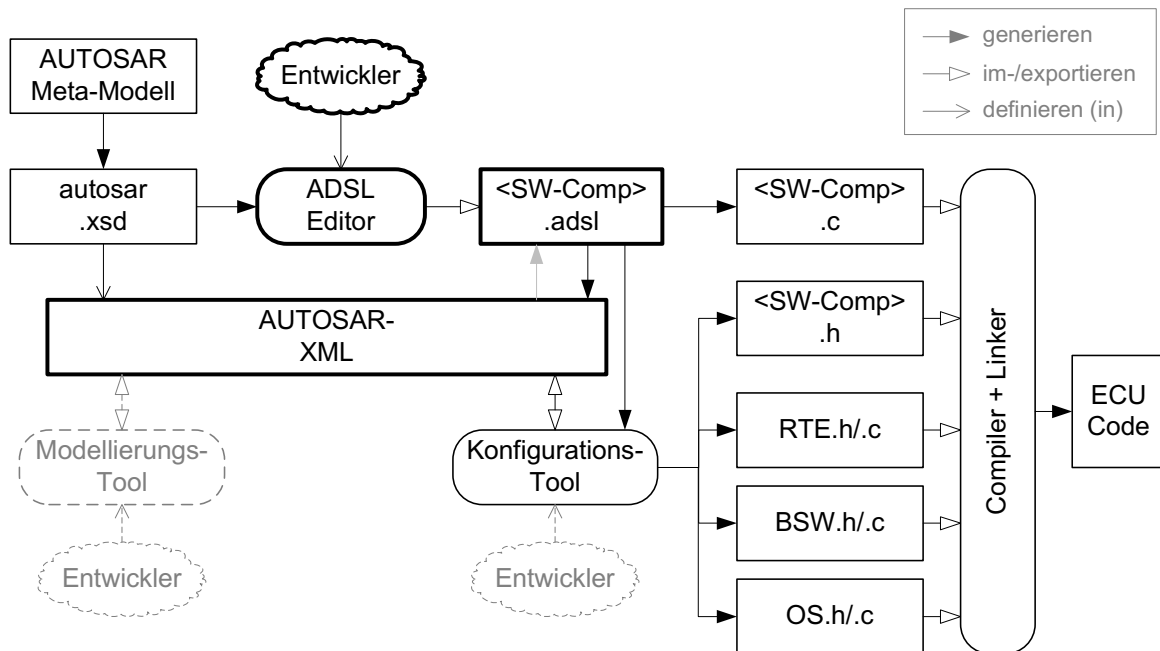


Abbildung 3: AUTOSAR-Prozess (vereinfacht) bei Verwendung einer textuellen DSL

Die Aspekte des Modells, der Konfiguration sowie der Implementierung werden automatisch durch den nachfolgenden Build-Prozess separiert. Dazu werden Generatoren gestartet, die z.B. ebenfalls vom DSL-Entwicklungssystem erzeugt werden können:

- Modellelemente wie Components, Runnables, Interfaces usw. sowie Konfigurationsinformationen wie z.B. das Taskmapping von Runnables werden in AUTOSAR-XML ausgegeben.
- Optional können zusätzliche, toolspezifische Konfigurationen erzeugt oder verändert werden, um die ADSL effizient in die verwendete Toolkette zu integrieren. Wie in [Mü07] gezeigt, kann es sogar möglich und sinnvoll sein, z.B. Quellcode nachträglich zu verändern, der von anderen Werkzeugen generiert wurde – etwa um die RTE mit Trace-Ausgaben zu instrumentieren.
- Die Implementierung einer Komponente wird als C-Datei <Komponente>.c generiert. Dieser Code muss nicht mehr manuell bearbeitet werden und ist sofort compilierfähig.

Der Entwickler einer Software-Komponente benötigt die Werkzeuge zur High-level Modellierung sowie zur Low-level Konfiguration nur noch für spezielle Aufgaben, die die DSL nicht abdecken kann oder soll, z.B. um grafische Übersichten zu erstellen.

Durch die enge Integration der DSL in die AUTOSAR-Toolkette ändert sich für den Entwickler in der Praxis gegenüber dem heutigen Ablauf des Programmierens, Compilierens und Linkens von C-Quellcode nichts Wesentliches.

## **6 Fazit**

Der Ansatz einer AUTOSAR-spezifischen Programmiersprache im Sinne einer textuellen DSL stellt einen vielversprechenden Schritt dar, der neben einer Steigerung der Akzeptanz von AUTOSAR bei den Software-Entwicklern auch zu einer effizienteren und wartbareren Implementierung von AUTOSAR-Komponenten führen kann.

Das erforderliche Lernen einer neuen Programmiersprache stellt keinen bedeutenden Aufwand dar, da die neuen AUTOSAR-Konzepte ohnehin erlernt werden müssen und die ADSL diese in natürlicher Weise umsetzt. Das vorgestellte Konzept erlaubt es darüberhinaus, eine spezifische AUTOSAR-DSL ganz nach Kundenwunsch zu erstellen. Den Vorteilen steht der Aufwand zur Entwicklung und Pflege der DSL gegenüber. Es konnte gezeigt werden, dass Sprachentwicklungswerkzeuge helfen, diesen Aufwand zu reduzieren.

## **Danksagungen**

Die Machbarkeit einer DSL für AUTOSAR wurde im Rahmen einer Masterarbeit des Instituts für Software Systems Engineering (SSE, [www.sse-tubs.de](http://www.sse-tubs.de)) der Technischen Universität Braunschweig bei der LINEAS Automotive GmbH evaluiert [Ap07]. Der Autor dankt Herrn Catalin Apostu für die Realisierung des Prototypen sowie Herrn Prof. Bernhard Rumpe und Herrn Hans Grönniger vom SSE für die gute Zusammenarbeit und die Möglichkeit zur Nutzung des MontiCore Frameworks.

## **Literaturverzeichnis**

- [Ap07] Apostu, C.: Konzeption und Realisierung einer textuellen domänenspezifischen Sprache am Beispiel AUTOSAR. Masterarbeit, TU Braunschweig, SSE, März 2007.
- [Gr06] Grönniger, H.; Krahn, H.; Rumpe, B.; Schindler, M.; Völkel, S.: MontiCore 1.0: Framework zur Erstellung und Verarbeitung domänenspezifischer Sprachen. Technische Universität Braunschweig, Institut für Software Systems Engineering, Informatik-Bericht 2006-04.
- [Mü07] Müller, J.-V.: Domain Specific Configuration Generators – How to Simplify Your Development Process. Embedded World, Nürnberg, 2007.
- [SV05] Stahl, T.; Völter, M.: Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management. Dpunkt Verlag, 2005.