

# Modellbasiertes, automatisiertes Testen mechatronischer Regelungssysteme

## Durchgängiges Testen während des gesamten Entwicklungsprozesses

**D**ie Mehrzahl der Tests, die bei der Entwicklung von Regelelektronik zur Sicherstellung der Qualität notwendig sind, erfordert automatisiertes Testen. Obwohl ca. 30 bis 40 Prozent des Gesamtaufwands von Entwicklungsprojekten in das Testen investiert werden, fehlt es oftmals an der Unterstützung durch geeignete Methoden und Tools. Ein großer Teil der heutigen Werkzeuge für das automatisierte Testen fokussiert auf den Test der Software selbst. Solche Testwerkzeuge können jedoch erst dann angewendet werden, wenn bereits eine ausführbare Implementierung der zu entwickelnden Funktionen vorliegt, beispielsweise in Form von C-Code. Dieses Vorgehen wird auch als ‚Unit-Test‘ bezeichnet. Durch ein modellbasiertes Vorgehen sind Tests bereits in sehr frühen Entwicklungsphasen möglich; bevor eine erste Implementierung der zu entwickelnden Funktionen als C-Code vorliegt, können Tests auf Basis eines ausführbaren Modells der zu testenden Funktion oder des Steuergeräts durchgeführt werden. Der folgende Beitrag stellt eine Lösung vor, welche die modellbasierte Entwicklung mit dem automatisierten Testen verbindet.

### Entwicklung mechatronischer Regelungssysteme

Die Entwicklung mechatronischer Regelungssysteme folgt typischerweise dem ‚V-Modell‘ (Abbildung 1). Das V-Modell besteht aus den folgenden Schritten [1]:

- Funktionsentwurf: Entwicklung von Steuerungs- und Regelungsfunktionen und -algorithmen und Test der Funktionen am simulierten Fahrzeug

**► Autor**

Dr.-Ing. Dipl.-Math. KLAUS LAMBERG ist Produktmanager Test- und Experimentiersoftware bei der dSPACE GmbH; Technologiepark 25, D-33100 Paderborn Fon: 05251/1638-0, Fax: 05251/66529 E-Mail: klamberg@dspace.de

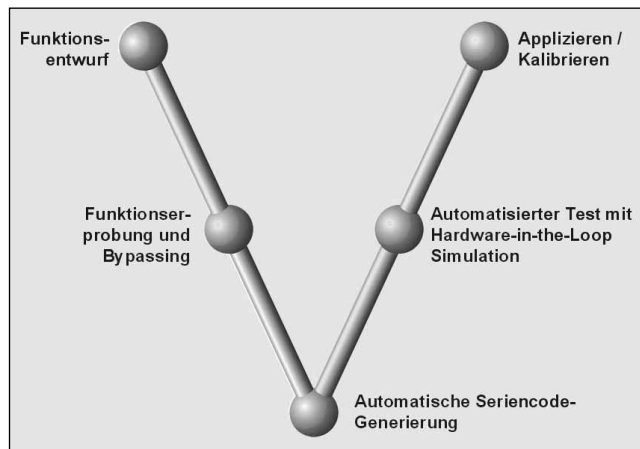


Abb. 1: Entwicklungsprozess für mechatronische Regelungssysteme

- Funktionserprobung und Bypassing: Erprobung der zu entwickelnden Funktion auf einem Prototyping-System im realen Fahrzeug oder am Prüfstand
- Automatische Seriencodegenerierung: Implementierung der Funktion auf dem Zielprozessor des Steuergeräts mittels automatischer Codegenerierung
- Automatisierter Test mit ‚Hardware-in-the-Loop‘-Simulation: automatisiertes Testen des Steuergeräteprototypen und seiner Funktion in einer virtuellen bzw. simulierten Umgebung
- Applizieren / Kalibrieren: Feinabstimmung der Funktionen und Algorithmen durch Einstellen von Steuergeräteparametern während der Erprobung am Prüfstand oder im Fahrzeug

entwicklungsmuster zum nächsten das komplette V-Modell mindestens einmal durchlaufen.

### Automatisiertes Testen

Heute wird vielfach manuell getestet. Dabei gibt ein Benutzer Testdaten zur Stimulation des Testobjekts vor und erfasst die Reaktion des Testobjekts zur anschließenden Auswertung. Bei der HIL-Simulation beispielsweise interagiert der Benutzer mit dem Gesamtsystem bestehend aus dem Steuergeräteprototypen und dem HIL-System, der das Verhalten des Steuergeräts in Echtzeit simuliert. Der Benutzer gibt Modellgrößen vor und erfasst die Reaktionen des Steuergeräts durch Beobachtung und Aufzeichnung entsprechender Zustandsgrößen des Modells, die von dem angeschlossenen Steuergerät direkt oder indirekt beeinflusst werden.

Das V-Modell wird während der gesamten Entwicklung iterativ angewendet. So wird in der Regel schon von einem Steuergeräte-

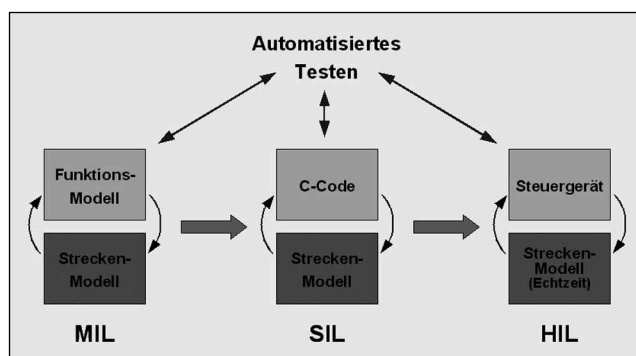


Abb. 2: Automatisiertes, modellbasiertes Testen in den unterschiedlichen Entwicklungsphasen

Zunehmend wird die Interaktion des Benutzers durch Testprogramme ersetzt. Solche Testprogramme übernehmen die Aufgabe der Stimulation und der Signalerfassung und -auswertung. Somit lassen sich eine Vielzahl von Tests automatisiert, d.h. auch über Nacht oder am Wochenende durchführen („Lights-out-Tests“). Dabei wird nicht nur die Testtiefe erhöht, sondern auch die Gesamtzeit, die für das Testen benötigt wird, erheblich reduziert. Bei der Entwicklung einer Getriebesteuerung konnten beispielsweise die Prüfzeiten für die Endstufendiagnose durch Testautomatisierung auf ein Zehntel des ursprünglichen Aufwands reduziert werden [2].

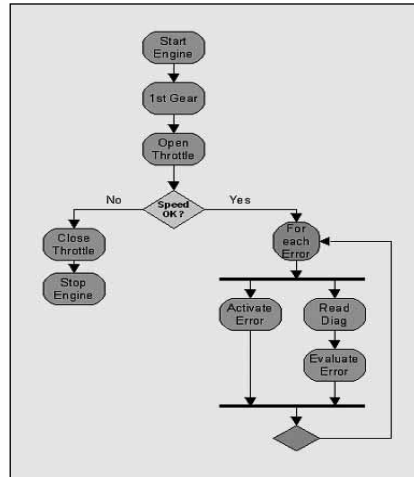


Abb. 3: Grafische Darstellung eines Testablaufs

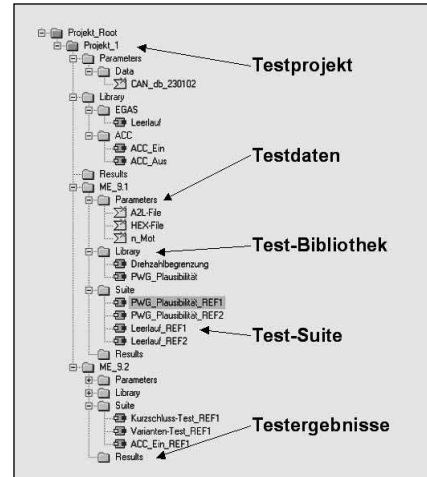


Abb. 4: Beispiel einer Testprojektstruktur

### Modellbasiertes Testen

Ein Nachteil der HIL-Simulation ist, dass bereits ein Prototyp des zu testenden Steuergeräts verfügbar sein muss, damit die notwendigen Tests durchgeführt werden können. Zukünftig wird automatisiertes Testen deshalb nicht mehr nur auf die HIL-Simulation beschränkt bleiben. Testen kann schon viel früher im Entwicklungsprozess einsetzen, wenn die zu testende Funktion oder das zu testende Steuergerät als ausführbares Simulationsmodell vorliegen, beispielsweise in ‚Simulink‘ oder ‚Stateflow‘. Dazu wird das Funktionsmodell mit einem Streckenmodell des Fahrzeugs verbunden. Das Funktionsmodell bildet dann die ‚Unit Under Test‘ (UUT), während das Streckenmodell die gesamte Umgebung der UUT simuliert. Weil das Funktionsmodell dabei innerhalb einer kompletten Regelschleife betrieben wird, spricht man häufig von ‚Model-in-the-Loop‘ (MIL). Bereits in dieser frühen Phase können umfangreiche Funktionstests durchgeführt werden. Ebenso sind strukturelle Tests, wie z.B. Modellabdeckungstests möglich.

Im Gegensatz zu MIL wird die UUT in einer ‚Software-in-the-Loop‘-Umgebung (SIL) nicht durch ein Funktionsmodell, sondern durch den C-Code repräsentiert, der später im

Steuergerät implementiert wird. Auch der C-Code kann mit dem Streckenmodell verbunden und somit in der simulierten Umgebung betrieben werden. Auch in dieser Phase werden Funktionstests durchgeführt, diesmal auf Basis des C-Codes. Ebenso sind strukturelle Tests möglich, wie z.B. Codeabdeckungstests.

Erst der letzte Schritt innerhalb dieses Prozesses ist die Hardware-in-the-Loop-Simulation [3]. Von großem Vorteil ist, dass Funktionstests, die bereits in einer MIL- oder SIL-Phase definiert und durchgeführt wurden, in der HIL-Phase wiederholt werden können, dann jedoch am realen Prototypen des zu testenden Steuergeräts. Typischerweise werden bei der Hardware-in-the-Loop-Simulation weitere Tests durchgeführt, wie z.B. Diagnose-tests und Tests ganzer Steuergerätenetzwerke.

Dieses Vorgehen, das durch die Verwendung von Funktions- und Streckenmodellen geprägt ist, und durch den Einsatz von Testprogrammen wie oben beschrieben ergänzt wird, bezeichnet man als automatisiertes, modellbasiertes Testen. Abbildung 2 zeigt das Vorgehen beim automatisierten, modellbasierten Testen in den verschiedenen Entwicklungsphasen.

### Grafische Testentwicklung

Testprogramme werden heute in der Regel mithilfe von Skriptsprachen geschrieben. Die Verwendung von Skriptsprachen ist äußerst flexibel. Fortgeschrittene Nutzer können ihre eigenen Bibliotheken anlegen und somit Funktionen und Tests wiederverwenden. Andererseits erfordert die manuelle Programmierung zunächst das Erlernen der Skriptsprache. Für weniger geübte Anwender kann die Testerstellung in einer Skriptsprache umständlich und fehleranfällig sein. Daher ist es von Vorteil, wenn Tests auf einer höheren Abstraktionsebene beschrieben werden können. Dies kann durch die grafische Testentwicklung erreicht werden. Ein grafischer Testeditor ermöglicht eine kurze und steile Lernkurve durch den Anwender. Die Testentwicklung wird schneller und somit effizienter, mehr Tests können in kürzerer Zeit entwickelt werden. Zusätzlich vereinfacht eine grafische Testrepräsentation das Navigieren in komplexen Testsequenzen mit mehreren Hierarchieebenen.

Abbildung 3 zeigt ein Beispiel einer grafischen Testdarstellung. Die verwendete Semantik lehnt sich an UML-Aktivitätsdiagramme

### LESETIPP

Sie suchen nach bestimmten Autoren?

Das Autorenverzeichnis (Griffmarke D.05) gibt einen alphabetischen Überblick über alle in dieser Ausgabe vertretenen Autoren und Co-Autoren

publish industry  
TECHNIK KOMMUNIZIEREN

Gollierstraße 23 · 80339 München, Germany · Fon +49/89/500383-0 · Fax +49/89/500383-10 · info@publish-industry.net · www.publish-industry.net

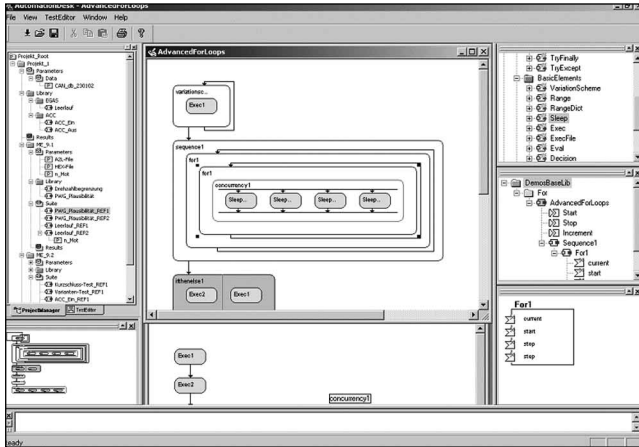


Abb. 5:  
'AutomationDesk'

an. Die Bedeutung der Unified Modeling Language (UML) [4]) in diesem Zusammenhang und insbesondere im automotiven Umfeld steigt immer mehr, sowohl für die Beschreibung von Softwarearchitekturen, als auch für die Testspezifikation.

### Automatisierungsschnittstellen und -bibliotheken

Die Effektivität eines Automatisierungskonzepts wird grundlegend durch die verfügbaren Automatisierungsfunktionen bestimmt. Diese Automatisierungsfunktionen definieren den Funktionsumfang, den ein Anwender aus einer Testsequenz heraus nutzen kann. Typische Beispiele dafür sind Funktionen für den Zugriff auf das Simulationsmodell, Funktionen für die Verwaltung und Bedienung von Fehlersimulations-Hardware (z.B. Relaiskarten in einem HIL-Simulator) und Funktionen für den Zugriff auf den Diagnosespeicher des zu testenden Steuergeräts. Weitere Funktionsumfänge, die beim automatisierten Testen häufig benötigt werden, sind Schnittstellen zu anderen Geräten und Programmen, z.B. zur Auswertung von Tests und Schnittstellen zur automatischen Report-Generierung. Innerhalb eines Tests werden diese Funktionen aufgerufen und verwendet. Sinnvoll ist es, wenn die Funktionen für die grafische Testentwicklung in einer Bibliothek bereitstehen. Der Anwender kann sie dann sehr einfach in seine Testsequenz einbauen, indem er sie unter Verwendung der Maus einfach aus der Bibliothek in die grafische Testdarstellung zieht. Generell lassen sich zwei Arten von Bibliotheken unterscheiden:

- ‚Built-in‘-Bibliotheken stellen grundlegende Funktionalität bereit, um daraus neue Sequenzen und Tests zusammenzustellen. Beispiele dafür sind Kontrollstrukturen (sequentielle oder nebenläufige Strukturen), Bedingungen und Schleifen

wie z.B. ‚If-Else‘- oder ‚While‘-Konstrukte, algebraische und logische Operatoren und Schreib- und Lesefunktionen für den Zugriff auf das zugrundeliegende Simulationsmodell.

- Kundenspezifische Bibliotheken enthalten Erweiterungen, die durch den Anwender gemacht werden. Typischerweise handelt es sich um generische Sub-Sequenzen, wie z.B. das Anfahren eines bestimmten Betriebspunkts, die in verschiedenen Testsequenzen benötigt und wiederverwendet werden sollen.

Solch ein Bibliothekskonzept hat einen entscheidenden Vorteil: ein Testentwickler kann nicht nur vordefinierte Automatisierungsschritte zu Sequenzen höherer Funktionalität zusammenfügen. Er kann diese Aggregationen auch zurück in die Bibliothek stellen, um sie in anderen Testsequenzen oder Projekten wiederzuverwenden. Nach und nach wachsen die verfügbaren Bibliotheksumfänge, die Testentwicklung wird schneller und somit effizienter.

### Testprojektmanagement

Für den Test eines einzelnen Steuergeräts können hunderte oder gar tausende von Tests notwendig sein [5]. Diese Tests müssen nicht nur entwickelt und ausgeführt werden. Sie müssen auch gespeichert und reproduzierbar verwaltet werden. Die große Zahl von Testergebnissen – jeder Testlauf erzeugt neue Ergebnisse – muss ebenfalls administriert werden. Basierend auf den Ergebnissen werden automatisiert Testreports erzeugt. Die Speicherung, Verwaltung und Administration der großen Zahl von Tests zusammen mit den Testdaten und den Testergebnissen erfordern Möglichkeiten der Strukturierung und des Managements von Testprojekten.

Abbildung 4 zeigt ein Beispiel für eine Testprojektstruktur. Die Testdaten und Testergebnisse werden gemeinsam mit den Testse-

quenzen dargestellt. Es ist auch erkennbar, dass zwischen Bibliotheken und Suites unterschieden wird. Wie bereits erläutert wurde, dienen die Bibliotheken zur Ablage von Testsequenzvorlagen. Diese können in anderen Testsequenzen oder Projekten wiederverwendet werden. Die Suites hingegen enthalten die ausführbaren Testsequenzen.

Die Möglichkeit des Testprojektmanagements – ausgehend von einer strukturierten Darstellung des gesamten Testprojekts – ist von zentraler Bedeutung. Für die Beherrschung der beim automatisierten Testen anfallenden immensen Zahl von Tests, Testdaten und Testergebnissen ist das Testprojektmanagement daher ein wichtiges Mittel.

### Zusammenfassung

Bei der Entwicklung mechatronischer Regelungssysteme wird das automatisierte Testen immer wichtiger. Dies gilt nicht nur im Zusammenhang mit der Hardware-in-the-Loop-Simulation, sondern zunehmend auch für frühere Entwicklungsphasen. Schlüsselbegriffe in diesem Zusammenhang sind Model- und Software-in-the-Loop. Der wachsende Bedarf nach automatisiertem Testen erfordert neue Ansätze der Entwicklung und Verwaltung großer Testprojekte im gesamten Entwicklungsprozess. Die vorgestellten Lösungsansätze, grafische Testentwicklung, erweiterbare Automatisierungsbibliotheken und Testprojektmanagement bilden die wichtigsten konzeptionellen Eckpunkte des Tools AutomationDesk [6].

### Literatur

- [1] Lamberg, K.; Wältermann, P.: Einsatz der HIL-Simulation zum Test von Mechatronik-Komponenten in der Fahrzeugtechnik. 2. Tagung Mechatronik im Automobil, HdT, München, 2000.
- [2] Gühmann, C.; Riese, J.: Testautomatisierung in der Hardware-in-the-Loop Simulation. VDI-Berichte Nr. 1672, 2002.
- [3] Lamberg, K.: Testen mit System – Moderne Lösungen für die Hardware-in-the-Loop-Simulation. ATZ/MTZ Automotive Electronics, 2002.
- [4] OMG: <http://www.uml.org>.
- [5] Lamberg, K., Richert, J.; Rasche, R.: A New Environment for Integrated Development and Management of ECU Tests. SAE 2003-01-1024, 2003.
- [6] Produktinformationen zu dSPACE AutomationDesk: <http://www.dspace.de>

Beitrag als PDF im Internet:

[www.duv24.net](http://www.duv24.net)

more @ click TK4B0703

