Einsatz automatischer Testvektorgenerierung im modellbasierten Test

Sadegh Sadeghipour, Meike Lim

IT Power Consultants
Gustav-Meyer-Allee 25
13355 Berlin
sadegh.sadeghipour@itpower.de
meike.lim@itpower.de

Abstract: In der Welt der modellbasierten Softwareentwicklung hat die automatische Testvektorgenerierung die Forschungsphase bereits durchschritten. Dementsprechend sind einige Tools für die Generierung von Testvektoren auf dem Markt der modellbasierten CASE Tools verfügbar. Dieser Beitrag beschreibt verschiedene Anwendungsszenarien für die automatische Generierung von Testvektoren auf Modellebene.

1 Einleitung

Die Methoden und Tools für die automatische Generierung von Testvektoren zur Überdeckung des Programmcodes sind derzeit noch Gegenstand der Forschung. Folglich werden sie noch nicht in der industriellen Praxis der Softwareentwicklung eingesetzt. In der Welt der modellbasierten Softwareentwicklung hat jedoch die automatische Testvektorgenerierung die Forschungsphase bereits durchschritten. Einige Tools für die Generierung von Testvektoren, die auf einer Auswahl von effizienten Technologien basieren, sind auf dem Markt der modellbasierten CASE Tools verfügbar. ATG [OS04], Conformiq Test Generator [Ve04], beide basierend auf Statecharts, und Reactis Tester [Re04], basierend auf Simulink/Stateflow sind drei Beispiele.

Reactis hat sich bereits seinen Weg in die industrielle Praxis gebahnt und wird von Entwicklern und Testern der automobilen Steuergeräte-Software verwendet. Nach einer Analyse der Modellstruktur werden zunächst zufällig Testvektoren generiert. Anschließend wird die Methode ‚Guided Simulation' angewandt, bei der Testvektoren generiert werden, die zu noch nicht überdeckten Zielen führen. Zu diesem Zweck werden verschiedene allgemeine Überdeckungskriterien, wie Decision und Condition Coverage, sowie Simulink- oder Stateflow-spezifische Kriterien, wie Überdeckung von bedingten Subsystemen oder Zustands und Transitionsüberdeckung, definiert.

Das Hauptproblem bezüglich des Einsatzes der automatischen Testvektorgenerierung ist die Frage der Testauswertung, d.h. die Frage, ob die mit den erzeugten Testvektoren

durchgeführten Tests erfolgreich sind (Problem des Testorakels). Um die Tests auswerten zu können, müssen die automatisch generierten Testvektoren funktional interpretiert werden. Im Allgemeinen ist dies jedoch eine komplexe Aufgabe. Deshalb ist die Hauptfrage nach Ansicht des Testingenieurs und des Verantwortlichen für die Qualitätssicherung bezüglich der automatischen Generierung von Testvektoren auf folgende Weise zu stellen: Reduziert der Einsatz von automatischen Testvektorgeneratoren den Aufwand des modellbasierten Testprozesses? Oder präziser formuliert: Wie sollte ein Generator von automatischen Testvektoren innerhalb des modellbasierten Testprozesses eingesetzt werden, um einen maximalen Nutzen zu erreichen? Dieser Beitrag beschreibt verschiedene Anwendungsszenarien für den Einsatz einer automatischen Testvektorgenerierung auf Modellebene zur Steigerung der Effizienz und zur Verbesserung der Qualität des Testprozesses.

2 Einsatzszenarien für automatische Testvektorgenerierung

In diesem Abschnitt werden die folgenden Einsatzszenarien für die automatische Testvektorerzeugung vorgestellt.

1. Back-to-Back-Test und Regressionstest
2. Effektive Teststrategie
3. Entwicklungsbegleitende Tests
4. Funktionale Tests

Die funktionale Interpretation der automatisch generierten Testvektoren und folglich das Orakelproblem wird in den ersten zwei Szenarien vermieden. Im dritten Fall muss eine ‚leichte' Version des Orakelproblems gelöst werden. Im letzten Szenario wird das Testverhalten vor der Testdatenerzeugung spezifiziert. Die automatisch generierten Testvektoren sind somit in der Spezifikation des Testverhaltens eingebettet. Die Testverhaltensspezifikation stellt deshalb gleichzeitig die funktionale Interpretation der generierten Testvektoren dar.

2.1 Back-to-Back- und Regressionstests

Back-to-Back-Tests prüfen die Äquivalenz zwischen den verschiedenen Repräsentationsformen des Testobjekts, z.B. zwischen dem Modell und dem daraus generierten Programmcode. Regressionstests stellen sicher, dass die am Testobjekt vorgenommenen Modifikationen die schon korrekt umgesetzten Funktionalitäten älterer Versionen ebenfalls erfüllen. In beiden Fällen dienen die bestätigten Ergebnisse früherer Tests als Referenzdaten.

Ein Back-to-Back- oder Regressionstest wird als erfolgreich bewertet, wenn der Nachweis der (partiellen) Äquivalenz zwischen dem Testobjekt und einer früheren schon geprüften Version desselben (Referenzobjekt) erbracht wird. Ein solcher Nachweis erfolgt durch den Vergleich der Outputs des Testobjekts mit dem Referenzobjekt, wobei sowohl Test- als auch Referenzobjekt mit den gleichen Inputs ausgeführt werden. Diese

durch die Testfälle bestimmten Inputs brauchen deshalb nicht individuell interpretiert zu werden. Jedoch müssen sie so umfassend sein, dass die beanspruchte (partielle) Äquivalenz zwischen dem Testobjekt und dem Referenzobjekt im Falle von erfolgreichen Tests gerechtfertigt ist. Eine automatisch generierte Testsuite zur Überdeckung struktureller Elemente des Modells wäre also durchaus für Back-to-Back- und Regressions-Tests geeignet (Abbildung 1). Das Erreichen der strukturorientierten Überdeckungskriterien garantiert die Breite und die Tiefe des Tests und reduziert gleichzeitig den Aufwand für die Ermittlung von Testfällen.

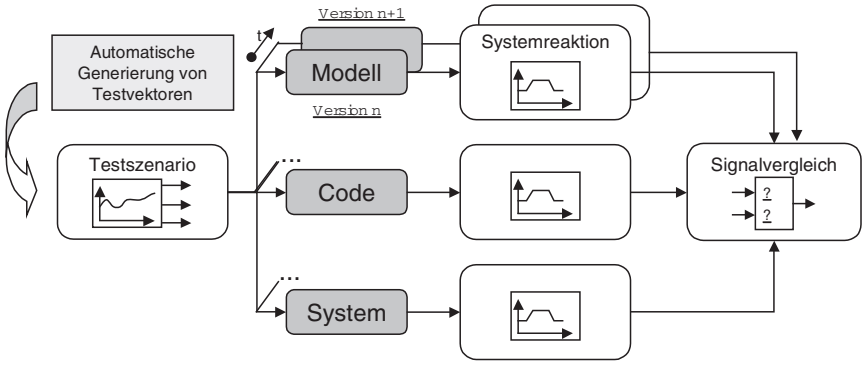

Abbildung 1: Einsatz der automatischen Generierung von Testvektoren bei Back-To-Back- und Regressionstests

Die Auswertung von Back-to-Back- und Regressionstests, d.h. der Vergleich der Outputs des Testobjekts mit den Referenzdaten, kann auch automatisiert werden. Beispielsweise ist dies mit dem Tool MEval möglich, das Zeitverschiebungen und Amplitudenabweichungen erkennt und getrennt voneinander analysiert [CSW05].

2.2 Effektive Teststrategie

Eine Teststrategie ist effektiv, wenn die auf dieser Strategie basierenden Tests Fehler im Testobjekt identifizieren [Be95]. Um die Wahrscheinlichkeit der Fehlerentdeckung zu erhöhen, wird im Allgemeinen eine geeignete Kombination von einzelnen Testmethoden als eine effektive Teststrategie erachtet.

Eine bekannte effektive Teststrategie innerhalb der konventionellen Softwareentwicklung ist die Kombination des funktionalen und des strukturorientierten Tests [Gr95]. Diese Strategie wurde dem modellbasierten Test angepasst [Co04]. Demnach wird nach der Spezifikation funktionaler Testfälle ein Modellüberdeckungskriterium bestimmt und die Modellüberdeckung während der funktionalen Testdurchführung gemessen. Im nächsten Schritt werden neue Testfälle spezifiziert, welche die noch nicht überdeckten Modellelemente überdecken. Genau für diese Aufgabe kann ein automatischer Testvektorgenerator eingesetzt werden, um die noch nicht überdeckten Modellelemente zu überdecken (Abbildung 2). Im Vergleich zu einer manuellen Ermittlung von Testvektoren wird der Testaufwand dadurch erheblich reduziert. Der funktionale Erfolg der Tests und deren funktionale Bedeutung kann nun vernachlässigt werden, da die funktionalen Tests bereits in der ersten Stufe definiert und durchgeführt wurden. Die

bereits in der ersten Stufe definiert und durchgeführt wurden. Die automatisch generierten strukturellen Testfälle werden als erfolgreich beurteilt, wenn keine offensichtlichen Fehler, wie z.B. Overflows oder Deadlocks, während des Modelltests auftreten.

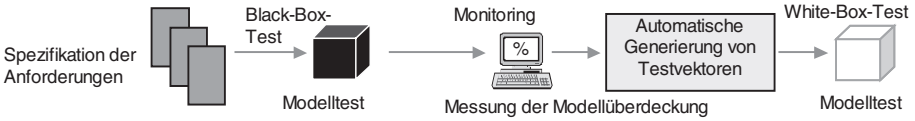

Abbildung 2: Einsatz der automatischen Testvektorgenerierung innerhalb einer effektiven modellbasierten Teststrategie

2.3 Entwicklungsbegleitende Tests

Im Rahmen einer Bottom-Up Entwicklungsstrategie sollten die Entwickler unmittelbar die kleinen, von ihnen entwickelten prüfbaren Module testen. Diese Module entsprechen einzelnen Algorithmen oder Funktionen der gesamten Software, die zu entwickeln ist. Aufgrund beschränkter Zeit und kurzfristiger Liefertermine verzichten Entwickler oft auf solche ‚Entwicklungsbegleitenden Tests' und akzeptieren damit zwangsläufig eine geringe Qualität des entwickelten Teils der Software. Die an dieser Stelle eventuell verursachten Fehler werden dann gegebenenfalls später während des Modul- oder Systemtests entdeckt, was jedoch höhere Kosten für ihre Behebung verursacht.

Automatische Testvektorgeneratoren unterstützen Entwickler bei der Generierung von Testvektoren für kleine prüfbare Einheiten. Da die Entwickler am besten mit der Struktur der von ihnen entwickelten Module vertraut sind, können sie die automatisch generierten Testvektoren leicht interpretieren und die entsprechenden Tests zuverlässig beurteilen.

2.4 Funktionale Tests

Die automatische Testvektorgenerierung kann auch für funktionale Tests, die auf der Anforderungsspezifikation der zu entwickelnden Software basieren, verwendet werden. Dies ist jedoch nur möglich, wenn der Testvektorgenerator dem Benutzer die Möglichkeit bietet, das Testverhalten zu modellieren und parallel zum Testobjekt zu simulieren (Abbildung 3). Reactis beispielsweise bietet dem Anwender mit dem ‚validator objective' eine solche Möglichkeit [Re04]. Natürlich ist in diesem Fall eine intellektuelle Arbeit des Testingenieurs erforderlich, um das Testverhalten aus der Anforderungsspezifikation abzuleiten. Jedoch ist dieser Aufwand inhärent zum funktionalen Testen. Der nächste Schritt, nämlich die Definition von Testvektoren zur Überdeckung des Testverhaltens, die im Falle der manuellen Durchführung sehr aufwändig ist, wird vom automatischen Testvektorgenerator übernommen.

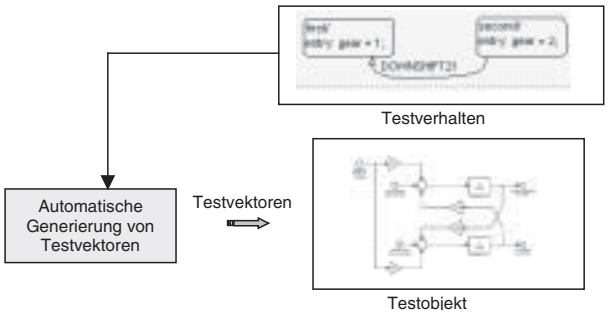

Abbildung 3: Einsatz der automatischen Testvektorgenerierung für funktionale Tests

3 Zusammenfassung

In diesem Beitrag wurden vier Szenarien für den Einsatz modellbasierter Testvektorgenerierung beschrieben. Vor allem ermöglicht die automatische Testvektorgenerierung eine erhebliche Zeit- und Aufwandsersparnis bei Regressions-, Back-to-Back-Tests. Im Falle der Existenz einer Qualitätsanforderung bezüglich des Erreichens eines vorgegebenen Modellüberdeckungsgrades ist das Einsetzen des Testvektorgenerators gemäß dem zweiten Einsatzszenario (effektive Teststrategie) sinnvoll. Die Spezifikation des Testverhaltens gemäß dem vierten Einsatzszenario eignet sich insbesondere für die sicherheitsrelevanten Funktionen, bei denen eine hohe Testtiefe zu erreicht ist.

Die in diesem Artikel präsentierten Ergebnisse wurden teilweise im Rahmen des Projekts IMMOS erarbeitet, das vom Bundesministerium für Bildung und Forschung gefördert wird (Projektnummer 01ISC31).

Literaturverzeichnis

[Be95] Beizer, B.: Black-Box Testing – Techniques for Functional Testing of Software and Systems. John Wiley & Sons, New York, 1995.

[Co04] Conrad, M.: Modell-basierter Test eingebetteter Software im Automobil – Auswahl und Beschreibung von Testszenarien. Dissertation, Deutscher Universitäts-Verlag, Wiesbaden, 2004.

[CSW05] Conrad, M.; Sadeghipour, S.; Wiesbrock, H.-W.: Automatic Evaluation of ECU Software Tests. In Proc. SAE 2005 World Congress, Detroit, USA, 2005.

[Gr95] Grimm, K.: Systematisches Testen von Software - Eine neue Methode und eine effektive Teststrategie, Dissertation, GMD-Bericht Nr. 251, R. Oldenburg Verlag, München 1995.

[OS04] OSC – Embedded Systems AG: ATG (Produktinformation). http://www.osc-es.de/products/en/atg.php, 2004.

[Re04] Reactive Systems, Inc.: Reactis® User's Guide V 2004.2. www.reactive-systems.com, 2004.

[Ve04] Verifysoft Technology GmbH: Conformiq Test Generator (Produktinformation). http://www.verifysoft.com/de_conformiq_testgenerator.html, 2004.