Universelle Architektur zur Testautomatisierung

Methodik für Funktionstests mechatronischer Produkte

Für die Automatisierung des Funktionstests mechatronischer Produkte gibt es viele verschiedenartige Werkzeuge. Diese fokussieren stets die Lösung spezieller Aufgaben und unterscheiden sich in Bedienphilosophie, Syntax und Semantik. Somit entsteht bei den Herstellern eine Vielfalt von Werkzeugen, was die Effizienz und die Anwenderakzeptanz des automatisierten Testens negativ beeinflusst. Durch die Integration vorhandener Testkomponenten in flexibel nutzbare Testsysteme mit einheitlicher Bedienung lässt sich dieses Problem lösen. Die universelle Testsystem-Architektur dient dabei als Grundlage zur Realisierung von Testsystemen und ermöglicht, eine hohe Effizienz der Testautomatisierung zu erreichen.

SCHLAGWÖRTER Mechatronische Produkte / Funktionstest / Testautomatisierung / Testwerkzeug / Testsystem

Universal Architecture for Efficient Test Automation – Methodology for Functional Testing of Mechatronic Products

For automated functional testing of mechatronic products, a lot of various test tools are available. Typically, they focus on unique specific tasks and have different user interfaces, syntax and semantics. This results in a diversity of the applied tools and leads to negative impacts on user acceptance, costs, and thus the efficiency of automated testing. The integration of already existing tools into flexible test systems with a uniform user interface helps to solve these problems. The universal test system architecture can be used as a basis for the realization of test systems and to improve the overall efficiency of test automation.

KEYWORDS Mechatronic products / functional testing / test automation / test tool / test system

DMITRY KOROTKIY, KLAUS BENDER, Technische Universität München

Moderne mechatronische Produkte zeichnen sich durch Variantenanzahl und Funktionalität aus. Gerade die eingebettete Software als Bestandteil eines mechatronischen Produkts ermöglicht, eine hohe Flexibilität in den Produkten zu erreichen. Mit zunehmendem Softwareanteil steigen zwangsläufig die Komplexität und somit die Risiken der Produktentwicklung. Verschiedene Qualitätssicherungsmaßnahmen helfen, rechtzeitig Gefahren für den Produkterfolg zu erkennen und so die Fehlerfolgekosten in der Nutzungsphase des Produkts zu minimieren. Dem Funktionstest im Rahmen des System-Abnahmetests kommt deshalb eine besondere Bedeutung zu. Als abschließende Qualitätssicherungsmaßnahme prüft der Funktionstest die Erfüllung funktionaler Anforderungen beziehungsweise das Verhalten des mechatronischen Produkts gemäß den Kundenanforderungen im Lastenheft und wird für die Bewertung der Produktqualität und -funktionalität verwendet [1].

Die Steigerung der Produktqualität kann nur dann wirtschaftlich erfolgen, wenn die dazu erforderlichen Testprozesse automatisiert werden. Aufgrund verschiedener produkt- und herstellerspezifischer Anforderungen an den Funktionstest wurden zur Testautomatisierung diverse Werkzeuge entwickelt. Bei Herstellern mechatronischer Produkte sind solche Werkzeuge oft zahlreich und unsystematisch im Einsatz. Demzufolge führt die Verwendung mehrerer Werkzeuge neben hohen Kosten auch zu einem erheblichen personellen Aufwand durch das Vorhalten von Bedien-Know-how, was letztendlich die Wirtschaftlichkeit und Akzeptanz seitens der Anwender mindert. Als Resultat sinkt die Effizienz des automatisierten Testens.

Um die Effizienz des automatisierten Funktionstests zu erhöhen, ist es nötig, unter Wiederverwendung vorhandener Testkomponenten die Anzahl verschiedenartiger Werkzeuge zu reduzieren und neue flexible Testsysteme mit einheitlicher Bedienung zu entwickeln [3]. Mit dieser Aufgabe sind verschiedene Herausforderungen verbunden, die ohne eine grundlegende Systematik kaum zu lösen sind. In diesem Beitrag wird eine allgemeingültige Testsystem-Architektur präsentiert, die eine universelle Grundlage zur Realisierung von Testsystemen für den Funktionstest mechatronischer Produkte darstellt.

1. LÖSUNGSANSATZ

Die allgemeine Aufteilung von Testsystemfunktionen umfasst zwei wesentliche Komponenten: Testautomat und Testbett. Der Testautomat übernimmt die festgelegten Testaufgaben, wie zum Beispiel die Organisation und Verwaltung eines Testprojekts sowie die Ausführung und Auswertung von Testfällen. Das Testbett dient als Schnittstelle zwischen dem Testautomaten und dem Testobjekt. Dabei werden testrelevante Daten erzeugt oder erfasst und notwendige Umgebungen des Testobjekts nachgebildet. Die in Bild 1 vorgestellte universelle Testsystem-Architektur erweitert diese Einteilung von Testsystemen um fünf Funktionsebenen und ist in Anlehnung an das doppelkegel-förmige Automatisierungsmodell [2] aufgebaut.

Die Architekturebenen sind Testmanagement, Testausführung und -auswertung, Testbettsteuerung, Testobjektstimulation und -beobachtung sowie Testobjektumgebung. Sie sind lösungsneutral definiert und umfassen die wichtigsten Aufgaben bei der Testdurchführung. Dies ermöglicht die eindeutige funktionale Einordnung existierender Werkzeugkomponenten entlang der Testsystem-Architektur und dient gleichzeitig als universelle Grundlage zur Testsystemrealisierung.

Durch die kegelförmige Darstellung deutet die Testsystem-Architektur die Vielfalt der für das Testmanagement verwendeten Werkzeuge an, ebenso die Vielfalt der von den mechatronischen Produkten abhängigen Testanforderungen. Zusätzlich teilt die universelle Testsystem-Architektur alle Testaufgaben in organisatorische (dispositive) und ausführende (operative) Aufgaben ein und gibt eine Skalierung der vorhandenen Datenmenge und der erforderlichen Reaktionsgeschwindigkeiten bei der Testdurchführung an. Die organisatorischen Aufgaben sind in der Regel wenig zeitkritisch und haben

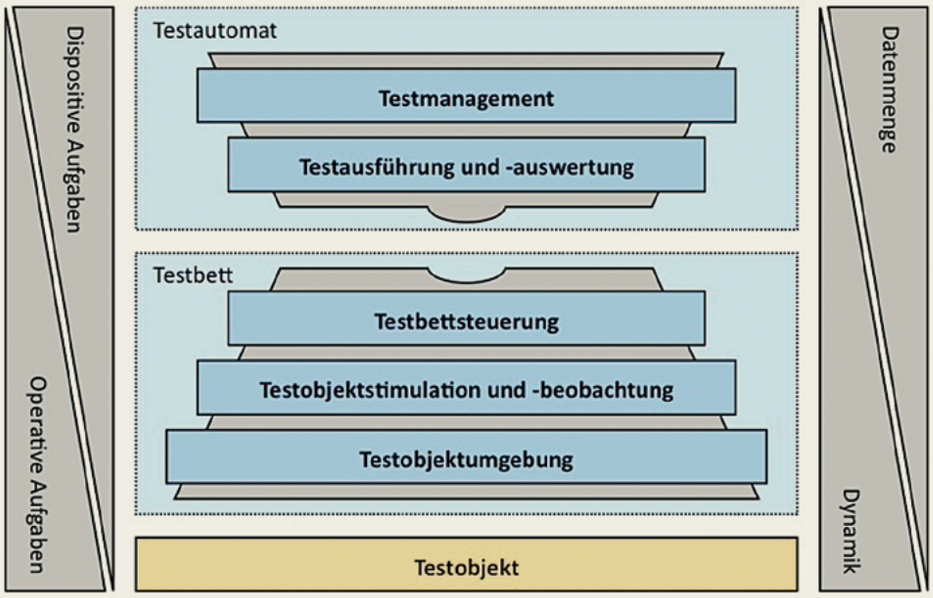

BILD 1: Universelle Testsystem-Architektur

typischerweise große Datenmengen zu bearbeiten, wie die Erstellung und Verwaltung von Testfällen und Testergebnissen auf der Testmanagementebene. Der begrenzende Faktor ist hierbei der Mensch. Bei der Stimulation und Beobachtung des Testobjekts liegen die geforderten Reaktionszeiten oft im Milli- und Mikrosekunden-Bereich, was eine besonders hohe Dynamik beim Datenaustausch zwischen Testobjekt und Testsystem erfordert.

2. BESCHREIBUNG DER ARCHITEKTUREBENEN

Die Testmanagementebene bildet die unmittelbare Schnittstelle zum Anwender und unterstützt diesen primär bei organisatorischen und verwaltenden Aufgaben im Testprozess. Hierzu zählen beispielsweise die Testfallerstellung, die Strukturierung und Parametrierung von Testfällen, die Zusammenstellung von Testszenarien sowie die Verwaltung von Testergebnissen und -protokollen.

Die Testsystemebene Testausführung und -auswertung hat die Aufgabe, Testfälle gemäß der Testfallbeschreibung auszuführen. Die implementierten Testfälle werden dabei in einzelne Testschritte zerlegt und zur endgültigen Ausführung an die Testbettsteuerung übergeben. Die resultierenden Reaktionen jedes Testschritts werden protokolliert und die im Testfall definierten Soll- und Ist-Bedingungen miteinander verglichen. Anhand des Vergleichs aus beobachtetem und erwartetem Verhalten des Testobjekts lässt sich das Testverdikt (PASS/FAIL/INCONCLUSIVE) bilden. Das schematische Verhalten eines Testsystems bei der Ausführung und Auswertung von Testfällen zeigt Bild 2.

Die Testbettsteuerung hat die Aufgabe, die korrekte Ausführung der einzelnen Testschritte sicherzustellen. Dazu beeinflusst sie gezielt die vorhandenen Testbettkomponenten zur Stimulation und Beobachtung des Testobjekts sowie der Testobjektumgebung, siehe Bild 3. Außerdem entkoppelt die Testbettsteuerung den Testautomaten vom Testbett, um die notwendigen Echtzeitanforderungen beim Funktionstest erfüllen zu können.

Die Erfassung und Generierung von Signalen zwischen Testobjekt und Testbett übernimmt die Testobjektstimulation und -beobachtungsebene. Die Stimulationskomponente setzt Befehle aus der Testbettsteuerungsebene in reale Signale zum Testobjekt um. Die Beobachtungskomponente empfängt die Daten vom Testobjekt und überträgt sie als Variable an die Testbettsteuerungsebene, siehe Bild 4. Dementsprechend dient diese Ebene primär dazu, verschiedene informationstechnische und physikalische Schnittstellen der Testobjektumgebung anzusprechen, um damit auf das Testobjekt zugreifen zu können.

Die Testobjektumgebung hat letztendlich die Aufgabe, das Testobjekt in den gewünschten dynamischen und somit testbaren Zustand zu überführen. Hierzu zählen vor allem die Stromversorgung sowie verschiedene testrelevante Prozessdaten, die nicht von der Testobjektstimulation und –beobachtungsebene erzeugt werden können beziehungsweise gemäß der Testspezifikation nicht erzeugt werden sollen. Die Prozessdaten werden direkt von Umgebungskomponenten entnommen oder, falls diese zum Testzeitpunkt nicht vorhanden oder verfügbar sind, durch passende Simulationsmodelle nachgebildet. Aus Sicht des Testobjekts darf sich hierbei kein Unterschied zwischen dem Datenaustausch mit der Produktivumgebung und der Einbindung in die Testobjektumgebung ergeben.

Testobjekte sind verschiedenartige softwareintensive mechatronische Produkte, also zum Beispiel Geräte oder Komponenten technischer Produktionsprozesse als Tei-

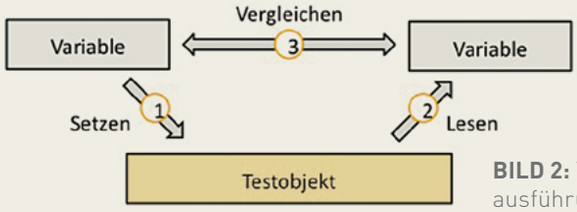

BILD 2: Testschritte der Testausführung und -auswertung

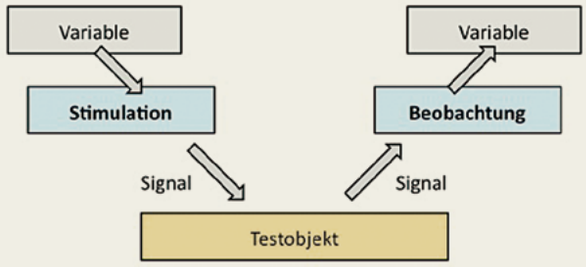

BILD 4: Stimulation und Beobachtung des Testobjekts

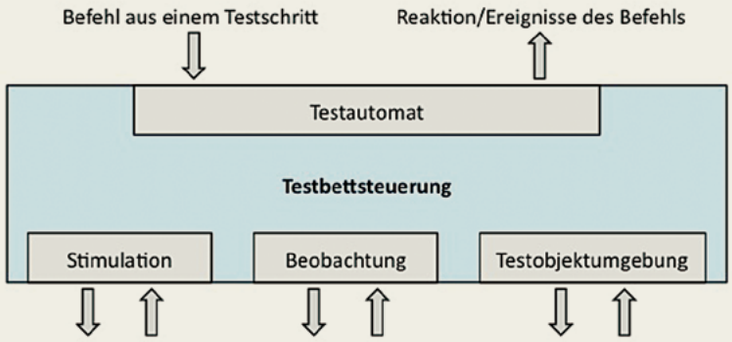

BILD 3: Schnittstelle der Testbettsteuerung

le eines dynamischen Gesamtsystems. Hierzu zählen intelligente Sensoren, Aktoren und eingebettete Steuerungen der Investitionsgüterindustrie, Automobilindustrie sowie Luft- und Raumfahrtindustrie.

3. ANWENDUNG DER UNIVERSELLEN TESTSYSTEM-ARCHITEKTUR

Das Vorgehen aus Bild 5 skizziert die Anwendung der universellen Testsystem-Architektur als Grundlage zur Realisierung konkreter Testsysteme:

1 | Die für den Funktionstest mechatronischer Produkte bisher verwendeten Werkzeuge sollen zunächst systematisiert werden. Dadurch lassen sich vorhandene Werkzeugfunktionalitäten und Verbesserungspotenziale bei der Realisierung von flexiblen und einheitlichen Testsystemen identifizieren. Die Systematisierung erfolgt durch die Einordnung vorhandener Werkzeuge beziehungsweise derer Funktionalität entlang der Architekturebenen.

2 | Die Schwerpunkte der Anforderungsanalyse liegen in der allgemeinen Beschreibung des Testobjekts, der Identifikation vorhandener Schnittstellen, der Feststellung der zu prüfenden Funktionalität und in der Ableitung der Anforderungen an das Testsystem. Bei der Analyse sollen vor allem die Fragen WAS? und WIE? des automatisierten Funktionstests geklärt sowie die Anforderungen bezüglich der Flexibilität und Einheitlichkeit des Testsystems berücksichtigt werden.
Als Nächstes werden mögliche Testszenarien betrachtet. Abhängig von gestellten Anforderungen müssen hierbei Testziele, Kriterien und Randbedingungen festgelegt, wie Qualitätsansprüche, Normen und Richtlinien, und entsprechende Testfälle definiert werden. Um das optimale Aufwand/Nutzen-Verhältnis beim Testen zu erreichen, sind die Testfälle schließlich zu priorisieren.
Die Anforderungsanalyse sollte anhand der universellen Testsystem-Architektur durch die sukzessive Festlegung und Verfeinerung dispositiver und operativer Aufgaben entsprechend den Ebenen des zu entwickelnden Testsystems erfolgen. Dies garantiert die Vollständigkeit der durchzuführenden Anforderungsanalyse.

3 | Durch die Gegenüberstellung der erzielten Ergebnisse der Systematisierung von Werkzeugen und der Anforderungsanalyse kann möglicherweise noch fehlende Werkzeugfunktionalität identifiziert werden. Dies dient als Grundlage zur Werkzeugauswahl, welche anhand eines geeigneten Vorgehens, wie zum Beispiel in [3], erfolgen kann. Alle Werkzeuge sollten bezüglich der Kriterien Verfügbarkeit, Funktionalität und Erweiterbarkeit analysiert werden.

4 | Nach Abschluss der Werkzeugauswahl kann ein Testsystem-Konzept erstellt werden. Die universelle Testsystem-Architektur dient dabei als Grundlage zur Konzeption. Entlang der Architekturebenen lassen sich die für das Testsystem verwendeten Werkzeuge strukturieren. Falls die Kriterien nicht oder nur teilweise von gewählten Testwerkzeugen erfüllt werden, müssen die Werkzeuge angepasst, andere beschafft oder neue implementiert werden. Ebenso sind die zum Testen notwendigen, aber nicht-vorhandenen oder nicht-verfügbaren Komponenten des Testsystems, wie Simulationsmodelle, zu implementieren. Abschließend werden die Werk-

zeuge und Komponenten zu einem homogenen Gesamttestsystem integriert. Der Aufwand für die Testsystemumsetzung lässt sich erheblich reduzieren, wenn eine geeignete Entwicklungs-Plattform und das Know-how bereit stehen.

5 | Die Testsystemrealisierung endet mit einer Bewertung. Das Ziel der Bewertung ist, Verbesserungsmöglichkeiten zu identifizieren. Hierbei soll neben der Funktionalität auch die Qualität des Testsystems geprüft werden. Bei der Qualitätsprüfung müssen die Gesamt- und Einzelfunktionen sowie das koordinierte Zusammenwirkung aller Werkzeuge und Komponenten verifiziert werden. Bei der Bewertung der Funktionalität wird das Testsystem mit bestehenden Anforderungen verglichen. Ferner sind das Testsystem und gegebenenfalls die Anforderungen anzupassen.

4. EVALUIERUNG UND ZUSAMMENFASSUNG

Um die Tragfähigkeit der universellen Testsystem-Architektur und die Anwendbarkeit des definierten Vorgehens zu bestätigen, wurden Testsysteme für den Funktionstest verschiedener mechatronischer Produkte realisiert. Unter Verwendung von zwei Werkzeugen (MoTest als Testautomat und LabVIEW als Testbett) konnten hierbei die geforderte Flexibilität und einheitliche Bedienung von Testsystemen umgesetzt werden. Die ausführliche Beschreibung von Beispielen ist in [3] und [4] präsentiert.

Zusammengefasst bietet die universelle Testsystem-Architektur eine durchgängige Methodik zur Realisierung von Testsystemen für den Funktionstest mechatronischer Produkte. Ihre klare Definition und Strukturierung zusammen mit den Vorgehensbeschreibungen ermöglichen eine einfache Anwendung im Testprozess. Unter Wiederverwendung vorhandener Werkzeuge unterstützt die universelle Testsystem-Architektur deren Integration in flexibel nutzbare Testsysteme mit einheitlicher Bedienung und erhöht dadurch die Effizienz des automatisierten Funktionstests.

MANUSKRIPTEINGANG
23. 02.2010

DANKSAGUNG

Der Beitrag stellt die wesentlichen Ansätze und Ergebnisse des Verbundforschungsprojekts TeSiM – Testsystem-Architektur für eingebettete Software in der Mechatronik vor. Das Projekt TeSiM wurde vom bayerischen Staatsministerium für Wirtschaft, Infrastruktur, Verkehr und Technologie im Rahmen des Förderprogramms „Informations- und Kommunikationstechnik" gefördert und vom VDI/VDE Innovation + Technik betreut. Das Projektkonsortium besteht aus den Firmen ascolab GmbH, Baumüller Anlagen-Systemtechnik GmbH & Co. KG, ITQ GmbH, Knowledge Department GmbH & Co. KG, Océ Printing Systems GmbH und Softing AG sowie dem Lehrstuhl für Informationstechnik im Maschinenwesen der TU München.

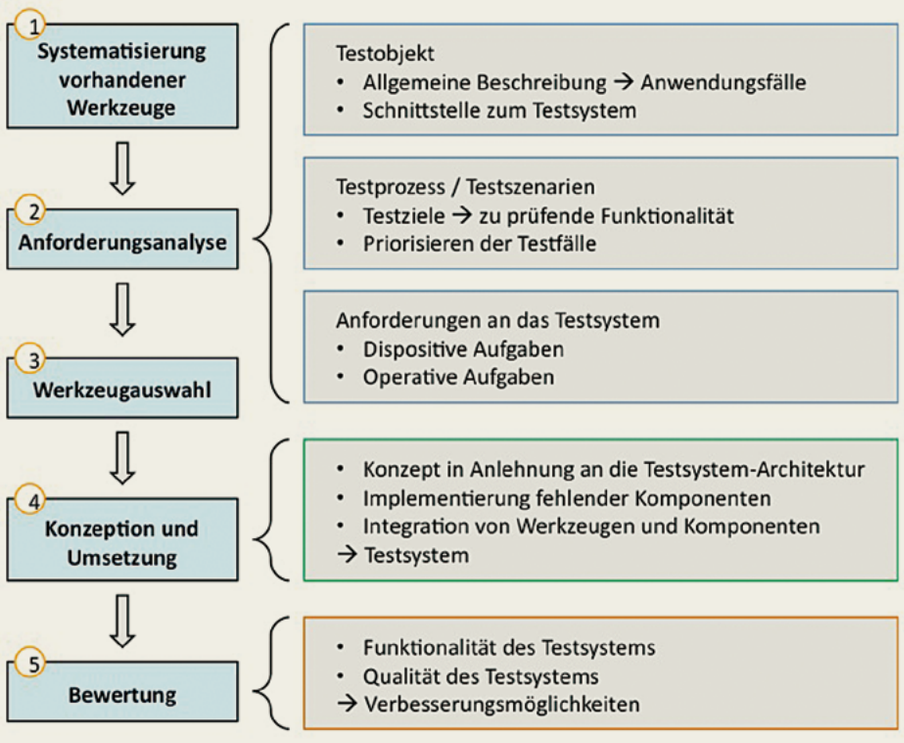

BILD 5: Vorgehen zur Realisierung von Testsystemen

REFERENZEN

[1] Bender, K. (Hrsg.): Embedded Systems – qualitätsorientierte Entwicklung, Berlin, Springer Verlag, 2005

[2] Vogel-Heuser, B.; Kegel, G.; Bender, K. und Wucher, K.: Global Information Architecture for Industrial Automation, Automatisierungstechnische Praxis – atp, Heft 1-2, 2009, S. 108-115

[3] Korotkiy, D.: Universelle Testsystem-Architektur in der Mechatronik, Göttingen, Sierke Verlag, 2010

[4] Bender, K.; Korotkiy, D. und Treffny, R.: Entwicklung eines Testsystems in der Praxis – Leitfaden zur Anwendung der universellen Testsystem-Architektur, Lehrstuhl itm, 2009 (http://www.itm.tum.de/downloads/TeSiM_Leitfaden.pdf)

[5] Korotkiy, D.; Dettmering, H.: Universal test system architecture in mechatronics – an approach for systematization of today's existing test tools, IEEE International Conference on Industrial Technology, ICIT 2009, 10.-13. Februar, Melbourne, Australia, 2009

AUTOREN

Dr. **DMITRY KOROTKIY** (geb. 1980) studierte Wirtschafts- mathematik an der Südrussischen Staatlichen Technischen Universität in Nowotscherkassk und ist seit Oktober 2003 wissenschaftlicher Mitarbeiter am Lehrstuhl für Informationstechnik im Maschinenwesen (itm) der Technischen Universität München.
2005 promovierte er am Moskauer Institut für Kraftfahrzeugwesen und Straßenbau. Am Lehrstuhl itm beschäftigt er sich im Rahmen seiner zweiten Promotion mit der Erforschung neuer Methoden zur Qualitätssicherung softwareintensiver interdisziplinärer Produkte und Systeme.

Technische Universität München
Lehrstuhl für Informationstechnik im Maschinenwesen,
Boltzmannstraße 15, D-85748 Garching b. München,
Tel. +49 89 289 164 48, E-Mail: korotkiy@itm.tum.de

Prof. Dr. **KLAUS BENDER** (geb. 1943) ist Emeritus des Lehrstuhls für Informationstechnik im Maschinenwesen (itm). Er gehört zahlreichen technisch-wissenschaftlichen Gesellschaften und Verbänden an. Seine Themenschwerpunkte sind Automation, Feldbus-Kommunikation und Mechatronik.

Technische Universität München
Lehrstuhl für Informationstechnik im Maschinenwesen,
Boltzmannstraße 15,D-85748 Garching b. München,
Tel. +49 89 289 164 00,E-Mail: bender@tum.de