Handhabung von Varianz in Simulink aus funktionsorientierter Sicht

Benjamin Gutekunst, Jens Weiland

Hochschule Reutlingen
Fakultät Technik
D-72762 Reutlingen
benjamin.gutekunst@reutlingen-university.de
jens.weiland@reutlingen-university.de

Abstract: Aufgrund der steigenden Zahl softwarebasierter Funktionen kommt in der Automobilindustrie der baureihenübergreifenden Wiederverwendung dieser Funktionen und der damit verbundenen Handhabung von Varianz eine essentielle Bedeutung zu. In den Entwicklungsbereichen existieren für die modellbasierte Softwareentwicklung mit Simulink erste Ansätze zur Handhabung von Varianz auf Basis elementarer Simulink-Blöcke. Im Kontext einer funktionsorientierten Sichtweise zeigen sich in der Praxis in Bezug auf die systematische Handhabung ganzer Funktionsmodule allerdings eine Reihe gravierender Einschränkungen. Der vorliegende Artikel stellt aufbauend auf den Eigenschaften existierender Ansätze ein Konzept zur Handhabung variabler Funktionsmodule in Simulink und dessen Anwendung vor.[1]

1 Einleitung

In eingebetteter automotive Software spiegelt sich die ganze Vielfalt heute verfügbarer Funktionen im Fahrzeug wider. Das Auftreten dieser Vielfalt – und der damit einhergehenden Varianz – ist vielschichtig aufgrund einer Vielzahl an Fahrzeuganforderungen und -konfigurationen, nicht zuletzt auch aufgrund einer steigenden Vernetzung im Fahrzeug und mit der Umwelt [SZ03]. Um der Varianz in eingebetteter automotive Software Rechnung zu tragen ist für eine wirtschaftliche Betrachtung die Wiederverwendung, z.B. von Spezifikationen, Architekturen, Komponenten oder auch Dokumentation und Tests, essentiell. Eine besondere Herausforderung liegt somit in der Handhabung wiederverwendbarer Artefakte[2].

Ein Trend, der sich in den letzten Jahren in der Automobilindustrie etabliert hat, ist die modellbasierte Entwicklung eingebetteter Softwarefunktionen [DW07]. Ein häufig verwendetes Softwareentwicklungswerkzeug für die Modellierung von Fahrzeugfunktionen ist die MATLAB Werkzeugkette mit den Erweiterungen Simulink und

[1] Das vorgestellte Konzept wurden im Rahmen des Forschungsprojektes ESPA entwickelt, welches durch das Bundesministerium für Wirtschaft und Technologie gefördert wird.
[2] Im Rahmen der Standardisierung von Softwarearchitekturen im Fahrzeug wird durch die AUTOSAR-Initiative [AU11] aktuell ein umfassender Schritt in Richtung Wiederverwendung vollzogen.

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4
weitere Artikel online:
http://informatik2011.de/519.html

Stateflow. Simulink-Modelle sind aus der Verschaltung von Elementarblöcken und Statecharts aufgebaute Signalflussgraphen. Komplexe Teilfunktionen können durch „Subsysteme“ gekapselt werden, die wiederum als Signalflussgraphen modelliert sein können. Effiziente Codegeneratoren, wie der Realtime Workshop Embedded Coder von The Mathworks oder TargetLink von dSpace, ermöglichen es die Qualität der so entstandenen komplexen Modelle bis in den Seriencode zu übernehmen.

Die Produktlinientechnik ermöglicht in der Automobilindustrie durch eine systematische Wiederverwendung der hohen Varianz im Fahrzeug zu begegnen. Auch in der modellbasierten automotive Softwareentwicklung ist die Produktlinientechnik unumgänglich. In Simulink existieren erste produktiv eingesetzte Ansätze zur Anwendung der Produktlinientechnik auf Basis elementarer Simulink-Blöcke [CK04, DL08, TD11, WK09].

In der Praxis hat sich jedoch gezeigt, dass im Rahmen der Entwicklung eines Simulink-Modells Funktionen mit der auf Varianz ausgerichteten Modellarchitektur „vermischt“ werden. Eine eindeutige Kapselung wiederverwendbarer Funktionsmodule – und somit eine strikte Trennung von Belangen zwischen variablen Funktionsmodulen und der Modellarchitektur – ist nicht möglich. Zwar werden in den neuesten Simulink-Versionen spezielle variantenspezifische Blöcke für eine funktionsorientierte Sicht bereitgestellt. Diese sind jedoch aufgrund erheblicher Einschränkungen in Bezug auf den Zeitpunkt, die sog. *Bindezeit*, wann Varianz entfernt und somit ein konkretes Funktionsmodul ausgewählt werden kann, nur begrenzt einsetzbar.

Gerade im Automobilbereich spielt dieser Zeitpunkt aber eine wichtige Rolle; soll z.B. eine optionale Funktion im Simulink-Modell im Zuge der Codegenerierung komplett entfernt werden, so dass für sie kein Code auf dem Steuergerät existiert, oder soll die Funktion lediglich über einen Parameter abgeschaltet werden, so dass sie gegebenenfalls zu einem späteren Zeitpunkt, z.B. in der Servicewerkstatt aktiviert werden kann. Der Zeitpunkt, zu dem Varianz entfernt werden kann, hat somit einen erheblichen Einfluss auf den Umfang der eingebetteten Software, die sich auf dem Steuergerät befindet. Aufgrund der hohen Stückzahlen von Steuergeräten haben selbst kleine Unterschiede im Umfang der Software einen erheblichen Einfluss auf die Produktionskosten der Steuergeräte. Eine flexible Wahl des Zeitpunktes erlaubt es für jede Varianz somit einen optimalen Zeitpunkt im Modell festzulegen.

Im Folgenden wird ein Konzept vorgestellt, welches einerseits eine Kapselung variabler Funktionsmodule ermöglicht, andererseits aber auch eine flexible Wahl des Zeitpunktes sicherstellt, wann Varianz entfernt werden soll. In Abschnitt 2 werden zunächst aktuelle Konzepte zur Modellierung von Varianz in Simulink beschrieben. Darauf aufbauend wird in Abschnitt 3 ein Konzept zur Handhabung variabler Funktionsmodule unter Berücksichtigung flexibler Bindezeiten vorgestellt. Die Anwendung dieses Konzeptes ist in Abschnitt 4 beschrieben. Abschnitt 5 fasst die Ergebnisse zusammen. Das beschriebene Konzept basiert auf der Werkzeugkette MATLAB R2006b, TargetLink 3.0 und pure::variants for Simulink.

ISBN 978-3-88579-286-4

2 Aktuelle Konzepte zur Modellierung von Varianz in Simulink

In Simulink stehen eine Reihe von Modellelementen zur Verfügung, die in den Softwareentwicklungsbereichen der Automobilindustrie zur Modellierung von Varianz verwendet werden. Diese lassen sich den zwei nachfolgenden Kategorien zuordnen:

a) Modellelemente zur Modelladaption

Das Entfernen von Varianz führt bei Verwendung dieser Modellelemente zu einer strukturellen Änderung des Simulink-Modells. Im Kontext von Varianz stellt Simulink aktuell den *Model Variant*-Block (ab Version R2009b) und *Variant Subsysteme* (ab Version R2010b) zur Verfügung[3].

Der *Model Variant*-Block stellt eine variantenspezifische Erweiterung des *Model Referencing* dar. Er ermöglicht Simulink-Modelle zu referenzieren, die als separate .mdl-Dateien zur Verfügung stehen und extern verwaltet werden können – z.B. unter einem eigenen Versionsmanagement.

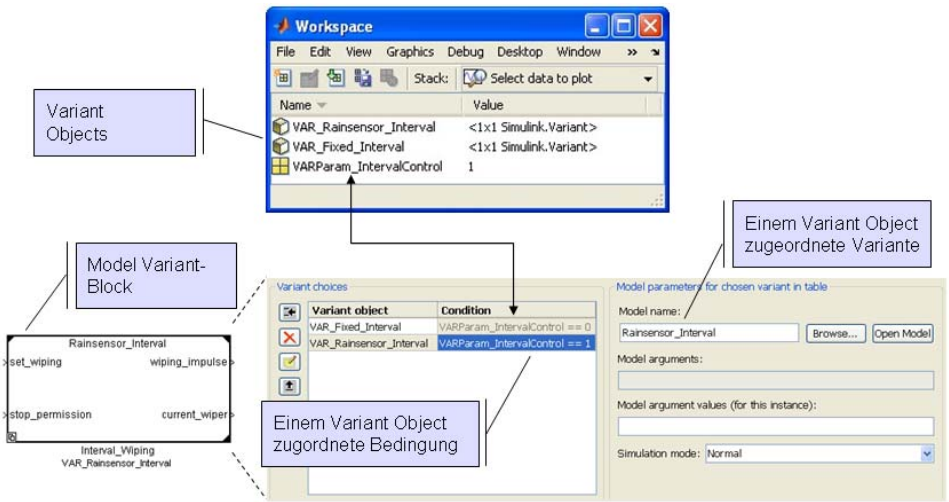

Abbildung 1: Struktur des Model Variant Blocks (gemäß [MW10])

Im Kontext von Varianz stellt die Verwendung eines Model Variant-Blocks in einem Simulink-Modell einen Variationspunkt dar; die referenzierten Teilmodelle stellen die selektierbaren Varianten dar. Zur Auswahl zwischen den Modellvarianten verwendet der

[3] Im Automobilbereich existieren Erweiterungen auf der Kombination von regulären *Subsystemen* und MATLAB-Funktionen, die ein variantenspezifisches Laden von Teilmodellen erlauben. Diese Konzepte sind mit denen des Model Variant-Blocks vergleichbar, weshalb sie hier nicht weiter betrachtet werden.

Model Variant-Block variantenspezifische Datenobjekte[4], die sog. *Variant-Objects*. Diese Datenobjekte werden im Workspace in Form eines Parameters instanziiert (s. Abbildung 1, oben) und im Model Variant-Block referenziert. Jedem Variant-Object ist eine Bedingung zugeordnet (s. Abbildung 1, unten). Ist diese Bedingung erfüllt, wird die spezifische Modellvariante (s. Abbildung 1, unten) in das Modell geladen. Es darf nur eine Bedingung zu wahr evaluieren. Zur Codegenerierung müssen die Schnittstellen zwischen dem Model Variant-Block und seinen referenzierten Modellvarianten identisch sein.

Das *Variant Subsystem* stellt eine variantenspezifische Erweiterung des *Configurable Subsystems* dar, das als Variationspunkt im Simulink-Modell instanziiert wird. Im Gegensatz zum Configurable Subsystem wird für das Variant Subsystem keine separate Bibliothek benötigt – die alternativen Modellvarianten werden innerhalb des Variant Subsystems in Form von Subsystemen modelliert. Die Auswahl einer Modellvariante wird auch hier über Variant-Objects und deren Bedingungen gesteuert.

Die Modellelemente zur Modelladaption ermöglichen die Kapselung von Funktionsmodulen. Variationspunkte und deren Modulvarianten sind auf der Basis dieser Blöcke eindeutig im Simulink-Modell als solche erkennbar. Ein wesentlicher Nachteil liegt im Zeitpunkt, zu dem Varianz entfernt werden kann. Varianz auf der Basis von Model Variant-Blöcken und Variant Subsystemen kann bei Verwendung von TargetLink lediglich auf Modellebene, bei Verwendung des Real Time Workshop Embedded Coder spätestens bei der Kompilierung des generierten Sourcecodes entfernt werden.

b) Konditionale Modellelemente

Konditionale Modellelemente ermöglichen die Steuerung von Varianz bis auf Signalebene auf Basis einer Bedingung. Beispiele für entsprechende Modellelemente sind Signal Routing-Blöcke (*Switch*- / *Multiport Switch*-Block) oder logische Gatter (*Logical Operator*-Block).

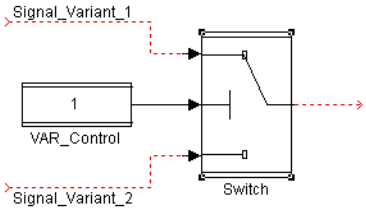

Abbildung 2: Modellierung von Varianz auf Signalebene am Beispiel eines *Switch*-Blocks

Die Eingangssignale repräsentieren alternative Varianten, aus denen über eine Bedingung eine Variante selektiert wird. Für die Bedingung bietet sich die Verwendung

[4] Datenobjekte sind Instanzen von Simulink-Datenklassen, deren Eigenschaften über Methoden und Attribute beschrieben werden.

ISBN 978-3-88579-286-4

eines Kontrollblocks an. Dieser kann beispielsweise als *Constant*-Block realisiert sein. Je nach Wert des Kontrollblocks wird die zugehörige Signalvariante verwendet. Abbildung 2) zeigt einen *Switch*-Block, dessen Ausführung über einen *Constant*-Block gesteuert wird. Der *Constant*-Block kann die selektierte Variante als Wert beinhalten oder in Form eines Parameters referenzieren.

Um konditionale Modellelemente zur Auswahl einer Signalvariante von der „regulären" Steuerung des Signalflusses zu unterscheiden sind weiterführende Konzepte erforderlich. [DW09] stellen ein Konzept vor, wie konditionale Modellelemente auf der Basis einer Blockannotation variantenspezifisch erkannt und gesteuert werden können.

Im Unterschied zu den Modellelementen zur Modelladaption existieren für konditionale Modellelemente Konzepte, um deren Varianz flexibel zu verschiedenen im Automobilbereich relevanten Zeitpunkten zu entfernen [TL06, BW09]. Im Hinblick auf die Modellierung von Funktionsmodulen zeigt sich in der Praxis jedoch, dass Funktionen und Varianz auf der Grundlage konditionaler Modellelemente im Simulink-Modell vermischt werden. Auf diese Weise entstehen Abhängigkeiten, die eine eigenständige Weiterentwicklung der Funktionen extrem aufwändig und fehleranfällig machen.

Einen ersten Schritt hin zur Modellierung variabler Funktionsmodule bildet das *Enabled Subsystem*. Einerseits kann durch das Enabled Subsystem ein Funktionsalgorithmus gekapselt werden, anderseits gehört das Enabled Subsystem zu den konditionalen Modellelementen, dessen Funktionsalgorithmus über einen Kontrollblock aktiviert bzw. deaktiviert werden kann. Enabled Subsysteme eignen sich somit insbesondere zur Modellierung optionaler Funktionsmodule. Zur Modellierung einer Gruppe alternativer Funktionsmodule, aus der genau ein Funktionsmodul ausgewählt werden muss, und Kapselung dieser Varianz innerhalb eines Variationspunktes, d.h. in Form eines einzelnen Modellelementes, sind weiterführende Konzepte erforderlich, die im Folgenden vorgestellt werden.

3 Konzept zur Handhabung variabler Funktionsmodule

Auf Basis der betrachteten Konzepte zur Modellierung von Varianz ergeben sich für die Handhabung variabler Funktionsmodule im Wesentlichen die beiden folgenden Fragestellungen:

- Wie können variable Funktionsmodule an dezidierten Stellen im Simulink-Modell, vergleichbar den Modellelementen zur Modelladaption, in „gekapselter" Form auf Basis eines Modellelementes verwendet werden?

- Wie kann Varianz auf Basis der verwendeten Funktionsmodule im Simulink-Modell, vergleichbar den konditionalen Modellelementen, flexibel zu spezifizierten Zeitpunkten entfernt werden?

Im Sinne einer optimierten Codegenerierung sollten hierbei ausschließlich Simulink-native Modellelemente verwendet werden.

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4
weitere Artikel online:
http://informatik2011.de/519.html

3.1 Konzept zur Realisierung von Variationspunkten

Ausgangspunkt für die Verwendung variabler Funktionsmodule im Simulink-Modell ist der Variationspunkt. Dieser umfasst nach [Be03] einen „… abgegrenzten und damit klar identifizierbaren Bereich […], in dem Anpassungen vorgenommen werden …“. Im Kontext der Handhabung variabler Funktionsmodule wird der Variationspunkt mittels eines Templates realisiert. Das *Variationspunkt-Template* steht als variantenspezifisch annotiertes Subsystem zur Verfügung, welches im Modell zu einem Variationspunkt instanziiert und an seinen Kontext konfiguriert wird. Der Variationspunkt besteht im Wesentlichen aus den drei nachfolgenden Komponenten (s. Abbildung 3).

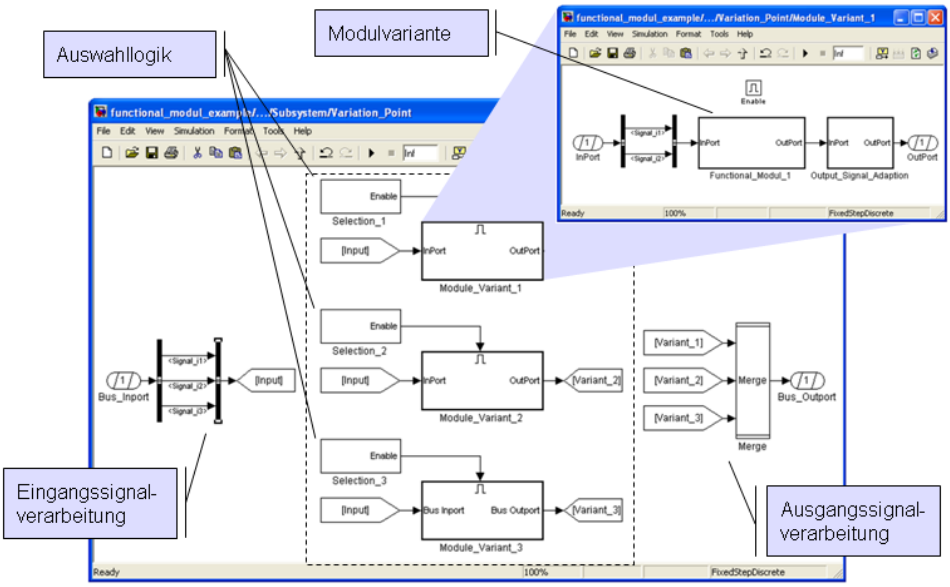

Abbildung 3: Struktur eines Variationspunktes

- Alternative bzw. optionale Funktionsmodule, sog. *Modulvarianten*, werden im Variationspunkt auf Basis von Enabled Subsystemen eingefügt. Die Enabled Subsysteme kapseln die Modulvarianten, indem sie nach außen eine einheitliche Schnittstelle bereitstellen. Auf diese Weise können Funktionsmodule ohne weitere Anpassungen und Einschränkungen bzgl. der Modellierung verwendet werden.

- Eine Auswahllogik steuert, inwieweit eine Modulvariante aktiviert oder deaktiviert wird und somit die Varianz eines Variationspunktes entfernt wird. Die Auswahllogik beeinflusst wesentlich den Zeitpunkt, wann die Varianz entfernt wird (s. Abschnitt 3.2).

- Die Eingangs- und Ausgangssignalverarbeitung verbindet den Variationspunkt (und somit die einzelnen Modulvarianten) mit dem umgebenden Simulink-Modell. Die Art und Weise der Modellierung der Eingangs- und Ausgangssignalverarbeitung stellt sicher, dass nach Auswahl einer gültigen Modulvariante das Simulink-Modell zu jedem Zeitpunkt ohne weitere Anpassungen simulierbar ist.

Das obige Konzept des Variationspunktes erlaubt die Kapselung variabler Funktionsmodule in einem variantenspezifischen Modellelement. Die Auswahl der zugehörigen Funktionsmodule findet im Rahmen der Instanziierung des Variationspunkt-Templates statt (s. Abschnitt 4). Die Auswahllogik wird in Abhängigkeit der Zahl der Modulvarianten generiert und konfiguriert. Die Struktur des Variationspunktes wird dem Entwickler gegenüber nicht dargestellt: Beim Öffnen eines maskierten Variationspunktes im Simulink-Modell wird direkt die aktuell selektierte Modulvariante geöffnet.

3.2 Konzept zur Spezifikation von Bindezeiten

Der Variationspunkt stellt eine Art Platzhalter dar, der im Laufe seines Lebenszyklus im Hinblick auf eine konkrete Variante konfiguriert werden muss; d.h. man muss sich für eine zugehörige Variante entscheiden. Diese Entscheidung wird als *Bindung* bezeichnet; der Zeitpunkt, zu dem diese Entscheidung getroffen wird, spezifiziert die *Bindezeit*.

In der Literatur existieren verschiedene Bindezeitmodelle, wie z.B. in [BF05, Ge03, KC90], insbesondere im Hinblick auf eingebettete automotive Software [FL02]. Das Bindezeitmodell, welches im Automobilbereich im Wesentlichen angewendet wird, basiert auf FODA [KC90] und unterscheidet die nachfolgenden Bindezeiten:

- *ModelConfigurationTime* kennzeichnet die Bindezeit, zu der aus einem variantenreichen Simulink-Modell auf Modellebene eine konkrete Modellvariante konfiguriert wird[5].

- Entscheidungen zur *CompileTime* werden zum Zeitpunkt der Kompilierung getroffen. Dies schließt auch das Preprocessing ein.

- Zur *LoadTime* wird die Entscheidung zu Beginn der Ausführung des Systems getroffen. Im Bereich eingebetteter Software betrifft dies beispielsweise Varianz, die beim „Start-Up“ des Systems aus der Systemumgebung oder mittels dezidierter Parametersätze entfernt wird.

- *RunTime* kennzeichnet die Bindezeit, bei der die Entscheidung für die eine oder andere Variante noch während der Ausführung des Systems getroffen werden kann.

In Simulink haben die Auswahl und Konfiguration der Modellelemente einen wesentlichen Einfluss auf die Bindezeit von Varianz. So wird Varianz auf der Basis

[5] Genau genommen muss zwischen der *ModelConfigurationTime* und einer *CodeGenerationTime* weiter differenziert werden. In der Regel wird allerdings zur Codegenerierung nur die Variabilität entfernt, die durch die Konfiguration des Simulink-Modells spezifiziert wurde.

ISBN 978-3-88579-286-4

konfigurierbarer Subsysteme ausschließlich auf Modellebene, d.h. noch vor der Codegenerierung, entfernt. Die in diesem Kontext verwendeten Enabled Subsysteme unterstützen in Bezug auf das obige Bindezeitmodell sämtliche Bindezeiten. Wie die Modellelemente im Seriencode abgebildet werden, ist allerdings abhängig von der Optimierung durch den verwendeten Codegenerator.

Im Rahmen der Codegenerierung wird Varianz auf Modellebene auf Daten- und Kontrollstrukturen im Sourcecode abgebildet, wie beispielsweise #define oder if-else-Konstrukte. Das Entfernen dieser Varianz im Sourcecode geschieht über Wertzuweisungen an Objekte (Variablen oder Konstanten), die in Bedingungen, wie z.B. in einer if-Anweisung, verwendet werden. Die Deklaration der Objekte ist daher entscheidend dafür, wann Varianz entfernt wird. Abbildung 4) zeigt dies am Beispiel des Enabled Subsystems.

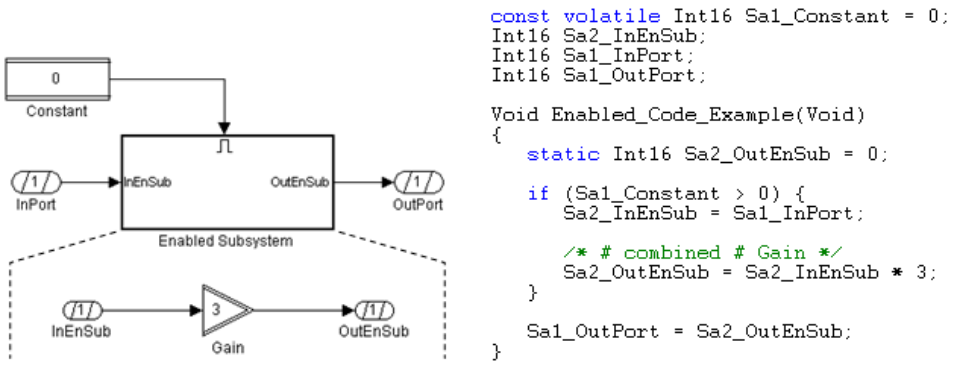

Abbildung 4: Einfluss der Konstantendeklaration auf die Codegenerierung

Wird der Constant-Block aus Abbildung 4) beispielsweise als Objekt der Form *const volatile* im Sourcecode deklariert, bedeutet dies für den Codegenerator, dass der Wert des Objektes noch auf dem Steuergerät, beispielsweise im Rahmen einer Interrupt-Bearbeitung, verändert werden kann. Das Enabled Subsystem wird somit auf jeden Fall in den Sourcecode übernommen. Wird der Constant-Block hingegen als lokales Objekt im Sourcecode deklariert, kann dessen Wert zu einem späteren Zeitpunkt nicht mehr verändert werden. Abhängig vom Wert des Constant-Blocks wird das Enabled Subsystem im Sourcecode berücksichtigt oder nicht. [BW09] beschreiben, wie auf Basis des TargetLink-Codegenerators die obigen Bindezeiten auf spezifische Deklarationen von Objekten im Sourcecode abgebildet werden können.

Im Kontext der Handhabung variabler Funktionsmodule wird die Konfiguration der Bindezeit im Simulink-Modell mittels der Auswahllogik gesteuert. Jeder Modulvariante – jeweils über ein Enabled Subsystem gekapselt – ist eine Auswahllogik zugeordnet. Die Auswahllogik ist durch einen für die Modulvariante eindeutigen Identifier gekennzeichnet. Abbildung 5) zeigt deren Aufbau am Beispiel der ersten Modulvariante.

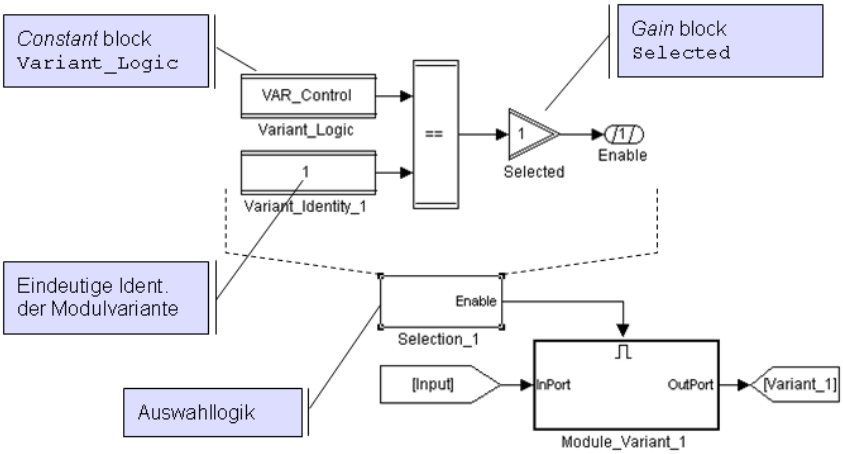

Abbildung 5: Struktur der Auswahllogik

Die Konfiguration der Bindezeit ist als zweistufiger Prozess realisiert. Einem Variationspunkt sind sämtliche alternativen Funktionsmodule zugeordnet. Über den *Gain*-Block `Selected` werden im Rahmen der Codegenerierung zunächst die Modulvarianten entfernt, die nicht in den Sourcecode übernommen werden sollen. Zu diesem Zweck ist der *Gain*-Block so konfiguriert, dass im Sourcecode ein lokales Objekt deklariert wird, dessen Wert nicht mehr verändert werden kann. Besitzt der Block den Wert 0, d.h. eine Multiplikation mit dem Wert 0, wird die zugehörige Modulvariante im Rahmen der Codegenerierung und -optimierung auf diese Weise im Sourcecode nicht abgebildet (s.a. *ModelConfigurationTime*). Diese Konfiguration findet insbesondere dann Verwendung, wenn die zugehörige Modulvariante ausschließlich zur Simulation verwendet werden soll.

Für Modulvarianten, die im Rahmen der Codegenerierung im Sourcecode berücksichtigt werden sollen, besitzt der *Gain*-Block den Wert 1 (s. Abbildung 5). Die Information, welche Modulvarianten für die Codegenerierung und welche ausschließlich zur Simulation verwendet werden sollen, wird im Rahmen der Instanziierung des Variationspunkt-Templates angegeben und im Variationspunkt gespeichert (s. Abschnitt 4).

Die Bindezeit der verbleibenden Modulvarianten findet über die Konfiguration des Constant-Blocks `Variant_Logic` statt. Dieser Block besitzt für jede Auswahllogik dieselbe Konfiguration in Bezug auf die zugeordnete Bindezeit und den Wert: Die dem Block zugeordnete Bindezeit entscheidet, wann die Varianz entfernt wird. Der Wert des Blockes, der über einen sog. *Variantenparameter* aus dem Modelworkspace referenziert wird, entscheidet über die Auswahl der auszuführenden Variante. Hierfür wird dieser Wert mit dem Identifier der Modulvariante verglichen.

4 Anwendung der Konzepte

Das oben beschriebene Konzept erlaubt eine systematische Vorgehensweise für die Wiederverwendung variabler Funktionsmodule. Stellvertretend sei nachfolgend die Modellierung und Konfiguration von Variationspunkten im Simulink-Modell betrachtet.

Ausgangspunkt der Modellierung eines variantenreichen Simulink-Modells auf der Basis variabler Funktionsmodule ist eine Verwaltung der Funktionsmodule z.B. in Form einer Variantenbibliothek (s. Abbildung 6, links oben). Hierbei wurde berücksichtigt, dass die Funktionsmodule auf Basis separater .mdl-Dateien verwaltet werden sollen, z.B. mittels eines externen Versionsmanagements. In der Variantenbibliothek sind die alternativen bzw. optionalen Funktionsmodule in Gruppen geordnet.

Die Verwendung der Funktionsmodule im Simulink-Modell geschieht nun auf Basis des Variationspunkt-Templates, welches in das Modell eingefügt wird (s. Abbildung 6, unten). Über einen variantenspezifischen Dialog (s. Abbildung 6, rechts unten) wird das Template anschließend konfiguriert. Der Dialog wird hierbei über eine Open-Callback-Funktion aufgerufen.

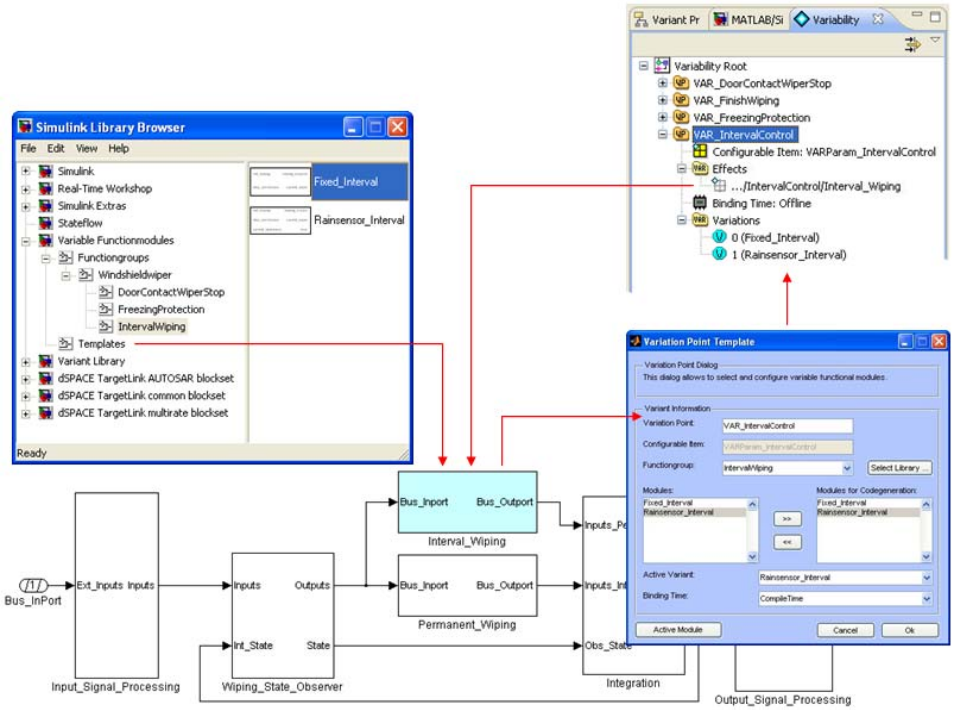

Abbildung 6: Modellierung von Varianz auf Basis variabler Funktionsmodule

ISBN 978-3-88579-286-4

Im Dialog werden zum Variationspunkt nachfolgende Informationen erfragt:

- Auswahl einer Gruppe variabler Funktionsmodule aus der Variantenbibliothek,
- Auswahl, welche Funktionsmodule aus dieser Gruppe ausschließlich zur Simulation bzw. auch zur Codegenerierung verwendet werden,
- die aktive Variante und
- die Bindezeit bezüglich der Funktionsmodule, welche zur Codegenerierung verwendet werden.

Aus diesen Informationen wird ein Variantenparameter im Modelworkspace angelegt und der Variationspunkt im Simulink-Modell aus dem Template mit der jeweiligen Auswahllogik, der Eingangs- und Ausgangssignalverarbeitung sowie der ausgewählten Gruppe variabler Funktionsmodule instanziiert.

Zur Darstellung der Varianzinformation existiert eine eigene Sicht auf das Simulink-Modell in Form einer Baumstruktur (s. Abbildung 6, rechts oben). Diese kann nun auf Basis der verfügbaren Informationen automatisch befüllt werden. Somit wird eine explizite Darstellung funktionaler Varianz im Simulink-Modell ermöglicht. Durch Auswahl eines Variationspunktes in dieser Sicht gelangt man direkt zum entsprechenden Block im Simulink-Modell.

Variable Funktionsmodule können selbst wieder Varianz beinhalten. Das vorgestellte Konzept integriert sich nahtlos in bereits existierende Konzepte zur Modellierung von Varianz auf Basis konditionaler Modellelemente (s. [DW09], [BW09]). Für die Implementierung des Konzeptes wurde das Werkzeug pure::variants for Simulink [PS11] verwendet. Neben der Modellierung von Varianz auf Basis konditionaler Modellelemente unterstützt dieses Werkzeug zudem das Management von Varianz in Form von Merkmalmodellen und die Definition von Abhängigkeiten zwischen Merkmalen und Variationspunkten im Simulink-Modell.

5 Zusammenfassung

Gerade bei einer rasant steigenden Zahl modellbasierter Funktionen ermöglicht das vorgestellte Konzept eine systematische Handhabung von Varianz in Simulink im Kontext einer funktionsorientierten Sichtweise. Im Gegensatz zu aktuell existierenden Ansätzen ermöglicht das vorgestellte Konzept die Kapselung wiederverwendbarer variabler Funktionsmodule zu Variationspunkten im Simulink-Modell bei gleichzeitiger flexibler Konfiguration bezüglich der im Automobilbereich relevanten Bindezeiten.

Das obige Konzept ergänzt sich nahtlos mit bisherigen Ansätzen zur Modellierung von Varianz in Simulink. Existierende Funktionen in Simulink können werkzeugunterstützt auf einfache Weise in das vorgestellte Konzept integriert werden. Durch die ausschließliche Verwendung nativer Simulink-Blöcke bleiben die Möglichkeiten des Codegenerators zur Optimierung des Sourcecodes erhalten.

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4
weitere Artikel online:
http://informatik2011.de/519.html

Das Konzept wird derzeit in das Werkzeug pure::variants for Simulink, welches aktuell in einem Serienprojekt der Mercedes Pkw-Entwicklung angewendet wird, integriert und an einem produktiv eingesetzten Modell evaluiert.

Das Bindezeitkonzept basiert hierbei auf verfügbaren Konzepten der Codegenerierung mit dSpace TargetLink. Im Wesentlichen steuert der Codegenerator die Optimierung des Sourcecodes und somit die Entscheidung, welche Codesegmente im Sourcecode erscheinen bzw. entfernt werden. Unabdingbar sind daher fest vorgegebene Bindezeiten, die auf spezifische Codemuster umgesetzt werden. Die Abbildung von Bindezeiten auf Codemuster im Kontext der Modellierung von Varianz bedarf auf jeden Fall weiterer Forschungsarbeit.

Literaturverzeichnis

[Be03] Beuche, D.: Composition and Construction of Embedded Software Families. Dissertation, Universität Magdeburg, 2003.

[BF05] Bayer, J.; Forster, T.; Kiebusch, S.; Lehner, T.; Ocampo, A.; Weiland, J.: Feature- und Entscheidungsmodell-basierte Varianteninstanziierung im PESOA-Prozess. PESOA-Report Nr. 21/2005, 2005.

[BW09] Beuche, D.; Weiland, J.: Managing Flexibility: Modeling Binding-Times in Simulink. 5th European Conference on Model-Driven Architecture Foundations and Applications (ECMDA), Enschede, The Netherlands, 2009.

[CK04] Creutzburg, U.; Kalix, E.: Process Integration of Model-Based Design and Production-Code Generation in the Multi-User / Multi-Project Development Environment at Continental Teves – Part 2. Proceedings of the Int. Automotive Conference, 2004.

[DL08] Dziobek, C.; Loew, J.; Przystas, W.; Weiland, J.: Von Vielfalt und Variabilität – Handhabung von Funktionsvarianten in Simulink-Modellen. Elektronik automotive, 2/2008.

[DW07] Dziobek, C.; Wohlgemuth, F.: Einsatz von AUTOSAR bei der Modellierung von Komfort- und Innenraumfunktionen. dSpace User Conference, München, 2007.

[DW09] Dziobek, C.; Weiland, J.: Variantenmodellierung und -konfiguration eingebetteter automotive Software mit Simulink. In: Tagungsband des Dagstuhl-Workshops: Modellbasierte Entwicklung eingebetteter Systeme V (MBEES 2009), 22.-24. April 2009, Dagstuhl, 2009, S. 36-45.

[FL02] Fritsch, C.; Lehn, A.; Strohm, T.: Evaluating Variability Implementation Mechanisms. Proceedings of the 2nd Int. Workshop on Product Line Engineering – The Early Steps (PLEES'02), Seattle, USA, 2002.

[Ge03] Geyer, L.: Variabilitätsmanagement in Produktfamilien. Dissertation, Universität Kaiserslautern, 2003.

[KC90] Kang, K.C.; Cohen, S.G.; Hess, J.A.; Nowak, W.E.; Peterson, A.S.: Feature Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Carnegie Mellon University, Pittsburgh, PA, 1990.

[MW10] The MathWorks, Inc.: Simulink 7 – User's Guide. The Mathworks, Inc., Natick, MA, 2010.

[PS11] pure-systems, pure::variants Eclipse Plugin User Guide, 2011

[SZ03] Schäuffele, J.; Zurawka, T.: Automotive Software Engineering – Grundlagen, Prozesse, Methoden und Werkzeuge. Vieweg, Wiesbaden, Juli 2003.

[TD11] Thomas, J.; Dziobek, C.; Hedenetz, B.: Variability Management in the AUTOSAR-based Development of Applications for In-Vehicle Systems. 5th International Workshop on Variability Modelling of Software-intensive Systems VaMoS'11, 27.-29.01.2011, Namur, Belgium, 2011.

[TL06] dSpace GmbH: TargetLink Advanced Practices Guide – for TargetLink 2.2. dSpace, Paderborn, 2006.

[WK09] Weber, M.; Kleinwechter, H.: Varianten in der Automobilelektronikentwicklung – Herausforderungen und Lösungsansätze. 3rd Model-Driven Development & Product Lines Konferenz 2009: Synergie and Experience, 23.-25. März 2009, Leipzig, 2009.

ISBN 978-3-88579-286-4