# Automatische Code-Generierung für sicherheitskritische Systeme

## Chancen und Nutzen durch geeignete Werkzeuge und Prozesse

Zukünftige Fahrzeuge werden in immer größerem Maß sicherheitskritische Systeme enthalten. Aufgrund hoher Änderungsfrequenzen finden bei der Software-Entwicklung vermehrt automatische Code-Generatoren Anwendung. Der heute für die Fahrzeugentwicklung verbindliche Sicherheitsstandard IEC61508 ist an der manuellen Software-Entwicklung ausgerichtet und bietet nur wenig Entscheidungshilfe bei Auswahl und Einsatz von Code-Generatoren für die Software sicherheitskritischer Systeme. Der Ansatz von ATENA Engineering basiert auf Erfahrungen und Standards aus der Luftfahrt und verwendet dSPACE TargetLink zur automatischen Code-Generierung für sicherheitskritische Systeme.

**Autoren:**

*Dipl.-Inform. Michael Jungmann ist Software-Entwickler bei der MTU Aero Engines GmbH in München und zuständig für Software-Entwicklungsprozesse und –werkzeuge für die Projekte der Tochtergesellschaft ATENA Engineering GmbH.*

*Dipl.-Math. Michael Beine ist Produktmanager von TargetLink bei der dSPACE GmbH in Paderborn.*

## 1 Sicherheitskritische Systeme und deren Software

Die Anzahl von sicherheitskritischen Systemen 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 Insassen des Fahrzeugs und andere Verkehrsteilnehmer darstellen.
Um die von solchen Systemen ausgehenden Gefahren zu minimieren, werden spezielle Entwicklungsstandards und –prozesse für sicherheitskritische Applikationen definiert und angewendet. Für die Automobilelektronik wurde dafür die Norm IEC61508 [1] als Richtlinie etabliert. Diese Sicherheitsnorm ist ein sehr allgemein gehaltener Standard, der branchen- oder projektspezifischer Detaillierungen bedarf. Im Bereich der Software-Entwicklung haben Untersuchungen gezeigt, dass der ursprünglich für die Luftfahrtbranche definierte Software-Entwicklungsstandard RTCA DO178B [2] eine auch für den Automobilbereich geeignete Konkretisierung der Sicherheitsnorm IEC61508 darstellt [3 u.a.].

Die Unit- bzw. Modul-Testphase 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 Tests der Funktionalität. Die in der jeweiligen Phase durchzuführenden Aktivitäten unterscheiden sich in den einzelnen Standards geringfügig. Allen Standards gemeinsam ist jedoch, dass nur die konsequente Durchführung aller Aktivitäten die vom Standard angestrebte Sicherheit ermöglicht. Kein einzelner Schritt dieses Prozesses für sich genommen erlaubt eine Aussage über die Qualität der entwickelten Software.

## 2 Automatische Code-Generierung im Entwicklungsprozess

Heute kommen bei der Software-Entwicklung in der Automobilindustrie vermehrt automatische Code-Generatoren zum Einsatz. Die Anwendung für sicherheitskritische Systeme ist bisher allerdings kaum verbreitet. Erstens werden ganz spezielle Anforderungen an den Code für sicherheitskritische Systeme gestellt. Und zweitens stehen viele Software-Lieferanten heute noch ganz am Anfang der Anwendung solcher Entwicklungsstandards und können die Besonderheiten der automatischen Code-Generierung nicht gleichzeitig berücksichtigen.

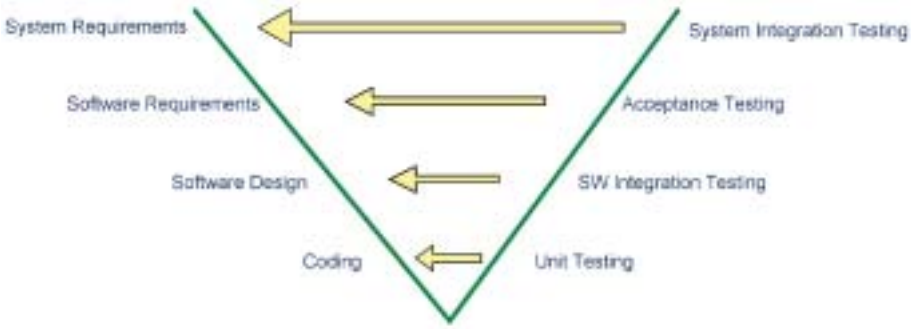


**Bild 1: V-Modell für Software-Entwicklung**

Der Software-Entwicklungsprozess nach RTCA DO-178B orientiert sich am gängigen V-Modell (siehe Bild 1).
Das V-Modell beschreibt auf der linken Seite den Implementierungspfad, der von Anforderungen auf hoher Ebene ausgehend, die Detaillierungsschritte bis zur Entstehung von Seriencode beschreibt und auf der rechten Seite den Verifikationspfad, der jeder Implementierungsphase eine horizontal gegenüberliegende Verifikationsphase zuordnet. Für die Betrachtung automatischer Code-Generierung sind in erster Linie die Phasen Coding und Unit-Testing von Interesse.

Impulse für den Einsatz automatischer Code-Generatoren auch für sicherheitskritische Systeme werden zuerst von Software-Herstellern ausgehen, die aufgrund ihres Engagements auch in anderen Branchen detaillierte Erfahrungen in der Anwendung der geeigneten Software-Entwicklungsprozesse haben.
Die Firma ATENA Engineering hat durch die enge Zusammenarbeit mit ihrer Muttergesellschaft MTU Aero Engines GmbH

**Bild 2: Von MTU Aero Engines entwickelter Triebwerksregler für ein Strahltriebwerk mit Nachbrenner**

die Möglichkeit, auf jahrzehntelange Erfahrung in der Entwicklung sicherheitskritischer Systeme aus dem Luftfahrtsektor zurückzugreifen. Beispielsweise wurden und werden von der MTU Aero Engines GmbH die Triebwerksregler für eine Zahl europäischer Luftfahrtprojekte entwickelt und gefertigt. Bei diesen Reglern handelt es sich um mehrkanalige Steuergeräte mit 4 bis 10 Prozessoren, Bild 2. Die für die Regelung erforderlichen Antwortzeiten liegen im Bereich von 2 msec. Die Entwicklung dieser Systeme erfolgt seit Jahren nach der RTCA DO178 [4] bzw. Vorläufern und projektspezifischen Ableitungen dieses Standards. ATENA selbst bietet ihre Engineering Dienstleistungen sowohl für die Luftfahrt als auch für Automobilhersteller und deren Zulieferer an. Durch diese Konstellation ist ATENA in der Position, bereits heute Software Entwicklungsstandards für sicherheitskritische Systeme in Automobilen umfassend anzuwenden.

Derzeit sind bei ATENA Fahrzeugsysteme in der Entwicklung, die entsprechend IEC61508 SIL3 eingestuft sind und deren Software-Anteil Umfänge bis zu 20 000 Programmzeilen haben.

Bei der Entwicklung solcher Systeme sprechen mehrere Gründe für den Einsatz eines automatischen Code-Generators. Einerseits sind hohe Änderungsraten für die aktuellen Projekte zu erwarten und andererseits liegt das Software-Design bereits als ausführbare Spezifikation vor, die ein automatischer Code-Generator mit einer geringeren Fehlerrate umsetzten kann.

## 3 Kosten-/Nutzenbetrachtung der Tool-Zertifizierung und Alternativen

Beim Einsatz eines Code-Generators ist die Frage zu klären, wie mit der Forderung der IEC61508 nach zertifizierten bzw. betriebsbewährten Tools umzugehen ist. Das Argument der Betriebsbewährtheit ist heute für SIL3-Entwicklungsprojekte nicht uneingeschränkt verwendbar, da bisher keine Referenzprojekte ähnlicher Komplexität vorliegen.

Die Alternative der Zertifizierung verlangt ein Vorgehen entsprechend eines nationalen oder internationalen Standards, ohne diesen näher zu bezeichnen. Ein dafür anwendbarer Standard ist der oben genannte RTCA DO-178B. Dieser lässt dem Anwender zwei Alternativen.

Einerseits ist es möglich, die Applikationssoftware allen vorgeschriebenen Verifikationsaktivitäten zu unterziehen, so als wäre sie ohne Zuhilfenahme eines automatischen Code-Generators entstanden. Dabei wird ein Verifikationsprozess angewandt, der auch Fehler finden kann, die durch einen automatischen Code-Generator entstanden sein könnten.

Andererseits könnte der Code-Generator selbst entsprechend der RTCA DO-178B qualifiziert werden, wofür im Gegenzug Reduzierungen bei den Verifikationsaktivitäten in Frage kommen. Diese Reduzierungen betreffen aber nur einzelne Aktivitäten der Unit- bzw. Modul-Testphase und sind projektspezifisch zu bestimmen, Bild 3. Eine generelle Auslassung dieser Verifikationsphase ist in der Regel nicht möglich. Die Qualifikation des Code-Generators setzt

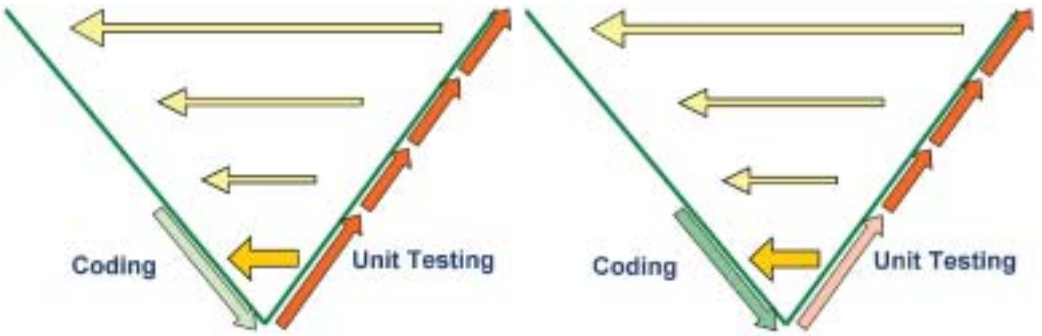


**Bild 3: Einsparungspotenzial bei der Verifikation bei Nutzung eines qualifizierten Code-Generators (rechts)**

das Vorhandensein eines qualifizierbaren Code-Generators voraus, d.h. die Entwicklungsdokumentation für den Code-Generator muss vom Tool-Hersteller entsprechend RTCA DO178B erstellt und ausgeliefert worden sein. 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.

Die Qualifikation des Code-Generators muss nach der gleichen Sicherheitseinstufung erfolgen, die auch für die Applikationssoftware gilt. Dadurch entsteht ein enormer Aufwand, der sich in den Kosten für den qualifizierbaren Code-Generator und der projektspezifischen Qualifikation niederschlägt. Der Preis eines qualifizierbaren Code-Generators liegt beim fünf- bis zehnfachen gegenüber dem nicht qualifizierbaren. Die projektspezifische Qualifikation verursacht Kosten von 50 000 bis 100 000 . Die Qualifikation des Code-Generators ist darüber hinaus sehr zeitaufwendig, so dass der Wartungszyklus für einen solchen Generator heute bei ca. zwei Jahren liegt, während für den nicht qualifizierten meist ca. 3 Monate ausreichen, um eine korrigierte Version zu erhalten.

Vor diesem Hintergrund hat sich ATENA entschieden, auch für sicherheitskritische Systeme keine qualifizierten 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 oben beschriebenen Nachteile mit sich bringen, werden als wichtig erachtet, wie zum Beispiel ein ISO/IEC15504 (SPiCE) [5] -konformer Entwicklungsprozess.

## 4 TargetLink als Codegenerierungswerkzeug für die Software-Entwicklung in sicherheitskritischen Anwendungen

TargetLink, der Code-Generator von dSPACE, integriert sich nahtlos in MATLAB und ermöglicht eine sichere Umsetzung des in Form von Simulink/Stateflow-Modellen vorliegenden Software-Designs in C-Code.

Die Korrektheit dieser Umsetzung lässt sich direkt mittels Simulation und durch den Vergleich mit Ergebnissen aus Referenz-Simulationen überprüfen. Diese Verifikation wird häufig als Model-in-the-Loop-Simulation (MIL) bezeichnet. Sie kann schrittweise erfolgen:

Zunächst durch eine Offline-Simulation auf dem Host-PC, auch Software-in-the-Loop-Simulation (SIL) genannt.

Dann durch eine Simulation in Echtzeit auf dem Zielprozessor unter Einbindung des tatsächlich verwendeten Compilers, auch Prozessor-in-the-Loop-Simulation (PIL) genannt, Bild 4.

Die Funktionshierarchie und Aufteilung auf Dateien können vom Anwender vorgegeben werden. Somit kann der Code für bestimmte Modellteile in separate C-Funktionen und -Dateien generiert werden. Diese Modellteile lassen sich auch inkrementell implementieren und testen, Bild 5. Das hat den Vorteil, dass bei Änderungen der Code für ungeänderte Modellteile und Software-Module nicht neu generiert und getestet werden muss.

Des Weiteren werden SIL- und PIL-Simulation durch eine Code-Abdeckungsana-

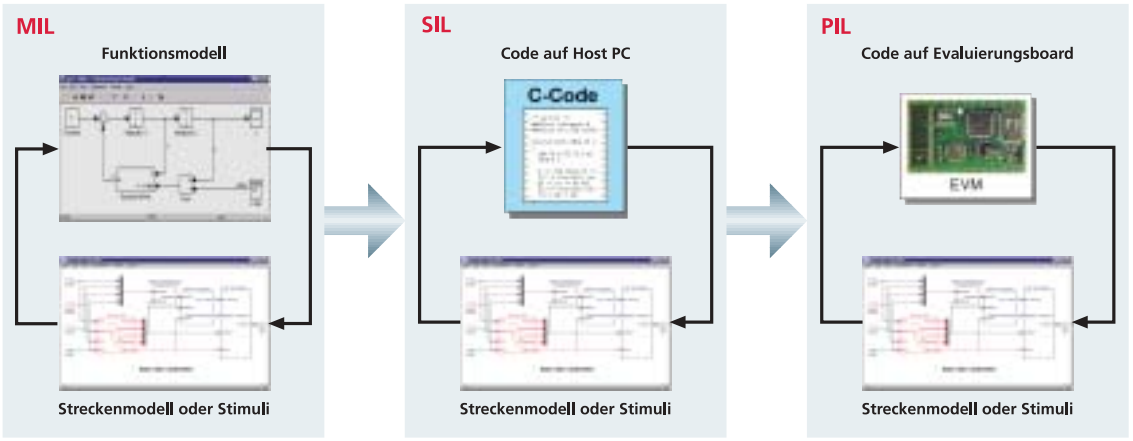


**Bild 4: MIL, SIL, PIL unterstützen die Unit-Testphase**

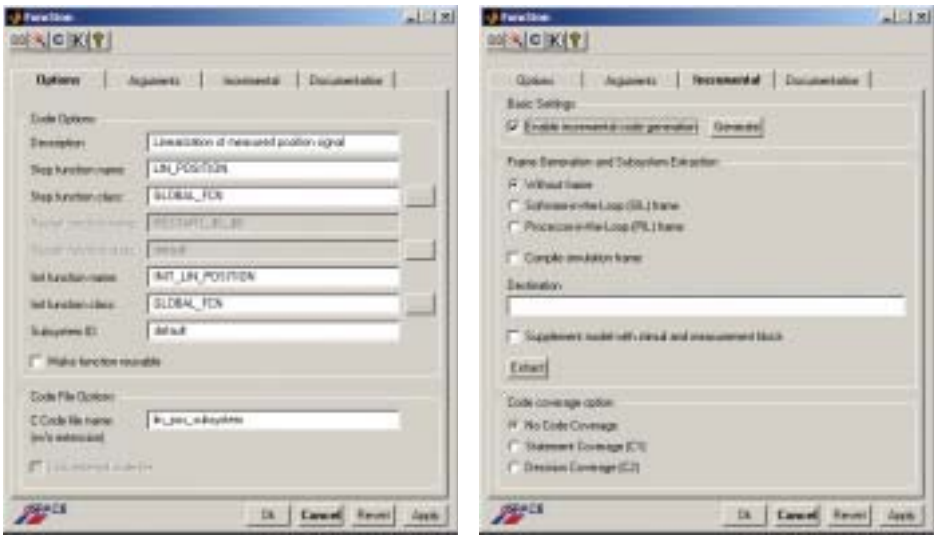

**Bild 5: Der TargetLink Function Block Dialog zur Auswahl von Funktionsnamen, Dateizugehörigkeit, Inkrementeller Code-Generierung und anzuwendendem Code-Abdeckungsmaß**

lyse ergänzt. Die zur Zeit unterstützten Abdeckungsmaße sind „Statement und Decision Coverage". Insgesamt wird so die Unit- bzw. Modul-Testphase, auch für inkrementell generierte Modellteile ausführlich unterstützt. Insgesamt bietet TargetLink umfassende Konfigurationsmöglichkeiten an, um den Anforderungen an Code für sicherheitskritische Systeme gerecht zu werden.

Besondere Aufmerksamkeit wird der Qualitätssicherung in der TargetLink-Produktentwicklung zu teil.

Für ein Maximum an Zuverlässigkeit und Qualität werden vor jeder Freigabe einer TargetLink-Version umfangreiche Tests durchgeführt:

Automatisches Durchlaufen einer Test Engine: Dabei werden mehrere hunderttausend Code-Muster getestet und mehrere Millionen Testpunkte durchlaufen. Diese Tests werden für jeden gesondert unterstützten Prozessor wiederholt.

Automatisches Durchlaufen einer Test Suite: Diese besteht aus weit über tausend Modellen. Durch Anwendung von unterschiedlichen Eingangswerten und durch Variation von Parametern ergeben sich mehr als hunderttausend Testfälle, deren Ergebnisse mit Erwartungswerten verglichen werden.

Halbautomatischer Systemtest:. Er dient zur Überprüfung der Installation und der Funktionsweise bei verschiedenen PC-Konfigurationen und dem Zusammenspiel mit verschiedenen MATLAB- und Compiler-Versionen.

Manueller Test an Kundenmodellen: Tests werden anhand von großen, realen Steuergeräte-Modellen durchgeführt und manuell ausgewertet.

Weiterhin wird vor der offiziellen Produktfreigabe eine Beta-Test-Phase mit ausgewählten Kunden durchgeführt. Wenn TargetLink dann an die Kunden ausgeliefert wird, hat es bereits eine gewisse Betriebsbewährtheit erlangt.

Darüber hinaus wird bei der Entwicklung von TargetLink besonders auf einen reifen Entwicklungsprozess Wert gelegt. Die Entwicklung von TargetLink erfolgt, wie von der Herstellerinitiative Software (HIS) gefordert, nach ISO/IEC15504 (SPiCE) [6]. Die Konformität wird durch einen unabhängigen Auditor überprüft. Der dabei angewandte Audit-Umfang wird ebenfalls von der HIS vorgegeben. Zusätzlich werden der Entwicklungsprozess sowie die Freigabetests durch ein internes, unabhängig arbeitendes dSPACE Software-Qualitätsmanagement beaufsichtigt.

Nach einer eingehenden Evaluierung von verfügbaren Codegenerierungswerkzeugen fiel die Entscheidung, TargetLink in einem SIL 3 Projekt einzusetzen. Maßgeblich für diese Entscheidung waren die technischen Eigenschaften und nicht zuletzt die hohe Produktqualität. Für die Einbindung von TargetLink in den Software-Entwicklungsprozess wurden verschiedene Anpassungen des Code-Generierungsprozesses von ATENA selbst 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 Komplexitätsmaße sowie der Ersatz von „function-like“ Makros und Funktionen der Standardbibliotheken durch eigene Funktionen. Darüber hinaus waren Maßnahmen zu treffen, derart komplexe Systeme in Teilsysteme zerlegen und separat übersetzen zu können und für diese Teilsysteme einheitliche Beschreibungsdateien für Applikationstools zu erzeugen. 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 TargetLink-Eigenschaften und -Einstellungen.

## 5 Erfahrungen

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 eingebettet. Diese garantiert die Einhaltung der Qualitätskriterien für sicherheitskritische Anwendungen. 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.

**Literaturhinweise:**


*[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., 1th Dec 1992*

*[3] Bauer,C.; Plawecki,D.:*
*IEC61508, Part 3 vs. RTCA/DO-178B*
*A comparative Study. Konferenz Anwendung des internationalen Standards IEC61508 in der Praxis,*
*Januar 2003*

*[4] RTCA/DO-178A*
*Software Considerations in Airborne Systems and Equipment Certification,*
*RTCA Inc., 22th Mar 1985*

*[5] ISO/IEC TR 15504:1998,*
*Information Technology – Software Process Assessment*

*[6] Wagner, Merkle, Bortolazzi, Marx, Lange:*
*Hersteller Initiative Software,*
*Automotive Electronics 1/2003*