

Automatisierter Closed-Loop-Testprozess für Steuergerätefunktionen

Verifikation und Validation von Softwaresystemen, die in ein technisches Umfeld eingebettet sind (Embedded Software), erfordern oftmals bis zu 50 Prozent des gesamten Entwicklungsaufwands. Aus diesem Grund entwickelte die IAV GmbH im Projekt SiLEST einen Prozess zum frühzeitigen und automatischen Testen von Motorsteuergerätefunktionen, das wiederholbar sowie rückverfolgbar ist. Dieser Prozess wird im Folgenden anhand der Ansteuerung eines zweistufigen Abgasturboladers für einen Dieselmotor vorgestellt.

1 Einführung

Funktionstests von Embedded Software sind aufwändig, da sie typischerweise einen Closed-Loop-Betrieb erfordern, das heißt die Rückkopplung von Messgrößen auf den Funktionseingang. Stand der Technik sind Hardware-in-the-Loop- (HiL-) Tests, die ein reales Steuergerät im Zusammenspiel mit einer simulierten Umgebung untersuchen. Nachteilig an HiL-Tests sind die für die Umgebungssimulation benötigte, teure Echtzeit-Hardware sowie die üblicherweise erst späte Verfügbarkeit der Steuergeräte-Hardware. HiL wird deshalb meistens erst für Integrationstests eingesetzt. Auch stehen Entwicklungssteuergeräte oft nur in geringer Stückzahl zur Verfügung, so dass HiL-Tests nur selten zeitgleich durchgeführt werden können.

Für möglichst frühzeitige und umfangreiche Tests der Embedded Software ist es deshalb wünschenswert, während des gesamten Entwicklungsprozesses begleitend automatisierte Regressionstests durchzuführen [1]. In frühen Entwicklungsphasen sind reine Simulationsmodelle zu verwenden (MiL: Model-in-the-Loop, SiL: Software-in-the-Loop), die unter anderem auch die Steuergeräte-Hardware nachbilden.

Die Benutzerakzeptanz neuer Testprozesse spielt eine zentrale Rolle. Der zusätzliche Nutzen muss mit minimalem Mehraufwand einhergehen, und die Tools müssen intuitiv bedienbar sein. Daher sind bestehende Modelle, Softwarewerkzeuge und Toolketten in die Testautomatisierung zu integrieren. Elementare Anforderungen an einen effizienten Testprozess sind zudem eine automatische Testauswertung sowie eine systematische Aufbereitung und Dokumentation der erzielten Testresultate. Des Weiteren müssen alle Tests und ihre Ergebnisse rückverfolgbar und jederzeit reproduzierbar sein.

2 SiLEST

Das Verbundvorhaben SiLEST wurde im Rahmen der Forschungsoffensive „Software Engineering 2006“ mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) gefördert. In dem Vorhaben arbeiten die Projektpartner DLR/Sistec, Fraunhofer First, Webdynamix GmbH, TU Berlin/MDT und IAV GmbH branchenübergreifend für die Luft- und Raumfahrt sowie die Automobilindustrie an der Umsetzung der genannten Ziele.

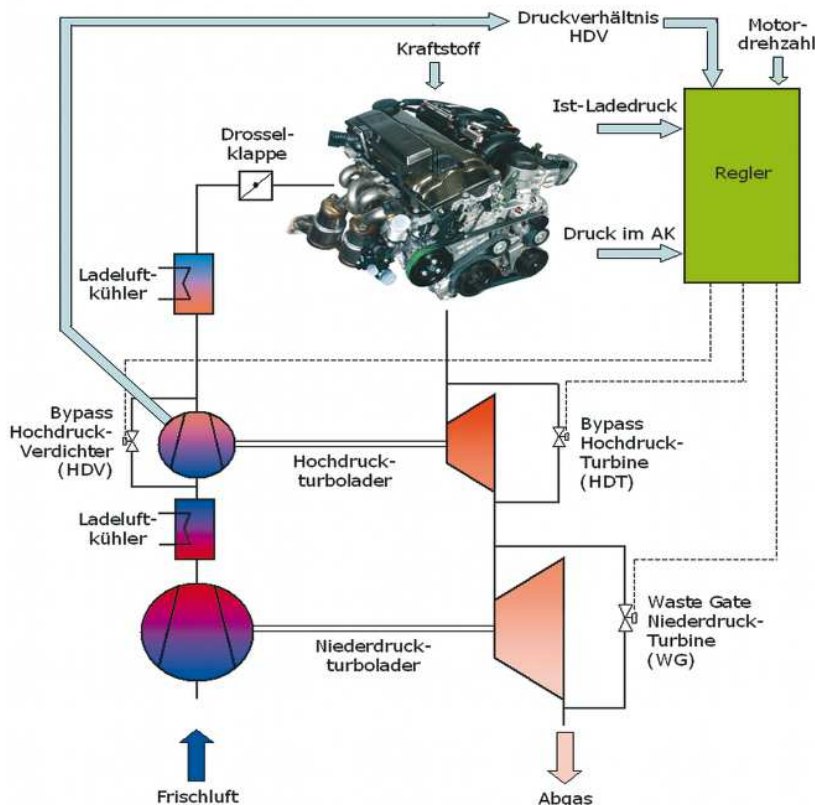


Bild 1: Prinzip der zweistufig geregelten Aufladung

Figure 1: Principle of supercharging controlled in two stages

Die Autoren



Dipl.-Ing.
Sven Rebeschies
ist Funktionsentwickler für Motor- und Getriebe-steuergeräte bei der IAV GmbH in Berlin.



Dr.-Ing.
Thomas Liebezeit
ist wissenschaftlicher Mitarbeiter am Fachgebiet Elektronische Mess- und Diagnosetechnik an der TU Berlin.



Dipl.-Ing.
Uzmee Bazarsuren
ist wissenschaftliche Mitarbeiterin am Fachgebiet Elektronische Mess- und Diagnosetechnik an der TU Berlin.



Prof. Dr.-Ing.
Clemens Gühmann
leitet das Fachgebiet Elektronische Mess- und Diagnosetechnik an der TU Berlin.

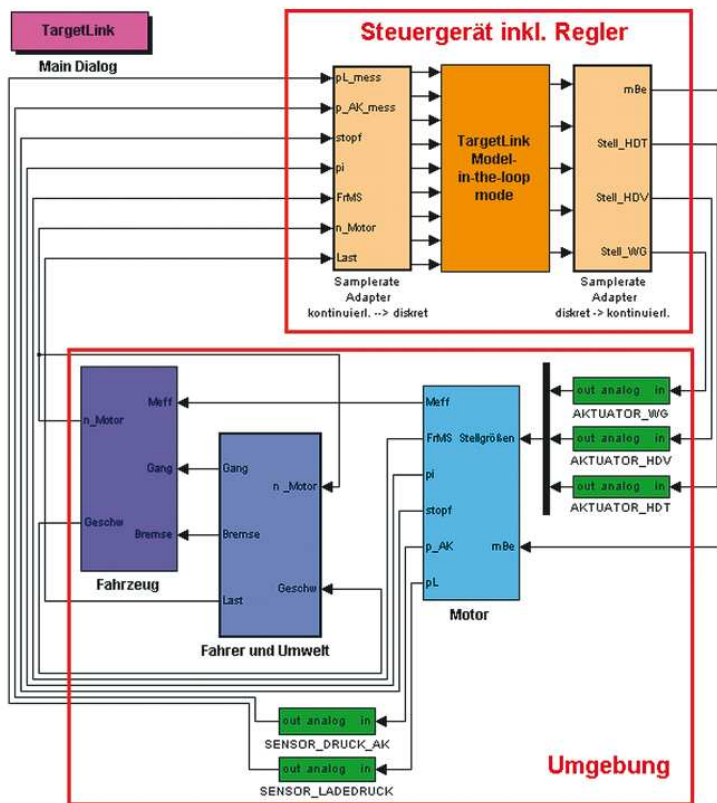


Bild 2: Simulink-Modell für die Regelung des zweistufig aufgeladenen Motors
Figure 2: Simulink model of the controller for a two-stage supercharged engine

Sie untersuchen zudem, welches Einsparpotenzial an Entwicklungskosten der Softwaretest eingebetteter Systeme innerhalb einer simulierten Umwelt besitzt und ermitteln die Einsatzmöglichkeiten und -grenzen des SiL-Testverfahrens. Der neue Testprozess soll HiL-Tests auf ein notwendiges Maß reduzieren.

Im Fokus des SiLEST-Projekts stehen funktionale White-Box-Tests komplexer Reglerfunktionen im Closed-Loop-Betrieb. Einen weiteren Schwerpunkt bildet die Simulation von Sensor- und Aktuatorstörungen, deren Auswirkungen auf das Gesamtsystem ebenfalls analysiert werden.

Die wesentlichen Bestandteile der hier präsentierten Testumgebung sind das Testsystem, die Testfälle, die Versionsverwaltung und die Testverwaltung, die im Folgenden beschrieben werden.

3 Testsystem

Dieser Beitrag stellt als praktische Anwendung die Ansteuerung eines zweistufigen Abgasturboladers für einen Dieselmotor vor. Bei der zweistufig geregelten Aufla-

dung (Prinzipbeschreibung in [2]) sind zwei unterschiedlich große Abgasturbolader, die für verschiedene Betriebsbereiche optimiert sind, in Serie geschaltet, **Bild 1**. Diese verdichten die Luft vor dem Einströmen in den Arbeitszylinder vor. Ziel der Regelung ist, dass sich der Ist-Ladedruck einem vorge-

gebenen Soll-Ladedruck angleicht. Neben dem Ist-Ladedruck wird auch der Druck im Auslasskrümmer gemessen. Als Steller fungieren das Waste Gate der Niederdruck-Turbine sowie die Bypasses des Hochdruck-Verdichters und der Hochdruck-Turbine.

Das zu untersuchende Testsystem besteht hier aus Matlab/Simulink, auf dem das im **Bild 2** dargestellte Modell simuliert wird. Das Modell umfasst:

- das Steuergerätemodell mit dem Regler für den zweistufig aufgeladenen Motor einschließlich Steuergerätecharakteristik (Task-Management, Scheduler, Leerlaufregler, Einspritzmengenberechnung) und Schnittstellen zur Umgebung
- das Umgebungsmodell mit Teilmodellen für Sensorik/Aktuatorik, Motor, Fahrzeug (Getriebe, Kupplung, Längsdynamik usw.), Fahrer und Umwelt (Steigung, Wind etc.).

Der Regler liegt als ausführbare Spezifikation (MiL) oder Steuergerätecode (SiL) vor. Die physikalischen Zusammenhänge des zweistufig aufgeladenen Dieselmotors sind mit enDyna Themos [3, 4] umgesetzt.

Für die Modellierung des Nominal- sowie des Fehlerverhaltens von Sensoren und Aktuatoren wird die in SiLEST entwickelte Simulink-Bibliothek SALib (Sensors and Actuators Library) eingesetzt. Die SALib-Modelle bilden neben dem fehlerfreien Verhalten auch typische, in der Praxis auftretende Fehlerfälle wie Kabelbruch, Kurzschluss gegen Masse, additives Messrauschen oder Alterungseffekte nach, **Bild 3**.

Werden derartige Fehler in der Entwicklung nicht berücksichtigt, kann es in der Serie -bedingt durch Folgefehler - zum kompletten Ausfall mechatronischer Systeme kommen. Für die zu testenden Steuer-

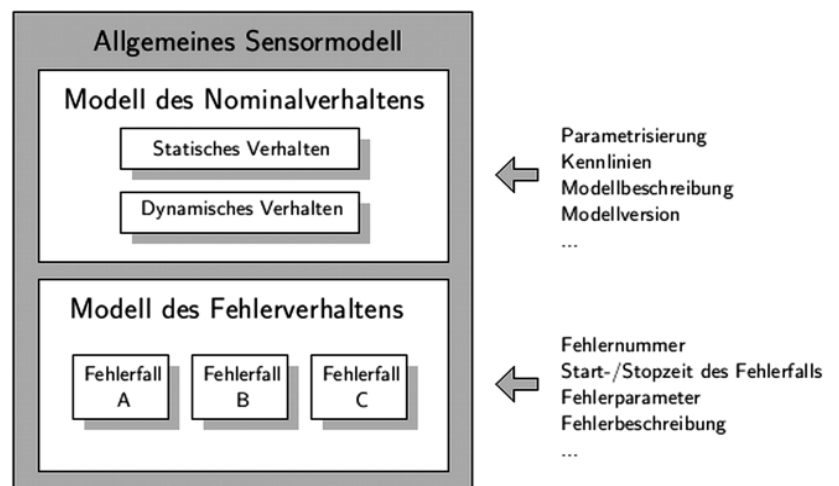


Bild 3: SALib-Sensormodell
Figure 3: SALib sensor model

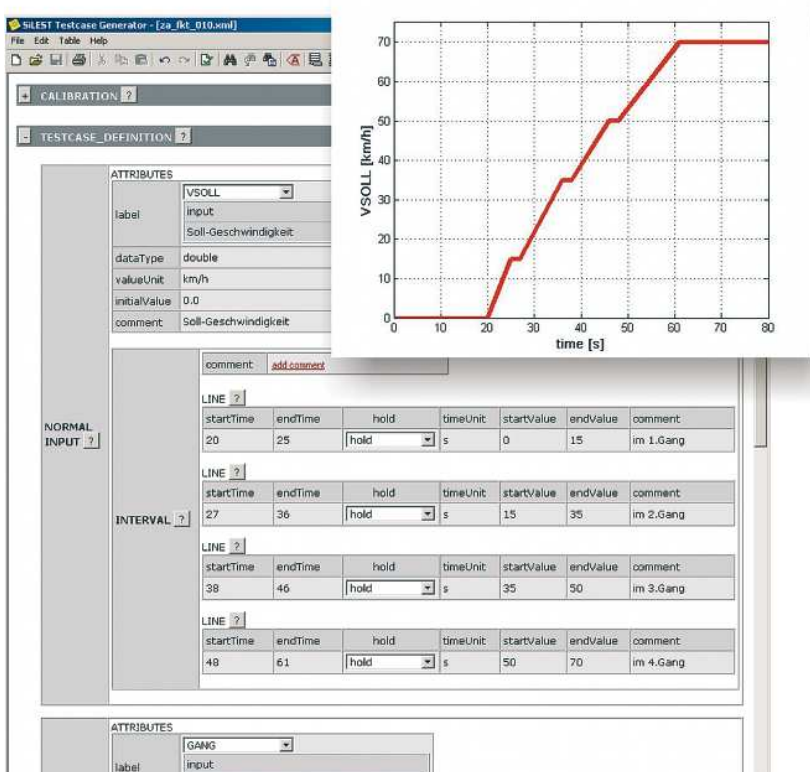


Bild 4: Testfall-Definition im Testfall-Editor und Geschwindigkeitsvorgabe

Figure 4: Test case defined in the test case editor and specification of velocity

geräte- und Umgebungsmodelle kann der Entwicklungsingenieur einen Satz geeigneter Sensoren und Aktuatoren auswählen und so parametrieren, dass verschiedene Fehler zur Simulationslaufzeit erzeugt werden.

Sowohl bei der MiL- als auch bei der SiL-Simulation laufen alle Modellteile auf einem einzigen PC. Die Simulation muss deshalb nicht in Echtzeit erfolgen.

4 Testfälle und Testkonfiguration

Neben dem korrekten Erreichen von stationären Arbeitspunkten ist vor allem das dynamische Systemverhalten von Interesse. Des Weiteren ist die Reaktion auf Sensor- und Aktuatorfehler von Belang. Deshalb sind Systemtests zu entwickeln, die auf diese Aspekte zugeschnitten sind.

Zur Definition der Tests wurde in SiLEST ein eigenes XML-Testformat entwickelt [5]. Ein Testfall besteht aus der Vorgabe von Eingangssignalverläufen, Parametern sowie Sensor- und Aktuatorfehlern. Zusätzlich enthält er für die automatische Auswertung die Definition der zulässigen Systemantworten inklusive Toleranzen oder Schranken. Das Testfallformat unterstützt sowohl eine analytische Signaldefinition

als auch eine über Wertepaare. Signale können zudem von anderen Signalen abhängig sein.

Auch für die benutzerfreundliche Eingabe der Testfälle ist gesorgt: Der SiLEST-eigene Testfall-Editor bietet dem Testersteller nur die jeweils möglichen Elemente mit ihren zugehörigen Attributen an und führt Syntaxüberprüfungen auf Basis des XML-Schemas durch. Zur grafischen Eingabe von Funktionsverläufen steht eine API-Schnittstelle zur Verfügung. Der Testersteller kann sich somit auf die Definition seiner Testfälle konzentrieren, ohne sich mit dem Format auseinandersetzen zu müssen. Für Nachsimulationen lassen sich zudem toolunterstützt Testfälle aus realen Messdaten generieren.

Neben den Testfällen existiert eine Testkonfiguration. Diese beschreibt das Testsystem und enthält Anweisungen zur Umsetzung der Testfälle auf dem Testsystem. Hierdurch sind die Testfälle unabhängig von der Testmethode (zum Beispiel MiL, SiL oder HiL) definiert.

Für den Regler der zweistufigen Aufladung wurden insgesamt 60 Systemtests identifiziert und definiert. Diese sind logisch in einer Testsuite organisiert, aus der hier beispielhaft ein Testfall gezeigt wird:

Während eines Anfahrvorgangs, der den ersten 80 s des EUDC-Testzyklus entspricht, wird nacheinander vom ersten bis zum fünften Gang geschaltet. Die Geschwindigkeit steigt von 0 bis maximal 70 km/h an und bleibt während der Gangwechsel für jeweils 2 s konstant. Der Ladedruck darf ab 2 s nach den Gangwechseln vom Sollverlauf um höchstens 50 mbar abweichen. **Bild 4** zeigt den Testfall-Editor mit einem Ausschnitt des Testfalls. Dargestellt ist die Definition des Geschwindigkeitsprofils, das zusätzlich als Plot abgebildet ist.

5 Versionsverwaltung

Um das Ziel einer vollständigen Nachverfolgbarkeit der Tests zu erreichen, ist es wichtig, alle verwendeten und erzeugten Daten zu archivieren. Dies sind:

- das Simulationsmodell inklusive seiner Initialisierungsdateien
- die Testfälle
- die Testkonfiguration
- die automatisch generierten Testreporte.

Im SiLEST-Projekt werden deshalb Versionsverwaltungswerkzeuge unterstützt, zum Beispiel das kommerzielle Tool StarTeam von Borland oder das Open-Source-Programm Subversion. Vergleichbare Programme anderer Anbieter sind ebenfalls problemlos anbindbar.

6 Testverwaltung

Neben der reinen Verwaltung der Dateien ist eine gezielte Bereitstellung der Testresultate wichtig. Die Durchführung der einzelnen Tests ist zu dokumentieren, und deren Resultate sind zu präsentieren. Die Historie der Tests ist ersichtlich, wenn jede Durchführung den neuen Teststatus hinzufügt ohne den alten zu löschen. Somit lässt sich eine Übersicht über den aktuellen Status der einzelnen Tests und den gesamten Entwicklungsfortschritt gewinnen.

Testverwaltungsprogramme wie die im SiLEST-Projekt unterstützten Quality Center (TestDirector) der Firma Mercury oder das Open-Source-Tool Testmaster bieten diese Funktionalität. Wird die Testverwaltung zudem mit einem Anforderungsmanagement verbunden, ergeben sich Rückschlüsse bezüglich der Erfüllung der gestellten Entwicklungsanforderungen.

Bild 5 zeigt einen Screenshot von Quality Center mit den Tests der zweistufigen Aufladung. Dargestellt ist eine Teilansicht der Testresultate für die Testsuite. Erkenn-

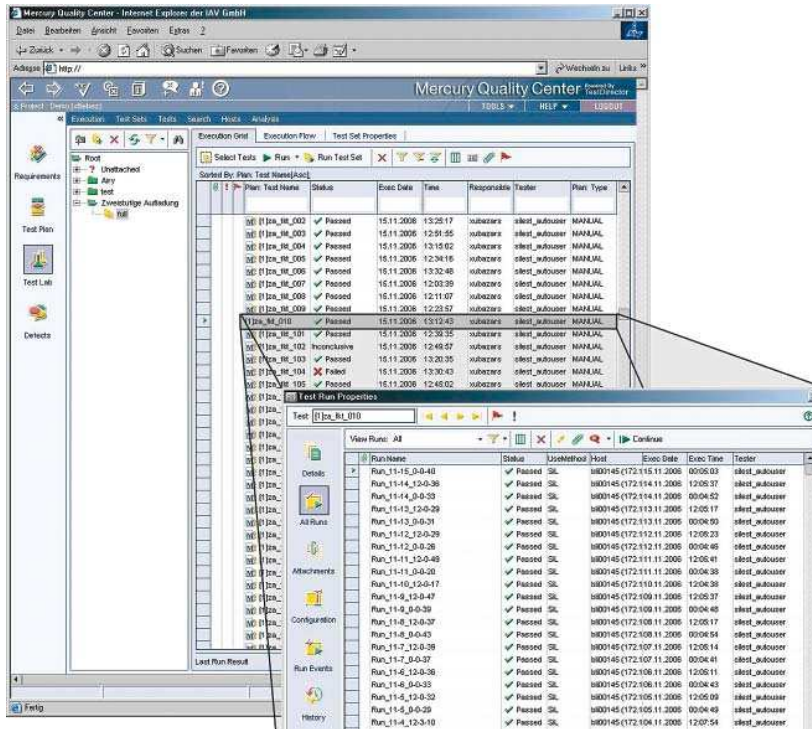


Bild 5: Quality Center mit Resultat-Ansicht und Historie eines Testfalls

Figure 5: Quality Center with result view and history of a test case

bar sind die einzelnen Testzustände (vgl. Kapitel 8) sowie der zugehörige Tester und das Datum der Durchführung. Für den vorgestellten Testfall ist zusätzlich ein Ausschnitt aus der Historie abgebildet. Dabei zeigt sich, dass der Test über die letzten Regressionen hinweg unverändert „Passed“ ist.

7 Testautomatisierung und Testablauf

Für Regressionstests ist eine automatisierte Ausführung wichtig, die nach dem Starten

ohne jede weitere Interaktion abläuft. Das zentrale Element dafür ist die im SiLEST-Projekt entwickelte Testablaufsteuerung [6]. Diese Java-Applikation koppelt diverse Standardtools (vgl. Kapitel 3, 5 und 6). Das offene Schnittstellenkonzept mit seiner Plug-in-Technik ermöglicht die Anbindung anderer beziehungsweise weiterer Programme.

Der prinzipielle Ablauf eines Testfalls im SiLEST-Prozess ist im Bild 6 dargestellt. Er wird durch ein Pythonscript in der Testablaufsteuerung festgelegt und ist dementsprechend weitestgehend flexibel anpassbar.

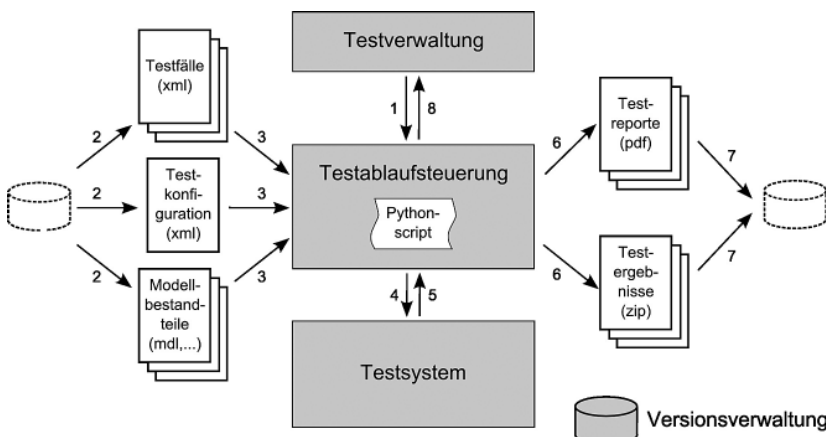


Bild 6: Testablaufsteuerung

Figure 6: Test sequence control

Der Aufruf der Testablaufsteuerung erfolgt für Regressionstests als geplante Windows-Task. Das hier vorgestellte Beispiel läuft täglich alle zwölf Stunden.

Die Testablaufsteuerung erhält beim Aufruf die zu verwendende Testkonfiguration und den Namen der einzusetzenden Testsuite. Aus der Testverwaltung besorgt sie sich zunächst die Links zu den Testfällen, die zur Testsuite gehören (1). Daraufhin werden diese gemeinsam mit den Modellbestandteilen aus der Versionsverwaltung ausgecheckt (2) und der Testablaufsteuerung übergeben (3). Diese setzt damit das Testsystem auf und stößt die Durchführung der Testfälle an (4). Dazu wird der Simulator geöffnet, das entsprechende Modell geladen und die Simulation initialisiert. Des Weiteren interpretiert die Testablaufsteuerung die Eingangssignale, Parameter und Fehlerbeschreibungen aus den Testfällen und setzt sie im Testsystem um. Schließlich wird die Simulation gestartet.

Nach Ablauf jedes Testfalls werden die für die Bewertung nötigen Informationen aus dem Testsystem abgefragt (5) und ausgewertet. Dies ist möglich, da in der Testfalldefinition neben den Eingangssignalen auch die erwarteten Ausgangssignale mit samt zulässiger Toleranzen beschrieben sind. Zur Dokumentation werden automatisch ein XML- sowie ein PDF-Testreport erstellt (6). Dessen Layout lässt sich problemlos an Kundenvorgaben anpassen.

Wesentliche Testergebnisse werden als ZIP-Datei gepackt und zusammen mit dem PDF-Report in die Versionsverwaltung eingekoppelt (7). Das Testresultat und die Links werden zuletzt in der Testverwaltung abgelegt (8). Im Anschluss wird der nächste Testfall bearbeitet.

8 Testzustände

Ein Testfall nimmt während seiner Durchführung verschiedene Zustände an, Bild 7. Mögliche Testresultate sind dabei „Passed“ und „Failed“ sowie „Inconclusive“ und „Aborted“. Während „Passed“ und „Failed“, ein klares Ergebnis der Testauswertung widerspiegeln, steht „Inconclusive“ für ein nicht entscheidbares Testresultat. Dies ist zum Beispiel dann der Fall, wenn ein von Vorbedingungen abhängiger Testfall nicht ausgeführt werden kann, weil die Vorbedingung zu Testbeginn nicht erfüllt ist. „Aborted“ zeigt an, dass ein Testlauf außerplanmäßig abgebrochen wurde. Das weist beispielsweise auf einen Absturz des Simulators hin.

9 Ergebnisse

Insgesamt erhalten 55 der 60 Testfälle aus der Testsuite der zweistufigen Aufladung das Resultat „Passed“, während vier fehlschlagen. Ein Test ist „Inconclusive“, weil er von einem fehlgeschlagenen Testfall abhängt.

Die Auswertung des untersuchten Testfalls erfolgt stückweise zwischen den Gangwechseln nach einer Übergangszeit von jeweils zwei Sekunden. **Bild 8** stellt den Verlauf des erzielten Regelfehlers des Ladedrucks dar. Es handelt sich dabei um einen Auszug aus dem generierten PDF-Testreport. Dargestellt ist die Teilauswertung für den Bereich nach dem letzten Gangwechsel. In diesem Beobachtungszeitraum darf die Regelabweichung für den Ladedruck eine Toleranz von ± 50 mbar nicht überschreiten. Es ergibt sich das Testresultat „Passed“, da sich das betrachtete Ausgangssignal im Testzeitraum komplett innerhalb der vorgegebenen Schranken bewegt.

Da der zeitliche Mehraufwand durch die Testablaufsteuerung gering ist, dominiert die Simulationsdauer die Gesamtdauer der Tests.

Für das komplexe Beispiel einer zweistufigen Aufladung zeigt sich der Nutzen des

hier vorgestellten Testprozesses. Eine große Anzahl von Testfällen kann automatisch und regelmäßig durchgeführt werden. Die Ergebnisse werden automatisch ausgewertet, für vertiefende Analysen stehen die archivierten Daten zur Verfügung. Die Zielstellung der nächsten Reglerversion ist es, alle Testfälle korrekt zu absolvieren und damit die gestellten Anforderungen vollständig zu erfüllen.

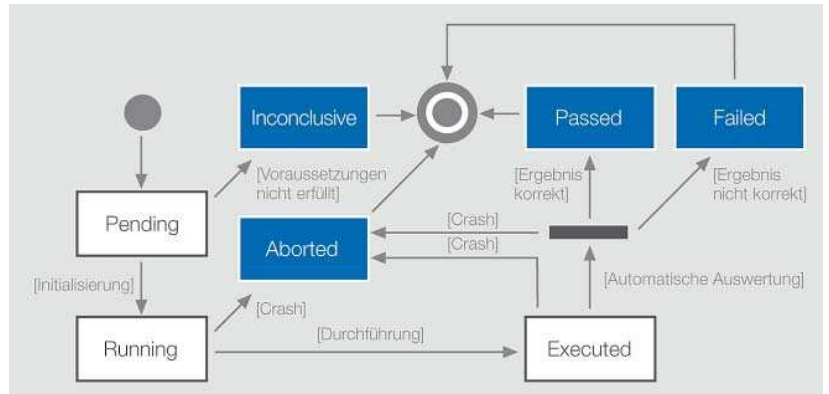


Bild 7: Zustände eines Testfalls während des Testlaufs

Figure 7: Statuses of a test case while a test is running

10 Fazit

Die Formalisierung von Tests und die Erstellung von Testfällen werden oftmals als erhöhter Aufwand angesehen. Dieses relativiert sich jedoch aufgrund der systematischeren Dokumentation und der vollständigen Nachverfolgbarkeit der Tests. Die SiLEST-Automatisierung erlaubt eine frühzeitige, häufige und intensive Testdurch-

Hier steht eine Anzeige.

 Springer

Hier steht eine Anzeige.

 Springer

führung parallel zum gesamten Entwicklungsprozess. Das ermöglicht bei nahezu unveränderten Kosten eine deutlich größere Testtiefe. Die Folge ist eine erhebliche Qualitätssteigerung der Tests und somit des entstandenen Produkts. Letztlich werden durch das frühzeitigere Auffinden von Fehlern enorme Kosten eingespart.

Der Einsatz entsprechender Tools, wie dem in SiLEST entwickelten Testfall-Editor, macht die Erstellung der Tests komfortabel und erfordert keine XML-Kenntnisse.

Mit SALib wurde eine Simulink-Bibliothek erstellt, mit der sich neben dem Nominalverhalten auch das Fehlerverhalten von Sensoren und Aktuatoren beschreiben lässt. Dadurch können Fehlfunktionen mechatronischer Komponenten besser als bisher bereits während der Funktionsentwicklung berücksichtigt werden. Auch dieses führt zu robusteren Gesamtsystemen und einer höheren Produktqualität.

Der im SiLEST-Projekt entwickelte Prozess für funktionale White-Box-Tests von Closed-Loop-Systemen verknüpft etablierte Standardtools und ermöglicht somit frühzeitiges, automatisches, wiederholbares und kostengünstiges Testen. Er ist durch die Verwendung von Testkonfigurationen unabhängig von der eingesetzten Testmethode. Damit steht ein flexibles Werkzeug für ein wesentlich verbessertes Testen des geschlossenen Regelkreisverhaltens in der Funktionsentwicklung zur Verfügung.

Literaturhinweise

- [1] Maibaum, O.; Rebeschies, S.: Test von adaptiven Softwaremechanismen zur Fehlerkompensation. Simulation und Test in der Funktions- und Softwareentwicklung für die Automobilelektronik, S.179-186. Renningen, Expert Verlag, 2005
- [2] Hiereth, H.; Prenninger, P.: Aufladung der Verbrennungskraftmaschine. Springer, 2003
- [3] IAV GmbH: http://www.iav.de/_downloads/de/handouts/powertrainmechatronik/THEMOS.pdf, 2006
- [4] DYNAware, TESIS: enDYNA Produktkatalog. TESIS DYNAware Technische Simulation Dynamischer Systeme GmbH, 2006
- [5] Liebezeit, Th.; Schumann, H.; Rebeschies, S.; Gühmann C.; Bazarsuren U.: XML-Format für den automatischen Software-in-the-Loop-Test. Moderne Elektronik im Kraftfahrzeug, S. 212-220. Renningen, Expert Verlag, 2006
- [6] Rebeschies, S.; Liebezeit, Th.; Bazarsuren, U.: Automated Closed-Loop Testing of Embedded Engine Control Software. 11. Software & Systems Quality Conferences 2006, 7. ICS Test, 2006

For an English version of this article, see

ATZ elektronik
WORLDWIDE

For information on subscriptions, just call us or send an E-Mail or fax:
Vieweg Verlag Postfach 1546 D-65173 Wiesbaden
Tel. +49 5241 80-1968 | E-mail: vieweg@abo-service.info



Bild 8: Automatisch generierter Testreport

Figure 8: Automatically generated test report

Hier steht eine Anzeige.

 Springer