Einsatz von automatischer Code-Generierung für die Entwicklung von sicherheitskritischer Software

Michael Beine
dSPACE GmbH, Paderborn, Deutschland

Michael Jungmann
MTU Aero Engines GmbH, München, Deutschland

Einleitung

Die Anzahl sicherheitskritischer Systeme in Fahrzeugen nimmt rapide zu. Während bis vor wenigen Jahren der Ausfall eines Computersystems im Fahrzeug im schlimmsten Fall den Ausfall einer Funktion nach sich zog, wird bei zukünftigen Systemen eine fehlerhafte Reaktion auf einen Defekt häufig auch ein Sicherheitsrisiko für die Fahrzeuginsassen und andere Verkehrsteilnehmer darstellen.

Bei der Entwicklung sicherheitskritischer Systeme für automotive Anwendungen kann man auf Erfahrungen aus dem Luftfahrtsektor zurückgreifen. In der Luftfahrt werden seit einigen Jahrzehnten programmierbare Systeme für die Fluglageregelung, Triebwerksregelung, Fahrwerksteuerung usw. eingesetzt. In Bezug auf Sicherheit und Zuverlässigkeit sind die Anforderungen an diese Systeme vergleichbar mit den Anforderungen an Steer-by-Wire- oder Brake-by-Wire-Systeme, die derzeit in der Fahrzeugindustrie entwickelt werden.

Die Automobilindustrie setzt verstärkt modellbasierte Entwurfsmethoden und automatische Code-Generierung für die Software-Entwicklung ein, um den ständig steigenden Anforderungen, verursacht durch die wachsende Anzahl elektronischer Systeme, zunehmende Komplexität sowie Zeit- und Kostendruck, zu begegnen. Der Einsatz automatischer Code-Generatoren für die Entwicklung sicherheitskritischer Systeme ist bisher kaum verbreitet. Zum einen unterliegt der Code sicherheitskritischer Systeme ganz bestimmten Anforderungen. Zum anderen fangen viele Software-Hersteller gerade erst an, geeignete Entwicklungsstandards anzuwenden, und können somit nicht gleichzeitig die Einführung automatischer Code-Generierung in Angriff nehmen.

Jedoch verlangen gerade die Komplexität und die Funktionsanforderungen sicherheitskritischer Systeme den Einsatz moderner Werkzeuge für Entwicklung, Entwurf, Implementierung, Verifikation und Validierung solcher Systeme. Der automatischen Code-Generierung kommt dabei für die Implementierung eine Schlüsselrolle zu.

Sicherheitsnormen

Um die von sicherheitskritischen Systemen ausgehenden Gefahren zu minimieren, wurden spezielle Normen und Prozesse für deren Entwicklung definiert. Diese Sicherheitsnormen definieren einen Katalog von Maßnahmen, die eingehalten werden müssen, um die angestrebte Sicherheit zu erreichen. Die Maßnahmen können generell in drei Kategorien unterteilt werden: die Auswahl von Entwicklungsmethoden und Werkzeugen, die Implementierung sowie die Verifizierung und Validierung des Systems.

Die Normen unterscheiden sich bezüglich ihrer Blickwinkel auf das zu entwickelnde System. Einige befassen sich mit der Entwicklung des Gesamtsystems. Dazu gehören die IEC 61508 sowie die ARINC 653. Andere Normen wie die RTCA DO-178B und die DoD-2167A behandeln ausschließlich die Entwicklung der Software, sind dafür aber detaillierter und damit konkreter.

Die in der Automobilelektronik etablierte Norm ist die IEC 61508 [1]. Diese Sicherheitsnorm ist ein sehr allgemein gehaltener Standard, der branchen- oder projektspezifischer Detaillierungen bedarf. Im Bereich der Software-Entwicklung haben Untersuchungen gezeigt, dass die ursprünglich für den Luftfahrtsektor definierte Software-Entwicklungsnorm RTCA DO-178B [2] eine auch für den Automobilbereich geeignete Konkretisierung der Sicherheitsnorm IEC 61508 darstellt [3 et al]. Während die IEC 61508 beispielsweise

„Software Integrity Level“ (SIL) von 1 bis 4 definiert, kennt die RTCA DO-178B „Software Level“ von A bis E. Pendant zu SIL 3 ist Software Level A.

Software-Entwicklungsprozess für sicherheitskritische Systeme

Ein geeigneter Software-Entwicklungsprozess orientiert sich am gängigen V-Modell. Das V-Modell beschreibt auf der linken Seite den Implementierungspfad, der von der Aufstellung der Systemanforderungen ausgehend, die Detaillierungsschritte bis zur Erstellung des Seriencodes beschreibt. Auf der rechten Seite wird der Verifikationspfad dargestellt, der jeder Implementierungsphase eine horizontal gegenüberliegende Verifikationsphase zuordnet.

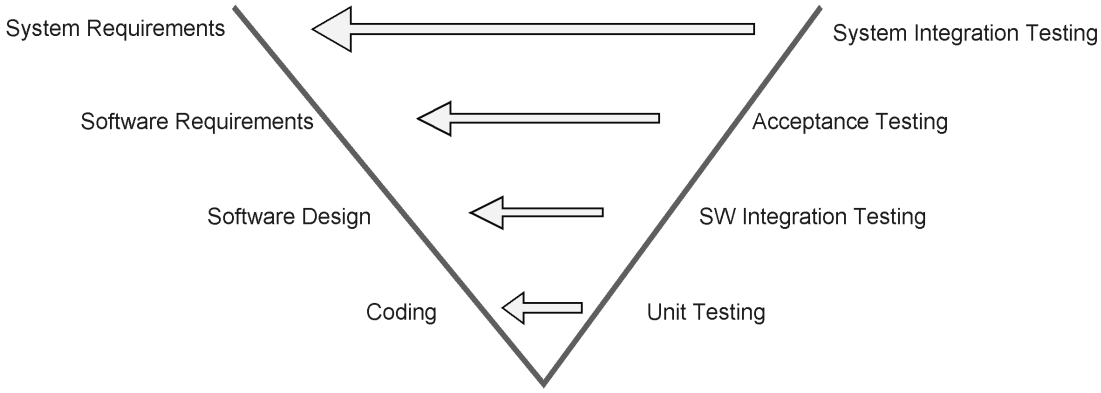

Abbildung 1: V-Modell für Software-Entwicklung.

Der Einsatz modellbasierter Entwurfsmethoden für die Implementierung bietet eine Vielzahl von Vorteilen[4]. Zudem liefern modellbasierte Entwicklungswerkzeuge eine ausführbare Spezifikation und erfüllen somit die Anforderungen der IEC 61508 bezüglich einer semi-formalen Methode für die Spezifikation. Im Gegensatz zu einer rein textuellen Spezifikation lässt eine ausführbare Spezifikation weit weniger Raum für Interpretationen, wodurch Fehlauslegungen und Fehler reduziert werden. Zudem wird ein durchgängiger Implementierungspfad gewährleistet, und mittels automatischer Code-Generierung gehen auch die Phasen Software-Entwurf und Codierung nahtlos ineinander über. Durch den modellbasierten Entwurf kann das Systemverhalten in jeder Phase der Implementierung bewertet und getestet werden, wodurch Fehler in einem frühen Stadium entdeckt und behoben werden können. Somit kann das Ergebnis jeder Entwicklungsphase durch Verifikation und Validierung abgesichert werden.

Die erste Phase der Verifikation ist die Unit- beziehungsweise Modultestphase. Sie dient der Verifikation des Seriencodes und der kleinsten funktionalen Blöcke der ausführbaren Software. Zu den Verifikationsaktivitäten dieser Phase gehören die statische Analyse und die dynamischen Funktionstests. Software-Integrationstests, Abnahmeprüfungen und Systemintegrationstests vervollständigen den Verifikationsprozess.

Die in den einzelnen Phasen durchzuführenden Schritte unterscheiden sich in den einzelnen Normen nur leicht. Allen Normen gemeinsam ist jedoch, dass nur die konsequente Durchführung aller Aktivitäten die von der Norm angestrebte Sicherheit ermöglicht. Kein einzelner Schritt dieses Prozesses für sich genommen erlaubt eine Aussage über die Qualität der entwickelten Software.

Vorteile der Automatischen Code-Generierung

Eckpfeiler des Software-Entwicklungsprozesses ist die Umsetzung des Software-Entwurfs in Code, die automatische Code-Generierung die Schlüsseltechnologie in dieser Phase der Entwicklung. Mehrere Gründe sprechen für den Einsatz automatischer Code-Generierung bei der Entwicklung sicherheitskritischer Systeme.

Der Einsatz automatischer Code-Generierung bietet sich vor allem an, wenn der Software-Entwurf schon als ausführbare Spezifikation vorliegt. Denn jeder manuelle Übersetzungsschritt ist fehleranfällig und zeitaufwendig. Die Gefahr, dass sich Fehler einschleichen ist insbesondere bei Iterationen und sukzessiven Änderungen hoch. Oftmals geht dabei die Konsistenz zwischen Spezifikation und dem Code verloren. Beim Einsatz automatischer Code-Generierung ist das nicht der Fall. Ein Code-Generator übersetzt das Modell und Änderungen im Modell in Code zuverlässig, reproduzierbar und mit einer geringeren Fehlerrate als es

bei Handprogrammierern der Fall ist. Er stellt sicher, dass der generierte Code immer mit dem Software-Entwurf übereinstimmt. Da üblicherweise auch die Dokumentation zusammen mit dem Code generiert werden kann, ist auch diese immer auf dem neuesten Stand. Somit kann mit automatischer Code-Generierung die Konsistenz von Software-Entwurf, Implementierung und Dokumentation gewährleistet werden.

Der Einsatz eines Code-Generators ist keine Garantie für fehlerfreie Software. Aufgrund der unendlichen Kombinationsmöglichkeiten von Modellierungsvarianten und Parametern gibt es keinen formellen Beweis, dass es nie Codierungsfehler geben wird. Jedoch macht ein sorgfältig entwickelt und getesteter Code-Generator wesentlich weniger Fehler als ein Handprogrammierer. Während einem Programmierer Flüchtigkeitsfehler unterlaufen können, liefert der Code-Generator jeden Tag dieselbe Qualität. Er lernt aus jedem korrigierten Fehler – und das ohne zu vergessen.

Anforderungen an den Code-Generator

Die Qualität und Zuverlässigkeit des Code-Generator und des generierten Codes sind von allergrößter Bedeutung, vor allem für die Entwicklung von sicherheitsgerichteter Software.

Zertifizierung von Entwicklungswerkzeugen

Die IEC 61508 fordert an einer Stelle den Einsatz von zertifizierten bzw. betriebsbewährten Entwicklungswerkzeugen, mindestens jedoch eine Begründung für den Einsatz eines Werkzeuges. Zertifizierung verlangt ein Vorgehen entsprechend einer nationalen oder internationalen Norm. Die IEC 61508 selbst macht jedoch keine Aussage darüber, wie eine solche Zertifizierung auszusehen hat und erfolgen muss.

Die RTCA DO-178B drückt sich da genauer aus. Sie macht eine klare Aussage, wann eine Zertifizierung nötig ist. Nämlich genau dann, wenn Prozessschritte durch den Einsatz eines Entwicklungswerkzeuges eliminiert, reduziert oder automatisiert werden, ohne dass der Output des eingesetzten Werkzeugs im Detail überprüft und getestet wird. Bezogen auf einen Code-Generator, dessen Output ja der generierte Code ist, welcher in der horizontal gegenüberliegenden Verifikationsphase, dem Unit Test überprüft wird, bedeutet dies, dass das Einsparpotential durch eine Code-Generator Zertifizierung nur in der Unit-Testphase liegt. Diese Reduzierungen betreffen aber nur einzelne Aktivitäten der Unit-Testphase und sind projektspezifisch zu bestimmen. Eine generelle Auslassung dieser Verifikationsphase ist in der Regel nicht möglich.

Die Zertifizierung des Code-Generators setzt das Vorhandensein eines zertifizierbaren Code-Generators voraus. Beispielsweise muss die Entwicklungsdokumentation für den Code-Generator vom Tool-Hersteller entsprechend der Norm erstellt worden sein und ausgeliefert werden. Das bedeutet, dass für den Einsatz eines Code-Generators in einem Projekt gemäß IEC 61508 SIL 3 ein Entwicklungsprozess nach RTCA DO-178B Level A für den Code-Generator notwendig ist. Darüber hinaus ist eine projektspezifische Qualifikation des Code-Generators nötig, die das Zusammenspiel von Code-Generator, Compilation System und Zielprozessor überprüft und nur für diese Konfiguration gültig ist. Demnach ist diese Form der Zertifizierung systemspezifisch.

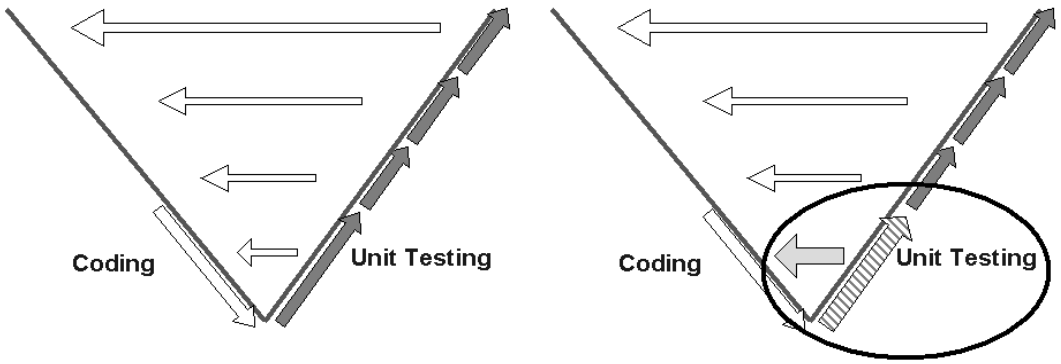

Abbildung 2: Einsparungspotential beim Einsatz eines zertifizierten Code-Generators.

Dadurch entsteht ein enormer Aufwand, der sich in den Kosten für den zertifizierbaren Code-Generator und der projektspezifischen Qualifikation niederschlägt. Der Preis eines zertifizierbaren Code-Generators kann leicht beim fünf- bis zehnfachen gegenüber dem nicht zertifizierbaren liegen. Die projektspezifische

Qualifikation verursacht Extra-Kosten, die nochmals 50 000 bis 100 000 Euro betragen können. Die Zertifizierung des Code-Generators ist darüber hinaus sehr zeitaufwendig, so dass der Wartungszyklus für einen solchen Code-Generator heute bei ca. zwei Jahren liegt, während für den nicht zertifizierten meist ca. 3 Monate ausreichen, um eine korrigierte Version zu erhalten.

Die Alternative besteht darin, die Applikationssoftware allen vorgeschriebenen Verifikationsaktivitäten zu unterziehen, so als wäre sie ohne Zuhilfenahme eines automatischen Code-Generators entstanden. Der dafür eingesetzte Verifikationsprozess kann sogar Fehler finden, die durch einen automatischen Code-Generator entstanden sein könnten. Die zuvor beschriebenen Nachteile bleiben bei diesem Ansatz außen vor und zudem ist er für die meisten Projekte kostengünstiger. Diese Alternative ist besonders attraktiv, wenn die Verifikation weitgehend automatisiert erfolgt.

Software-Qualitätsstandards

Ein Verzicht auf Zertifizierung heisst jedoch nicht, dass Abstriche an der Qualität gemacht oder in Kauf genommen werden. Systematische Entwicklung und Test sind für einen Code-Generator, welcher ja selbst eine komplexe Software ist, von allergrößter Bedeutung. Prozessororientierte Entwicklungsstandards wie CMM, ISO 9001 und ISO/IEC 15504 bilden einen Rahmen für die Beherrschung der Komplexität in der Software-Entwicklung.

Eine für die Entwicklung eines Code-Generators anwendbare Norm ist die ISO/IEC 15504, auch bekannt unter dem Namen SPICE ("Software Process Improvement Capability Determination") . Von einer Arbeitsgruppe der ISO/IEC auf Basis bestehender Qualitätsnormen definiert, vereint SPICE Ansätze aus ISO 9001 und CMM. SPICE verlangt regelmäßige, formelle Prozess-Assessments durch unabhängige Organisationen. Aus den Ergebnissen der Assessments werden Schritte zur Prozessverbesserung abgeleitet. Wird diese Norm eingehalten und ein hoher Übereinstimmungsgrad erreicht, stehen die Chancen gut, dass die produzierte Software qualitativ hochwertig ist.

Daher verlangt die Hersteller Initiative Software (HIS), ein Zusammenschluss der deutschen Automobilhersteller Audi, BMW, DaimlerChrysler, Porsche und Volkswagen, dass die Steuergeräte-Entwicklung und die Entwicklung von Entwicklungswerkzeugen in einem SPICE konformen Entwicklungsprozess erfolgen[6].

Anforderungen an den Code

Der Code für sicherheitskritische Anwendungen muss mehrere, ganz bestimmten Anforderungen erfüllen. Dazu gehören zum Beispiel:

- Begrenzung auf einen Teil der eingesetzten Programmiersprache, der für sicher gehalten wird
- Einschränkung der Verwendbarkeit von Kontroll- und Datenstrukturen auf genau spezifizierte Varianten
- Einhaltung genau definierter Sichtbarkeitsregeln für Funktionen und Daten
- Einhaltung vordefinierter Komplexitätsmaße
- Lesbarkeit, Wartbarkeit und Testfähigkeit des Codes

Solche Anforderungen an die Code-Qualität sind zum Beispiel in der MISRA C Richtlinie festgelegt [7]. MISRA C ist eine allgemein zugängliche Norm für die Verwendung von C-Code in Steuergeräte-Projekten bis zu SIL 3, definiert von der britischen „Motor Industry Software Reliability Association". Diese Norm wird unter dem Titel "Guidelines for the use of the C language in vehicle based software" geführt, ist aber allgemein unter dem Schlagwort "MISRA C" bekannt.

Darüber hinaus darf der generierte Code nicht wesentlich mehr Speicher und Ausführungszeit benötigen als manuell programmierter Code. Mangelhafte Effizienz war lange Zeit der Grund, warum automatische Code-Generierung nicht für die Serienproduktion eingesetzt wurde.

Prozessintegration

Damit die Vorteile automatischer Code-Generierung voll ausgeschöpft werden können, muss der Code-Generator optimal in den Entwicklungsprozess und in die Werkzeugkette eingebunden sein. Eine enge Integration und ein nahtloses Zusammenspiel mit dem Modellierungswerkzeug sind selbstverständlich. Des Weiteren sind offene Schnittstellen und die Möglichkeit zur Automatisierung wichtig für das Zusammenspiel

mit anderen Werkzeugen, das Automatisieren von Arbeitsschritten und das Vorbereiten und Unterstützen späterer Entwicklungsphasen, wie zum Beispiel der angrenzenden Unit-Testphase.

TargetLink als Code-Generierungswerkzeug für sicherheitskritische Software

Die in der Automobilindustrie am meisten verbreiteten Entwurfs- und Simulationswerkzeuge sind MATLAB®, Simulink® und Stateflow®. Datenflussmodelle werden in Simulink beschrieben, wohingegen Stateflow für den Kontrollfluss eingesetzt wird. Simulink und Stateflow sind integriert und lassen eine Mischung beider Typen in einem Modell zu. TargetLink, der Seriencode-Generator von dSPACE, integriert sich nahtlos in MATLAB und ermöglicht eine zuverlässige Umsetzung der in Form von Simulink/Stateflow-Modellen vorliegenden Software-Entwürfe in C-Code.

Test- und Verifikationsmöglichkeiten

Ein wesentlicher Vorteil der modellbasierten Entwicklung ist die Möglichkeit der frühen Verifikation mittels Simulation. TargetLink unterstützt unterschiedliche Simulationsmodi, durch die die Richtigkeit der Implementierung und damit des von TargetLink generierten Codes getestet werden kann. Dies geschieht durch das Vergleichen der Simulationsergebnisse mit den Resultaten aus Referenzsimulationen, welche auch als Model-in-the-Loop-Simulation bezeichnet werden. Diese Verifikation kann schrittweise durchgeführt werden:

- Zuerst wird der generierte Code mit einem Host-Compiler kompiliert und auf dem Host-PC ausgeführt, dies wird auch als Software-in-the-Loop-Simulation bezeichnet.
- Dann wird der Code auf einem Evaluierungsboard, welches den auch im Steuergerät verwendeten Zielprozessor enthält, simuliert. Dafür wird der generierte Code mit dem im Projekt verwendeten Target-Compiler kompiliert. Dieser Simulationsmodus heisst Processor-in-the-Loop-Simulation.

In allen unterstützten Simulationsmodi können die Signalverläufe mit TargetLink aufgezeichnet werden. Diese Signalverläufe können gespeichert und übereinander gelegt werden. Sie geben direktes, visuelles Feedback und ermöglichen weitergehende Analysen.

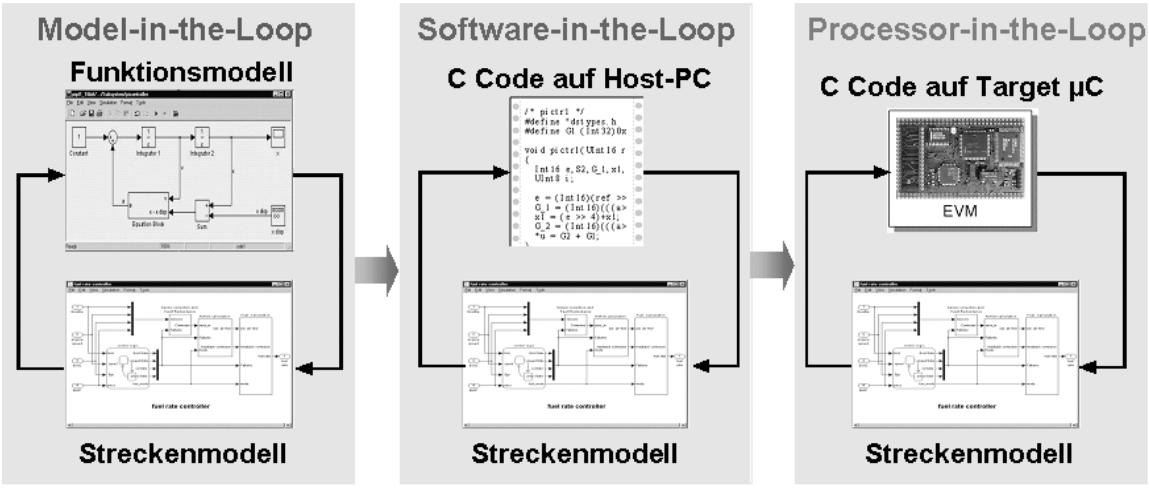

Abbildung 3: Model-, Software- und Processor-in-the-Loop-Simulation unterstützen den Steuergeräte-Test.

Der Anwender hat volle Kontrolle über die Aufteilung des generierten Codes in Funktionen und Dateien. Somit kann Code für bestimmte Modellteile in separate C-Funktionen und Dateien generiert werden. Diese Modellteile können zusätzlich inkrementell implementiert und getestet werden. Das hat den Vorteil, dass Code für Modellteile und Software-Module, die sich nicht geändert haben, nicht neu generiert und getestet werden muss.

Software- und Processor-in-the-Loop-Simulationen werden durch die Code-Abdeckungsanalyse ergänzt. Zurzeit werden 'Statement Coverage' und 'Decision Coverage' unterstützt. Insgesamt wird somit der Unit-Test umfassend unterstützt.

Code-Merkmale und -Eigenschaften

TargetLink ist speziell für die Generierung von Seriencode ausgelegt und entwickelt worden und kann es ohne weiteres mit der Leistungsfähigkeit von Handprogrammierern in Bezug auf Speicherbedarf und Ausführungsgeschwindigkeit aufnehmen. Zahlreiche Kunden-Benchmarks und Anwenderberichte zeigen, dass die Ergebnisse von TargetLink und die Ergebnisse eines Handprogrammierers nicht allzu weit auseinander liegen.

Der von TargetLink generierte Code kann auch Code-Reviews unterzogen werden. TargetLink-Code ist gut lesbar und ausführlich kommentiert. So wird man beispielsweise über die Kommentare vom Code zurück an die entsprechende Stelle ins Modell geführt. Unnötige Makros, Funktionsaufrufe, kryptische Benennungen und Ähnliches werden vermieden. Durch die umfassenden Konfigurationsoptionen kann der Anwender Variablen-, Funktions- und Dateinamen frei vorgeben. Zudem ist er bei der Code-Partitionierung in Funktionen und Dateien flexibel, um die Struktur logisch und übersichtlich gestalten zu können.

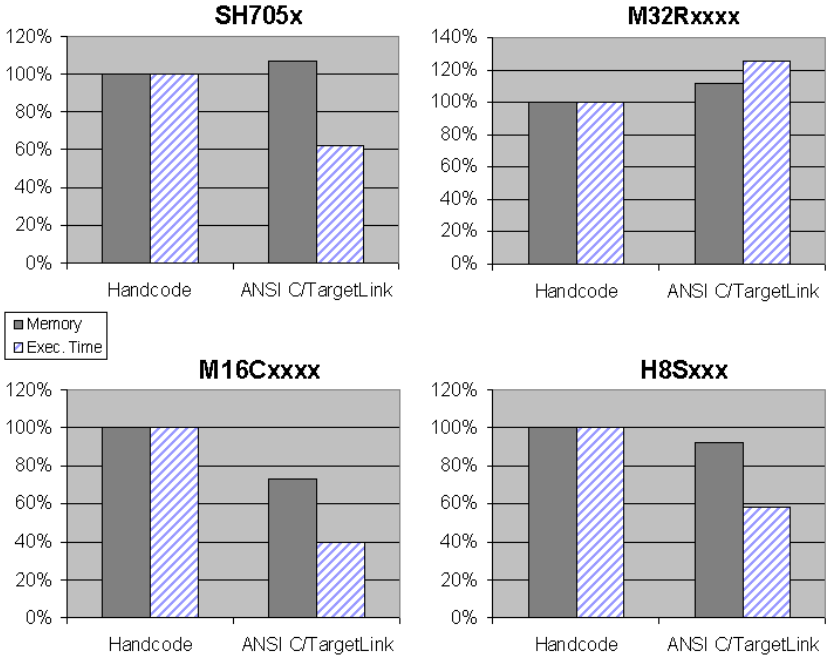

Abbildung 4: Kunden-Benchmark zum Vergleich von TargetLink-Code und handgeschriebenem Code auf unterschiedlichen Prozessoren.

Mit TargetLink generierter Code hält einen Großteil der 127 MISRA-Regeln ein. MISRA erlaubt ausdrücklich Abweichungen von der Norm, solange diese technisch begründet und dokumentiert sind. dSPACE stellt ein solches Dokument bereit, in dem die Abweichungen zu MISRA dargestellt sind [8].

Prozessautomatisierung

TargetLink wird mit einer umfassenden und komplett dokumentierten API ausgeliefert. Dadurch ist der Zugriff auf alle Eigenschaften und Einstellungen in TargetLink sowie die Automatisierung aller Prozesse möglich. Mittels „Hook-Funktionen“ besteht die Möglichkeit in allen Prozessphasen Anwenderaufgaben auszuführen. Beispielsweise direkt vor oder nach der Code-Generierung, vor oder nach der Kompilierung für den Host und für den Zielprozessor, vor oder nach dem Herunterladen auf das Evaluierungsboard für die PIL-Simulation sowie vor oder nach der Generierung von ASAP2- oder Applikationsdateien.

Maßnahmen zur Qualitätssicherung

Besondere Aufmerksamkeit wird der Qualitätssicherung in der TargetLink Produktentwicklung zu teil. Um ein Maximum an Zuverlässigkeit und Qualität zu gewährleisten, wird vor jeder Freigabe einer TargetLink-Version ein mehrstufiger Testprozess durchlaufen.

- Automatisches Durchlaufen einer Test-Engine: Mehrere hunderttausend Code-Muster werden getestet und einige Millionen Testpunkte durchlaufen. Diese Tests werden für jeden unterstützten Prozessor gesondert durchgeführt.
- Automatisches Durchlaufen einer Test-Suite: Dieser Test wird auf weit mehr als tausend Modellen durchgeführt. Durch Anwendung von unterschiedlichen Eingangswerten und durch Variation von Parametern ergeben sich mehr als hunderttausend Testfälle, deren Ergebnisse mit Erwartungswerten verglichen werden.
- Halbautomatisierter Systemtest: Dient zur Überprüfung der Installation und der Funktionsweise bei verschiedenen PC-Konfigurationen sowie dem Zusammenspiel mit verschiedenen MATLAB- und Compiler-Versionen.
- Manueller Test an Kundenmodellen: Tests werden anhand von großen, realen Steuergeräte-Modellen durchgeführt und manuell ausgewertet.

Zusätzlich findet vor der offiziellen Produktfreigabe eine Beta-Testphase mit ausgewählten Kunden statt. Somit hat TargetLink, wenn es an Kunden ausgeliefert wird, bereits eine gewisse Betriebsbewährtheit erlangt.

Darüber hinaus wird besonders auf einen reifen Entwicklungsprozess Wert gelegt. Die Entwicklung von TargetLink erfolgt, wie von der HIS gefordert, nach ISO/IEC 15504 (SPICE). Die Konformität wird durch einen unabhängigen Auditor überprüft. Der dabei angewandte Audit-Umfang wird ebenfalls von der HIS vorgegeben. Zusätzlich werden der Entwicklungsprozess und die Freigabe-Tests von einem internen, unabhängig arbeitendem dSPACE Software-Qualitätsmanagement beaufsichtigt.

Anwendungsbeispiel

Seit seiner Markteinführung 1999 wird TargetLink in zahlreichen Steuergeräteprojekten weltweit eingesetzt. TargetLink-Code ist in verschiedenen Anwendungen in Serie gegangen, von denen einige auch sicherheitsbezogen sind. Zum Beispiel kam TargetLink erfolgreich bei der Entwicklung einer Kabineninnendruckregelung zum Einsatz. Von TargetLink generierter Code wurde nach RTCA DO-178B Software Level A, dem höchsten Sicherheitslevel, welcher besagt, dass ein Funktionsausfall katastrophale Folgen für das Flugzeug hat, zertifiziert .

Nachfolgend wird der Einsatz von TargetLink bei ATENA Engineering für die Entwicklung sicherheitskritischer Software gemäß IEC 61508 SIL 3 beschrieben.

Erfahrungen in der Entwicklung sicherheitskritischer Systeme bei ATENA

Durch die enge Zusammenarbeit mit ihrer Muttergesellschaft MTU Aero Engines GmbH kann ATENA auf jahrzehntelange Erfahrung in der Entwicklung sicherheitskritischer Systeme aus dem Luftfahrtsektor zurückgreifen. Zum Beispiel wurden und werden von der MTU Aero Engines GmbH die Triebwerksregler für zahlreiche europäische Luftfahrtprojekte entwickelt und gefertigt. Bei diesen Reglern handelt es sich um mehrkanalige Steuergeräte mit vier bis zehn Prozessoren. Die für die Regelung erforderlichen Antwortzeiten liegen im Bereich von 2 ms. Die Entwicklung dieser Systeme erfolgt seit Jahren nach der RTCA DO-178A [10] beziehungsweise entsprechenden Vorläufern und projektspezifischen Ableitungen dieses Standards. ATENA selbst bietet ihre Engineering-Dienstleistungen nicht nur für die Luftfahrtindustrie sondern auch für Fahrzeughersteller und ihre Zulieferer an. Dadurch ist ATENA bereits heute in der Position, Software-Entwicklungsstandards für sicherheitskritische Systeme auch in Fahrzeugen umfassend anzuwenden.

Erfahrungen mit Automatischer Code-Generierung

MTU Aero Engines verfügt genauso wie ATENA über umfassende Erfahrungen auf dem Gebiet der automatischen Code-Generierung. Beide Firmen haben in der Vergangenheit unterschiedliche Code-Generatoren im Detail evaluiert. Bei diesen Evaluierungen stand, unabhängig davon, ob der Code-Generator in der Luftfahrt oder in automotiven Projekten eingesetzt werden sollte, die Anwendbarkeit auf die Entwicklung sicherheitskritischer Systeme nach den Normen RTCA DO-178B Level A bzw. IEC 61508 SIL 3 im Vordergrund. Zentrale Punkte bei den Evaluierungen waren:

- Integration in den Gesamtentwicklungsprozess
- Qualität und Effizienz des generierten Codes
- Flexible und umfassende Konfigurationsoptionen des Code-Generators

TargetLink im Einsatz bei ATENA

Aufgrund der zuvor beschriebenen Nachteile hat sich ATENA entschieden, auch für sicherheitskritische Systeme keine zertifizierten Code-Generatoren einzusetzen. Stattdessen erfolgt eine vollständige Verifikation der Applikationssoftware, wobei der Verifikationsprozess bestmöglich automatisiert wird. Dennoch wird natürlich besonderer Wert auf die Qualität und Zuverlässigkeit des eingesetzten Code-Generators gelegt. Qualitätsmerkmale, die nicht die beschriebenen Nachteile mit sich bringen, beispielsweise ein ISO/IEC 15504 (SPICE)-konformer Entwicklungsprozess, werden als wichtig erachtet.

Bei ATENA war für den modellbasierten Entwurf die Verwendung von MATLAB, Simulink, Stateflow von The MathWorks bereits etabliert. Aus dem Grund wurde nach einem Code-Generator gesucht, der sich nahtlose in diese Modellierungswerkzeuge integriert. Eine eingehende Evaluierung der verfügbaren Code-Generatoren führte dazu, sich für die Software-Implementierung eines sicherheitskritischen Systemes nach der IEC 61508 SIL 3 für TargetLink zu entscheiden. Bei dem betreffenden System handelt es sich um eine automotive Anwendung für alternative Kraftstoffkonzepte. Das entwickelte Simulink-Modell besteht aus ca. 120 000 Teilsystemen. Die Software-Komponenten bestehen aus bis zu 25 000 Programmzeilen. Die Hauptgründe für die Auswahl von TargetLink waren die technischen Leistungsmerkmale sowie die hohe Produktqualität.

Für die Einbindung von TargetLink in den Software-Entwicklungsprozess wurden verschiedene Anpassungen des Code-Generierungsprozesses von ATENA vorgenommen. Diese dienten der weiteren Automatisierung des Generierungsprozesses und dem Erreichen der geforderten Software-Qualität für sicherheitskritische Anwendungen. Dazu gehört die stark eingeschränkte Verwendung von Pointern und Interrupts, die Einhaltung verschiedener Komplexitätskriterien sowie der Ersatz von funktionsähnlichen Makros und Funktionen der Standardbibliotheken durch eigene Funktionen. Darüber hinaus wurden Maßnahmen getroffen, derart komplexe Systeme in Teilsysteme zerlegen und separat übersetzen zu können sowie für diese Teilsysteme einheitliche Beschreibungsdateien für Applikationswerkzeuge zu erzeugen. Zudem wurde zur Vorbereitung des Unit-Tests auch die Testrahmen-Software für den Test einzelner Module automatisch generiert. Unterstützt wurden diese Anpassungen durch die TargetLink-API. Diese erlaubt, sämtliche Prozesse zu automatisieren und bietet gleichzeitig Eingriffsmöglichkeiten in den einzelnen Prozessphasen sowie Zugriff auf alle Eigenschaften und Einstellungen von TargetLink.

Der Software-Entwicklungsprozess, dessen Implementierungsphase maßgeblich von dSPACE TargetLink unterstützt wird, wird nun seit November 2002 bei ATENA angewandt. Der automatischen Code-Generierung kommt dabei ein hoher Stellenwert zu, da es gelungen ist, ca. 80% des gesamten Seriencodes aus Simulink-Modellen zu erzeugen. Der Code-Generator ist dabei in eine projektspezifische Werkzeugkette eingebunden. Diese garantiert die Einhaltung der Qualitätskriterien für sicherheitskritische Anwendungen.

Das finale Produkt der ersten Entwicklungsphase wurde mittlerweile für Integrationstests an den OEM ausgeliefert. Während dieser ersten Entwicklungsphase konnten folgende Erfahrungen gesammelt werden:

- Die modellbasierte Anforderungsanalyse und Entwurfsphase führten zu einer ausführbaren Spezifikation, mit der das System frühzeitig verifiziert werden konnte. Daraus resultierte eine ausgereifte Spezifikation. Spätere Software-Tests ergaben nur noch wenige Fehler, welche ihren Ursprung in der Spezifikation hatten.
- Die automatische Code-Generierung erlaubte die Implementierung des Software-Entwurfs mit sehr geringer Fehlerrate und hoher Effektivität besonders bei kleineren Änderungen.
- Die Qualität des aus dem Software-Entwicklungsprozess bei ATENA mit TargetLink als Schlüsselkomponente generierten Codes genügt den Anforderungen für sicherheitskritische Anwendungen.

Basierend auf den bei ATENA gesammelten Projekterfahrungen und den bei der MTU Aero Engines erzielten Evaluierungsergebnissen ist der Einsatz von TargetLink auch für zukünftige Luftfahrtprojekte bei der MTU Aero Engines geplant.

Zusammenfassung

Die Automatische Code-Generierung bietet zusammen mit dem modellbasierten Entwurf viele Vorteile für den Anwender und kann für die Entwicklung sicherheitskritischer Systeme eingesetzt werden. Eingebettet in einen entsprechenden Entwicklungsprozess ist sie geeignet für die Erstellung von Software, die nach

Sicherheitsnormen wie der IEC 61508 SIL 3 eingestuft und bewertet wird. Es gibt mehrere Anforderungen an den Code-Generator bezüglich Qualität und technischer Leistungsmerkmale, die erfüllt werden müssen.

TargetLink, der Seriencode-Generator von dSPACE, erfüllt diese Anforderungen und ist deshalb auch ein geeigneter Code-Generator für die Entwicklung sicherheitskritischer Systeme.

Bei ATENA Engineering fiel die Wahl aufgrund der technischen Leistungsmerkmale und der hohen Produktqualität auf TargetLink. Gleichzeitig hat sich ATENA dazu entschieden, einen Code-Generator mit weniger Qualifizierungsoptionen einzusetzen und dafür die Applikationssoftware vollständig zu verifizieren. Diese Alternative wird als kosteneffektiver angesehen, als einen zertifizierten Code-Generator einzusetzen.

ATENA und dSPACE stehen in engem Kontakt, um bei Folgeversionen von TargetLink die Integration in die Werkzeugkette weiter auszubauen und sicherheitskritische Aspekte in der Code-Generierung noch stärker zu berücksichtigen.

Referenzen

1. IEC 61508 Functional Safety of Electrical/Electronic/Programmable Electronic Safety Related Systems, IEC 1998
2. RTCA/DO-178B Software Considerations in Airborne Systems and Equipment Certification, RTCA Inc., 1. Dez. 1992
3. Bauer,C.; Plawecki,D.: IEC 61508, Part 3 vs. RTCA/DO-178B A Comparative Study. Konferenz Anwendung des internationalen Standards IEC 61508 in der Praxis, Januar 2003
4. Bechberger: Modellbasierte Softwareentwicklung für Steuergeräte, Automotive Electronics 2/2000
5. ISO/IEC TR 15504:1998, Information Technology - Software Process Assessment
6. Wagner, Merkle, Bortolazzi, Marx, Lange: Hersteller Initiative Software, Automotive Electronics 1/2003
7. MISRA Guidelines for the Use of the C Language in Vehicle Based Systems, April 1998
8. Thomsen, T.: Integration of International Standards for Production Code Generation, SAE Technical Paper 03AE-32, 2003
9. Aloui, Andreas: C Code Reaches New Heights at Nord-Micro, dSPACE Newsletter, 1/2002
10. RTCA/DO-178A Software Considerations in Airborne Systems and Equipment Certification, RTCA Inc., 22. März 1985

Autoren

Michael Beine
dSPACE GmbH – Produktmanager TargetLink
Technologiepark 25
33100 Paderborn
mailto:mbeine@dspace.de

Michael Jungmann
MTU Aero Engines GmbH
Engine Systems Design/Software Development
Dachauer Straße 665
80995 München
mailto:Michael.Jungmann@muc.mtu.de