Standardisierte Blockbibliothek zum modellbasierten Steuergeräte-Softwareentwurf als Basis für den Modellaustausch zwischen Entwicklungstools

Achim Wohnhaas, debis
Rainer Moser, FKFS
Peter Brangs, BMW

1 Zusammenfassung

Die Zielsetzung im Projekt MSR-MEGMA ist die Erarbeitung eines Bibliothek-Standards zur modellbasierten Entwicklung von Steuergerätefunktionen als Basis für den grafischen Modellaustausch zwischen Entwicklungstools.

MSR (Manufacturer Supplier Relationship) ist ein firmenübergreifendes Komitee zur Unterstützung gemeinsamer Entwicklungen zwischen Automobilherstellern und Zulieferern von Elektroniksystemen.

In diesem Bericht werden zunächst die generellen Möglichkeiten des Modellaustauschs und der Modellkopplung zwischen Entwicklungstools vorgestellt. Des weiteren wird die im MSR-MEGMA für die Entwicklung von Steuergerätesoftware verfolgte Vorgehensweise mit einer einheitlichen, grafischen Blockbibliothek im Detail erläutert.

Außerdem werden technische Besonderheiten der betrachteten Entwicklungstools und deren Einflüsse auf die Konvertierbarkeit der Modelle aufgezeigt.

Der Fokus richtet sich dabei auf Funktionen und Strukturen im speziellen Umfeld der Funktionsentwicklung für Steuergeräte.

2 Ausgangssituation und Zielsetzung

Die Entwicklung elektronischer Funktionen in Kraftfahrzeugen nimmt einen zunehmend hohen Stellenwert in der Automobilentwicklung ein. Der Umfang, die Komplexität und der Vernetzungsgrad der Elektronik steigen sehr stark an. Zukünftige Fahrzeuge werden sich in steigendem Maße durch Softwarefunktionen von ihren Wettbewerbern abheben und herstellerspezifische Eigenschaften werden per Software ins Fahrzeug gebracht. Aus dieser Situation heraus wird der Einsatz rechnergestützter Methoden in der Entwicklung von Steuergerätesoftware unumgänglich. Eine Neuordnung der Zusammenarbeit zwischen Elektronik-Zulieferer und Hersteller ist erforderlich. Von den Entwicklungstools ist gefordert, dass sie diesen firmenübergreifend verteilten Entwicklungsprozeß sowohl von der technischen Seite der Schnittstellen als auch aus Sicht der Prozeßorganisation optimal unterstützen.

Im Arbeitskreis MSR-MEGMA soll der Einsatz der Entwicklungstools durch die Verfügbarkeit einer zugeschnittenen Standardbibliothek vorangebracht werden und

damit die Schnittstellenproblematik zwischen unterschiedlichen Entwicklungstools für die Entwicklung von Steuergerätesoftware gelöst werden.

3 Toolkopplung und Toolschnittstellen

Kopplungen und Schnittstellen zwischen Entwicklungstools sind eine große Forderung der Entwickler und eine ebenso große Herausforderung an die Toolhersteller. Die Notwendigkeit unterschiedliche Tools einzusetzen ergibt sich durch

- Unzulänglichkeiten der Tools den gesamten Entwicklungsprozeß durchgängig zu unterstützen,
- die notwendige Zusammenarbeit verschiedener Entwicklungsabteilungen mit ihren gewachsenen Strukturen (Tools, Know-How und Modelle),
- die gemeinsame Entwicklungsarbeit von Hersteller und Zulieferer mit unterschiedlichen Tools,
- den Wunsch für jede Entwicklungsaufgabe das geeignetste Tool einzusetzen.

Die Toolhersteller stehen diesen Wünschen und Forderungen wohl aufgeschlossen gegenüber. Eine perfekte und lückenlose Schnittstellenlösung kann es schon wegen der teils sehr unterschiedlichen Arbeitsweise und den spezifischen Features der Tools nicht geben, durch die sich die Tools im Wettbewerb behaupten müssen.

Es gibt jedoch einige funktionierende und einsatzgeprüfte Möglichkeiten für Toolkopplung und Modellaustausch, die im folgenden kurz dargestellt werden sollen. **Bild 1** stellt einige Möglichkeiten symbolisiert dar.

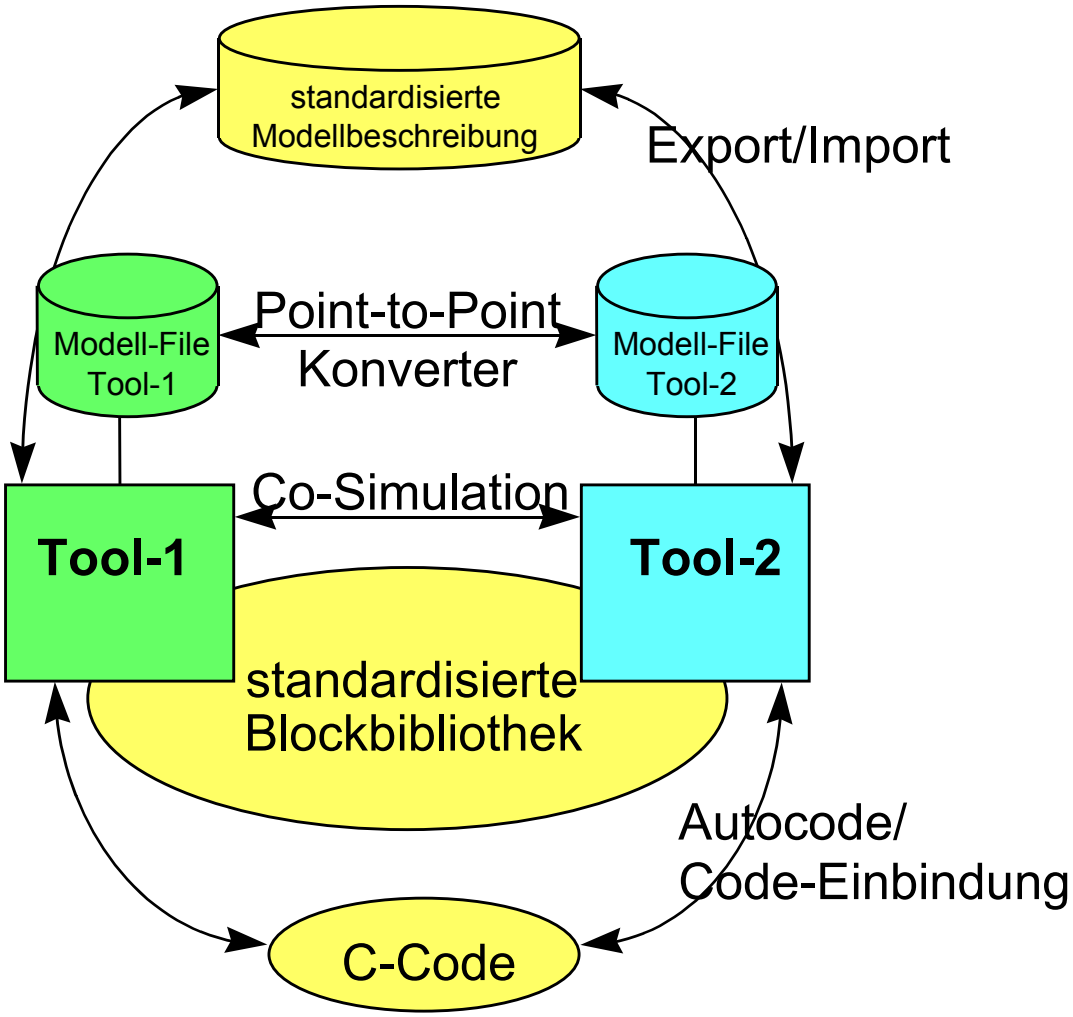


Bild 1: Modellaustausch und Schnittstellen in Entwicklungstools

3.1 Co-Simulation (Backplane)

Simulation eines über mehrere Plattformen (Tools) verteilten Modells. Der Datenaustausch erfolgt beispielsweise über Shared Memory oder über eine Netzwerkverbindung.

Es gibt sowohl von Drittanbietern speziell angefertigte als auch von den Toolherstellern in den Tools integrierte Lösungen.

3.2 Kopplung mit Remote Function Calls

Aufruf von Funktionen (z.B. numerischen Solvern) in einem anderen Tool. Diese Lösung ist dann erforderlich, wenn speziell programmierte oder per C-Code importierte Modellteile mit den im eingesetzten Tool verfügbaren Funktionen nicht oder nur unbefriedigend lösbar bzw. simulierbar sind.

3.3 Code-Generierung und Code-Einbindung

Die in Frage kommenden Entwicklungstools bieten automatisierte Codegenerierung und Möglichkeiten zur Einbindung von Code als Block im Modell. Es können damit Funktionen, die im Quell-Tool entwickelt wurden, im Ziel-Tool eingesetzt werden. Änderungen oder Weiterentwicklungen der eingebundenen Funktion erfordern den zeitaufwendigen Umweg über das Quell-Tool mit neuer Code-Generierung.

3.4 Point-to-Point Konverter

Konvertertools setzen auf den proprietären Modellbeschreibungssprachen der Tools auf und konvertieren eine Beschreibung in diejenige des Ziel-Tools. Für Tools deren Modellspeicherung in einem geschlossenen Format (binär oder in Datenbanken) erfolgt, kann dieser Konverter nur vom Toolhersteller selbst erstellt und verfügbar gemacht werden. Einige Konvertertools sind als „Unsupported Tools“ bereits verfügbar, decken jedoch nur einen eingeschränkten Funktionsumfang ab. Siehe auch Erfahrungsbericht in Kap. 7.

3.5 Standardisierte Beschreibungssprache oder Austauschformat

Die Tools unterstützen eine standardisierte Beschreibungssprache (z.B. VHDL-AMS) oder ein Austauschformat und bieten einen Import und Export in dieses Format, das die vollständige Beschreibung von Modell, Grafik und Netzliste enthält. Diese Lösung wurde bisher von den Toolherstellern nicht realisiert und ist nach Einschätzung des MEGMA-Arbeitskreises auch nicht in Aussicht.

3.6 Modellaustausch über standardisierte Blockbibliothek

Modellaustausch auf Basis einer standardisierten Entwicklungsbibliothek mit entsprechenden Konvertierungsmöglichkeiten, die in den Tools integriert sind. Die standardisierte Bibliothek sichert die Verfügbarkeit funktional und grafisch identischer Blöcke für einen speziellen Anwendungsbereich – hier modellbasierte Steuergeräte-

Softwareentwicklung – und ist damit einheitliche Arbeitsgrundlage in den eingesetzten Entwicklungstools. Diese Variante wird vom MSR-MEGMA verfolgt und ist im folgenden Abschnitt beschrieben.

Bei allen angesprochenen Möglichkeiten der Modellkonvertierung ergeben sich Probleme durch Funktionen und Strukturen die im Ziel-Tool nicht verfügbar sind. Diese müssen entweder von einem besonders intelligenten Konvertiertool oder Importfilter umgesetzt oder in manueller Nacharbeit behandelt werden.

4 Die standardisierte Blockbibliothek als Arbeitsbasis und Voraussetzung für den Modellaustausch

Das Ziel im MSR-MEGMA Projekt ist die Spezifikation einer in allen relevanten Entwicklungstools für modellbasierten Steuergeräte-Softwareentwurf implementierten, einheitlichen Blockbibliothek. Diese Blockbibliothek ist die sowohl funktional als auch grafisch einheitliche und notwendige Arbeitsbasis für den Steuergeräte-Softwareentwurf und damit nach Ansicht des Arbeitskreises MEGMA eine wesentliche Voraussetzung für einen funktionierenden Modellaustausch zwischen Entwicklungstools.

Die MSR-Bibliothek beschränkt sich auf physikalische Steuergerätefunktionen (vgl **Bild 2**). Implementierungsinformationen (Abbildung der physikalischen Daten auf Datentypen der Steuergeräte), sowie Modelle der Regelstrecke werden nicht betrachtet.

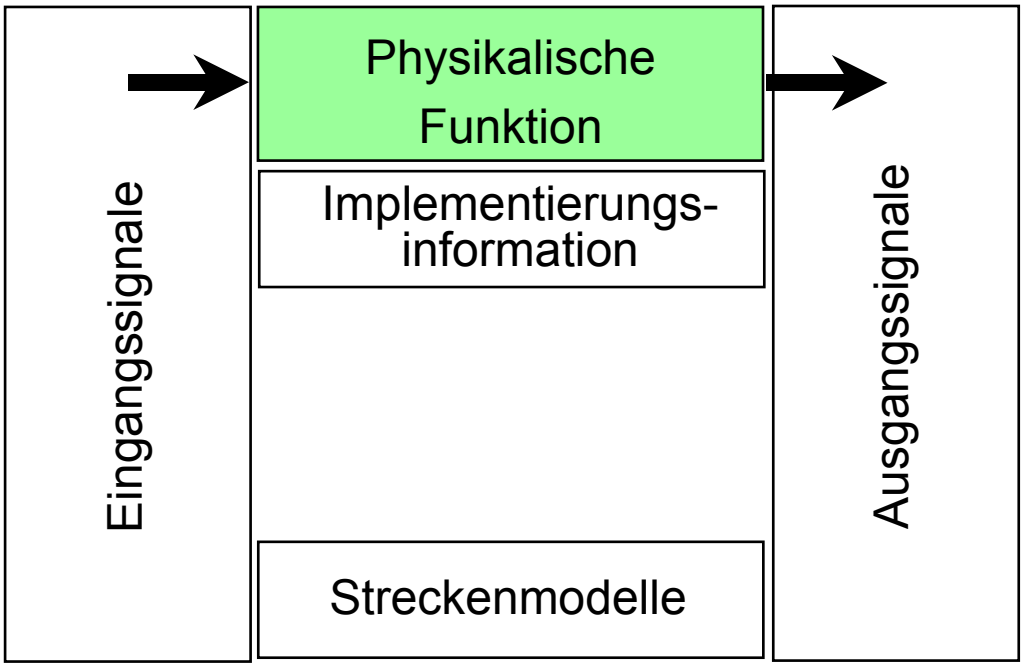


Bild 2: Standardbibliothek im MSR-MEGMA

Allen notwendigen und speziellen Anforderungen der Kfz-Steuergeräteentwicklung wird mit der MSR-Standardbibliothek Rechnung getragen, so dass damit alle betrachteten Entwicklungstools gleichermassen für den modellbasierten Entwurf von Steuergerätesoftware einsetzbar sind. Die folgenden weiterführenden Gründe sind für die Erstellung einer MSR-Blockbibliothek zu nennen:

- Die in der Standardbibliothek spezifizierten Blöcke orientieren sich an den Erfordernissen im Kfz-Elektronikumfeld und werden ohnehin von den Entwicklern benötigt, sofern sie nicht in den Tools direkt verfügbar sind. Die individuelle Erstellung der benötigten Blöcke durch den Entwickler wird damit vermieden.

- Alle Entwickler arbeiten mit denselben Basisfunktionen. Heterogenitäten innerhalb einer Firma oder sogar einer Entwicklungsmannschaft werden vermieden.

- Aufbau und Struktur der Software werden zwangsläufig ähnlich.

- Der Wiedererkennungswert der grafischen Modelle ist höher. Die zusätzliche Einführung von Modellierungs-/Entwurfsrichtlinien schafft eine weitergehende Einheitlichkeit und verbessert diesen Effekt.

- Eine manuelle Konvertierung von Modellen in andere Entwicklungstools ist durch die Verfügbarkeit funktional und grafisch identischer Blöcke durchführbar, ohne Experte im Quell- und Ziel-Tool zu sein.

Der Grundgedanke zur Modellkonvertierung im MSR-MEGMA ist, dass MSR-Bibliotheksblöcke mit verifiziert identischer Funktionalität und einheitlichen Schnittstellen in den Entwicklungstools einfach per Netzliste, Blockreferenzen und -position in ein anderes Tool übertragbar sind sofern keine inkompatiblen Kontrollstrukturen eingesetzt werden (siehe dazu nächster Abschnitt).

Der kritische Leser fragt sich, weshalb zur Modellkonvertierung nicht einfach eines der bereits verfügbaren Konvertierungstools eingesetzt und weiterentwickelt wird. Durch die eingeschlagene Vorgehensweise wird diese Lösung nicht ausgeschlossen – im Gegenteil, eine Adaption der bestehenden Point-to-Point Konvertier-Tools auf die MSR-Bibliothek ist wünschenswert. Hier liegt jedoch der Fokus auf der erforderlichen Weiterentwicklung hinsichtlich vollständiger Unterstützung der MSR-Bibliothek. Wesentlich für die Einsetzbarkeit einer Modellkonvertierung in der Kfz-Elektronikentwicklung ist eine exakte funktionale Übereinstimmung der konvertierten Modelle in jedem diskreten Rechenschritt. Dies kann nur durch den durchgängigen Einsatz der MSR-Standardbibliothek erreicht werden, wenngleich dadurch die Toolspezifischen Probleme ausgelöst durch spezielle Kontrollstrukturen und sonstige Besonderheiten und Features der Tools nicht gelöst werden.

Auf diese Problematik wird im folgenden Abschnitt eingegangen.

Die Standardbibliothek und Konzepte zum Modellaustausch werden von einem Expertenteam erarbeitet, bestehend aus Vertretern der Firmen Robert Bosch GmbH, BMW AG, Daimler-Chrysler, Hella KG und Siemens AG. Die zur Zeit betrachteten Entwicklungstools sind ASCET-SD, MATLAB/SIMULINK und MATRIX$_X$/SYSTEMBUILD.

5 Unterschiede und Besonderheiten der Tools

Die betrachteten Tools weisen einen hohen Überdeckungsgrad an ähnlichen Funktionen auf, der jedoch keine Aussage darüber zuläßt, wie groß die Unterschiede tatsächlich sind. Häufig sind die feinen Unterschiede auf den ersten Blick gar nicht sichtbar.

5.1 Modelleigenschaften

Die folgenden Modelleigenschaften sind bei der Entwicklung von SG-Software wesentlich:

- Es werden nur skalare Signale/Größen benötigt,
- SG-Software ist zeitdiskret,
- Hierarchien werden zur Strukturierung (grafisch und funktional) eingesetzt,
- Datentypisierung ist im späteren Steuergerätecode erforderlich,
- Teilmodelle mit mehreren Zeitrastern sind üblich,
- Steuergrößen und applizierbare Parameter als Signale im Modell
- Zustandsautomaten zur Beschreibung reaktiver Systemteile sind erforderlich,

Im MSR-MEGMA wurde hierzu folgendes festgelegt:

Datentypen werden zu einem späteren Zeitpunkt im Hinblick auf Verfügbarkeit in den Entwicklungstools gesondert betrachtet.

Zustandsautomaten sind in den Entwicklungstools sehr unterschiedlich gelöst und befinden sich noch stark in der Weiterentwicklung. Durch den Einsatz der Zustandsgrafen ändert sich die Modellstruktur. Es setzt sich unter anderem der Einsatz der Zustandsgrafen als reines Control Chart im modellbasierten Softwareentwurf durch.

Eine Behandlung der Zustandsgrafen ist deshalb für einen zukünftigen Modellaustausch erforderlich. Es ist die Einschätzung des Arbeitskreises, dass in den nächsten Jahren die grundlegenden Features von Zustandsgrafen in allen Entwicklungstools in kompatibler Form verfügbar sein werden.

5.2 Hierarchiebildung und Strukturierung

Im folgenden sind die verschiedenen Hierarchieformen der Tools und deren Eigenschaften aufgelistet:

- SYSTEMBUILD:
  - Hierarchielement *SuperBlock*: Zentrales Hierarchieelement in SYSTEMBUILD, das je nach Typ unterschiedliche Zeitattribute besitzt. Ein *SuperBlock* ist oberstes Element eines Modells.

Typ	Zeitattribute
Continuous	Zeitkontinuierliches Raster, vererbt Zeitraster an Subblöcke (Basisblöcke, andere *SuperBlocks*), instantiierbar
Discrete	Zeitdiskretes Raster, vererbt Zeitraster
Procedure	Erbt Zeitraster von *SuperBlock* bzw. *Condition Block*, erzeugt Funktion, referenzierbar (function-call)
Trigger	Externe Rastervorgabe (flankengesteuert)
Enable	Eigenes Zeitraster, Externe Aktivierung des *SuperBlocks* (wertgesteuert)

  - Hierarchieelement *Condition block*: Hierarchieelement für *Procedure-Superblocks*, die über Bedingungen angestoßen werden. Erbt das Zeitraster.

- ASCET-SD:
  - *Hierarchy*: Grafische Zusammenfassung von Blöcken ohne zeitliche oder funktionale Eigenschaften.
  - *Module*: Einfach instantiierbar mit verschiedenen diskreten Zeitrastern.
  - *Class*: Mehrfach instantiierbar mit verschiedenen diskreten Zeitrastern.

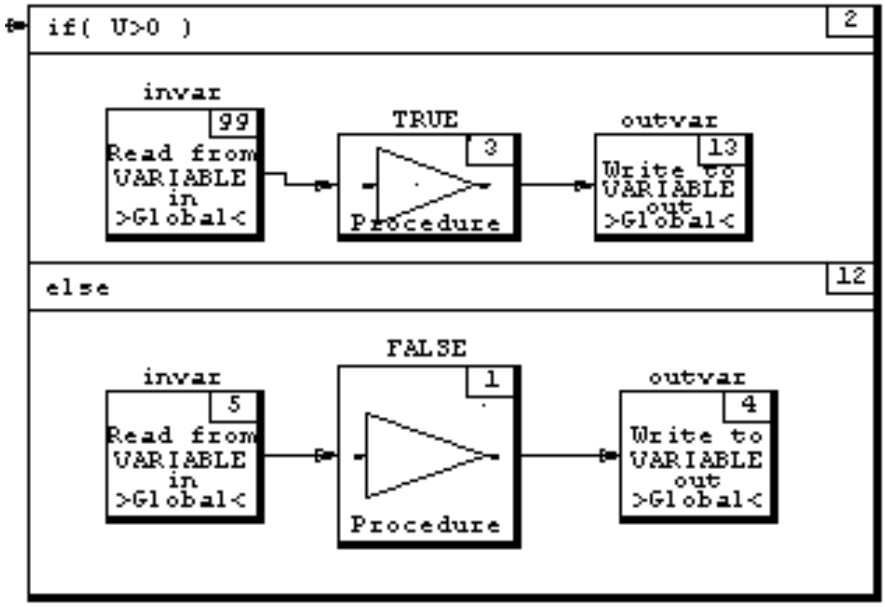
- SIMULINK:
  - *Subsystem*: Grafische Zusammenfassung von Basisblöcken oder anderen *Subsystems* (ohne zeitliche oder funktionale Eigenschaften).

5.3 Kontrollstrukturen

Steuergerätesoftware läßt sich häufig mit einfachen Kontrollflußelementen wie Schalter (Switch) beschreiben. Komplexe Kontrollstrukturen wie sie in den Tools verfügbar sind, ermöglichen gegebenenfalls eine elegante oder effiziente Beschreibung, tragen aber nicht zur einfachen Austauschbarkeit und Transparenz der Modelle bei. Dabei handelt es sich meist um softwarenahe Konstrukte wie IF-THEN-ELSE oder SWITCH-CASE. Es sind jedoch gerade die Kontrollstrukturen, hinter denen sich eine Tool-Philosophie verbirgt über die sich ein Tool vom Wettbewerber abhebt. Der Entwickler, der sich ausschließlich in einem Tool bewegt, schätzt diese Vorteile und nutzt bzw. benötigt sie sogar zwingend, um bestimmte Funktionalität zu erreichen. Verbietet man den Einsatz dieser toolspezifischen Möglichkeiten wird es Akzeptanzprobleme bei den Entwicklern geben. Dieser Zielkonflikt ist durch die Einführung von Modellierungsrichtlinien mit Überzeugungsarbeit zu überwinden, da eine vollständig automatisierte Austauschbarkeit ohne gewisse Einschränkungen aus heutiger Sicht nicht möglich scheint.

Ein einfaches Beispiel zeigt **Bild 3**, in dem sich die Unterschiede in der Grafik zwischen SYSTEMBUILD und ASCET-SD offenbaren.

MATRIX$_X$ / SYSTEMBUILD



ASCET-SD

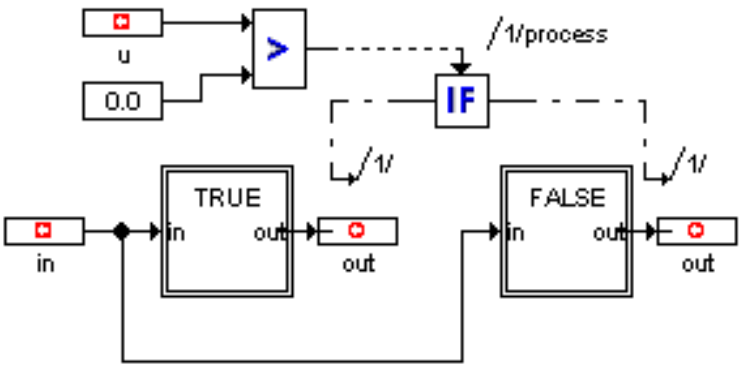


Bild 3: Grafische Unterschiede bei Kontrollstrukturen zwischen den verschiedenen Tools (If-Then-Else-Block).

5.4 Varianten und konfigurierbare Blöcke

Es ist Tool-Philosophie ob überwiegend mit statischen oder konfigurierbaren Blöcken gearbeitet wird. Existieren im einen Tool mehrere Varianten einer Funktion als statische Blöcke, so exisitiert im anderen Tool ein generischer Block der für jede Variante entsprechend konfigurierbar ist.

5.5 Signalfluß

Wesentlich für die Transparenz der Modelle ist die durchgängige Sichtbarkeit und Verfolgbarkeit der Signalflüsse im Modell. Insbesondere müssen in der SG-Software alle applizierbaren Parameter im Signalfluß und damit an der Schnittstelle zu Basisblöcken auftauchen. In SIMULINK beispielsweise arbeiten viele Blöcke mit internen Parametern, die in der Steuergeräte-Software als applizierbare Parameter benötigt werden

Diese Blöcke sollen durch entsprechend erweiterte Blöcke aus der MSR-Standardbibliothek ersetzt werden.

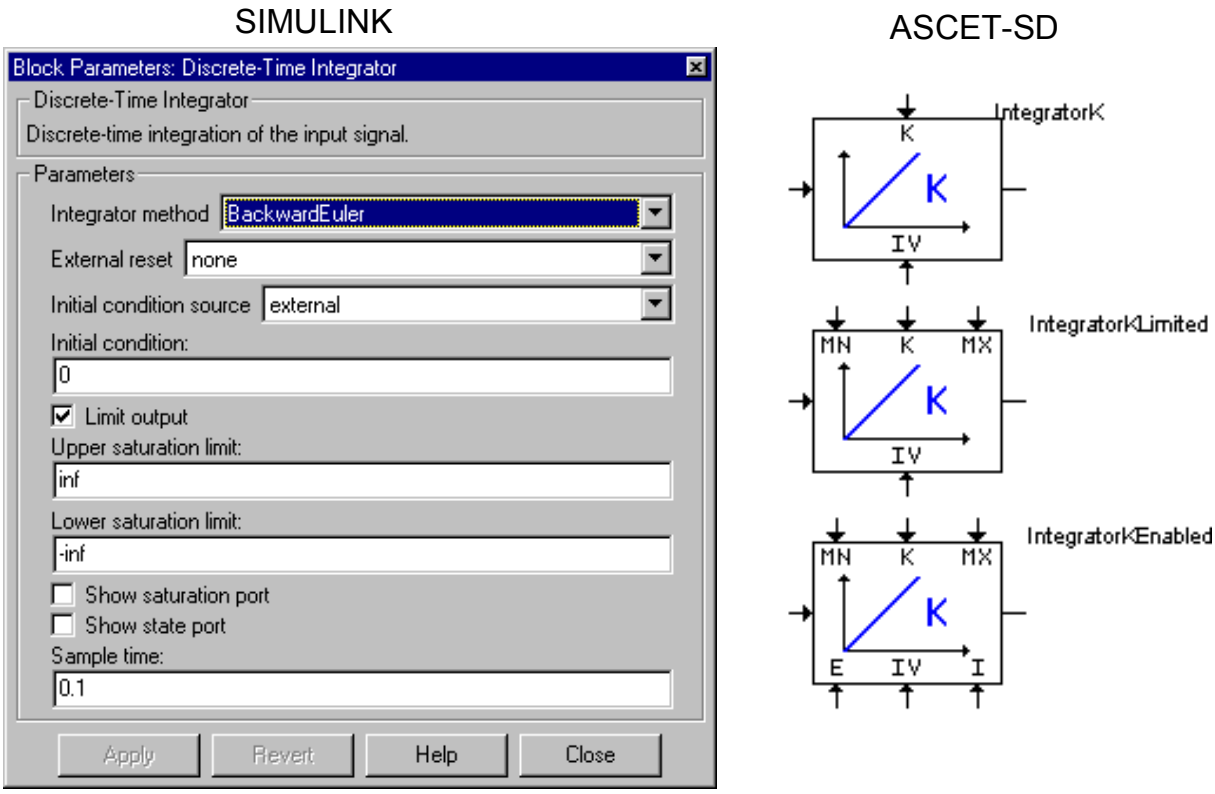


Bild 4: Abbildung der Varianten verschiedener Blöcke durch interne Parameter bzw. externe Signaleingänge (Simulink-Interface und Varianten in ASCET-SD).

5.6 Timing und Berechnungsreihenfolge

Im Timing zeigt sich die unterschiedliche Philosophie und die Flexibilität der Tools sehr deutlich. In SYSTEMBUILD wird das Zeitverhalten am SuperBlock festgemacht. In SIMULINK sind interne Parameter der Blockelemente für das Timing konfigurierbar und in ASCET-SD hat der Entwickler alle Freiheiten. Letzteres gilt insbesondere auch für die Festlegung der Berechnungsreihenfolge, die automatisiert datengetrieben oder manuell beliebig veränderbar sein kann. ASCET-SD arbeitet dabei von den Ausgängen zu den Eingängen rückwärts, d.h. ein Ausgang fordert die Berechnung aller Blöcke, die seinen Eingang beeinflussen an. Im Gegensatz dazu arbeitet SIMULINK datengetrieben (vom Eingangssignal zum Ausgangssignal) und bietet dem Anwender keine Eingriffsmöglichkeit in die Berechnungsreihenfolge. In SYSTEMBUILD steht dafür beispielsweise ein SequencerBar zur Verfügung, der eine grafisch sichtbare Trennung der einzelnen Berechnungsblöcke darstellt.

Im Detail ergeben sich Timing-Unterschiede durch unterschiedliche werkzeuginterne Berechnungsreihenfolgen (vgl. **Bild 5**). Zum Beispiel werden in SIMULINK und SYSTEMBUILD zuerst die Ausgänge aktualisiert, bevor die internen Zustände neu berechnet werden. Dies kann zu unterschiedlichen Simulationsergebnissen führen, da in ASCET-SD diese Berechnungsreihenfolge vom Anwender angepasst werden kann. Dies erschwert einen automatisierten Austausch der verschiedenen Modelle.

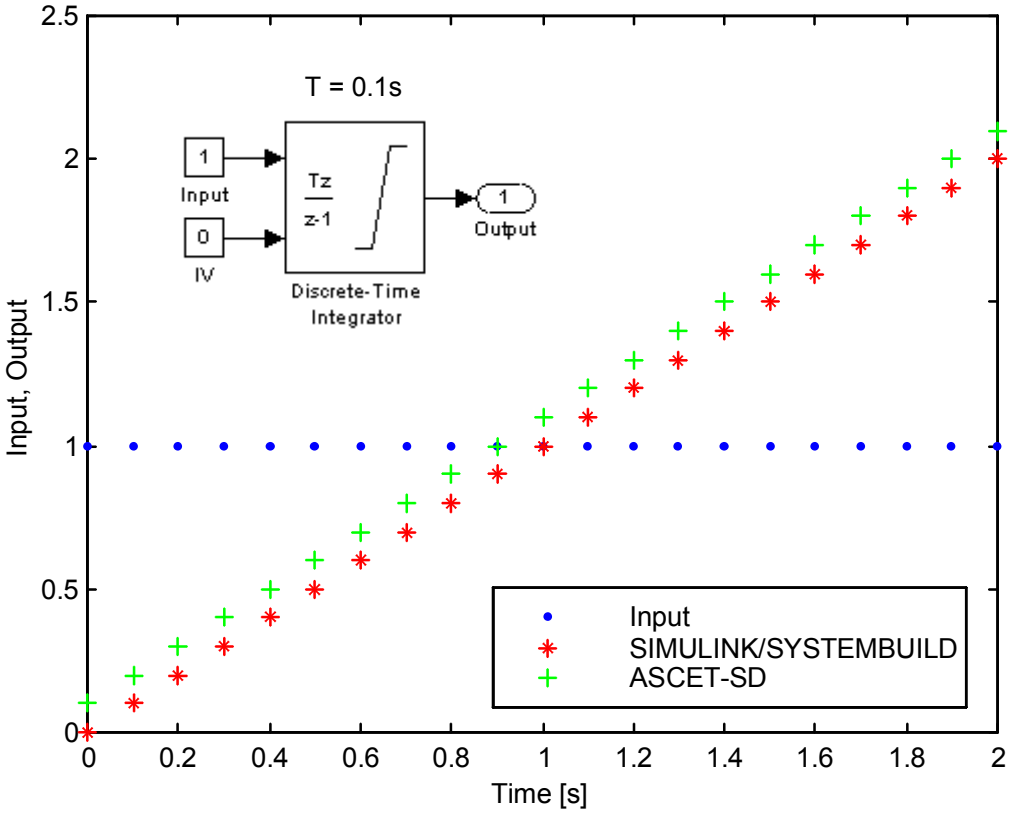


Bild 5: Timing-Unterschied beim Integrator aus den Standardbibliotheken der verschiedenen Werkzeuge.

5.7 Parameter

Aus Sicht der Steuergeräteentwicklung werden durch applizierbare Parameter unterschiedliche Steuergerätevarianten realisiert. Im Hinblick auf den Modellaustausch müssen diese Parameter als Teile des Modells übertragen werden. Hierbei sind Standardschnittstellen (z.B. SGML basiert) zwischen den Tools gefragt.

Funktionale Unterschiede bestehen hauptsächlich in der Art der Berechnung der Ausgänge (lineare Interpolation bzw. Tabellensuche) und dem Verhalten außerhalb des definierten Eingangswertebereichs (lineare Extrapolation, Begrenzung) von Kennlinien und Kennfeldern.

6 Konzepte und Überlegungen des MSR-MEGMA zum Modellaustausch

6.1 Anforderungen

Die Anforderungen an einen grafischen Modellaustausch wurden wie folgt festgelegt:

- Hin- und Rückkonvertierung ohne funktionale Verluste
  (Quell-Tool -> Ziel-Tool -> Quell-Tool)
- Unidirektionale Konvertierung ohne grafische Verluste bei Basisblöcken
  (Menge der unterstützten Basisblöcke wird von MSR-MEGMA festgelegt)
  Hohe Wiedererkennbarkeit der Funktion ist gewünscht.
- Einschränkungen unterstützter tooleigener Kontrollstrukturen und Einführung von Modellierungsrichtlinien die von der Konvertierung unterstützt werden.
  Es werden keine MSR-eigenen Kontrollelemente spezifiziert

6.2 Vorgehensweise

Der automatisierte Modellaustausch stützt sich aus der oben genannten Motivation heraus auf eine MSR-Blockbibliothek, die alle benötigten Elemente (Basisblöcke) zur modellbasierten Beschreibung von Steuergerätefunktionen enthält. Die prinzipielle Funktionsweise ist wie folgt beschrieben:

- Eine Netzliste beschreibt die Lage und Verbindung der Elemente.
- Eine Abbildungsvorschrift bildet ein Element im Quell-Tool auf ein funktional identisches Element im Ziel-Tool ab.
- Unbekannte Blöcke werden als Dummy mit entsprechenden Schnittstellen konvertiert.
- Unterschiedliche Varianten der Elemente werden über ein Varianten-Mapping behandelt.

- Es gibt eine Vorschrift zur Nachbildung des Zeitverhaltens.
- Ein Verfahren deckt die Behandlung hierarchischer Strukturen ab.
- Kontrollstrukturen werden nach folgenden Möglichkeiten behandelt:
  - Kontrollfluss wird nur mit Schaltern erstellt (Switch, Mux).
  - Einfache Kontrollstrukturen sind zugelassen und Konvertierbar
  - Komplexe Kontrollstrukturen werden nach strengen MSR-Modellierungsrichtlinien eingesetzt und vom Konvertiertool umgesetzt.
  - Zustandsgraphen werden in allen Tools einheitlich eingesetzt.

Technisch gesehen lässt sich der Modellaustausch auf Basis der MSR-Blockbibliothek auf unterschiedliche Weise realisieren.

1. Es besteht die Möglichkeit ein auf die MSR-Automotive Anwendungen zugeschnittenes **Austauschformat** zu schaffen, das alle Informationen die zum Modellaustausch erforderlich sind enthält. Dieses Austauschformat wird vom MSR-MEGMA entwickelt und von den Toolherstellern unterstützt. Es besteht aus der hierarchisch strukturierten Beschreibung der Modelle und der in den Modellierungsrichtlinien zugelassenen Kontrollstrukturen. Der Arbeitskreis MEGMA hat exemplarisch ein solches Austauschformat auf Basis einer SGML-Document-Type-Definition erstellt.

2. Eine zweite Möglichkeit besteht darin, die von den Toolherstellern teilweise verfügbaren **Point-to-Point** Konverter so weiterzuentwickeln, dass sie die MSR-Bibliothek und die in Modellierungsrichtlinien angegebenen Kontrollstrukturen vollständig abdecken.

3. Als dritte Möglichkeit könnte vom MSR in Eigenleistung ein Konvertiertool nach dem Vorbild der verfügbaren Point-to-Point Konverter erstellt werden, das intern mit dem o.g. Austauschformat arbeitet und die proprietären Formate der betrachteten Toolhersteller unterstützt.

6.3 MSR-Blockbibliothek

Eine Bibliothek der MSR Automotive-Blöcke wurde vom MSR-MEGMA Expertenteam spezifiziert. Ziel ist die Abbildung eines Elements im Quell-Tool auf ein funktional identisches Element im Ziel-Tool.

Der Entwickler hat damit zukünftig in den betrachteten Tools eine MSR- Standardbibliothek zur Verfügung, mit der er seine Steuergerätefunktionen aufbauen kann. Ebenso kann er mit den standardmäßig im Tool verfügbaren Elementen arbeiten, sofern diese in der vom MSR-MEGMA festgelegten Menge für einen Modellaustausch zugelassen sind.

Der Arbeitskreis MSR-MEGMA empfiehlt die Herausgabe von Modellierungsrichtlinien, die der Entwickler beachten muss, um sowohl den Modellaustausch zu sichern als auch den Wiedererkennungseffekt zu steigern.

6.4 Variantenmapping

Ein Varianten-Mapping verringert die Anzahl der MSR-Blöcke und ermöglicht die Nutzung der tooleigenen Elemente (vgl. **Bild 6**).

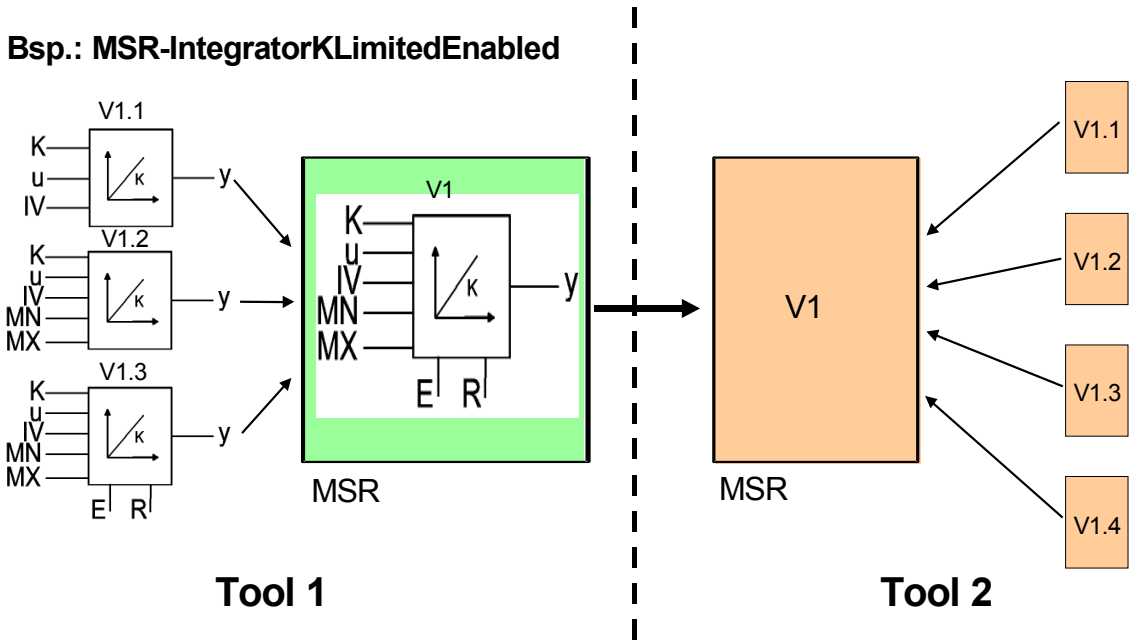


Bild 6: Variantenmapping beim Modellaustausch, am Beispiel von 3 Integratorvarianten.

6.5 Zeitverhalten und Hierarchien

Die oben erläuterten Unterschiede der Tools im Zeitverhalten sind eine Herausforderung an einen automatisierten Modellaustausch bei identischer Funktionalität.

Daneben verlangt die unterschiedliche Behandlung der Zeitraster (Abtastraster, sample time) in den einzelnen Tools auch teilweise eine andere Strukturierung des Modells (z.B. SYSTEMBUILD):

- In SYSTEMBUILD erfolgt eine Ordnung nach Zeitrastern (Superblock). Funktionen mit unterschiedlichen Zeitrastern liegen daher in getrennten Superblöcken. Externe Trigger sind ebenfalls möglich.

- In ASCET-SD sind Zeitraster beliebig an Ausgänge und speichernde Elemente gebunden. Rückwärtiges Verfolgen der Struktur bis zur nächsten (davor liegenden) Speicherzelle ergibt ein Raster und wird beispielsweise in SYSTEMBUILD in einen Superblock gewandelt.
- Zeitraster in SIMULINK sind an Eingangssignale und speichernde Elemente gebunden oder durch externe Trigger vorgegeben.

Die entsprechenden hierarchischen Strukturen der Tools werden aufeinander abgebildet. So ergeben sich aus *SuperBlocks* in SYSTEMBUILD *Subsystems* in SIMULINK und *grafische Hierarchien* in ASCET-SD. Die Verwendung von Klassen wurde in ASCET-SD ausgeschlossen.

6.6 Konvertierungsbeispiel Leerlaufregler (ohne Kontrollstrukturen)

Die Konvertierungsregeln für Modelle ohne Kontrollstrukturen wurden manuell an einem Beispiel nachvollzogen. Das Ergebnis entsprach den Erwartungen. Es konnte in allen Werkzeugen funktionale Gleichheit erzielt werden. Die grafische Präsentation war in den Tools teilweise sehr ähnlich und entsprach dem erwünschten Wiedererkennungswert (**Bild 7 und 8**).

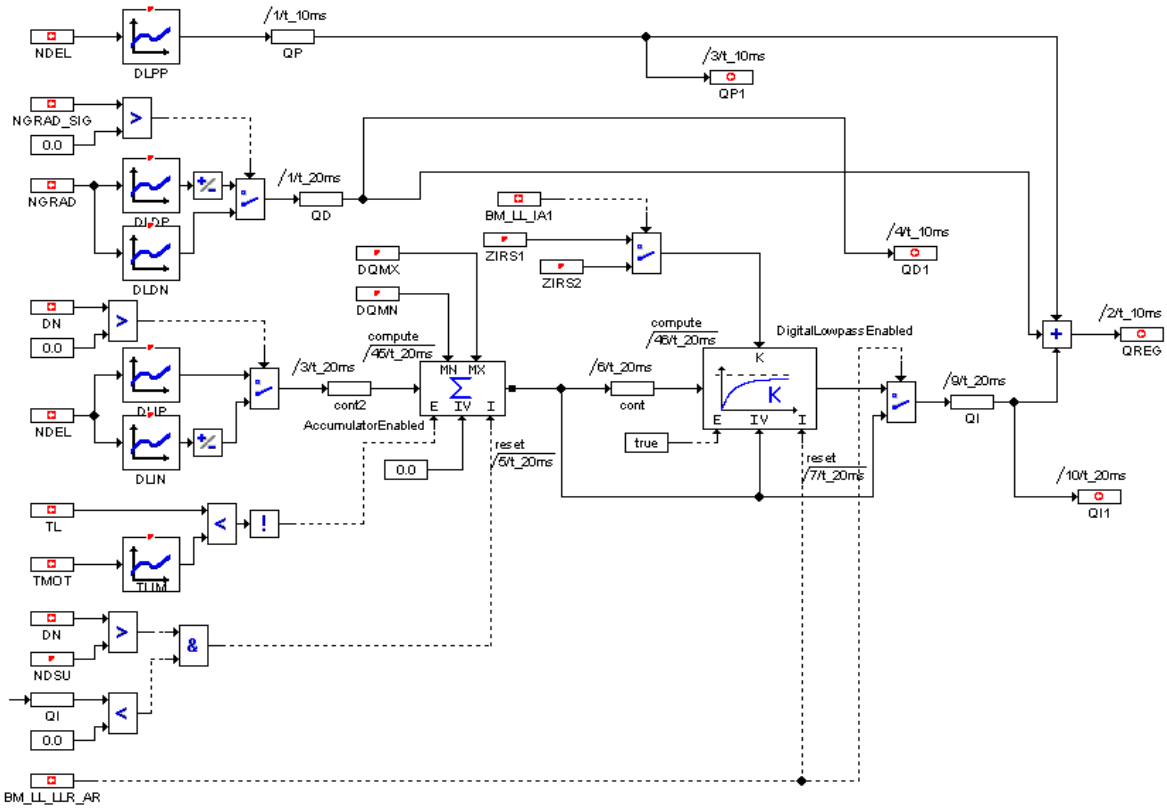


Bild 7: Leerlaufregler in ASCET-SD.

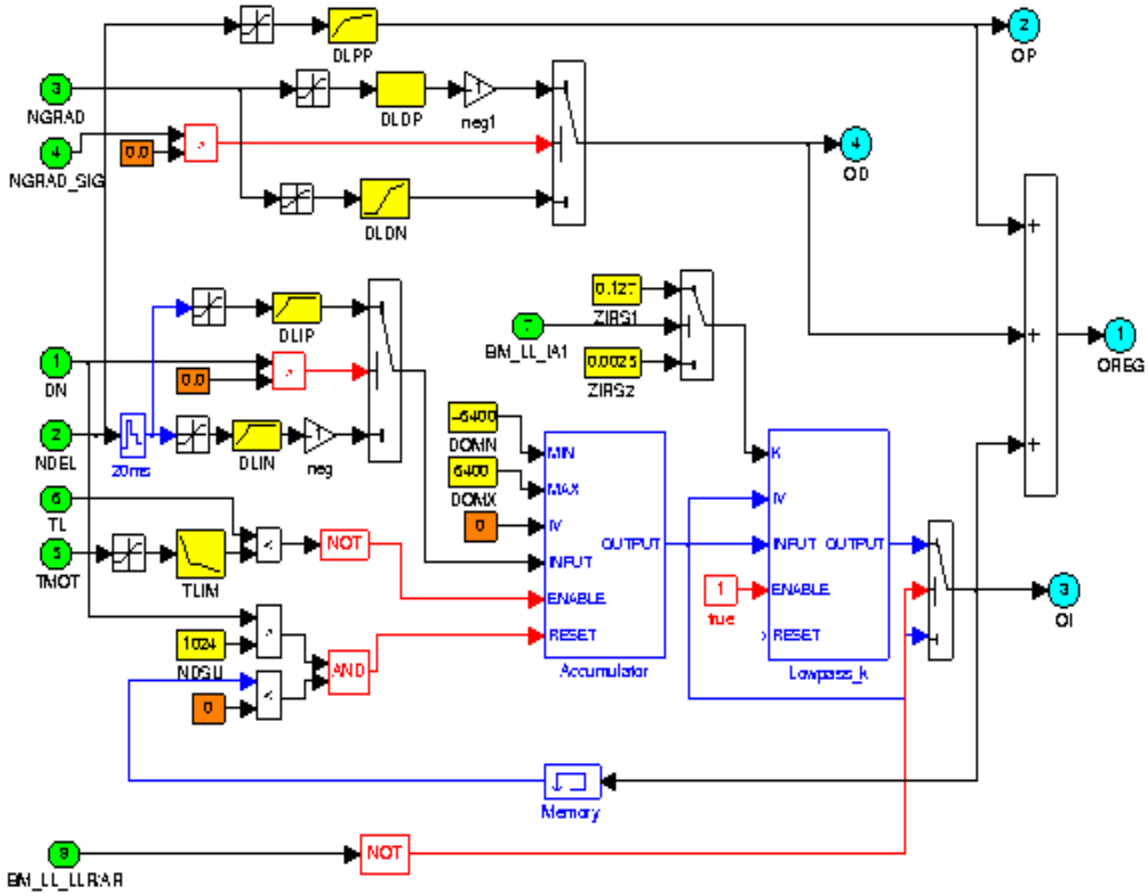


Bild 8: Leerlaufreglerbeispiel von ASCET-SD nach SIMULINK konvertiert.

7 Erfahrungsbericht Point-to-Point Konverter

Wie in Kap. 3 bereits erwähnt ist, sind einige Point-to-Point Konvertierungswerkzeuge zur unidirektionalen Übersetzung der Modelle zwischen zwei Werkzeugen verfügbar. Konkret sind dies zum aktuellen Zeitpunkt die folgenden Lösungen:

7.1 Matlab/Simulink nach MatrixX/Systembuild

Die Konvertierung SL2SB von Matlab/Simulink nach MatrixX/Systembuild wird von der Firma Windriver/ISI mitgeliefert. Die Funktion des Werkzeugs konnte in der verfügbaren ersten Version nur für das vorgegebene Beispiel nachgewiesen werden, so dass eine echte Bewertung nicht stattfinden konnte.

7.2 MatrixX/Systembuild nach Matlab/Simulink

Die Konvertierung von MatrixX/Systembuild nach Matlab/Simulink wird von Mathworks als Toolbox auf der MATLAB-CD mitgeliefert. Die Toolbox heißt SB2SL und ist in MATLAB R11.1 in der Version 2.0 verfügbar. Sie arbeitet mit MATLAB 5.3 und SIMULINK 3.0 und Systembuild-Dateien (ASCII-Format) der Versionen 5.0 und 6.0. Die aktuelle Systembuild-Version 6.1 wird noch nicht unterstützt.

Arbeitsweise von SB2SL:

SB2SL führt eine blockweise Zuordnung durch. Einfach ist es für Systembuild-Blöcke, für die ein äquivalenter Block in Simulink existiert. Für die anderen Blöcke in Systembuild exisiert eine spezielle SB2SL-Library in Simulink, die mit maskierten Subsystemen die Systembuild-Blöcke nachbildet.

Funktionsumfang von SB2SL 2.0:

- BlockScripts werden als C-Code in S-Functions übersetzt
- Vektorsignale in MatrixX werden in skalare Signale in Simulink aufgelöst.
- die graphische Darstellung (Layout) funktioniert mit Einschränkungen
- Konvertierung von kontinuierlichen und zeitdiskreten Blöcken

Einschränkungen von SB2SL 2.0:

- Datentypen werden nicht unterstützt
- Einschränkungen bei Read-to-/Write-from-Variable
- Unterschiedliches Zeitverhalten von "Triggered Subsystems"
  Systembuild liefert den aktualisierten Output im Gegensatz zu Simulink erst im darauffolgenden Integrationsschritt.
- Einschränkungen bei triggered Superblocks

Nicht konvertiert werden

- Zustandsgraphen
- MathScript Blöcke und UserCode Blöcke
- Animationsblöcke
- Condition-Blöcke

Erfahrung/Einschätzung:

Im grossen und ganzen funktioniert der Import in Simulink für Standardblöcke brauchbar. Als Preprozessor für eine einmalige Konvertierung mit anschließender händischer Nachbereitung ist SB2SL geeignet. Entscheidende Einschränkung ist die fehlende Konvertierung von Condition-Blöcken, sowie die Auflösung von Vektorsignalen auf skalare Einzelsignale an den Blöcken.

7.3 Matlab/Simulink nach Ascet-SD

Der Konverter wird als Bestandteil des sog. Matlab Integration Package (MIP) ausgeliefert.

Der Modell-Import in Ascet-SD erfolgt derzeit auf Basis einer Matlab/Simulink Blockbibliothek (in Ascet-SD), die eine eins-zu-eins Abbildung der Simulink Blöcke in Ascet SD ermöglicht. Eine Abbildung auf Standard Ascet-SD Blöcke ist nicht möglich.

Einschränkungen der Konvertierung

- es werden derzeit nur zeitdiskrete Blöcke übertragen
- es werden nur skalare Signale (keine Vektoren!) übertragen
- nicht erkannte Blöcke werden durch Dummy-Blöcke dargestellt
- die graphische Darstellung funktioniert mit Einschränkungen (die Blöcke überlappen sich)
- die Modellgröße, bzw die Anzahl der Signale ist vermutlich begrenzt, da bei größeren Modellen der Import versagt

Nicht konvertiert werden

- S-Funktionen,
- Zustandautomaten,
- und triggered/Enabled Blöcke.

Erfahrung/Einschätzung:

Im grossen und ganzen funktioniert der Import von Matlab/Simulink Modellen in Ascet-SD für Basisblöcke und kleinere Beispiele. Für grössere Modelle ist er derzeit nicht geeignet. Entscheidende Einschränkung ist die fehlende Konvertierung von S-Funktionen, die generelle Abbildung auf spezielle, Nicht-Standardblöcke in Ascet-SD, sowie die fehlende Behandlung von Vektorsignalen und die Begrenzung der Modellgrösse.

8 Untersuchung von Kontrollstrukturen

Wesentliches Kontrollflusselement für die modellbasierte Entwicklung von Steuergerätefunktionen ist die IF-THEN-ELSE Bedingung. Die Umsetzung der Kontrollflussstrukturen in den Werkzeugen kann durch unterschiedliche Modellierungen erfolgen, z.B. if-then-else kann in Simulink als Enabled-Subsystem oder als Stateflow-Konstrukt ohne Zustände (Entscheidungsbaum) abgebildet werden. In der Praxis hat sich die Darstellung durch Stateflow durchgesetzt, da der Anwender hierbei sehr übersichtlich, strukturierte Verzweigungen darstellen kann.

Zunächst muß die Frage beantwortet werden, ob eine Konvertierung überhaupt technisch möglich ist und wenn ja welche Art von Konvertierung, d.h. mit welcher Genauigkeit bzgl. grafischer und funktionaler Information. Um eine funktionale Konvertierbarkeit nachzuweisen, werden im folgenden Templates von einfachen If-Then-Blöcken für die einzelnen Werkzeuge erstellt.

Ziel des MEGMA-Arbeitskreises ist die blockorientierte Darstellung der Kontrollflussstrukturen. Um eine blockorientierte Darstellung zu erhalten, die in allen Werkzeugen den größtmöglichen Wiedererkennungseffekt besitzt, wurde das if-then-Konstrukt in drei Teile aufgelöst:

1. Condition: ein beliebig komplexer Ausdruck bzw. Hierarchie, die als Ausgang eine logische Variable besitzt.

2. If-Block: eigentlicher Kern des Blocks, der den Eingang auswertet und bei wahrem Inhalt ein Ereignis bzw. einen Funktionsaufruf generiert.

3. Action: beliebig komplexe Ausdrücke in einer Hierarchie, die als ein Funktionsaufruf ausgewertet werden. Dieser Block besitzt in jedem Fall einen Aufrufeingang (Ereignis, Funktionsaufruf) und weitere beliebige Signalein- und -ausgänge.

Ein MEGMA-If-Block-Template hat damit folgende Gestalt:

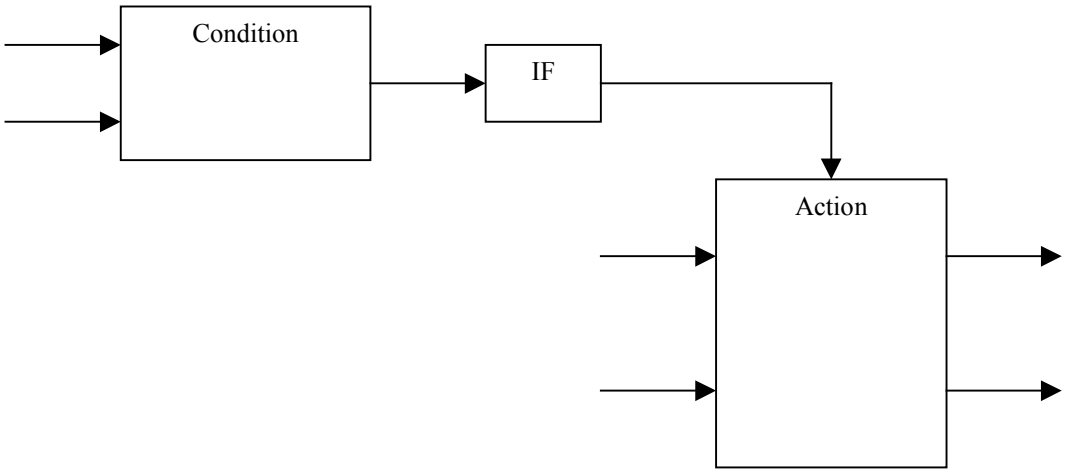


Bild 9: Darstellung des MEGMA-If-Block-Templates.

Der if-Block selbst besteht ausschließlich aus einem Eingang vom Typ logical (boolean) und einem Ausgangssignal, das eine Art Ereignis bzw. Aktivierungssignal darstellt. Die beiden weiteren Blöcke sind bereits Bestandteil der Bibliothek.

Im folgenden wird auf Basis des MEGMA-If-Blocks versucht eine äquivalente Darstellung in den verschiedenen Werkzeugen zu erzielen.

8.1 If-Block in ASCET

Die vorgegebene Dreiteilung läßt sich auch in ASCET umsetzen und hat dann folgende Gestalt.

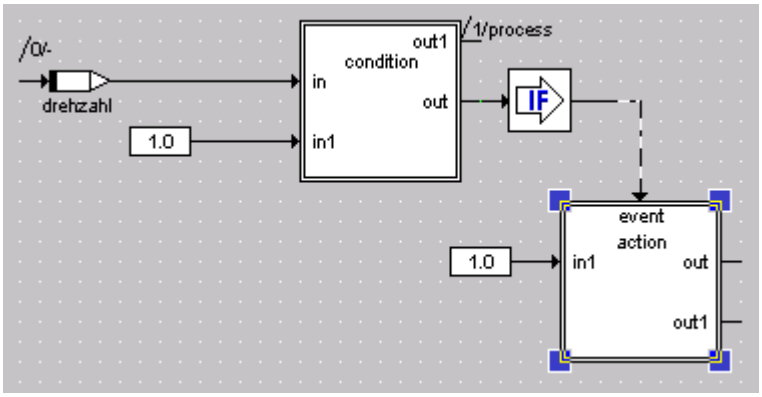


Bild 10: Abbildung des if-then-Blocks in ASCET.

Bei der Speicherung eines if-Blocks in ASCET müssen zur Weiterverarbeitung hinsichtlich einer Konvertierung alle vom Block action aktivierten Elemente gesucht und in einer Action-Hierarchie abgespeichert werden.

8.2 If-Block in SIMULINK

In Simulink läßt sich ebenfalls eine äquivalente Darstellung finden.

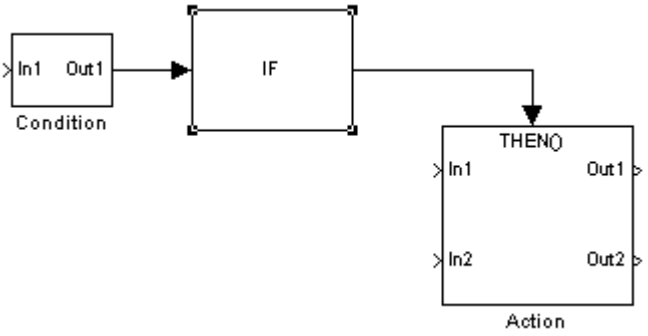


Bild 11: Abbildung des if-then-Blocks in Simulink.

Der If-Block in Bild 11 kann beliebig dargestellt werden (Stateflow oder Enable).

8.3 If-Block in MatrixX

In MatrixX/Systembuild läßt sich die Dreiteilung nicht erzielen, da hier für Kontrollfluss Container angeboten werden (s. Bild 12).

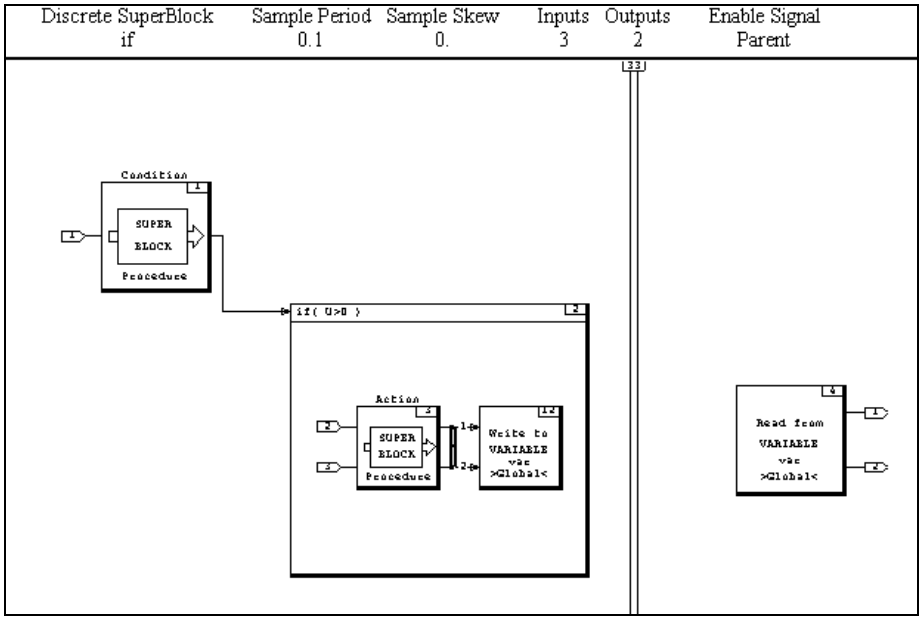


Bild 12: Abbildung des If-Blocks in Systembuild.

8.4 Zusammenfassung

Prinzipiell ist eine funktionale Konvertierung für Modelle möglich. Funktionale Inhalte sind: Rechenraster, Rechenreihenfolge, Kontrollflüsse, Basisblöcke. Eine detaillierte Betrachtung dieser Aspekte ist für einen praktikablen Austausch unumgänglich!

- Die grafisch identische Darstellung o.g. Inhalte in allen Tools ist nicht machbar
- Problematisch ist, daß importierte Modelle nicht die tooltypische Semantik anwenden.

Dies wird an folgendem Beispiel näher erläutert:

Ein Anwender modelliert in Simulink beliebige Kontrollflüsse mittels Stateflow ohne Zustände (reiner Kontrollfluß). Das Modell besteht aus beliebig vielen Entscheidungspfaden, in denen jeweils maximal zwei Funktionsaufrufe generiert werden können. Eine Konvertierung dieses Blocks wird im folgenden untersucht. In MatrixX kann zukünftig mittels BetterState eine äquivalente Darstellung des Blocks erfolgen. Bei ASCET wird es zumindest in naher Zukunft (incl. Version 4.0) keine Unterstützung dieser Entscheidungsbäume geben, so daß eine Auflösung der Pfade in einzelne ASCET-Blöcke erfolgen muß (pro Pfad ein if-Block bzw. If.then-else-elseif-Block). Dies ist unter der Voraussetzung möglich, daß eine Konvertierung der Matlab/Simulink-Sprachkonstrukte in den Aktionen und Bedingungen der Pfade nach allen Werkzeugen möglich ist.

Eine mögliche Lösung wäre die Restriktion des Simulink-Anwenders auf vordefinierte Teil-Blöcke (Basis-Charts) und daraus zusammengesetzte Stateflow-Charts was jedoch nach Ansicht des Arbeitskreises MEGMA auf Akzeptanzprobleme beim Anwender stossen dürfte.

Faktisch ist ein Stateflow-Block dann für ASCET ein generischer Block. Bei einer Rückkonvertierung nach Simulink ergibt sich weiterhin das Problem der Zusammenfassung der einzelnen möglicherweise abgeänderten ASCET-Kontrollflußelemente. Aus heutiger Sicht wird bezweifelt, daß ein in ASCET-Einzelteile aufgelöstes und dort weiterentwikkeltes „Stateflow-Chart“ wieder automatisch nach Simulink in ein Stateflow-Chart rückkonvertiert werden kann.

9 Ausblick

Der Arbeitskreis MSR-MEGMA wird die Implementierung der MSR-Bibliothek durch die Toolhersteller begleiten. Vorrangiges Ziel ist die schnelle Verfügbarkeit der MSR-Bibliothek in allen betrachteten Entwicklungstools. Die Einführung der MSR-Bibliothek in den Entwicklungsabteilungen der beteiligten Firmen schließt sich an. Damit einhergehen werden notwendige Erweiterungen und Anpassungen im Sinne eines Änderungsmanagements.

Begleitend wird sich der Arbeitskreis MEGMA in der Erstellung von Modellierungsrichtlinien für den modellbasierten Steuergeräte-Softwareentwurf engagieren. Ein möglicher erster Schritt ist die Ausarbeitung einer Wunschliste aller benötigten Kontrollstrukturen für Steuergerätesoftware und die entsprechende Umsetzung in allen Entwicklungswerkzeugen.

Eine Festlegung der weiteren Vorgehensweise hinsichtlich des gewünschten Modellaustausches wird getroffen. Angestrebt wird eine Lösung die von den Toolherstellern unterstützt wird. Hierzu besteht der Wunsch die technischen Möglichkeiten mit den Experten der Toolhersteller zu diskutieren.

In die weiteren Arbeiten fließen die neuen Features und Möglichkeiten der Entwicklungstools insbesondere hinsichtlich Zustandsgrafen und verfügbarer Datentypen mit ein.

10 Literaturverzeichnis

/1/ E. Perenthaler, B. Weichel, T. Hirth, P. Rauleder (1998):
MSR-Standards in der Praxis; VDI-Berichte 1415, 1998.

/2/ http://www.msr-wg.de

/3/ A. Wohnhaas, R. Moser: Modellaustausch zwischen Steuergeräte-Entwicklungstools auf Basis einheitlicher, Graphischer Blockbibliotheken, 3. Stuttgarter Symposium 1999, ISBN 3-8169-1751-8, Expert Verlag.

/4/ A. Wohnhaas: Rechnergestützte Steuergeräte-Softwareentwicklung. ATZ, Bd 99 (1997), Nr. 12: S. 744-754.

/5/ SYSTEMBUILD User's Guide, Integrated Systems Inc., 3260 Jay Street, Santa Clara (CA).

/6/ ASCET-SD, User's Guide, ETAS GmbH&Co.KG, Borsigstraße 10, 70469 Stuttgart.

/7/ Using SIMULINK, The MathWorks, Inc., 24 Prime Park Way, Natick (MA).