

# Systematisch zur sicheren Software

## Modellbasierte Software-Entwicklung für sicherheitskritische Systeme

In der Luftfahrt, in der Eisenbahntechnik und zunehmend auch im Automobilbereich wird die Entwicklung der Software von Anforderungen an die funktionale Sicherheit und Integrität bestimmt, die in verschiedenen Standards und Normen festgeschrieben sind. In der Luftfahrt ist das beispielsweise der Standard DO-178B, in den meisten anderen Anwendungsbereichen die internationale Norm IEC 61508. Die Zertifizierung nach dieser Norm ist nahezu untrennbar mit automatischer Codegenerierung verbunden.

Von Jakob Gärtner

Im Bereich der Embedded-Systeme hat sich für die Entwicklung der Applikationssoftware der modellbasierte Ansatz weitgehend durchgesetzt. Modellbasierende Entwicklung bedeutet, dass der Ingenieur bereits in einem frühen Stadium das Verhalten seines Designs betrachten kann, sofern er ein Werkzeug zur Verfügung hat, welches ihm ausführbare Modelle zur Verfügung stellt. Besitzt er zudem noch einen automatischen Codegenerator, dann kann er sein Modell auch in Quellcode überführen, den er im Idealfall

unverändert auf seine Zielplattform übertragen kann.

Unter der Voraussetzung, dass eine geeignete Modellierungstechnik gewählt wurde, profitiert man erheblich von automatischer Codegenerierung:

- Die Modellierung findet in einer Weise statt, die es dem Systemingenieur erlaubt, sein Design in geeigneter Weise darzustellen (Datenflussdiagramme, Zustandsautomaten).

- Die Codegenerierung erfüllt die Anforderungen des Software-Ingenieurs an Effizienz, deterministisches Verhalten

und sicheren Aufbau (z.B. keine dynamische Speicherallokation) des Codes.

- Neue Prozesse, die das Erreichen der definierten Sicherheitsanforderungen garantieren, können aufgesetzt werden.
- Die Codierungszeiten verringern sich dramatisch (von Wochen auf Sekunden).
- Der Aufwand für die Verifikation des Modells reduziert sich erheblich, da ein geeigneter Codegenerator die Übereinstimmung von Code und Modell garantiert.
- Wenn das Werkzeug geeignete Methoden zur Verifikation des Modells zur Verfügung stellt, können Probleme und Fehler sehr früh gefunden und behoben werden – lange, bevor die erste Zeile Code geschrieben wurde.
- Die Änderungszyklen werden entsprechend kürzer.

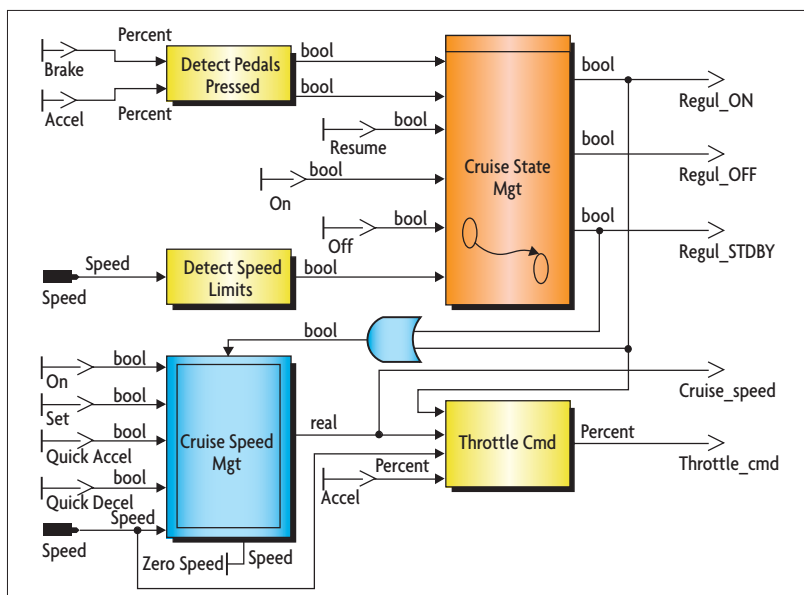
### ■ Von den Anforderungen zum Software-Spezifikationsmodell

Am Beispiel einer einfachen Anwendung (Cruise Control) werden nachfolgend die Schritte einer modellbasierten Entwicklung dargestellt. Zunächst gilt es, ein Design zu erstellen. Hierzu wird eine bewährte Notation wie z.B. das Blockdiagramm in *Bild 1* gewählt. Auf den ersten Blick sieht man Eingänge, Ausgänge, Blöcke und Linien. Dargestellt sind hier Datenflüsse. Die Eingänge werden zyklisch gelesen. Die Linien transportieren die Daten zu den Blöcken. In den Blöcken werden die Daten verarbeitet, über die Linien weitergeleitet, bis schließlich die Ergebnisse wieder auf die Ausgänge geschrieben werden.

Das Besondere an dieser Art der Modellierung ist, dass die Notation Vollständigkeit, Zeit- und Datenkonsistenz erzwingt. Hinter den grafischen Blöcken befindet sich eine formale Modellbeschreibungssprache. Auf diese Weise können gewisse Modelleigenschaften garantiert werden, das Verhalten kann auf Modellebene verifiziert werden und der Code kann zur Implementierung auf der Zielhardware generiert werden.

Der Entwickler kann aber weiter seinen Entwurf auf intuitive Weise auf-

**Bild 1.**  
Das erste Design einer Anwendung – hier eine Cruise Control – entsteht auf der Grundlage eines Blockdiagramms.



bauen. Durch die formale Notation ist das Werkzeug in der Lage, selbstständig auf die Einhaltung der Modellierungsregeln zu achten, die durch die gewünschten Eigenschaften des Designs impliziert werden. Betrachtet man den Block „Cruise Speed Mgt“ näher, so ist zu sehen, dass eine weitere Hierarchieebene enthalten ist, die das Design weiter verfeinert (Bild 2). Hier hat das Tool ein Problem aufgespürt: Es gibt eine unzulässige Rückkopplung im Modell. Die Fehlermeldung zeigt, dass die Modelle tatsächlich auf der logischen Ebene „denken“. Die einzelnen Blöcke werden nicht nach einer bestimmten Reihenfolge abgearbeitet – vielmehr kommt es alleine

auf die definierten, durch die Abhängigkeiten der Datenflüsse bestimmten logischen Zusammenhänge an. Das Modell ist nicht imperativ, sondern rein deskriptiv beschrieben.

Nun wird das Modell so verändert, dass der logische Bezug zwischen den Blöcken *Compute Target Speed* und *Compute Throttle* explizit definiert wird. Dies geschieht durch Einfügen eines *FBY-Operators* („followed by“). Das Modell liest sich nun an der Stelle so, ganz informell ausgedrückt: „Compute Throttle“ followed by „Compute Target Speed“. Damit ist der logische Zusammenhang eindeutig definiert, der Zirkelschluss aufgehoben. In diesem kleinen Modell wird auch sicht-

bar, wie sich funktionale Gleichzeitigkeit (zwischen *Set Regulation Mode* und *Compute Target Speed*) und funktionale Abfolgen darstellen lassen.

Die Darstellung in Datenflüssen wird ergänzt durch Zustandsdiagramme („Safe State Machines“). Für diese Automaten gilt analog zu den Blockdiagrammen, dass hier das Verhalten und die logischen Zusammenhänge beschrieben werden. Semantik und Verhalten dieser Modelle sind streng deterministisch; auch hier steckt eine formale Modellbeschreibungssprache dahinter. Dieser Zustandsautomat ist transparent in das Modell eingebunden

### ■ Funktionale Sicherheit und IEC 61508

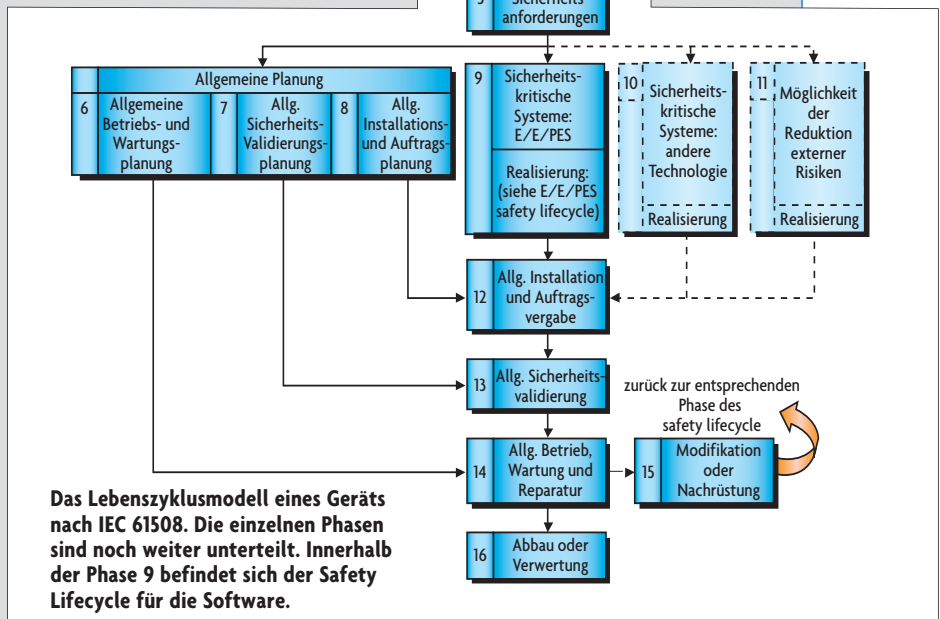
Der Standard IEC 61508 betrachtet den gesamten Lebenszyklus eines E/E/PES (Elektrisches/Elektronisches/Programmierbares Elektronisches System). Der Standard IEC 61508 verwendet spezifische Begriffe und Definitionen:

- **Equipment under Control (EUC):** Das System, welches der Zertifizierung unterzogen wird.
- **Sicherheit:** Freiheit von einem nicht akzeptablen Risiko für Gesundheit oder Leben, entweder direkt durch Versagen des Systems oder indirekt durch Schädigung von Gütern oder der Umwelt.
- **Funktionale Sicherheit:** Der Teil der Sicherheit, der von einer korrekten Funktion des Systems abhängt.
- **Safety Integrity Level (SIL):** wird mit Hilfe von Methoden der Hazard- und Risikoanalyse ermittelt. Man spricht von vier Safety Integrity Levels (SIL1 bis SIL4), wobei SIL4 die höchsten Ansprüche an Entwicklung und Verifikation des Systems stellt.

IEC 61508 definiert die Methoden und Ziele, deren Einhaltung die Voraussetzung zur Zertifizierung eines Systems ist:

- Die Norm verfolgt einen riskobasierten Ansatz, um die Anforderungen an funktionale Sicherheit und Integrität zu definieren.
- Sie legt für das betrachtete System ein globales Lebenszyklusmodell (Bild) zugrunde.

- Sie betrachtet das Gesamtsystem, bestehend aus Hard- und Software, und verschiedene Fehlerklassen (systematische, zufällige, common-cause, erkannte, nicht erkannte u.a.).

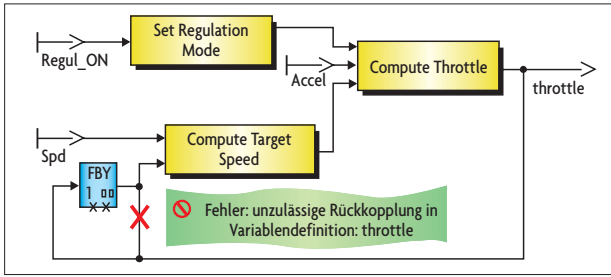


- IEC 61508 bestimmt sowohl Anforderungen zur Vermeidung als auch zur sicheren Beherrschung des Auftretens von Fehlern.

- Sie beschreibt die Maßnahmen und Techniken, die zur Sicherstellung des Erreichens eines bestimmten SIL verwendet werden müssen.

Für jede einzelne Phase innerhalb des Lebenszyklus sind Entwicklungs-

und Verifikationsaktivitäten vorgeschrieben, die mit Hilfe eines V-Modells dargestellt werden. Die einzelnen Verifikations-, Test- und Validierungsschritte sind detailliert in Tabellen aufgeschlüsselt und variieren je nach erforderlichem SIL. Typischerweise fällt mehr als die Hälfte des gesamten Software-Entwicklungsaufwands bei Verifikation und Test an, wenn die Anwendung nach IEC 61508 zertifiziert werden soll.



**Bild 2.** Die statische Modellanalyse zeigt zyklische Abhängigkeit. Erst nach Einfügen eines „Followed by“-Operators (FBY) ist das Modell fehlerfrei.

und verbirgt sich als Block *Cruise State Mgt* im Cruise-Control-Modell.

Die hier kurz beschriebene Modellierungsmethode besitzt wichtige Eigenschaften, die die Sicherheitsanforderungen der relevanten Standards erfüllen:

- Die Modellierungssprache ist ein Ergebnis von mehr als 20 Jahren wissenschaftlicher Forschung. Die Interpretation eines Modells hängt weder von einem Tool noch vom Nutzer ab. Sie ist eindeutig definiert.
- Die Modellierungssprache ist sehr einfach und geht auf einige wenige einfache Grundregeln zurück.
- Die Modellierungssprache erzwingt die Definition sicherheitsrelevanter Eigenschaften: explizite Initialisierung, Datentypen, logische Konsistenz.
- Funktionale Gleichzeitigkeit kann intuitiv dargestellt werden; Deadlocks und Race Conditions sind ausgeschlossen.
- Die Anforderungen aus der IEC 61508-3, A-2 und B-5 an ein formales Modell werden erfüllt.

Dieses Modell kann nun noch weiter verfeinert werden. Es könnte zum Beispiel gefordert sein, das funktionale Modell auf ein bestimmtes Zielsystem zu portieren, welches eine Festkomma-Implementierung benötigt. Der User wählt nun die entsprechenden Parameter aus und reichert sein Modell mit den nötigen Informationen an (*Bild 3*).

Das Werkzeug prüft nun die Konsistenz der Eingaben und propagiert dann die Datentypen und Skalierungen durch die einzelnen Datenflüsse und Funktionen. Wenn der Anwender sicher ist, dass die erreichte Implementierung seinen Anforderungen entspricht, kann er ein Implementierungsmodell generieren lassen, in dem die Skalierungen nun durch die passenden Datentypen und Skalierungsoperationen (z.B. Bitshift) umgesetzt sind.

### Modellbasierte Verifikation stellt Übereinstimmung mit Anforderungen sicher

Das Design kann nun auf Modellebene geprüft werden. IEC 61508 schreibt detailliert vor, für welchen SIL (safety integrity level) welche Maßnahmen

erforderlich sind. Eine wesentliche Maßnahme ist das Testen des Modells gegen die Anforderungsspezifikation. Aus den Anforderungen ist einerseits die Anwendung zu entwickeln. Andererseits ist es erforderlich, aus der gleichen Anforderungsbeschreibung Testfälle abzuleiten sowie für diese Testfälle erwartete Ergebnisse zu erarbeiten. Gegen diese Testspezifikation wird nun das Modell geprüft. Diese Tests müssen automatisiert werden können. Die Testfälle werden in Szenarios beschrieben, mit denen die Simulation des Modells stimuliert wird. Falls dabei Probleme auftreten, ist eine Fehlersuche nötig. Dazu kann das Modell interaktiv ausgeführt werden. Debugging-Möglichkeiten wie Breakpoints und Stop Conditions sowie grafische Animation des Modells helfen bei dieser Aufgabe.

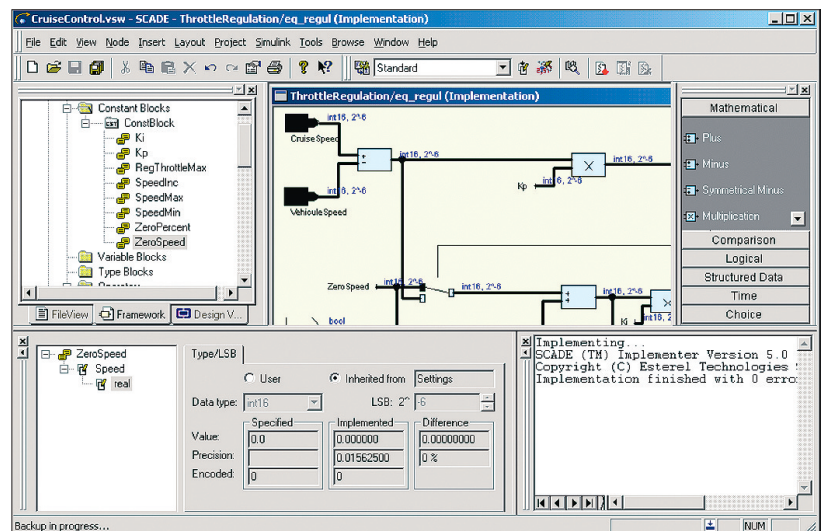
Ein Abgleich der Anforderungen mit dem Modell durch Integration von Anforderungsmanagement-Werkzeugen (DOORS) und die Messung der Testabdeckung auf dem Modell mit Hilfe der „Model Test Coverage“-Analyse (MTC) garantieren, dass alle Anforderungen umgesetzt und vollständig getestet wur-

den und kein toter oder deaktivierter Code im Modell enthalten ist.

Zur Optimierung der Festkomma-Implementierung kann es nötig sein, die minimal und maximal auftretenden Datenwerte an allen Variablen aufzuzeichnen, um realistische Daten für die Skalierung dieser Variablen auf dem Zielsystem zu erhalten. Die aufgezeichneten Daten können direkt vom Implementierungswerkzeug verwendet werden.

### Formale Verifikation und automatische Codegenerierung

Einen komplementären Ansatz zum Testen stellt die formale Verifikation dar. Der Entwickler möchte beispielsweise sicher sein, dass seine Cruise Control nicht regelt, wenn der Fahrer bremst. Das Werkzeug betrachtet nun das Modell Cruise Control daraufhin, ob für alle möglichen Sequenzen von allen möglichen Permutationen von Eingangswerten die oben genannte Eigenschaft erfüllt ist. Dies geschieht nicht durch Testen, sondern durch eine mathematische Analyse der durch die Modelle dargestellten Zustandsräume. Es können logische, arithmetische und zeitliche Zusammenhänge analysiert werden. Wenn das Werkzeug feststellt, dass die geforderte Eigenschaft nicht unter allen Bedingungen besteht, so stellt es dem Entwickler das Simulationsszenario zur Verfügung, welches zur Verletzung der Eigenschaft führt.



**Bild 3.** Nach der Modellierung werden Implementierungsdetails festgelegt. Der Anwender wählt Details aus und setzt die nötigen Parameter.

Aus dem nunmehr gegen die Anforderungen verifizierten Modell wird automatisch Code generiert. Dies geschieht mit einem geeigneten Codegenerator, welcher nach IEC 61508, Teil 1 und 3 zertifiziert und für den Einsatz unter allen SIL (safety integrity levels) qualifiziert ist. Damit ist sichergestellt, dass der generierte Code ein exaktes Abbild des Modells ist, und zwar in allen Aspekten der Struktur und des Verhaltens. Außerdem ist der Code deterministisch. Die gleiche Sequenz von Eingangssignalen führt immer zur gleichen Sequenz von Ausgangssignalen.

Auch bei der Sicherheit des generierten Codes gibt es keinen Zweifel. Bei der Generierung wird eine Untermenge von C ohne jegliche Seiteneffekte verwendet. Es gibt keine dynamische Speicherallokation, keine Zeigerarithmetik und keine Schleifen, die ins Endlose laufen können. Der generierte Code ist immer auf das Modell zurückführbar.

Im Verlauf einer Softwarezertifizierung kommt der Anwender daher in den Genuss folgender Vorteile:

- Der generierte Code muss nicht gegen die Datenflussmodelle verifiziert werden. Codecoverage-Analyse ist damit ebenfalls überflüssig.
- Statische Codeanalyse ist überflüssig, ebenso wie konstruktive White-Box-Tests.
- Wegen des vollständig statisch definierten Verhaltens ist eine Programmflusskontrolle für den generierten Code nicht nötig.

Der generierte Code genügt auch strengen Anforderungen an Effizienz, Leistungsfähigkeit und Portabilität. Eine Validation Suite für Compiler und Linker wird verwendet, um den Schritt zum Objektcode abzusichern.

## ► Einsatz in der Praxis

Die vorgestellte Methode um die Toolkette SCADE von Esterel Technologies ([www.esterel-technologies.com](http://www.esterel-technologies.com)) hat sich seit den frühen 90er Jahren in der europäischen Luftfahrtindustrie bewährt. Der Codegenerator KCG 4.2 ist nach der DO-178B bis Level A qualifiziert. Im Airbus A380 sind mehr als 50 Subsysteme mit der SCADE-Suite implementiert worden, darunter die primären und sekundären Flugsteuerungssysteme.

Mit der Zertifizierung des KCG 4.2 nach IEC 61508 durch den TÜV Süd wird dieser Ansatz auch für andere Industrien zugänglich. Die Toolkette SCADE Drive bringt diese Technologie auch in den Automobilbereich. Integration von UML und Simulink, Konfigurations- und Anforderungsmanagement ermöglichen die Einbindung in bestehende Prozesse. Esterel Technologies ist Premium Member des AUTOSAR-Konsortiums. jk



**Jakob Gärtner**

**ist Technical Director und Senior Consultant bei der Esterel Technologies GmbH. Er beschäftigt sich mit formalen Methoden zur modellbasierten Entwicklung, modellbasierten Verifikation und zertifizierten Codegenerierung. Davor hat er zehn Jahre Erfahrung als selbstständiger Consultant für sicherheitskritische Software im Luftfahrt- und Marinebereich gesammelt.**  
► E-Mail: [info.de@esterel-technologies.com](mailto:info.de@esterel-technologies.com)