Modellbasierte Testgenerierung aus Spezifikationen mit parallelem Verhalten

Jan Krause, Christian Diedrich

ifak - Institut für Automation und Kommunikation e.V. Magdeburg
Werner-Heisenberg-Str. 1
39106 Magdeburg
jan.krause@ifak.eu, christian.diedrich@ifak.eu

Abstract: In diesem Beitrag wird eine Methode zur Generierung von Testfällen aus modellbasierten Verhaltensspezifikationen vorgestellt, wobei auch die Spezifikation von parallelem Verhalten unterstützt wird. Zur Realisierung werden etablierte Verfahren der Petri-Netz-Theorie und der constraintbasierten Programmierung benutzt. Auf der Grundlage eines entwickelten einfachen farbigen Petri-Netz-Modells (SPeNAt) können damit Testfälle mit parametrierbarem Abdeckungsgrad generiert werden, wobei auch sehr hohe, insbesondere für die Entwicklung sicherheitskritischer Systeme relevante, Abdeckungsgrade (wie z. B. „alle möglichen alternativen Pfade“, MC/DC an den Bedingungen, etc.) garantiert werden können.

1 Einleitung

In den letzten Jahren wurde eine Reihe von Methoden zur Testgenerierung entwickelt. Grundlage dieser Methoden ist immer eine formale Spezifikation bzw. ein formales Spezifikationsmodell. Meist arbeiten diese entwickelten Testgenerierungsmethoden nicht auf dem gesamten Zustandsraum des Spezifikationsmodells, wodurch auch die Testgenerierung für komplexe Modelle ermöglicht wird. Im Regelfall wird das Modell mit Eingaben gefüttert und die Modellreaktion wird protokolliert, wodurch sich das gewünschte Verhalten bzw. die Testfälle ergeben. Die Modelleingaben werden durch genetische Algorithmen (z. B. in [PSO08]), intelligente Heuristiken (z. B. in [SW07]) und/oder durch Methoden der constraintbasierten Programmierung (z. B. in [PL01]) ermittelt. Diese Prozedur wird solange fortgesetzt, bis die gewünschten, vorher spezifizierten Testziele (z. B. Abdeckungsgrad der Spezifikation) erreicht werden. Allerdings können mit diesen Methoden meist nur strukturelle Spezifikationsabdeckungen wie „alle Zustände“ bzw. „alle Transitionen“ erreicht werden. Höhere Abdeckungsgrade (z. B. „alle möglichen alternativen Pfade“) können nicht garantiert werden, da nicht auf dem gesamten Zustandsraum des Spezifikationsmodells operiert wird. Gerade in der Entwicklung sicherheitsrelevanter Systeme ist diese erreichte Spezifikationsabdeckung mitunter zu gering. Auch sind Testgenerierungsmethoden, die explizit die Spezifikation parallelen Verhaltens (auch auf Komponentenebene) unterstützen, den Autoren nicht bekannt. Durch die zukünftig erwartete Zunahme von mehrkernigen Systemarchitekturen wird allerdings die Spezifikation parallelen Verhaltens auch auf Komponentenebene vermehrt benötigt werden.

Die in dieser Arbeit vorgestellte Methode zur Testfallgenerierung erreicht höhere Spezifikationsabdeckungen (u. a. „alle möglichen alternativen Pfade“) der generierten Testfälle. Sie basiert und erweitert die Arbeiten aus [Ulr98] und [KHD08]. Erreicht wird die hohe Spezifikationsabdeckung durch die Analyse des Spezifikationsmodells mit bekannten und etablierten Methoden der constraintbasierten Programmierung und der Petri-Netz-Theorie, wodurch explizit die Beschreibung von parallelem Verhalten in der modellbasierten Spezifikation (auch auf Komponentenebene) unterstützt wird. Dafür wurde ein spezielles (Low-Level-)Petri-Netz-Modell (Sicheres Petri-Netz mit Attributen – SPeNAt) entwickelt, auf welches das Spezifikationsmodell abgebildet werden muss.

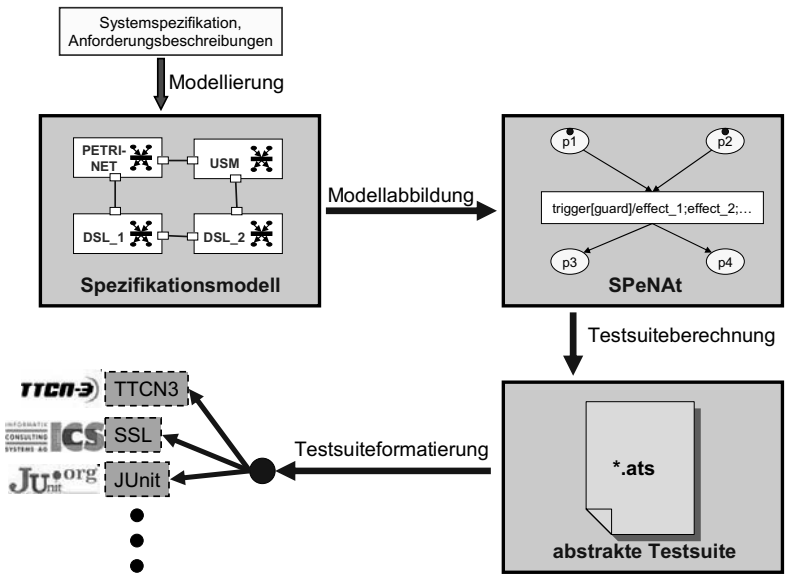

Abbildung 1: Schritte zur Testgenerierung und –formatierung

In Abbildung 1 werden die benötigten Schritte zur Testgenerierung und Testformatierung übersichtlich dargestellt. Alle Schritte werden durch entsprechende Software-Werkzeuge unterstützt, wobei etablierte Modellierungswerkzeuge und prototypische Implementierungen des ifak benutzt werden können. Auf der Grundlage der Systemspezifikation und/oder der beschriebenen Anforderungen wird zunächst das geforderte Systemverhalten modelliert und durch ein Spezifikationsmodell beschrieben. Dieses Spezifikationsmodell bildet die Grundlage für die Testgenerierung und spätere Testformatierung. Daher ist es von immenser Wichtigkeit, dass das Spezifikationsmodell die Systemspezifikation semantisch korrekt abbildet. Der Nachweis dieser Eigenschaften kann z. B. mit Methoden der formalen Verifikation (u. a. Model Checking [CGP99]) erfolgen, was in dieser Arbeit allerdings nicht weiter berücksichtigt wird. Vielmehr wird davon ausgegangen, dass das Spezifikationsmodell korrekt ist und zur Testgenerierung verwendet werden kann. Für die Modellierung des geforderten Systemverhaltens können dabei verschiedene Notationen wie z. B. Petri-Netze, UML-Zustandsmaschinen (USM), UML-Aktivitätsdiagramme und/oder domänenspezifische Sprachen (DSL) verwendet werden.

In der Praxis hat sich dabei die UML-Zustandsmaschine für die Modellierung reaktiven System- bzw. Komponentenverhaltens etabliert, wobei auch paralleles Verhalten mit einer UML-Zustandsmaschine modelliert werden kann.

2 Sicheres Petri Netz mit Attributen (SPeNAt)

Grundlage der entwickelten Methode zur modellbasierten Testfallgenerierung ist ein spezielles Petri-Netz-Modell (Sicheres Petri Netz mit Attributen – SPeNAt). Das Modell eines SPeNAts ist sehr einfach gehalten, wodurch Abbildungsvorschriften für gängige und etablierte Modellierungsnotationen wie z. B. UML-Zustandsmaschinen, UML-Aktivitätsdiagramme, DSLs möglich sind. Aktuell sind Abbildungsvorschriften für ein Spezifikationsmodell bestehend aus UML-Zustandsmaschinen definiert und in [KHD08] näher beschrieben.

Bei einem SPeNAt handelt es sich um ein sicheres Stellen/Transitionen Netz (siehe z. B. [ERV96]), welches um einige Aspekte erweitert wurde. So kann ein SPeNAt Attribute besitzen, die jeweils durch eine Datenstelle repräsentiert werden und durch einen (Daten-)Typ und einen eindeutigem Namen charakterisiert sind. Die Transitionen erhalten eine Beschriftung, die in Syntax und Semantik der Beschriftung einer Transition eines UML-Zustandsautomaten entspricht. Das bedeutet im Einzelnen, dass einer SPeNAt-Transition ein (externes, parametriertes) Ereignis, ein Guard sowie eine Menge an Aktionen zugeordnet sind. Eine SPeNAt-Transition schaltet, wenn ihre Vorgänger-Stellen markiert sind, ihr zugeordnetes (externes) Ereignis erscheint und ihr Guard zu *true* ausgewertet wird. Schaltet eine SPeNAt-Transition, werden ihre zugeordneten Aktionen ausgeführt und ihre Nachfolger mit entsprechenden Markierungen belegt.

In Abbildung 2 ist ein Beispiel für ein SPeNAt vor und nach dem Schalten einer Transition angegeben. Hier wird auch eine weitere Eigenschaft eines SPeNAts dargestellt. Datenstellen sind mit einer Transition immer durch eine Schlinge verbunden und dadurch immer markiert, da sie auch Teil der Anfangsmarkierung sind. Ob eine Datenstelle mit einer Transition verbunden ist, wird durch die Beschriftung der Transition (Guard und Aktionen) determiniert. Die Markierungen einer Datenstelle können durch verschiedene Werte des Typbereichs der Datenstelle erfolgen. Dieses Konzept für eine Petri-Netz-Markierung wird in der Literatur auch „farbige Token“ (siehe insb. [Jen09]) genannt, weshalb ein SPeNAt auch als einfaches farbiges Petri-Netz betrachtet werden kann.

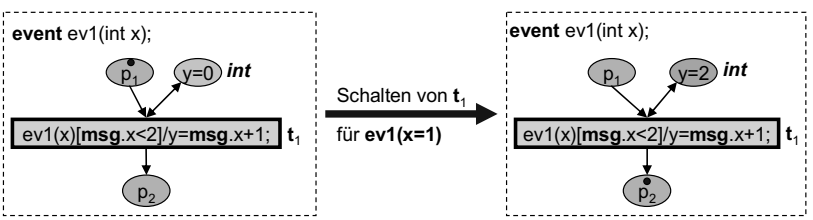

Abbildung 2: SPeNAt vor und nach dem Schalten einer Transition

3 Testgenerierung auf Grundlage der Spezifikationsmodellanalyse

Die Identifizierung der Testfälle wird auf Grundlage der Verhaltensanalyse des Spezifikationsmodells nach dessen Abbildung auf ein SPeNAt durchgeführt. Das Verhalten eines SPeNAt wird dabei durch die sukzessive Konstruktion des Entfaltungsnetzes berechnet. Dafür wird der Algorithmus aus [ERV96] verwendet und für die speziellen Eigenschaften eines SPeNAts angepasst [KHD08]. Die Arbeiten in [KHD08] wurden dahingehend erweitert, dass nun auch externe Ereignisse mit Parametern berücksichtigt werden können. In [KHD08] konnte das entsprechende Petri-Netz-Modell zwar auch schon Attribute verwenden (z. B. an den Guards der Transitionen), allerdings konnten nur externe Ereignisse ohne Parameter die Transitionen schalten. Erreicht wird diese Erweiterung durch die Verwendung von Methoden der constraintbasierten Programmierung, d. h. bei der Überprüfung der Schaltbarkeit einer Transition wird neben der benötigten Markierung der entsprechenden Vorgängerstellen auch ein Bedingungserfüllungsproblem (Constraint-Satisfaction-Problem – CSP) aufgestellt und ausgewertet. Dadurch können auch diese Informationen in der Präfixberechnung und damit in der darauf aufbauenden Testfallgenerierung mit berücksichtigt werden. Durch entsprechende umfassende Modellierung des geforderten Verhaltens im Spezifikationsmodell (z. B. Berücksichtigung des Fehlerverhaltens durch Modellierung mit *else*-Transitionen) werden somit Abdeckungsgrade nach dem v. a. bei der Entwicklung von sicherheitskritischen Systemen geforderten Kriterium MC/DC (siehe u. a. [TSWP10]) an den Bedingungen der SPeNAt-Transitionen erreicht.

Beim Entfaltungsnetz – präziser dem kompletten Anfangsteil des Entfaltungsnetzes (Präfix) – eines Stellen/Transitionen-Netzes handelt es sich um ein zyklenfreies, einfaches Bedingungen/Ereignisse-Netz [ERV96], das alle möglichen erreichbaren Markierungen des ursprünglichen Petri-Netzes codiert. Im Gegensatz zum Erreichbarkeitsgraphen eines Petri-Netzes werden aber im Entfaltungsnetz alle parallelen Ausführungen beibehalten, wodurch der Zustandsraum insbesondere bei parallelem Verhalten deutlich kompakter dargestellt wird [ERV96].

Die Eigenschaft des Entfaltungsnetzes, alle möglichen alternativen Prozesse zu enthalten [ERV96], wird bei der Testfallidentifizierung ausgenutzt. Wird jedem Prozess ein Testfall zugeordnet, wird ein Spezifikationsabdeckungsgrad „alle möglichen alternativen Pfade“ erreicht, welcher insbesondere bei der Entwicklung von sicherheitskritischen Anwendungen benötigt wird. Auf der Grundlage der sukzessiven Konstruktion des Entfaltungsnetzes können aber auch andere (einfachere) Testziele, wie z. B. Abdeckungsgrad „alle Transitionen“ o. ä., leicht identifiziert und somit auf den jeweiligen Anwendungsfall maßgeschneiderte Testfälle erstellt werden.

Ein sehr einfaches funktionales Anwendungsbeispiel wird in Abbildung 3 vorgestellt, um die Funktionsweise der Testfallgenerierung auf der Basis eines Spezifikationsmodells zu demonstrieren. Durch das Spezifikationsmodell werden drei Aktivitäten (Prozesse) durch entsprechende SPeNAt-Transitionen beispielhaft spezifiziert, wobei zwei Aktivitäten parallel ausgeführt werden können, aber vor der dritten Aktivität beendet sein müssen.

Insgesamt ist für dieses einfache Beispiel nur ein Testfall für eine vollständige Spezifikationsabdeckung notwendig (siehe rechten Teil aus Abbildung 3), wobei die konkreten benötigten Eingabeparameter aus der Auswertung des jeweiligen CSP berechnet werden. Durch die Verwendung der Entfaltungskonstruktion des SPeNAt-Spezifikationsmodells für die Testfallgenerierung kann das spezifizierte parallele Verhalten für die Testfallbeschreibung beibehalten werden. Somit sind bei gleicher Spezifikationsabdeckung deutlich weniger Testfälle notwendig als bei der Verwendung von Testfällen, die nur sequentielles Verhalten beschreiben können.

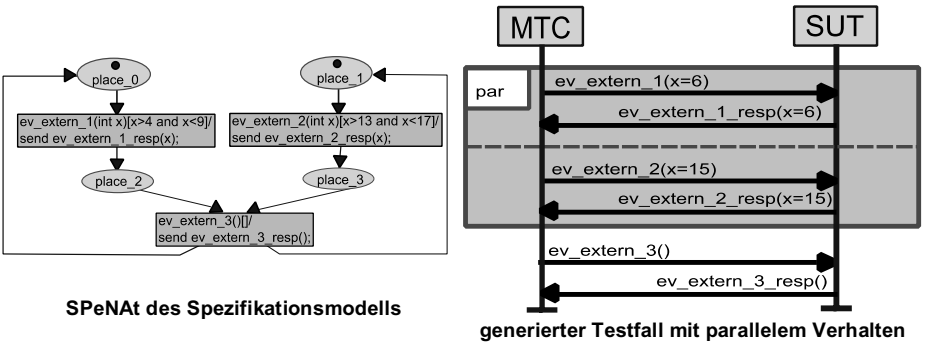

Abbildung 3: Funktionales Anwendungsbeispiel

An diesem Punkt besteht ein Testfall der generierten Testsuite aus einer Sequenz von SPeNAt-Transitionen mit entsprechenden Eingabeparametern, wobei Transitionen auch parallel in einem Testfall ausgeführt werden können. Für die konkrete Testrealisierung muss diese abstrakte Testsuitebeschreibung in ein Format gebracht werden, das im entsprechenden Testprozess verwendet werden kann. Dazu sind Informationen nötig, die sich aus dem jeweiligen Einsatzkontext ergeben. So müssen z. B. Datentypen und/oder Ereignisse bzw. Signale, die im Spezifikationsmodell auf einem eher abstrakten Niveau definiert wurden, auf verwendbare Strukturen der Testnotation des verwendeten Testwerkzeugs abgebildet werden.

Ein bekanntes und in der Praxis immer mehr etabliertes Format zur Testbeschreibung ist die durch die ETSI standardisierte *Testing and Test Control Notation* (TTCN-3), für die es mittlerweile auch mehrere Werkzeuge zur Testrealisierung gibt. Bei der Verwendung von TTCN-3 als Testnotation können die generierten Testsequenzen durch einfache Abbildungsregeln erzeugt werden. Allerdings müssen dann die Modellelemente auf die vorliegenden Gegebenheiten des SUT (Systems Under Test) bei der Implementierung des TTCN-3-SUT-Adapters angepasst werden, um eine automatische Realisierung des Tests zu ermöglichen. Die Spezifikation von parallelem Verhalten innerhalb eines Testfalls wird auch in TTCN-3 aktuell nicht vollständig unterstützt (partiell durch das *interleave*-Statement), ist aber in einer der nächsten Versionen (durch ein *par*-Statement) vorgesehen, so dass für die Testrealisierung aktuell noch auf nicht standardbasierte Werkzeuge zurückgegriffen werden muss.

4 Zusammenfassung

In diesem Papier wurde eine Methode zur Testfallgenerierung auf der Grundlage formaler Spezifikationen unter Berücksichtigung von parallelem Verhalten vorgestellt. In Abhängigkeit von den spezifizierten Testzielen können mit dieser Methode sehr hohe Spezifikationsabdeckungen (z. B. „alle möglichen alternativen Pfade“, MC/DC-Abdeckung an den Bedingungen) erzielt werden. Erreicht wird das durch die Verwendung eines einfachen, farbigen Petri-Netz-Modells (SPeNAt) und dessen Analysemöglichkeiten durch die Kombination von etablierten Methoden der Petri-Netz-Theorie und der constraintbasierten Programmierung. Falls erforderlich, wird auf dem gesamten Zustandsraum operiert, um die gewünschten Abdeckungsgrade zu erreichen. Dabei wird als Zustandsraumkodierung das komplette Präfix des Entfaltungsnetzes verwendet, der zum Einen eine kompakte Darstellung des Zustandsraumes erlaubt und zum Anderen eine einfache Identifizierung aller möglichen alternativen Prozesse – und damit der Testfälle – der Verhaltensspezifikation ermöglicht.

Literaturverzeichnis

[CGP99] E. M. Clarke, O. Grumberg, D. A. Peled: Model Checking, MIT-Press, 1999.

[ERV96] J. Esparza, S. Römer, W. Vogler: An improvement of McMillan's unfolding algorithm, In Proceedings of TACAS'96, pages 87-106, 1996.

[Jen09] Jensen, K.: Coloured Petri Nets: Modeling and Validation of Concurrent Systems, Springer Verlag, Berlin, 2009

[KHD08] J. Krause, A. Herrmann, Ch. Diedrich: Test case generation from formal system specifications based on UML State Machines. atp - International 01/2008, Oldenbourg-Verlag, 2008.

[PSO08] F. Pinte, F. Saglietti, N. Oster: Automatic Generation of Optimized Integration Test Data by Genetic Algorithms in Software Engineering, Workshop proceedings of MOTES'08, 2008.

[PL01] A. Pretschner, H. Lötzbeyer: Model Based Testing with Constraint Logic Programming: First Results and Challenges, Proceedings of WAPATV'01, Toronto, 2001.

[SW07] H. Schlingloff, S. Weißleder: Deriving Input Partitions from UML Models for Automatic Test Generation; In: Models in Software Engineering, Holger Giese (Ed.), Springer LNCS 5002, 2007.

[TSWP10] http://www.testingstandards.co.uk/glossary.htm, 2010.

[Ulr98] A. Ulrich: Testfallableitung und Testrealisierung in verteilten Systemen, Dissertation "Otto-von-Guericke"-Universität Magdeburg, 1998.