Ein Framework für die Testautomatisierung bei Fahrer-Assistenz-Systemen

Vladimir Entin, Klaus Meyer-Wegener

Department Informatik, Lehrstuhl für Informatik 6 (Datenbanksysteme)
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstraße 3, 91058 Erlangen
vladimir.entin@informatik.uni-erlangen.de
klaus.meyer-wegener@informatik.uni-erlangen.de

Abstract: In dieser Arbeit wird ein Framework für die Testautomatisierung bei Fahrer-Assistenz-Systemen vorgestellt, das im Rahmen eines INI.FAU-Projekts (www.ini.fau.de) prototypisch realisiert worden ist. Es wird auf zwei Probleme der Testautomatisierung eingegangen: Testprozessbeschreibung und Schaffung einer geeigneten Architektur zur Testprozessausführung. Der vorgestellte Ansatz ist ebenfalls auf andere Bereiche anwendbar, in denen eine heterogene Tool-Landschaft und eine sehr große Menge aufgezeichneter Messwerte als Test-Eingangsdaten eingesetzt werden.

1 Testvorgänge bei der Entwicklung von Fahrer-Assistenz-Systemen

Die modellbasierte Funktionsentwicklung in der Automobilindustrie hat in den letzten Jahren breite Akzeptanz gefunden. Etablierte Entwicklungs- und Testverfahren wie Modell-basierte Simulation mit künstlich erzeugten Eingangsdaten sind jedoch nicht imstande, reale Testfahrten vollständig zu ersetzen. Die zu entwickelnde Funktion wird dafür in Form eines C-Codes auf die Steuereinheit übertragen. Während anschließender Testfahrt beobachtet der Funktionsentwickler die Verhaltensweise seines Algorithmus. Im Bereich Vorentwicklung von Fahrer-Assistenz-Systemen (FAS, z.B. Adaptive Cruise Control) wird der Funktionscode nicht ausschließlich aus Modellen abgeleitet, sondern es werden komponentenbasierte Entwicklungsumgebungen zu seiner Erzeugung eingesetzt. Dazu gehört beispielsweise das Automotive Data And Time Triggered Framework (ADTF). Es erlaubt es dem Benutzer, die entwickelten Software-Module in die Bus- und Hardware Architektur von Versuchsfahrzeugen zu integrieren [Sc07]. Die Testfahrten stellen einen hohen Kostenfaktor dar und können deshalb nicht jedes Mal durchgeführt werden, wenn ein neuer Software-Stand der zu testenden Funktion vorliegt. Aus diesem Grund werden Messwerte wie beispielsweise CAN-Botschaften im Binärformat abgespeichert. Die Größe einer solchen Datei kann bis zu siebzig Gigabyte pro Stunde Messfahrt betragen. Grund dafür sind Videoaufzeichnungen, die zum Zweck der Bildverarbeitung unkomprimiert abgespeichert werden müssen. Die Messdateien, beziehungsweise zeitliche Ausschnitte (Szenarien) davon, werden in einer Szenarien-Datenbank verwaltet [En06]. Durch die Aufzeichnung aller Messwerte ist der

Funktionsentwickler imstande, Systemtests [Li06] nach dem Blackbox-Prinzip im Labor durchzuführen. Die charakteristische Eigenschaft solcher Tests besteht in der Art der Testfall-Definition. Diese geschieht, indem eine Menge von Fahrmanövern festgelegt wird, die während einer Messfahrt bei bestimmten Wetterbedingungen und Fahrbahnzuständen abgefahren werden müssen. Bei den klassischen Systemtests setzt man hingegen auf automatisierte oder manuelle Ableitung von Testfällen samt zugehöriger Test-Eingangsdaten von der Systemspezifikation [Li 06]. Zur Testdurchführung sind Konfigurationsschritte notwendig, die mit steigender Anzahl an Messdateien einen hohen manuellen Aufwand für den Funktionsentwickler bedeuten. Ohne einen automatisierten Ansatz ist also ein umfangreicher Systemtest in der Vorentwicklung von Fahrer-Assistenz-Systemen (FAS) kaum möglich.

2 Realisierung der Testautomatisierung

Zur Realisierung des automatisierten Testablaufs sind grundsätzlich zwei Probleme zu lösen. Zum einen muss eine flexible Prozessbeschreibung geschaffen werden, die es erlaubt, die unterschiedlichsten am Testablauf beteiligten Anwendungen zu parametrisieren und die Datenflüsse zwischen diesen darzustellen. Zum anderen muss eine geeignete Softwarearchitektur entworfen werden, die die Ausführung des im ersten Schritt beschriebenen Testprozesses erlaubt.

2.1 Testprozessbeschreibung

Gerade bei einem automatisierten Ablauf mehrerer Anwendungen, die Daten untereinander austauschen, ist es notwendig, im Vorfeld zahlreiche Parametrisierungen vorzunehmen. Oftmals bestehen gewisse Abhängigkeiten zwischen den Konfigurationsparametern, die eine schrittweise Vorgehensweise bei der Parametrisierung erfordern. Beispielsweise können keine Szenarien-IDs aus der Datenbank automatisiert ausgelesen werden, wenn man davor die Zugangsparameter wie beispielsweise Port oder Serveradresse für die letztere nicht spezifiziert hat. Außerdem weisen unterschiedliche Anwendungen verschiedenste Parametersätze auf, weshalb vor der Parametrisierung zunächst festgelegt werden muss, welche Anwendungen am Testvorgang beteiligt sind. Um der variablen Testprozess-Zusammensetzung gerecht zu werden, aber auch Integrierbarkeit diverser Anwendungen in den automatisierten Testvorgang zu gewährleisten, wurde der Ansatz der Process-Driven Architecture (PDA) gewählt [Mu07]. PDA unterscheidet zwischen domänenspezifischer Prozessmodellierung, systemspezifischer Prozessmodellierung und plattformspezifischer Modellierung. Durch schrittweise Abbildung domänenspezifischer Modelle auf systemspezifische und anschließend systemspezifischer auf plattformspezifische wird der Testprozess soweit beschrieben, dass er vom entsprechenden Framework ausgeführt werden kann (siehe Abschnitt 2.2). Bei der domänenspezifischen Prozessmodellierung geht es darum, das Testprozessmodell mithilfe von abstrakten Typen (z.B. Entwicklungsumgebung, Testdaten-Speicherort) sowie deren Verbindung untereinander

durch Datenflüsse darzustellen. Bei der Abbildung auf das systemspezifische Prozessmodell müssen für jeden abstrakten Typ die Anwendungsbezeichner angegeben werden. Anschließend geht es darum, das systemspezifische Prozessmodell zu erstellen, indem die anwendungsabhängigen Parametrisierungssätze teilweise in mehreren Schritten angegeben werden. Letztlich muss das systemspezifische Prozessmodell um plattformspezifische Eigenschaften ergänzt werden. So müssen u.a. für jede Anwendung Framework-Plugins samt Pin-Verbindungen (siehe Abschnitt 2.2) angegeben werden. Somit entsteht das plattformspezifische Prozessmodell. Anschließend wird das plattformspezifische Modell vom Testframework [Iß07] zur Ausführung gebracht.

2.2 Framework-Architektur zur Testprozessausführung

Da der automatisierte FAS-Testprozess weitgehend ohne Interaktion mit dem Entwickler verläuft und auf Dateiaustausch zwischen den beteiligten Anwendungen basiert, wurde als Architekturgrundlage für das Framework das Pipes-and-Filters-Muster gewählt [Bu00], wobei Filters als Plugins realisiert worden sind. Dadurch wird zum einen die bereits erwähnte Integrierbarkeit der Anwendungen in den Testprozess garantiert. Zum anderen wird eine variable Testprozess-Zusammensetzung ermöglicht, indem bestimmte Plugins mehrfach in unterschiedlichen plattformspezifischen Prozessmodellen verwendet werden können. Das Framework implementiert drei wesentliche Mechanismen (Abb. 1).

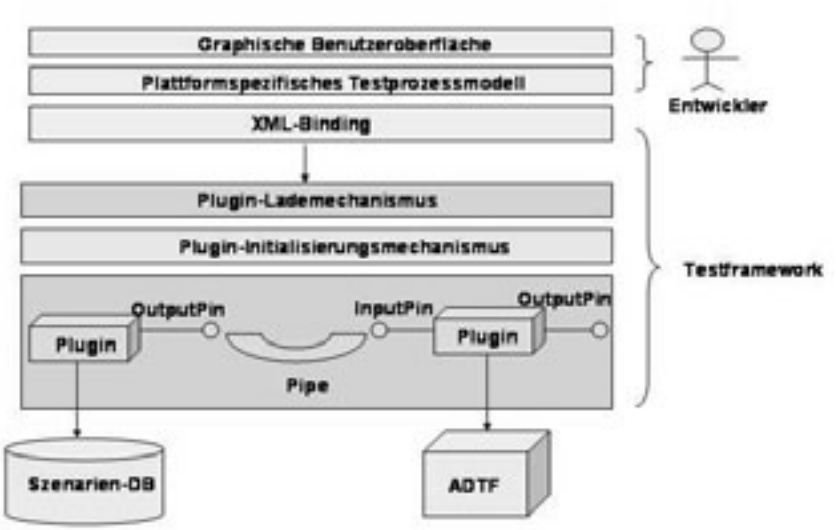

Abb. 1 Testframework-Architektur

Zunächst muss der Entwickler durch eine entsprechende graphische Oberfläche in mehreren Schritten das plattformspezifische Testprozessmodell erzeugen. Dieses wird als XML-Dokument dem Testframework übergeben. Letzteres liest über den XML-Data-Binding-Mechanismus [Xb07] zunächst die Informationen über die beteiligten Plugins aus. Anschließend werden die benötigten Plugins als Assemblies geladen. Bei der

Plugin-Initialisierung geht es darum, die Pipes zwischen den Plugins zu erzeugen. Im letzten Schritt wird die eigentliche Ausführung gestartet. In dieser werden Schritt für Schritt über entsprechende Plugins die am Testvorgang beteiligten Anwendungen angesteuert.

3 Zusammenfassung und Ausblick

Es wurde ein bei AUDI AG prototypisch umgesetztes Framework zur Testautomatisierung bei FAS in der Vorentwicklung vorgestellt. Dieses erlaubt, unterschiedliche Konstellationen von Anwendungen zu einem automatisierten Testvorgang zu bringen. Aktuell wird das Framework im Bereich der Bildverarbeitung insbesondere bei der Verfolgung von Fahrzeugen verwendet. Die ersten Erfahrungen haben gezeigt, dass die Auswahl einer minimal notwendigen Szenarienmenge entscheidend für die Akzeptanz des Frameworks ist. Außerdem hat sich eine formularbasierte graphische Oberfläche zur Erstellung der Testprozessbeschreibung als wenig flexibel erwiesen. In der Zukunft soll eine auf Kombination aus Workflow und Formular basierende graphische Oberfläche geschaffen werden. Der vorgestellte Ansatz ist keineswegs auf den automobilen Bereich beschränkt. Der Einsatz eines solchen Frameworks wäre überall dort denkbar, wo eine heterogene Tool-Landschaft und eine sehr große Menge aufgezeichneter Messwerte zum Testen und Entwickeln von Software eingesetzt werden, wie etwa in der Medizin.

Literaturverzeichnis

[Bu00] Buschmann, F.: Pattern-orientierte Software-Architektur. Addison-Wesley. München, 2000.

[En06] Entin, V.: Datenbank zur Verwaltung von Fahrszenarien. Diplomarbeit. Universität Erlangen-Nürnberg, Audi Electronics Venture GmbH. Gaimersheim, 2006.

[Iß07] Ißbruecker, M.: Konzeption und Erstellung einer Umgebung zur Anfertigung und Ausführung von Testprozessen für Fahrer-Assistenzsysteme. Diplomarbeit. Westsächsische Hochschule. Zwickau, 2007.

[Li06] Liggesmeyer, P.: Software-Qualität. Spektrum-Verlag Heidelberg. Berlin, 2002.

[Mu06] Müller, S.: Modellbasierte IT-Unterstützung von wissensintensiven Prozessen. Dissertation. Universität Erlangen-Nürnberg, Erlangen 2007.

[Sc07] Schabenberger, R.: ADTF: Framework für Fahrerassistenz-und Sicherheitssysteme. Audi Electronics Venture GmbH. VDI 13. Internationaler Kongress “Elektronik im Kraftfahrzeug” Baden Baden, 2007.

[Xb07] Bourret, R.: XML Data Binding Resources. www.rpbourret.com/xml/XMLDataBinding.htm Letzter Aufruf: 16.12.2007.