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) 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;
```

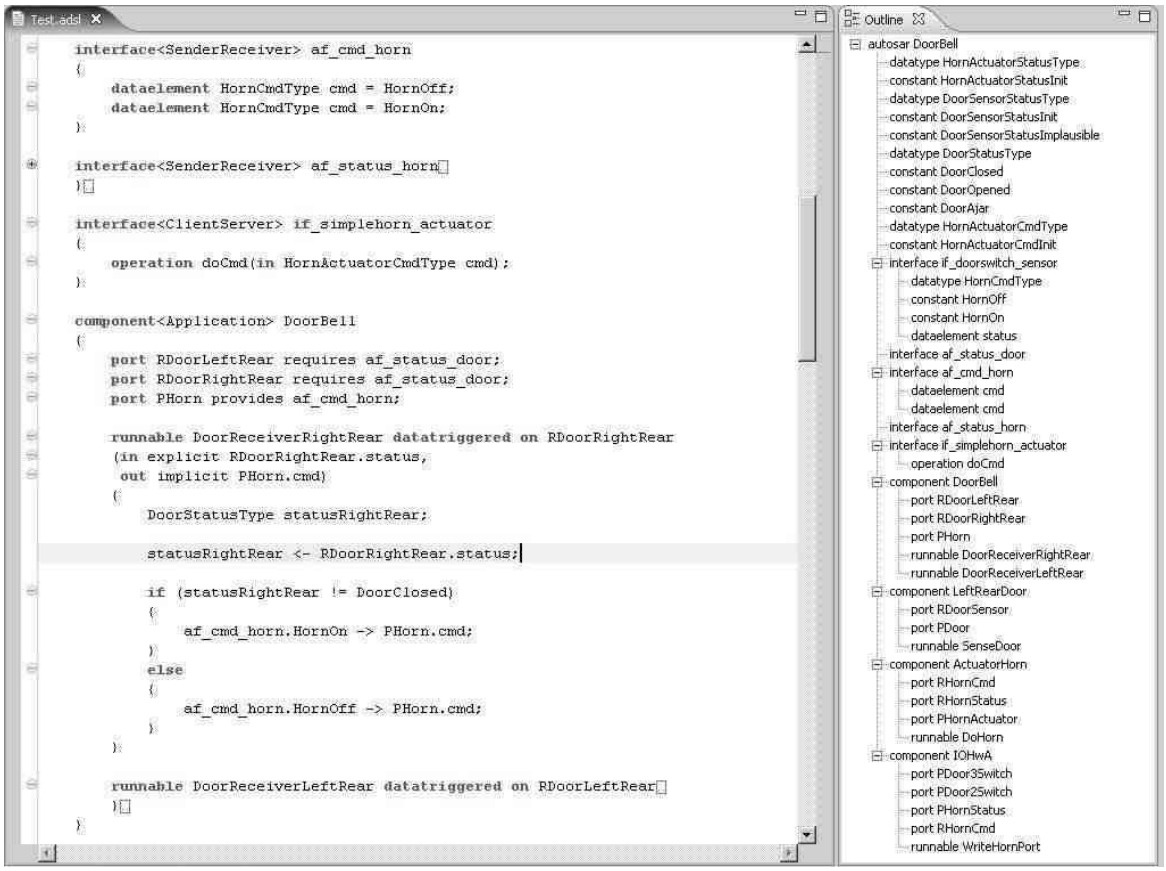

Abbildung 2: Generierter AUTOSAR-DSL-Editor mit Syntaxhervorhebung und Strukturbaum

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.

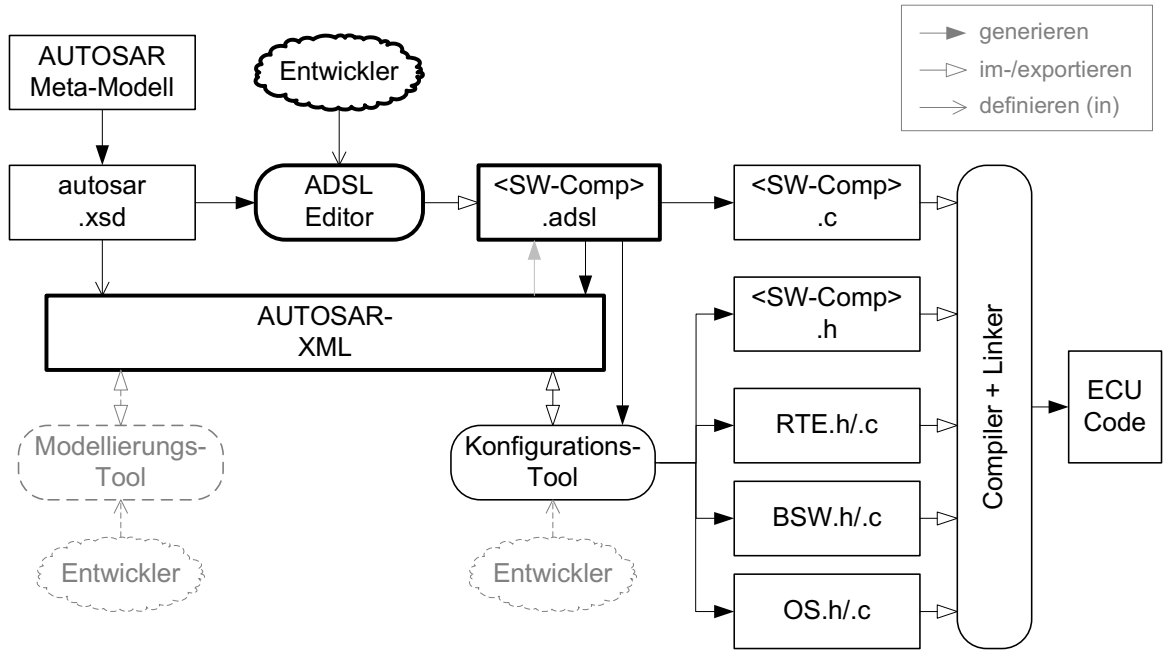

Abbildung 3: AUTOSAR-Prozess (vereinfacht) bei Verwendung einer textuellen DSL

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) 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.