

Automatische Test- und Validierungswerkzeuge im Einsatz

Modelchecking und automatische Testfallgenerierung im modellbasierten Entwicklungsprozess

Exponentielles Wachstum der Komplexität von eingebetteten Steuerungssystemen erfordert die Einführung neuer Testmethoden im Rahmen des Entwicklungsprozesses. Dieser Artikel stellt die Anwendung innovativer Test- und Validierungswerkzeuge auf der Basis formaler Methoden vor.

HANS JÜRGEN HOLBERG, GUIDO SANDMANN



Dipl.-Inf. HANS JÜRGEN HOLBERG,
Leiter Consulting und
Dipl.-Inform. GUIDO SANDMANN
Leiter Vertrieb, OSC - Embedded
Systems AG, Oldenburg

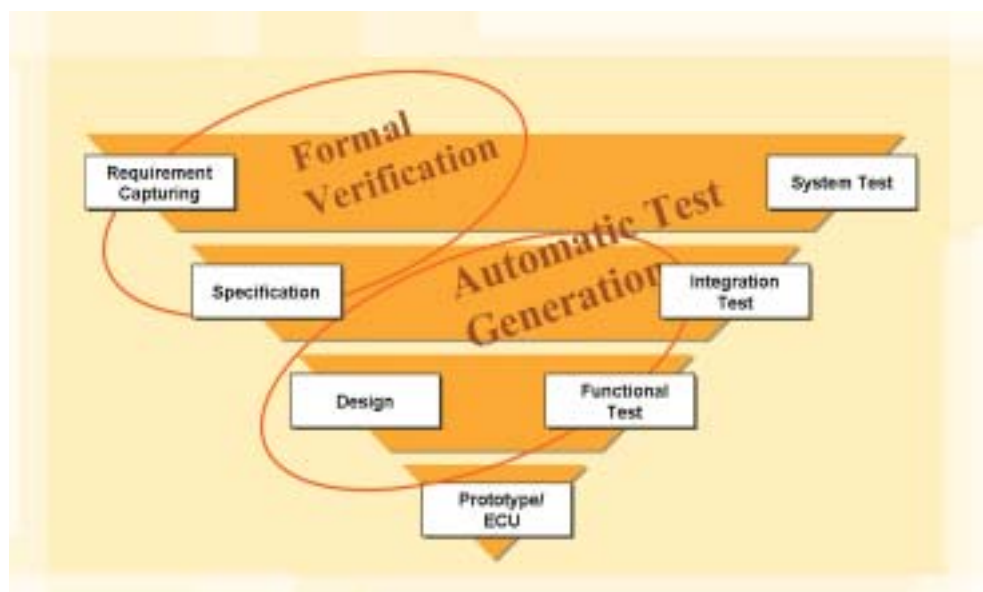


Abb. 1: Test- und Validierungstechniken im V-Modell

Der Entwicklungsprozess von eingebetteten Steuerungssystemen vollzieht aktuell den Weg von der direkten Programmierung aus Anforderungen hin zum modellbasierten Entwicklungsprozess auf der Basis ausführbarer Modelle. Dieser Weg eröffnet dem Entwickler die Chance, dem rasant steigenden Komplexitätsgrad moderner Steuergeräte, bei gleichzeitigen Anforderungen wie schnelle Innovationszyklen oder Einsparungen bei den Entwicklungskosten, zu begegnen. Schon sehr früh im Entwicklungsprozess ist er in der Lage, durch Simulation des Modells auf echtzeitfähigen

gen Rapid-Prototyping-Umgebungen Fehler in seinem Modell aufzudecken. Aber gerade die rasch wachsende Zahl der Funktionen und die damit verbundene exponentielle Zunahme der Komplexität führt den Entwickler an seine Grenzen, wenn er die Validierung der Steuergerätemodelle nicht werkzeugunterstützt durchführen kann.

Die modellbasierte Entwicklung

Grundlage für belastbare Tests von Steuergeräten oder von einzelnen Funktionen sind Prozesse,

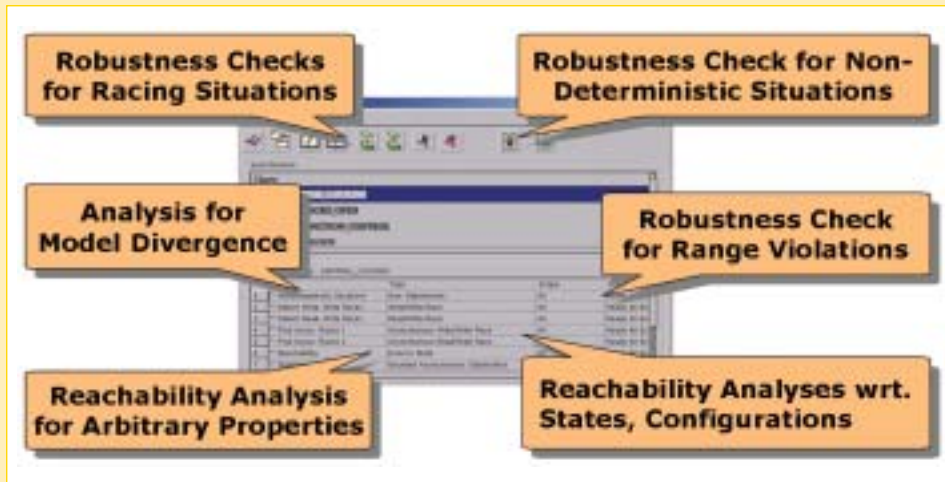


Abb. 2: Standardanalysen als ‚Push-Button‘-Technologie

die den gesamten Entwicklungszyklus umfassen. In den verschiedenen Industriedomänen wie der Automobil-Industrie, der Luft- und Raumfahrt- und auch der Bahnindustrie hat sich in diesem Zusammenhang das V-Modell etabliert. Die Abb. 1 zeigt die einzelnen Phasen der Entwicklung nach dem V-Modell:

‚Requirement Capturing‘, ‚Specification‘, ‚Design‘, ‚Prototype/ECU‘ entlang des absteigenden Astes und den Test- und Validierungsphasen ‚Function Test‘, ‚Integration Test‘ und ‚System Test‘ entlang des aufsteigenden Astes. Der wesentliche Vorteil eines modellbasierten Entwicklungsprozesses ist die ausführbare Spezifikation, wie sie moderne CASE-Tools, z.B. Statemate von I-Logix oder Matlab/Simulink von The Mathworks, anbieten. Im UML-Bereich hat sich z.B. das Werkzeug Rhapsody der Firma I-Logix etabliert. Eine ausführbare Spezifikation erlaubt es dem Entwickler von eingebetteten Systemen bereits in sehr frühen Phasen die Funktionalität seines Designs durch graphisch animierte Simulation zu überprüfen. Der Entwickler ist also deutlich vor der Implementierung in der Lage, festzustellen, ob sich sein Modell entsprechend der funktionalen Vorgaben verhält.

Betrachtet man an dieser Stelle nun den exponentiellen Anstieg der Komplexität bei der Entwicklung der Steuergeräte, dann kann man leicht erkennen, dass das auch nur annähernd vollständige Testen von Modellen per Simulation kaum durchführbar ist, denn die Erzeugung von Stimuli-Daten für die Simulation muss der Testingenieur in der Regel manuell oder bestenfalls halbautomatisch durchführen. Bereits bei kleinen Modellen mit wenigen Variablen wächst die Anzahl der Systemzustände sehr schnell in Größenordnungen jenseits von 2^{100} .

Dies macht deutlich, dass entsprechend viele Testvektoren zur vollständigen Abdeckung aller Szenarien generiert werden müssten. Der Testingenieur wird also im Wesentlichen Testfälle generieren, die die einzelnen Features seines Modells abtesten. Es sind aber gerade die uner-

warteten Situationen, die der Tester bei dieser Vorgehensweise außer Betracht lässt. Die Lösung dieses Problems sind automatische Test- und Validierungswerkzeuge, die auf der Basis von formalen Methoden in der Lage sind das Modell eines Steuergerätes auf Korrektheit vollständig zu überprüfen. Ein Blick auf das V-Modell soll verdeutlichen, wie die Werkzeuge die unterschiedlichen Rollen innerhalb des Entwicklungsprozesses unterstützen können. Auf der Basis der textuellen Anforderungen wird der Entwicklungsingenieur beginnen, diese Stück für Stück in Form eines Modells umzusetzen.

Mit steigendem Umfang wird er die Robustheit seines Modells überprüfen wollen. Aufgrund des Anwendungsszenarios muss das Werkzeug einfach zu bedienen sein (‚Push-Button‘-Technologie), damit es den Entwicklungsingenieur bei seiner täglichen Arbeit unterstützt. Dieses leistet z.B. das Werkzeug ModelChecker als Add-On-Werkzeug für das CASE-Tool Statemate. Zum Ende der Spezifikationsphase kommt es darauf an, dass das Modell eine entsprechende Reife hat und alle funktionalen Eigenschaften erfüllt sind, die während der Requirement-Capturing-Phase definiert wurden. Hier geht es um einen Zertifizierungsprozess der im Wesentlichen durch die Qualitätssicherung vollzogen wird. Dazu müssen die Requirements formalisiert werden, damit ein Modelchecker (MC) den Nachweis führen kann, dass das Modell diese Requirements erfüllt. Dieses bietet z.B. der ‚Statemate Model-Certifier‘. Wenn ein Spezifikationsmodell nachweislich alle an ihn gestellten Anforderungen erfüllt, dann ist dieses Referenzmodell eine optimale Quelle, daraus Testfälle für das Testen der späteren Implementierung zu gewinnen. Z.B. generiert das Werkzeug ‚Statemate ATG‘, automatisch eine Menge von Testvektoren, die für Tests späterer Implementierungen verwendet werden können. Auch im UML-Kontext existieren mittlerweile Werkzeuge, die den Entwicklungsingenieur bei der Modellierung unterstützen.

Modelchecking (MC)

Modelchecking bietet dem Entwickler von Steuergerätemodellen die Möglichkeit, Designs automatisch bezüglich benutzerdefinierter Eigenschaften zu überprüfen.

Diese Technologie ist eine formale Methode zur automatischen Verifikation von Modellen gegen beliebig definierte Eigenschaften. Der MC stellt eine vollständige Technologie dar, das heißt, sein Ergebnis ist zu 100 Prozent korrekt – das gleiche Ergebnis als ob das Modell unter allen denkbaren Umständen getestet worden wäre.

Wenn der MC das Ergebnis ‚true‘ bezüglich einer Eigenschaft liefert, so existiert kein Systemlauf des Modells, welcher der gegebenen Eigenschaft widerspricht. Der MC kann dies sicherstellen, da er einen mathematischen Beweis der gegebenen Eigenschaft auf dem Modell basierend durchführt. Findet der MC einen Fehler, so ist das Ergebnis ‚false‘ und es wird ein Beispiellauf generiert, der das Fehlverhalten aufzeigt. Diese Technologie kann nun im Entwicklungsprozess an verschiedenen Stellen eingesetzt werden. Diverse Anwendungsgebiete deckt die MC-Technologie der OSC Embedded Systems AG mit unterschiedlich ausgelegten Softwarepaketen ab.

Robustheitsanalysen

Während der Modellentwicklung im CASE-Tool bietet der MC eine Vielzahl von Standardanalysen (Abb. 2), die auf Knopfdruck auf ein Modell angewendet werden können.

Diese Standardchecks prüfen das Modell in Bezug auf dessen Robustheit. Einige Beispiele für solche Analysen sind:

- ▶ **Dead-Code-Analysen:**
Nichterreichbarkeitsanalysen von Modellteilen, z.B. von graphischen Zuständen
- ▶ **Hazard- und Racing-Analysen:** Schreib-Schreib- bzw. Schreib-Lese-Konflikte auf Variablen des Modells
- ▶ **Nicht-Determinismus Analysen:**
Diese Analyse findet Situationen in Zustandsautomaten in denen zwei oder mehrere Transitionen zu einem Zeitpunkt gleichzeitig feuern können.
- ▶ **Stabilitätschecks**
- ▶ **Überlaufanalysen**

Im Gegensatz zu konventionellen Ansätzen, prüft der MC hier dynamisch auf die Standardfehler und nicht statisch. Deshalb kann der MC dem Benutzer im Fehlerfall auch ein Fehlerbeispiel präsentieren, welches automatisch im Simulator ablauffähig ist.

Zusammen mit den üblichen Animationsfähigkeiten des CASE-Tools kann der Benutzer den Fehler und die Ursache direkt und schnell finden. ▶



Abb. 3: Vorteile der Modelcheck-Technologie

Automatisches Debugging

Die Analysewerkzeuge bieten zusätzlich zu den Robustheitsanalysen noch die Möglichkeit zu prüfen, ob ein beliebig benutzerdefinierter Systemzustand dynamisch erreichbar ist oder nicht. Hier definiert der Modellentwickler einfach den gewünschten Zustand mit Hilfe einer simplen Formel, die aus beliebigen Modellobjekten wie Variablen und graphischen Elementen bestehen kann und fragt den MC, ob und wie er erreichbar ist. Zum Beispiel soll ausgeschlossen werden, dass sich ein Modell gleichzeitig in zwei verschiedenen parallelen Zuständen befindet und eine Variable einen gewissen Schwellwert überschreitet. Sollte der Pfad existent sein, so wird ein entsprechendes Simulationsprogramm generiert, welches die Eingabevariablen so über die Zeit stimuliert, dass der gewünschte Fehlerfall sichtbar wird. Sollte der MC als Ergebnis ‚unerreichbar‘ angeben, so kann der Entwickler absolut sicher sein, dass dieser Fehlerfall niemals auf Modellebene auftreten wird.

Simulationsunterstützung

Die oben beschriebene Methode, selektiv Simulationsläufe generieren zu können, kann auch dazu verwendet werden, um komplizierte Vorläufe von manuellen Simulationsläufen zu generieren. Diese ersparen dem Entwickler bzw. Tester die komplizierte, zeitaufwendige Analyse, welche Eingabesequenzen zum gewünschten Simulationspunkt führen.

Definition und Beweis eindeutiger Eigenschaften

Nach der Debuggingphase soll das Design auf funktionale Korrektheit überprüft werden; das Ergebnis ist ein Referenzmodell für die spätere Implementierung. Wichtige Modelleigenschaften können mit Hilfe einer vordefinierten Pattern-Bibliothek einfach und intuitiv beschrieben werden. Diese vom Hersteller erweiterbare Pat-

tern-Bibliothek deckt einen sehr hohen Anteil industriell relevanter Eigenschaften ab und wurde auf Basis von Erfahrungen vor allem im Automobil-, Flugzeug- und Eisenbahnbereich erstellt. Diese vordefinierten Pattern geben die Eigenschaftsstruktur vor und müssen dann vom Benutzer mit entsprechenden Modellobjekten, wie zum Beispiel Variablen oder auch graphischen Elementen mit deren Parametern, definiert werden. Die Pattern decken z.B. Invarianten (‚es muss immer P gelten‘), Ordnungen (‚Q nur nach P‘) oder auch Realzeiteigenschaften (‚wenn P für N Zeiteinheiten gilt, dann muss Q innerhalb der nächsten M Zeiteinheiten gesehen werden‘) ab und erleichtern es dem Benutzer, temporallogische Eigenschaften zu formalisieren.

Hinzu kommt, dass der Benutzer bei Pattern immer genau weiß, was spezifiziert und bewiesen wurde. Nach der Definition der gewünschten Eigenschaft, kann diese per Knopfdruck vom MC analysiert werden.

Automatische Testfall Generierung (ATG)

Ist das Modell eines Steuergeräts erst einmal bezüglich wichtiger Eigenschaften verifiziert und als Referenzmodell ausgezeichnet, kann damit aber nicht auf eine korrekte Implementierung des Steuergeräts geschlossen werden. Auswirkungen der Kodierung (auch automatische), der Hardware, des Netzwerks und des Betriebssystems auf die Korrektheit des Steuergeräts sind auf diesem Weg noch nicht evaluiert worden. Hierfür bietet sich die Methode an, aus dem Referenzmodell selbst eine Menge von Testfällen automatisch zu generieren. Diese Testfälle beschreiben die Eingangsbelegungen des Steuergeräts über die Zeit mit den zu erwartenden Ausgangsbelegungen. Diese Testfälle können dann automatisch auf den Testumgebungen des implementierten Steuergeräts ausgeführt werden. Auch die Auswertung kann automatisiert werden, da auch die Erwartungswerte aus dem Modell ab-

geleitet werden. Die Güte der generierten Testfälle wird nach Modellüberdeckungskriterien gemessen und anhand des von ATG automatisch generierten Statistikberichts abgelesen. Die automatische Testfallgenerierung basiert auf einer Mischung verschiedener Basistechnologien. Hier kommen Heuristische Methoden, wie auch Modelcheck-Technologien zum Einsatz.

Zusammenfassung

Die oben beschriebenen OSC-Analysewerkzeuge sind für verschiedene Frontend-Werkzeuge wie Matlab/Simulink/Stateflow, Statemate und Rhapsody UML bereits im Einsatz und unterstützen den modellbasierten Entwicklungsprozess auf dem kompletten Weg hin zum Produkt. Modelchecking bildet das Bindeglied zwischen Anforderungen und Modell und ATG sichert den Weg vom Modell hin zur Implementierung ab. Modelcheck- und ATG-Werkzeuge garantieren einen durchgängigen Entwicklungsprozess, da Fehler die auf einer Ebene gemacht werden, auch nur auf dieser Ebene gefunden und beseitigt werden müssen. Konzept- und/oder Modellfehler werden durch Modelchecking früh gefunden und nicht erst auf dem Hardwareteststand, wo eine Behebung des Problems extrem teuer ist (Abb. 3). Fehler, die durch den konventionellen Testprozess nicht gefunden werden, können unter Umständen katastrophale Folgen nach sich ziehen. Eine vollständige Methode wie Modelchecking kann die Risiken minimieren und Entwicklungszeit sparen helfen. Mittlerweile nimmt das Testen mehr als die Hälfte der Gesamtentwicklungszeit in Anspruch. ATG bietet im Entwicklungsprozess eine große Chance diesen Aufwand zu minimieren.

Beitrag als PDF per ‚more@click‘:

