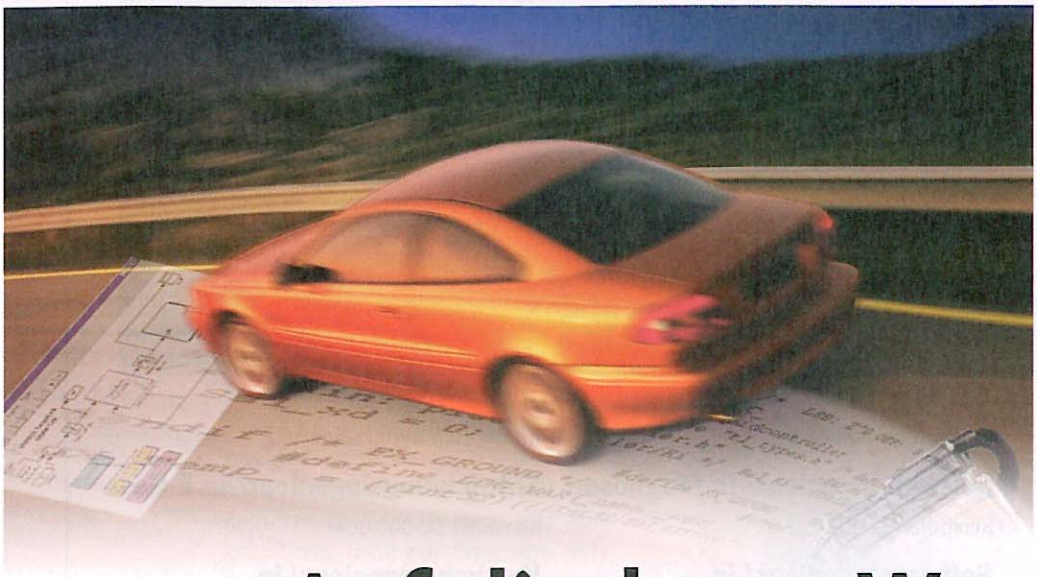


Auf direktem Weg zum Seriencode

Durchgängig entwickeln vom modellbasierten Entwurf oder Funktionsentwurf bis zum automatisch erzeugten Seriencode

Mit der steigenden Komplexität in der Steuergeräte-Entwicklung gestalten viele Automobilhersteller und deren Zulieferer auch ihre Entwicklungsprozesse neu. Modellbasierte Funktionsentwicklung und automatische Seriencode-Generierung in Interaktion mit Standards sind erfolgreiche Wegbereiter für einen durchgängigen und effizienten Entwicklungsprozess.

Von Michael Beine und Günther Gruhn

Fahrwerke werden aktiv, Bremsen werden intelligent – zahlreiche Bereiche in einem Automobil werden inzwischen mittels Elektronik in ihrer Leistungsfähigkeit verfeinert. Während in anderen Industriezweigen der Elektronik eher Stagnation zu erkennen ist, lassen die Kennzahlen aus der Automobilindustrie eher ein Konjunkturhoch vermuten. Dreißig Prozent Anteil an der gesamten Wertschöpfung eines Kfz und neunzig Prozent aller Neuerungen in einem Kfz gehen auf das Konto der Elektronik. Die Herausforderung ist dabei, der steigenden Komplexität bei gleichzeitig kürzer werdenden Entwicklungszeiten gerecht zu werden, ohne Kompromisse bei der Qualität einzugehen.

Der Einsatz von modernen Entwicklungsmethoden und -werkzeugen sowie die Optimierung des Entwicklungsprozesses bieten für viele Unternehmen die zur Zeit größten Potentiale, um den zukünftigen Anforderungen gerecht zu werden. Die Durchgängigkeit des Entwicklungsprozesses und das Zusammenspiel der eingesetzten Werkzeuge spielen hierfür eine entscheidende Rolle.

Wann ist ein Entwicklungsprozess durchgängig?

Es stellt sich zuallererst die Frage, wann ein Prozess als durchgängig bezeichnet werden darf. Zunächst einmal muss festgestellt werden, dass Entwicklungsprozesse immer zu einem Teil firmenspezifisch sind. Und jedes Unternehmen verfügt über eigene Bewertungsmaßstäbe seiner Entwicklungsprozesse. Dennoch gibt es eine Menge allgemeiner Anforderungen. Die folgenden Kriterien sollen als Anhaltspunkte dienen:

- Der Entwicklungsprozess verfügt über leicht handhabbare technische und organisatorische Schnittstellen. Technische Schnittstellen sind klar definiert und idealerweise standardisiert.
- Organisatorische Schnittstellen ergeben sich, wenn sich der Entwicklungsprozess über mehrere Organisationseinheiten erstreckt, z.B. über verschiedene Abteilungen hinweg oder auch firmenübergreifend vom Automobilhersteller zum Steuergeräte-Lieferant. In diesem Fall verfügen beide Organisationseinheiten auch ablauforganisatorisch über einen reibungslosen und eindeutig beschriebenen Prozess. Idealerweise setzen beide Organisationseinheiten darüber hinaus aufeinander abgestimmte Entwicklungswerk-

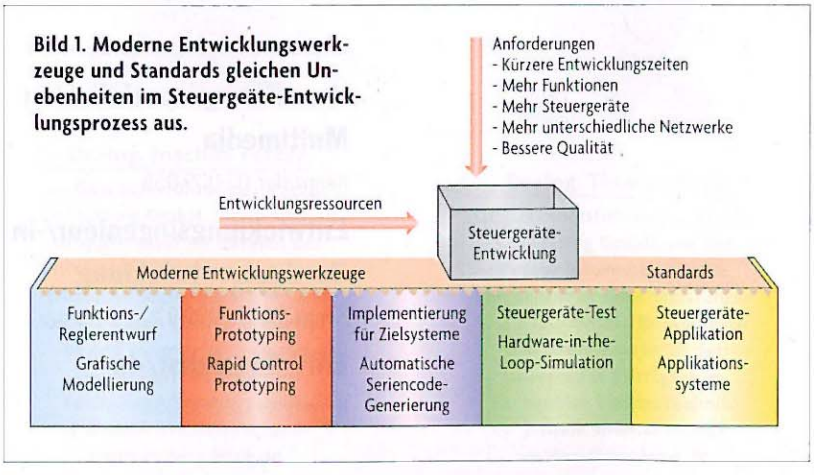


Bild 1. Moderne Entwicklungswerkzeuge und Standards gleichen Unebenheiten im Steuergeäte-Entwicklungsprozess aus.

zeuge ein. Wesentlich ist das optimale Zusammenwirken von technischen und organisatorischen Schnittstellen.

- Aufeinander abgestimmte Entwicklungswerkzeuge unterstützen den Entwicklungsprozess. Sie lassen in einfacher Weise notwendige Iterationen zwecks Optimierung, Verifikation oder Fehlerbehebung zu.

Ohne diese Abstimmung kommt es zu aufwendigen, teilweise manuellen Konvertierungen von Daten. Missverständnisse, Lücken in der Spezifikation oder gar Fehler sind die Folge. Und diese Probleme entstehen bei jeder Änderung bzw. Iteration aufs Neue. Eine durchgängige Werkzeugkette verhindert diese Probleme bereits im Ansatz.

Schließlich sollte der Prozess die Wiederverwendung von Daten ermöglichen und fördern. Anzuführen sind hier beispielsweise die Verwendung von Testdaten oder Testszenarien in mehreren Entwicklungsschritten (Prototyping, Steuergeräte-Test) oder auch die Wiederverwendung von Modellen aus der Entwurfs- und Prototyping-Phase als funktionale Spezifikation für den Seriencode-Generator.

Hieraus lassen sich die heutigen Bestrebungen zahlreicher Automobilhersteller ableiten und verstehen:

- Mehr offene Systeme und Standards definieren und zusammen mit der Zulieferindustrie etablieren.
- Durch mehr Standards zu mehr Werkzeug-Unabhängigkeit und somit zu besserem Investitionsschutz und zu einer besseren Integration der einzelnen Entwicklungsschritte kommen.
- Durch mehr Standards die Wiederverwendbarkeit von Software erhöhen.
- Mit neuen Entwicklungsmethoden einen ganzheitlichen Blick auf den Entwicklungsprozess ermöglichen.

Bild 1 verdeutlicht, dass die Steuergeräte-Entwickler durch steigende Anforderungen einem erhöhten Druck ausgesetzt sind. Der Einsatz von modernen Entwicklungswerkzeugen und Standards hilft, den Entwicklungsprozess zu glätten, um den erhöhten Anforderungen gerecht zu werden.

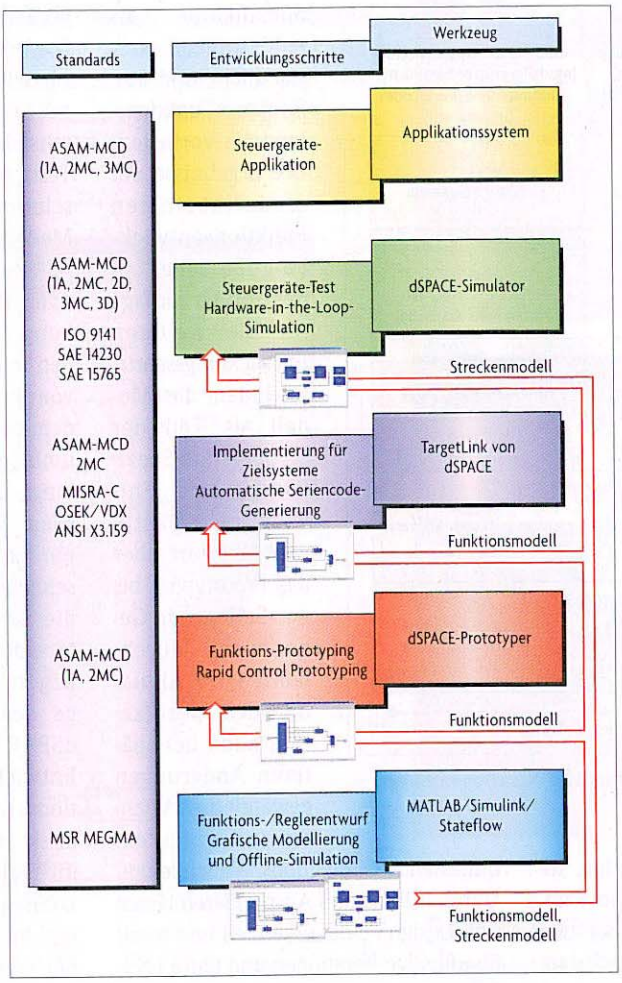


Bild 2. Standards schaffen exakt definierte Schnittstellen im Steuergeräte-Entwicklungsprozess. Werkzeuge setzen diese Standards um.

Standards für Modelle und Seriencode

Standards spielen eine Schlüsselrolle in Bezug auf die Durchgängigkeit eines Entwicklungsprozesses (*Bild 2*). Dazu wird nachfolgend auf ausgewählte Standards im Bereich der modellbasierten Funktionsentwicklung und der automatischen Seriencode-Generierung eingegangen. Als Sprachenstandard gibt es im Umfeld der grafischen Modellierung erfolgreiche Quasi-Standards. Diese haben sich als proprietäre Produkte die Marktdurchdringung meist über den Einsatz an Hochschulen bis in die Industrie erkämpft. Besonders etabliert ist hier das Entwicklungswerkzeug Matlab/Simulink/Stateflow von The MathWorks.

Erste Ansätze für eine Normierung bei der grafischen Modellierung zeigt der Standard MSR MEGMA. Vertreter der Automobilindustrie möchten mit diesem Standard den Blocksatz für grafische Modellierungswerkzeuge normieren. Im Bereich der Seriencode-Generierung ist die bedeutendste Norm für einen Code-Generator die Norm seiner Ausgabesprache. Im Normalfall ist dies die Programmiersprache C, die durch ISO/IEC 9899 bzw. die dazu identische Norm ANSI X3.159 international genormt wurde.

Ein weiterer wichtiger Standard, der an Bedeutung zunimmt, ist der britische MISRA-C-Standard (MISRA: Motor Industry Software Reliability Association). Dieser enthält Regeln zur Vermeidung von häufig gemachten Softwarefehlern.

Der OSEK/VDX-Standard fällt in den Bereich der Echtzeit-Betriebssysteme und ist damit auch relevant für die Seriencode-Generierung. Er definiert eine einheitliche Betriebssystem-Funktionalität und eine einheitliche Programmierschnittstelle der Betriebssystemdienste. Damit ist es einem Seriencode-Generator möglich, den Anschluss auch an diese Softwareschicht zu automatisieren. Die Vorteile des OSEK/VDX-Standards sind:

- Verkürzung der Entwicklungszeit,
- erhöhte Software-Qualität durch Wiederverwendung bereits getesteter Softwaremodule,
- optimale Speicherausnutzung durch Skalierbarkeit,
- Wiederverwendung von Software in anderen Umgebungen.

Als letzter Standard sei der ASAM-MCD-2MC-Standard erwähnt. Dieser spielt eine wichtige Rolle, um die vom Anwender bei der Seriencode-Generierung spezifizierten Variablen oder Parameter auch für die Steuergeräte-Ap-

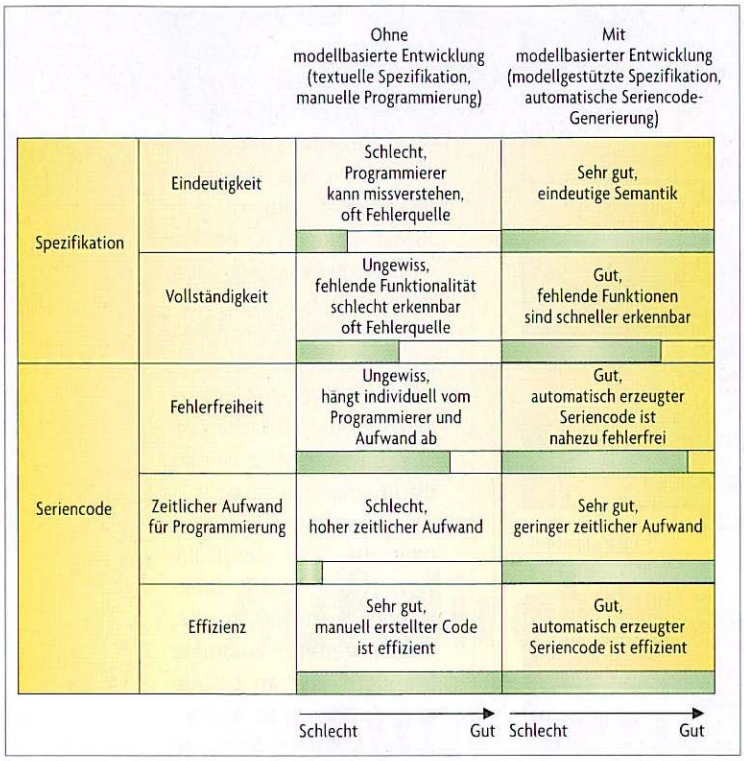


Bild 3. Modellbasierte Funktionsentwicklung hilft, unnötige Iterationen zu vermeiden und damit Kosten und Zeit zu sparen.

plikation zur Verfügung zu stellen. Informationen über Variablen und Parameter werden dem Applikationssystem über eine Beschreibungsdatei nach dem ASAM-MCD-2MC-Standard bekannt gegeben. Alle wichtigen Applikationssysteme unterstützen heute diesen Standard.

Viele der hier erwähnten Standards legen technische Schnittstellen fest. Als Resultat können Entwicklungswerkzeuge herstellerunabhängig aufeinander abgestimmt werden. Neben den Standards ist es aber auch die modellbasierte Funktionsentwicklung selbst, die einen wichtigen Beitrag für einen durchgängigen Entwicklungsprozess liefert.

Spezifikation als ausführbare Funktionsmodelle

Wesentliche Problembereiche im klassischen Steuergeräte-Entwicklungsprozess stellen inkonsistente Spezifikationen und mangelhafte Werkzeugunterstützung dar. Inkonsistente und nicht eindeutige, in Prosa verfasste Spezifikationen führen regelmäßig zu Fehlern, Missverständnissen und kostspieligen Iterationen. In einem idealen Prozess sollten die Spezifikation bzw. Teile der Spezifikation als eindeutige, verständliche und ausführbare Funktionsmodelle vorliegen. Die Ergebnisse der modellbasierten Funktionsentwicklung (Modelle) bilden die Grundlage für die weiteren Entwicklungsschritte. Indem das Modell als Teil der Steuergeräte-Spezifikation die Entwicklungsschritte vom Entwurf über das Prototyping bis zur Seriencode-Generierung durchläuft, wird Aktualität der Spezifikation auch bei späteren Änderungen gewährleistet. Wichtig sind außerdem umfassende Simulationsmöglichkeiten. Durch frühzeitiges Ausprobieren lassen sich Design-Fehler erkennen und somit überflüssige Iterationen und teure Fehlentwicklungen vermeiden.

Bild 3 verdeutlicht, dass modellbasierte Funktionsentwicklung und automatische Seriencode-Generierung den Prozess für die Erzeugung des Steuergeräte-Codes erheblich beschleunigen. Die Verwendung von bereits erprobten Modellen als Eingabe für den Seriencode-Generator, zusammen mit der Bereitstellung von einfachen Analyse- und Testmöglichkeiten, ermöglicht eine rasche und fehlerfreie Umsetzung des Modells in Steuergeräte-Code.

Wie bereits erwähnt, ist im automotiven Umfeld die Entwicklungsumgebung Matlab/Simulink/Stateflow für den modellbasierten Funktionsentwurf von Steuergeräten weit verbreitet. Zudem bietet diese Umgebung Zugriff auf umfangreiche Berechnungs-, Darstellungs- und Optimierungsverfahren. Vor allem aber das Vorhandensein moderner Entwicklungswerkzeuge für das schnelle Funktions-Prototyping und für die Seriencode-Generierung ist entscheidend. Besonders erfolgreich werden in diesen Bereichen die Werkzeuge der Firma dSPACE eingesetzt. dSPACE Prototyper ist ein flexibles Entwicklungssystem, mit dem Funktionsentwürfe ohne Programmierung direkt aus Simulink/Stateflow-Modellen im realen Fahrzeug erprobt werden können. Anschließend wird mit TargetLink, dem Seriencode-Generator der Firma dSPACE, Seriencode automatisch und direkt aus den zuvor erprobten Modellen generiert. Somit ist gleichzeitig Konsistenz von Spezifikation, Seriencode und Dokumentation gewährleistet.

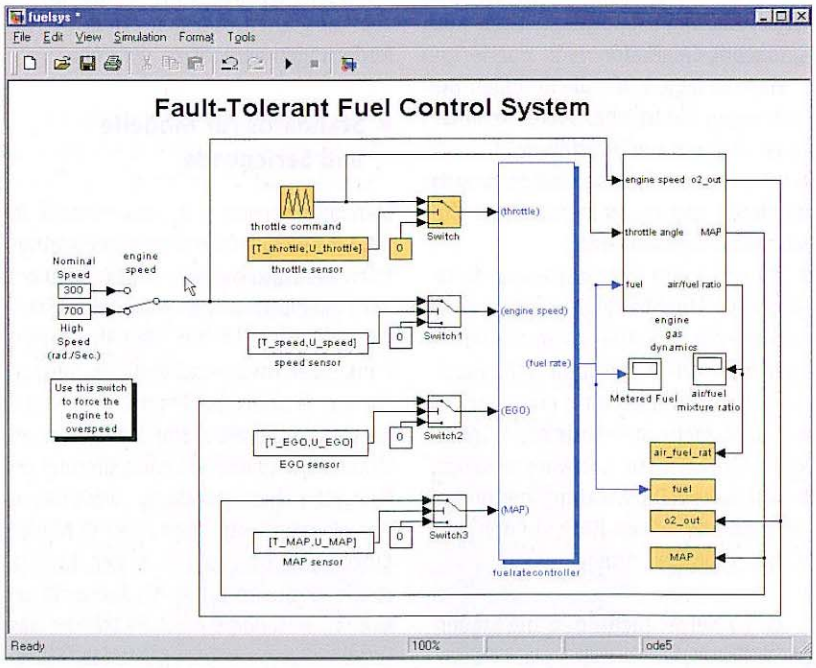
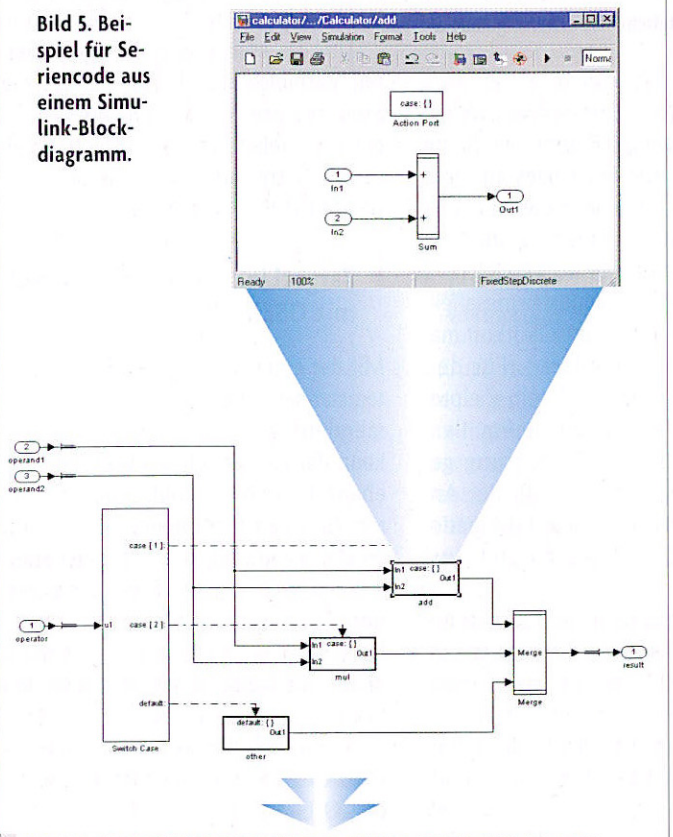


Bild 4. Simulink-Blockdiagramm als Basis für TargetLink.

Trotz der zuvor beschriebenen und allgemein anerkannten Vorteile der automatischen Code-Generierung war deren praktischer Einsatz im Bereich Seriencode lange Zeit nicht möglich. Der generierte Code war hinsichtlich Laufzeit und Speicherverbrauch nicht effizient genug. Der Einsatz von schnelleren, mit mehr Speicher bestückten Prozessoren,

Bild 5. Beispiel für Seriencode aus einem Simulink-Blockdiagramm.




```
/*********************************************************************\
   Model step function of subsystem(s): Calculator
\*********************************************************************/
Void Calculator(
   UInt16 Sbl_operator  /* LSB: 2^0 OFF:  0 MIN/MAX:  0 .. 65535 */,
   UInt16 Sbl_operand1  /* LSB: 2^0 OFF:  0 MIN/MAX:  0 .. 65535 */,
   UInt16 Sbl_operand2  /* LSB: 2^0 OFF:  0 MIN/MAX:  0 .. 65535 */,
   UInt16 * Sbl_result  /* pointer to LSB: 2^0 OFF:  0 MIN/MAX:  0 .. 65535 */)
{
   /* SwitchCase: Calculator/Switch Case */
   switch ( Sbl_operator ) {
      case 1: {

         /*
            Sum: Calculator/add/Sum
            # combined # TargetLink outport: Calculator/result_

            # combined # update of inport  for function Calculator/add.
            # combined # update of inport  for function Calculator/add. */
         *Sbl_result = (UInt16)(Sbl_operand1 + Sbl_operand2);

         break;
      }

      case 2: {

         /*
            Product: Calculator/mul/Product
            # combined # TargetLink outport: Calculator/result_

            # combined # update of inport  for function Calculator/mul.
            # combined # update of inport  for function Calculator/mul. */
         *Sbl_result = Sbl_operand1 * Sbl_operand2;

         break;
      }

      default: {

         /*
            Reference of merge block: Calculator/other/Constant: Calculator/Merge
            # combined # TargetLink outport: Calculator/result_ */
         *Sbl_result = 4711;

         break;
      }

   }
}
```

Modell	Seriencode
Modell ist grafisch und leicht verständlich	Seriencode muss lesbar sein
Modell ist hardwareunabhängig	Seriencode ist (in der Regel) prozessorabhängig (8, 16, 32 bit), optimiert für bestimmte Prozessorarchitektur
Modell ist wiederverwendbar	Seriencode ist nur sehr eingeschränkt wiederverwendbar
Modell verwendet physikalische Einheiten mit Gleitkomma-Arithmetik	Seriencode verwendet wahlweise Festkomma-Arithmetik mit skalierten Integerzahlen oder Gleitkomma-Arithmetik

Modell und Seriencode zeichnen sich durch mitunter gegensätzliche Anforderungen aus. Maßstab für den Seriencode ist alleine seine Effizienz.

um den Mangel an Effizienz auszugleichen, kam und kommt aus Kostengründen nicht in Frage. Somit galt die Effizienz lange Zeit als Herausforderung und ist auch heute noch das wichtigste Kriterium eines leistungsfähigen Seriencode-Generators. Eine Gegenüberstellung von Modelleigenschaften und Seriencode (siehe *Tabelle*) verdeutlicht, welche – manchmal sogar gegensätzlichen – Anforderungen der Seriencode-Generator erfüllen muss.

Der Seriencode-Generator TargetLink von dSPACE erfüllt diese Anforderungen und generiert hocheffizienten Code, der im Hinblick auf Laufzeiteffizienz und Speicherverbrauch vergleichbar ist mit handgeschriebenem Code.

Funktionsweise des Seriencode-Generators TargetLink

Die Basis für die automatische Seriencode-Generierung mit TargetLink sind Blockschaltbilder und Zustandsdiagramme, die in Simulink und Stateflow modelliert sind (*Bild 4*). Für die Seriencode-Generierung werden die Simulink-Blöcke und Zustandsdiagramme mit Eingabemöglichkeiten für die notwendigen Implementierungsinformationen versehen. Dazu zählen beispielsweise Angaben zur Skalierung oder die Bestimmung der Speicherklasse.

Mit der Gleitkomma-Simulation auf dem PC ist es möglich, Simulationsergebnisse als Referenz für den späteren Vergleich zum generierten Code aufzuzeichnen. TargetLink kann dazu während der Simulation die Signalverläufe eines jeden Blocks speichern. Anschließend erfolgt, sofern erforderlich, die Skalierung der Variablen als Vorbereitung auf eine spätere Festkomma-Arithmetik. Dies geschieht wahlweise automatisch oder manuell. Minima und Maxima der zuvor aufgezeichneten Signalverläufe können dabei als Wertebasis dienen.

Schließlich wird der Seriencode von TargetLink erzeugt (*Bild 5*). Die Simulation des generierten Codes auf dem PC gibt dem Anwender die Möglichkeit, direkt Implementierung und Spezifikation miteinander zu vergleichen, beispielsweise um Quantisierungseffekte bei Festkomma- und Gleitkomma-Arithmetik zu analysieren. Für den Test des Seriencodes auf dem Zielprozessor kann dieser auf einem Evaluierungs-Board zur Ausführung gebracht werden. Damit erhält der Anwender zusätzlich genaue Informationen zu Speicherverbrauch und Ausführungszeit.

Eine Besonderheit von TargetLink ist die Möglichkeit zur Erzeugung von prozessorspezifischem Code. TargetLink erzeugt wahlweise portablen ANSI-C-Code oder für verschiedene Prozessor-Compiler-Kombinationen optimierten Code. Je nach Anwendungsfall können so nochmals signifikante Verbesserungen hinsichtlich Effizienz und Speicherverbrauch erzielt werden. Möglich wird dies durch die Nutzung der unterschiedlichen Prozessor-Vorteile. TargetLink erzeugt dazu auf die jeweilige Prozessor-Architektur abgestimmte Codemuster, bedient sich compilerspezifischer Spracherweiterungen oder erzeugt direkt Assembler-Code.

Um die vom Anwender spezifizierten Variablen und Parameter für die Steuergeräte-Applikation zur Verfügung zu stellen, erzeugt TargetLink eine Beschreibungsdatei nach dem ASAM-MCD-2MC-Standard.

TargetLink im Zusammenspiel mit OSEK/VDX

Mit der Unterstützung des Betriebssystemstandards OSEK/VDX in der kommenden TargetLink-Version (*Bild 6*) kann der Anwender schon auf Modellebene Betriebssystemdienste und deren Attribute bestimmen. Dies hat sowohl Auswirkungen auf den generierten Code als auch auf die Konfiguration des Betriebssystemkerns. Für die Unterstützung von OSEK/VDX muss daher ein weiteres Werkzeug in den Spezifikations- und Generierungsprozess eingebunden werden. Zu diesem Zweck existiert die Beschreibungssprache OIL (OSEK Implementation Language). Diese Beschreibungssprache ist Teil des OSEK/VDX-Standards und

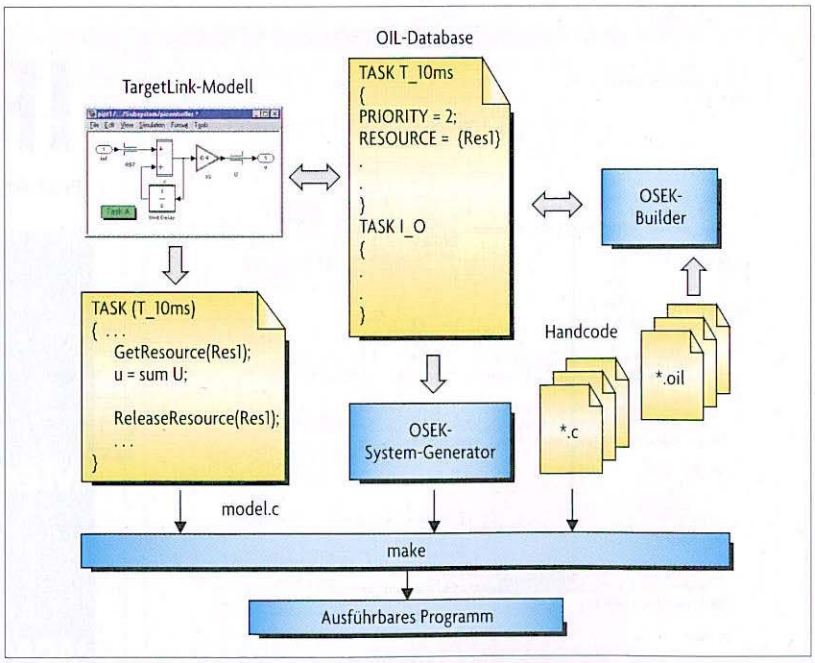


Bild 6. Zusammenwirken von OSEK/VDX-Betriebssystemwerkzeugen und TargetLink. Dieses Zusammenwirken ist mit der kommenden Version von TargetLink möglich.

```
TASK T_10ms
{
  PRIORITY = 4;
  SCHEDULE = NON;
  ACTIVATION = 1;
  AUTOSTART = FALSE;
  RESOURCE = Resource2;
  EVENT = Event2;
};

TASK T_100ms
{
  PRIORITY = 3;
  SCHEDULE = FULL;
  ACTIVATION = 1;
  AUTOSTART = FALSE;
  RESOURCE = Resource2;
};

TASK T_crankshaft
{
  PRIORITY = 3;
  SCHEDULE = FULL;
  ACTIVATION = 1;
  .
  .
```

Bild 7. Auszug aus einer OIL-Datei.

definiert Betriebssystemobjekte wie Tasks, Events, Messages oder Ressourcen. Die Beschreibung erfolgt in einer OIL-Datei (*Bild 7*) und kann mit einem ASCII-Editor editiert oder mit einem vom Hersteller des Betriebssystems bereitgestellten OIL-Editor bearbeitet werden.

Mit der Unterstützung des OSEK/VDX-Standards ist TargetLink in der Lage, die OIL-Datei zu lesen, zu interpretieren und sie auch zu ändern, womit eine weitere Möglichkeit zur Beschreibung der OIL-Datei entsteht. Schon auf Modellebene kann der Anwender Einstellungen zu Betriebssystemobjekten wie beispielsweise Tasks vornehmen. Diese Informationen werden von TargetLink nach der Bearbeitung direkt in die OIL-Datei geschrieben.

Mit der aktualisierten OIL-Datei kann sich der Anwender sofort einen passenden Betriebsystemkern vom System-Generator des Betriebssystemherstellers erstellen lassen. Der Anwender muss sich nicht mehr um die Konsistenz zwischen Blockdiagrammen, generiertem C-Code und dem Betriebssystem kümmern. Tippfehler, Syntaxfehler oder Fehlspezifikationen in der OIL-Datei kommen nicht mehr vor. Somit wird dank OSEK/VDX auch die Implementierung des C-Codes auf das Steuergerät erheblich vereinfacht. Die eingangs beschriebenen Durchgängigkeitsmerkmale für einen optimierten Entwicklungsprozess werden an dieser Stelle deutlich sichtbar.

Mit TargetLink erstellter Seriencode spart bei vielen Unternehmen mehr als 40 % der Entwicklungszeit für die betreffenden Funktionen ein. In Entwicklerteams entsteht dabei oft der Eindruck, über einen zusätzlichen Programmierer zu verfügen. Dieser Eindruck verstärkt sich, wenn Standards wie OSEK/VDX dabei helfen, TargetLink zusätzlich zu einem Integrationswerkzeug werden zu lassen.

Mit modernen Werkzeugen effektiver arbeiten

Bei der steigenden Komplexität in der Steuergeräte-Entwicklung spielen die Durchgängigkeit des Entwicklungsprozesses und das Zusammenspiel der eingesetzten Werkzeuge eine entscheidende Rolle. Moderne Entwicklungsmethoden wie modellbasierte Funktionsentwicklung und die automatische Seriencode-Generierung sind Meilensteine auf dem Weg zu einem durchgängigen Prozess. Standards leisten gerade im Bereich der technischen Schnittstellen ebenfalls einen wichtigen Beitrag. Sie bieten zudem die Möglichkeit, an sinnvollen Stellen proprietäre Wege zu verlassen, und tragen auf diese Weise zu mehr Investitionssicherheit bei.

Im Zusammenspiel mit einem entsprechenden organisatorischen Rahmen lassen sich diese Potentiale im Sinne einer effektiveren Steuergeräte-Entwicklung nutzen. *jk*

Literatur

[1] *Köster, L.; Thomsen, T.; Stracke, R.:* Connecting Simulink to OSEK/VDX: Automatic Code Generation for Real-Time Operating Systems with TargetLink. SAE Technical Paper 2001-01-0024, März 2001.

[2] *Zobiegala, C.:* Code-Generierung. Seminararbeit, FB Informatik, Universität Dortmund, November 1999.

[3] *Thomsen, T.:* Integration automotiver Standards in die Seriencodegenerierung. Technical Paper, Februar 2002.

[4] *Müller, A.:* Entwurfsmethodik und automatisierte Verteilung für Steuerungssoftware in einem verteilten Rechnersystem in der Automobilelektronik. Dissertation, Juli 1999.

[5] *Bechberger, P.:* Modellbasierte Softwareentwicklung für Steuergeräte. Automotive Engineering Partners Special, Februar 2000.

Dipl.-Math. Michael Beine

hat nach dem Studium der Technomathematik an der Universität-Gesamthochschule Paderborn bei der Firma dSPACE als Software-Entwickler begonnen. 1999 wechselte er innerhalb des Unternehmens für zwei Jahre zur amerikanischen Tochterfirma dSPACE Inc. nach Detroit, und war dort u.a. als Applikationsingenieur tätig. Seit seiner Rückkehr im Oktober 2001 ist er Produkt-Manager für TargetLink.

E-Mail: mbeine@dspace.de

Dipl.-Ing. (FH) Günther Gruhn

beendete 1987 sein Studium der Elektrotechnik an der Fachhochschule Bielefeld und entwickelte bis 1992 im Bereich der Automatisierungstechnik Software bei der Siemens-Nixdorf AG. Danach war er mehrere Jahre in der Informationsverarbeitung als Projektleiter tätig. 2000 trat er bei der dSPACE GmbH im Marketing als technischer Redakteur ein und betreut dort heute redaktionell den Produktbereich Seriencode-Generierung.

E-Mail: ggruhn@dspace.de