Von Vielfalt und Variabilität

Handhabung von Funktionsvarianten in Simulink-Modellen

Heutige Fahrzeuge sind geprägt durch eine Vielzahl von Funktionsvarianten. Diese Variabilität spiegelt sich zwangsläufig in der modellbasierten Software-Entwicklung wider. Unabdingbar sind daher Konzepte für einen systematischen Umgang mit der Variabilität in Funktionsmodellen. Die Trennung in zentrale und modellspezifische Variabilitätsinformation ermöglicht in Simulink eine uniforme Handhabung und somit eine explizite Darstellung verteilter modellbasierter Variabilität.

Von Christian Dziobek, Joachim Loew, Wojciech Przystas und Jens Weiland

Die Automobilindustrie ist geprägt durch eine hohe Zahl variabler Fahrzeugfunktionen. Vergleicht man die Funktionsvariabilität einer C-Klasse mit ihrem Absatz im Jahr 2006, so dürften in diesem Jahr statistisch gesehen keine zwei identischen Fahrzeuge produziert worden sein. Die Quelle der Variabilität ist vielschichtig: Von der Anpassung der Fahrzeuge an unterschiedliche Märkte mit ihren länderspezifischen Vorgaben und ihren kundenspezifischen Wünschen bis zu den technischen Möglichkeiten durch elektronische Systeme. Da mittlerweile ein Großteil der Funktionen im Fahrzeug software-basiert ausgeführt wird, nimmt die eingebettete Software eine zentrale Rolle bei der Realisierung der Fahrzeugfunktionen ein. Funktionsvariabilität im Fahrzeug führt somit zwangsläufig zu Variabilität in der Software.

Ein wirtschaftlicher Umgang mit der Variabilität erfordert eine gezielte Wiederverwendung von Software-Artefakten wie Architekturen, Spezifikationen, Komponenten oder auch Dokumentation und Tests. Diese kann jedoch nur dann erreicht werden, wenn strukturelle Variabilitäten innerhalb von Software-Artefakten und auch kompositorische Variabilitäten systematisch beschrieben werden.

Die Entwicklung eingebetteter Software im Automobilbereich geschieht zunehmend modellbasiert und durch den Einsatz von Codegeneratoren, etwa mit der Werkzeugkette Matlab/Simulink/Stateflow von The MathWorks. Diese grafischen Modellierungssprachen ermöglichen Spezifikation, Modellierung und Simulation signalfluss- und zustandsorientierter Systeme. Die Implementierung kann mit Codegeneratoren, wie dem Real-time Workshop Embedded Coder von The MathWorks oder TargetLink von dSpace, durchgeführt werden.

Simulink-Modelle sind aus der Verschaltung von Elementar- und Zustandsdiagramm-Blöcken aufgebaute Signalflussgraphen. Komplexe Teilfunktionen können durch Subsysteme gekapselt werden, die wiederum als Signalflussgraph modelliert sein können. Die Signalverbindungen der Blöcke repräsentieren die Daten, die während der Simulation eines Modells ausgetauscht werden. In Simulink stehen zahlreiche Blöcke für verschiedene Funktionen zur Verfügung; beispielsweise für logische Operationen oder das Routing von Signalen.

Die Signalflussgraphen beschreiben, vorrangig auf Basis der Elementarblöcke, den funktionalen Algorithmus für ein Echtzeit-System. In der Praxis zeigt sich jedoch, dass bei der Erstellung von wiederverwendbaren

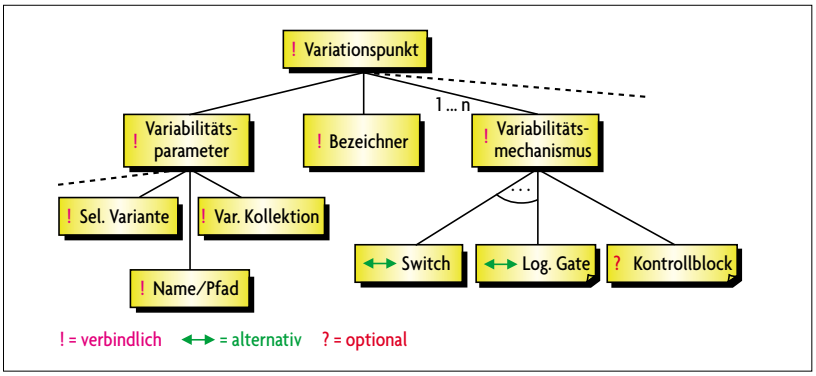

Bild 1. Struktur eines Variationspunktes.

und konfigurierbaren Funktionsmodulen der Basis-Algorithmus durch Aspekte der Modellierung und Konfiguration von Funktionsvarianten überlagert wird. Diese beiden Aspekte können auf Grund der heute nur unzureichend vorhandenen Beschreibungsmittel nicht eindeutig und prozesssicher beschrieben werden.

Der vorliegende Artikel beschreibt einen Ansatz, wie diese Defizite innerhalb von Signalflussgraphen, die mit Simulink/Stateflow erstellt wurden, verringert werden können. Der Ansatz betrachtet insbesondere, welche Informationen zu Elementarblöcken gespeichert werden müssen, um Variabilität zu beschreiben, und wie diese Informationen abgelegt werden können.

Erst die systematische Betrachtung von Variabilität ermöglicht

- die einheitliche Beschreibung und somit eine einheitliche Konfiguration von Funktionsvarianten;
- deren variantenspezifische Darstellung im Signalflussgraphen, d.h., die Unterscheidung zwischen normalen und variantenspezifischen Blöcken;
- das Erkennen von Abhängigkeiten zwischen Funktionsvarianten.

Die vorliegende Implementierung basiert auf der Werkzeugkette Matlab 7.1 und TargetLink 2.1. Dies stellt jedoch keine Einschränkung dar: Eine Implementierung der Konzepte ist auch rein auf der Matlab-Werkzeugkette möglich und wird an entsprechender Stelle erläutert.

Variabilitätsmodellierung in Simulink

Der folgende Ansatz basiert auf dem Produktfamilien-Engineering [1]. Der Ausgangspunkt der Beschreibung von Variabilität ist der Variationspunkt (**Bild 1**). Dieser kapselt die Variabilitätsinformationen konkreter Funktionsvarianten, mit denen das Simulink-Modell angereichert werden soll. Neben einem eindeutigen Bezeichner umfasst es im Wesentlichen den Variabilitätsparameter und den Variabilitätsmechanismus.

Der Variabilitätsparameter stellt die eigentliche „Stellschraube“ im Simulink-Modell dar, der konfiguriert wird, um eine Funktionsvariante auszuwählen. Er enthält eine Wertemenge, die Variantenkollektion, aus der er konfiguriert werden kann, und einen Wert aus dieser Wertemenge, die selektierte Variante.

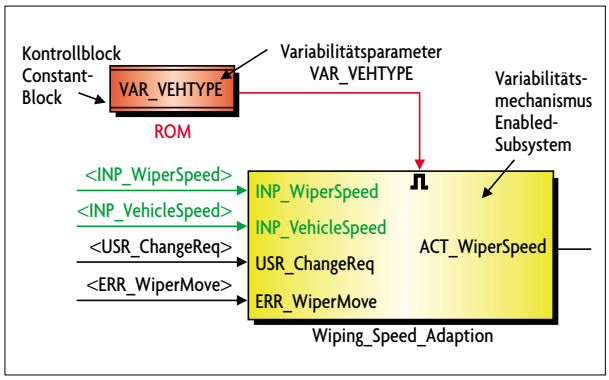

Bild 2. Beispiel eines Variationspunktes. Die Steuerung erfolgt hier über den Kontrollblock.

Der Variabilitätsmechanismus beschreibt, wie Funktionsvariabilität an einer konkreten Stelle im Simulink-Modell aufgelöst wird. Er stellt sicher, dass nach der Konfiguration des Variabilitätsparameters die selektierte Funktionsvariante ausgeführt wird. Über die Blockbibliothek stehen in Simulink mehrere Blöcke zur Verfügung, die zur Modellierung des Variabilitätsmechanismus herangezogen werden können. Beispiele hierfür sind:

- Bedingt ausführbare Subsysteme (Enabled-Subsystem, Function-Call-Subsystem).
- Subsysteme (If-Block, „Switch Case“-Block).
- „Signal Routing“-Blöcke (Switch-Block, „Multiport Switch“-Block, „Manual Switch“-Block).
- Logikgatter (AND-Block, OR-Block).
- Konfigurierbare Subsysteme (Configurable Subsystem).

Jeder dieser Blöcke hat hierbei seinen eigenen Mechanismus zum Auflösen variabler Funktionen. So wird der Funktionsumfang, der mittels eines Enabled-Subsystems gekapselt ist, je nach Wert des Enabled-Signals aktiviert oder deaktiviert. Das Enabled-Subsystem eignet sich auf diese Weise besonders zur Modellierung optionaler Funktionen. Auf ähnliche Weise können über die logische Verknüpfung mittels AND- oder OR-Block optionale Funktionen aktiviert oder deaktiviert werden. Beim Switch-Block wird über das Control-Signal eine Variante selektiert – vergleichbar der „Switch Case“-Kontrollstruktur in C –, wodurch sich der Switch-Block insbesondere zur Modellierung alternativer Funktionen eignet.

Die Mehrzahl dieser Blöcke erfordert ein Eingangssignal, welches die Ausführung des Blocks steuert. Zur Auswahl einer Variante bietet sich hier die Verwendung eines Kontrollblocks an. Dieser kann als Constant-Block oder als „Data Store Read“-Block realisiert sein. Je nach Wert des Kontrollblocks wird die zugehörige Variante ausgeführt.

Eine Ausnahme bildet hier das Configurable Subsystem, dessen Variante über den Wert des Blockparameters *BlockChoice* selektiert wird [2]. Dieser enthält allerdings eine Reihe von

Einschränkungen bezüglich des Bindezeitpunktes, des Debuggings und der Darstellung der Schnittstelle des konfigurierbaren Subsystems.

Bild 2 zeigt ein Enabled-Subsystem, dessen Ausführung über einen Constant-Block gesteuert wird. Der Constant-Block kann die selektierte Variante als Wert beinhalten. Der Variabilitätsparameter wäre in diesem Fall der Blockparameter *Value* des Constant-Blocks.

Da variable Funktionen an mehreren Stellen im Modell ihre Wirkung zeigen können, empfiehlt es sich jedoch, einen zentralen Variabilitätsparameter im Model- oder Base-Workspace anzulegen, der anschließend vom Kontrollblock referenziert wird. In Bild 2 ist dies der Workspace-Parameter VAR_VEHTYPE.

„Separation of concerns" der Variabilitätsinformation

Nachdem bekannt ist, welche Informationen zur Modellierung von Variabilität in Simulink relevant sind, stellt sich die Frage, wie diese Informationen abgelegt werden können. Ziel ist eine einheitliche Handhabung bezüglich der Variabilität, insbesondere der unterschiedlichen Variabilitätsmechanismen. Dabei interessiert in erster Linie, welche Variabilität im Simulink-Modell vorhanden ist (Variationspunkt); erst danach, wie sie aufgelöst wird (Variabilitätsmechanismus). Diese separate Betrachtung führt zu einer Trennung der Variabilitätsinformation in einen allgemeinen und einen spezifischen Anteil (**Bild 3**):

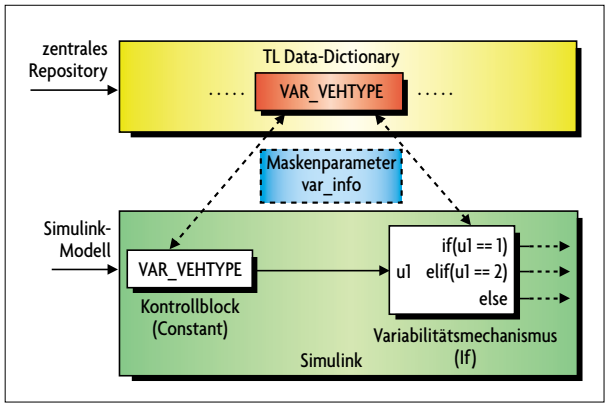

Bild 3. „Separation of concerns" bezüglich Variabilitätsinformationen.

- Die Referenzen auf Variabilitätsparameter, Variantenkollektion und selektierte Variante repräsentieren die allgemeinen Variabilitätsinformationen und können als Objekte gleichen Typs in einem zentralen Variantenbestand gespeichert werden. Die Objekte besitzen den eindeutigen Bezeichner eines Variationspunktes und sind so für verschiedene im Modell befindliche Kontrollblöcke und Variabilitätsmechanismen referenzierbar. Im vorliegenden Fall sind diese Daten als Objekt im TargetLink Data Dictionary modelliert. Alternativ können diese auch als Matlab-Struktur, Simulink-Datenklasse oder als Java-Objekt gekapselt und als Parameter im Base- oder Model-Workspace instanziiert werden.
- Der spezifische Anteil betrifft den Variabilitätsmechanismus und den zugehörigen Kontrollblock. Allen Blöcken, die Variabilität im Simulink-Modell ausdrücken, wurde ein Maskenparameter VAR_INFO hinzugefügt. Dieser kennzeichnet einen Block als variantenspezifisch. Mit dem Wert des Maskenparameters VAR_INFO wird zudem auf das zugehörige Objekt im zentralen Variantenbestand verwiesen. Maskenparameter haben den Vorteil, dass diesen ein eindeutiger Be-

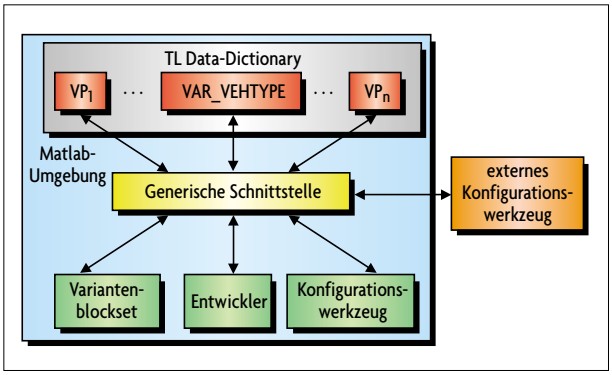

Bild 4. Generische Schnittstelle für einen transparenten Zugriff auf Variabilitätsinformationen.

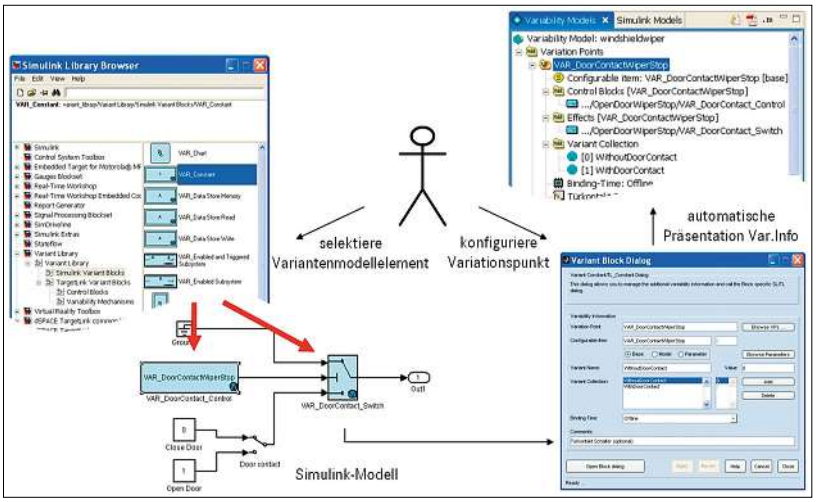

Bild 5. Entwurf variabler Funktionen mit Hilfe des Varianten-Blocksets.

zeichner für die Variabilitätsinformation zugeordnet werden kann. Blockparameter, wie Tag oder Description, können hingegen bereits für andere Zwecke verwendet worden sein.

Erst die Aufteilung der Variabilitätsinformation in einen gemeinsamen und einen spezifischen Anteil erlaubt einen uniformen Zugriff auf die Variabilitätsinformation. Dieser ist über eine generische Schnittstelle auf den zentralen Variantenbestand realisiert (**Bild 4**). Auf Basis dieser Schnittstelle können verschiedene Blöcke, die ihren eigenen Mechanismus zur Auflösung variabler Funktionen haben, als Variabilitätsmechanismus verwendet werden. Für diese Blöcke ist die Struktur eines Variationspunktes identisch.

Die generische Schnittstelle ermöglicht so einen transparenten Zugriff auf die Variabilität im Simulink-Modell. Der Zugriff kann direkt vom Entwickler über Matlab-Funktionen oder von Konfigurationswerkzeugen genutzt werden.

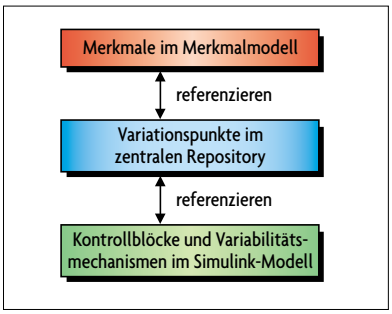

Bild 6. Abstraktionsebenen vom Merkmal- zum Simulink-Modell.

Zur Unterstützung des Modellierungsprozesses wurde ein eigenes Varianten-Blockset entwickelt. Dieses umfasst allgemeine Blöcke der Simulink-Bibliothek, die zur Modellierung von Variabilität speziell annotiert wurden. Der „Callback"-Parameter *OpenFcn* ermöglicht hierbei die Konfiguration der Blöcke über einen variantenspezifischen Dialog.

Anwendungsfälle im modellbasierten Entwicklungsprozess

Im Folgenden wird die Modellierung der Variabilität unter Verwendung des Variantenblocksets näher erläutert (**Bild 5**). Während des Entwurfs werden auf Basis des Variantenblocksets variantenspezifische Blöcke in das Simulink-Modell eingefügt (Bild 5, links unten). Über einen variantenspezifischen Dialog (Bild 5, rechts unten) werden diese Blöcke anschließend konfiguriert. Der Dialog wird über eine „Open Callback"-Funktion *OpenFcn* aufgerufen. Dabei werden bereits existierende Variationspunkte – Objekte aus dem gemeinsamen Variantenbestand – zur Auswahl angeboten. Bei einem neu anzulegenden Variationspunkt werden die Informationen des Variabilitätsparameters erfragt:

- Name und Pfad des Variabilitätsparameters,
- Werte, die der Parameter annehmen kann,
- die selektierte Variante.

Die Implementierung des Dialogs sieht vor, dass bei Verwendung von TargetLink-Blöcken über diesen Dialog direkt der TargetLink-Blockdialog zur Eingabe der blockspezifischen Information aufgerufen werden kann.

Zur Darstellung der Variabilitätsinformationen existiert eine eigene Sicht auf das Simulink-Modell in Form einer Baumstruktur (Bild 5, rechts oben). Diese kann auf Basis der verfügbaren Informationen automatisch befüllt werden und ermöglicht so eine explizite Darstellung der Variabilität. Durch Auswahl eines variantenspezifischen Blocks gelangt man direkt zum entsprechenden Block im Simulink-Modell.

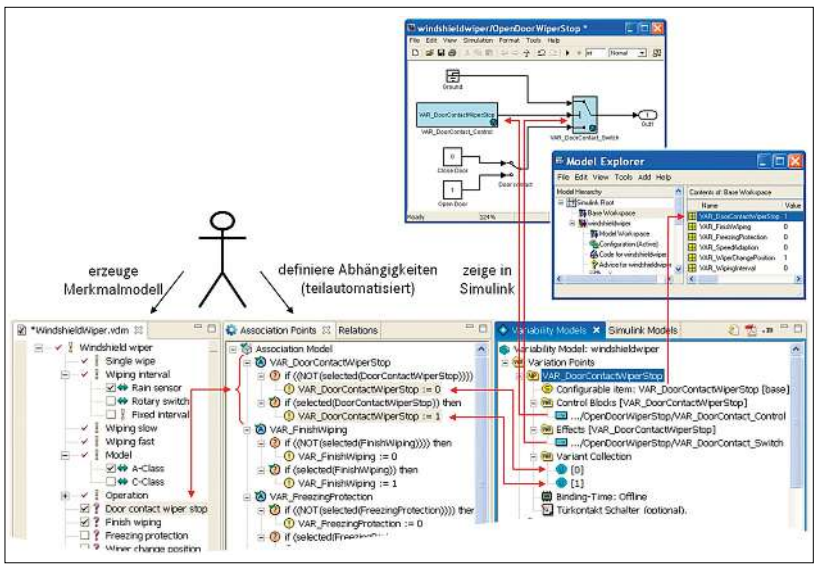

Bild 7. Konfiguration von Variabilität in Simulink-Modellen über Merkmal-Modelle am Beispiel der Türkontakt-Funktion.

Der transparente Zugriff auf die Variabilität im Simulink-Modell bietet vielfältige Möglichkeiten für Konfigurationswerkzeuge, etwa die Konfiguration über Merkmalmodelle. Diese bilden eine abstrakte Sicht auf Funktionsvarianten, unabhängig vom Simulink-Modell (**Bild 6**).

Merkmale spezifizieren gemeinsame und variable Konzepte einer Domäne aus anwendungsorientierter Sicht (in **Bild 7** die optionale Auswahl einer Türkontaktfunktion, welche die Scheibenwischer beim Öffnen einer Fronttür stoppt). Das Ergebnis ist ein umfassendes Modell dieser Konzepte sowie deren Abhängigkeiten von den Modellvarianten (Bild 7, links unten). Unterschieden werden verbindliche, alternative und optionale Merkmale sowie (1..n):m-Gruppenbeziehungen.

In [3] werden die Beziehungen zwischen den Merkmalen eines Merkmalmodells und den Variationspunkten eines Simulink-Modells beschrieben und wie diese definiert werden können (in Bild 7, unten Mitte als Association Modell bezeichnet). Diese Beziehungen beschreiben, welcher Wert einem variantenspezifischen Parameter im Simulink-Modell, bei Auswahl eines bestimmten Merkmals im Merkmalmodell zugewiesen wird. Für die Implementierung eines Konfigurationswerkzeugs wurde hierfür das Tool pure::variants von pure-systems integriert [4].

Soll ein variabler Funktionsumfang berücksichtigt werden, helfen obige Konzepte, die Qualität der modellbasierten Software weiter zu verbessern:

- Die Definition der Abhängigkeiten ermöglicht ein selektives Suchen nach verteilter Variabilität im Simulink-Modell. Dies beantwortet, welche Variationspunkte im Simulink-Modell durch welches Merkmal beeinflusst werden und von welchen Merkmalen ein Variationspunkt abhängt. So können über das Merkmalmodell gültige Konfigurationen, etwa als Parametersatz oder Konfigurationsanweisungen, automatisch erzeugt werden, um das Simulink-Modell zu konfigurieren.
- Das Auffinden von Widersprüchlichkeiten in der Konfiguration von Funktionsmodellen mit komplexen Variabilitäten wird durch die Kopplung der Information des Simulink-Modells mit denen des Merkmalmodells erleichtert.
- Die Variabilität im Simulink-Modell ist explizit sichtbar. Sie zeigt sich in der Darstellung von variantenspezifischen Blöcken im Simulink-Modell und über den gemeinsamen Variantenbestand, welcher sämtliche Variabilitäten im Simulink-Modell zentral verwaltet.

Die vorgestellten Konzepte wurden im Rahmen eines Forschungsprojektes entwickelt und werden aktuell in einem Serienprojekt der Mercedes-Pkw-Entwicklung angewendet. Hierfür wurden eigens ein Varianten-Blockset, eine API auf Basis von Matlab-Funktionen und ein Konfigurationswerkzeug implementiert. *sj*

Literatur

[1] *Czarnecki, K.; Eisenecker. U.:* Generative Programming – Methods, Tools, and Applications. Addison-Wesley, Boston, MA, 2000.

[2] *Weiland, J.; Richter, E.:* Konfigurationsmanagement variantenreicher Simulink-Modelle. A.B. Cremers et.al. (Hrsg.): INFORMATIK 2005 – Informatik LIVE!. Band 2. Beiträge der 35. Jahrestagung der Gesellschaft für Informatik e.V. (GI). 19. – 22. Septe|mber 2005. Kölln Verlag, Bonn, 2005.

[3] *Klengel, K.; Weiland, J.:* Merkmalbasierte Konfiguration variantenreicher Simulink-Modelle. H. Dörr, T. Klein: Unterlagen zum Workshop „Modellbasierte Entwicklung von eingebetteten Fahrzeugfunktionen", Modellierung 2006. 22. – 24. März 2006. Innsbruck, Österreich, 2006.

[4] pure-systems GmbH, pure::variants. Eclipse Plugin User Guide, 2007.

Dipl.-Ing. Christian Dziobek

studierte Allgemeine Elektrotechnik an der RWTH Aachen. Seit 1998 arbeitet er bei der Daimler AG und beschäftigt sich mit der Einführung von Methoden und Tools zur modellbasierten Funktionsentwicklung im Bereich E/E der Pkw-Serienentwicklung

christian.dziobek@daimler.com

Dipl.-Inf. Wojciech Przystas

studierte Informatik und Computerlinguistik an der Universität Stuttgart. Seit dem Jahr 2007 ist er Doktorand bei der Daimler-AG-Forschung und beschäftigt sich dort schwerpunktmäßig mit der Variantenkonfiguration modellbasierter eingebetteter Software.

wojciech.przystas@daimler.com

Dipl.-Ing. Joachim Loew

studierte Luft- und Raumfahrttechnik an der Universität Stuttgart. Seit 1998 arbeitet er bei der Daimler AG. Zu seinem Aufgabengebiet gehört die modellbasierte Funktionsentwicklung für Steuergeräte.

joachim.c.loew@daimler.com

Dipl.-Inf. Jens Weiland

studierte Informatik an der Universität der Bundeswehr München. Seit dem Jahr 2000 leitet er Forschungsprojekte in der Daimler-AG-Forschung. Sein Themenschwerpunkt ist die Variantenkonfiguration modellbasierter eingebetteter Software.

jens.weiland@daimler.com