Visualisierung überdeckter sowie zu überdeckender Modellelemente im modellbasierten Test

Florin Pinte, Francesca Saglietti, Achim Neubauer

Lehrstuhl für Software Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstraße 3
91058 Erlangen
pinte@informatik.uni-erlangen.de
saglietti@informatik.uni-erlangen.de
achim.neubauer@gmx.de

Abstract: Verfahren zur automatischen Testfallgenerierung aus UML-Modellen sind Gegenstand vieler aktueller Forschungs- und Entwicklungsarbeiten. Sowohl für die Bewertung dieser automatisch generierten Testfälle durch den Benutzer als auch für die Unterstützung der Fehlerfindung im Modell ist eine Visualisierung der dadurch erzielten Modellüberdeckung sehr hilfreich. Um dies zu ermöglichen, beschreibt dieser Artikel ein Verfahren zur Visualisierung der von modellbasierten Testfällen erreichten strukturellen Überdeckung bezüglich verschiedener Überdeckungskriterien.

1 Einleitung

Durch die immer größere Verbreitung modellbasierter Softwareentwicklung gewinnt heutzutage auch der modellbasierte Test immer mehr an Bedeutung. Dabei stellt die Generierung geeigneter Testfälle auch bei dieser Testart die aufwändigste Aktivität dar. Deshalb beschäftigen sich viele Forschungsansätze im Bereich des modellbasierten Testens mit der automatischen Generierung von Testfällen zur Erreichung einer möglichst hohen strukturellen Abdeckung des Modells.

Neben der automatischen Generierung der Testfälle ist die Visualisierung der dadurch bereits überdeckten Modellteile in vielfacher Hinsicht sehr hilfreich: weil sie den Testfortschritt optisch veranschaulicht, die Bewertung eines Testablaufs unterstützt und darüber hinaus durch das Hervorheben noch nicht überdeckter Modellelemente die Erkennung von Modellfehlern erleichtert. Deshalb bietet das im Rahmen des Projektes UnITeD (Unterstützung Inkrementeller TestDaten) entwickelte Werkzeug – neben der automatischen Generierung von Testfällen zur Überdeckung von Zustandsautomaten – auch die Visualisierung bereits überdeckter sowie noch zu überdeckender Modellelemente an.

Zwar unterstützen bereits bestehende Modellierungswerkzeuge wie z.B. Rhapsody ebenfalls die Visualisierung der durch Tests überdeckten Modellentitäten. Der im Folgenden beschriebene Visualisierungsansatz kennzeichnet allerdings darüber hinaus auch die Relevanz (notwendig bzw. optional) noch nicht überdeckter Elemente in Bezug auf eine Reihe von Komponenten- und Integrationstestüberdeckungskriterien, die in Kapitel 4 näher beschrieben werden.

Dieser Artikel ist wie folgt gegliedert: der zweite Kapitel gibt einen Überblick über das zugrunde liegende Projekt UnITeD. Anschließend bietet Kapitel 3 eine ausführliche Motivation zur Visualisierung überdeckter Modellelemente. Im vierten Kapitel werden die entwickelten Konzepte zur Visualisierung dargestellt. In Kapitel 5 wird dann die Umsetzung der Visualisierung als Plugin für das Modellierungswerkzeug MagicDraw beschrieben.

2 Automatische Testdatengenerierung mit UnITeD

Ansätze zur automatischen Generierung von Testfällen aus UML-Modellen beschränken sich in den meisten Fällen auf die Generierung von Testsequenzen (d.h. Abfolgen von Operationsaufrufen an Komponenten), ohne dabei die aufwändige Aktivität der Generierung zugehöriger Testdaten zu betrachten. Ziel des Projekts UnITeD ist die Unterstützung der automatischen Testfallgenerierung (Testsequenzen mit zugehörigen Eingabedaten). Dieses Ziel wird mittlerweile auch von diversen, aktuell laufenden Forschungsprojekten [We08, LI07] verfolgt, indem Testsequenzen mit zugehörigen Testdaten generiert werden.

Von diesen unterscheidet sich der hier beschriebene Ansatz einerseits durch die besonders im Hinblick auf den Integrationstest ausgerichteten Überdeckungskriterien, andererseits durch die Optimierung der Testfallmenge. Optimiert wird hinsichtlich einer möglichst hohen Überdeckung des Testobjekts nach bestimmten Überdeckungskriterien bei gleichzeitiger weitgehender Minimierung der Anzahl dazu erforderlichen Testfälle. Angesichts des Aufwandes der im Allgemeinen bei der Überprüfung der Testfallausführung anfällt ist Letzteres von besonderer Bedeutung. Zum Zwecke der Optimierung wendet das in UnITeD entwickelte Werkzeug genetische Algorithmen an, die sowohl den Komponenten- als auch den Integrationstest unterstützen. Dabei wird die Güte generierter Testfallmengen als gewichtete Summe der beiden Optimierungsziele bewertet. Zusätzlich wird auch die automatische Generierung und Optimierung von Regressionstestfällen unterstützt. Dabei werden die nach Modelländerungen noch gültigen Testfälle markiert und falls nötig optimierte zusätzliche Testfälle dazu generiert. Als Folgeprojekt des am gleichen Lehrstuhl entstandenen Projektes .gEAr, das mittels genetischer Algorithmen Testfälle auf Quellcodeebene erzeugt, überträgt UnITeD den Ansatz von .gEAr von der Code auf die Modellebene. Den Publikationen [PSO08, Pi08, Os07] können Einzelheiten zu den Verfahren entnommen werden. Anhand konkreter Beispiele konnte nachgewiesen werden, dass das in UnITeD entwickelte und umgesetzte Verfahren zu einer entscheidenden Effizienzsteigerung des modellbasierten Tests beiträgt.

3 Vorteile der Visualisierung

Die in diesem Artikel beschriebene Visualisierung bietet zwei wichtige Vorteile:

3.1 Unterstützung der Testfallbewertung durch Veranschaulichung des Testfortschritts

Die Veranschaulichung des erreichten Testfortschritts ermöglicht dem Benutzer eine schnelle Bewertung der generierten Testfallmenge hinsichtlich ihrer Modellabdeckung. Dies ist besonders wichtig, falls die gewünschte Abdeckung nicht erreicht wurde, weil somit eine Entscheidungsgrundlage geschaffen wird, um entsprechende Maßnahmen zu treffen. Ein Beispiel für solch eine Maßnahme ist die manuelle Generierung von Testfällen zur Überdeckung noch nicht erreichter Entitäten. Auch im Falle der Erzielung einer 100-prozentigen Überdeckung ist die Visualisierung hilfreich, da sie erlaubt, Modellteile anzuzeigen, die von einzelnen Testfällen aus der Testfallmenge überdeckt wurden. Somit wird die Bewertung eines bestimmten Testfalls der Menge unterstützt.

3.2 Unterstützung der Erkennung von Modellfehlern

Die durch modellbasierte Softwareentwicklung in Aussicht gestellten Vorteile kommen natürlich nur dann zur Geltung, wenn die der Implementierung zugrunde liegenden Modelle fehlerfrei sind. Um diese Modelle zu überprüfen, haben sich Methoden wie z.B. Inspections und Walkthroughs durchgesetzt. Durch die manuelle Natur dieser Methoden besteht die Gefahr, dass bei deren Anwendung bestimmte Teile des Modells übersehen werden. Gegenüber diesen manuellen Verfahren besteht der Vorteil bei der Anwendung modellbasierter Testfälle in der systematischen Abdeckung der Modellelemente. Durch die Visualisierung der durch Testfälle beschriebenen Abläufe durch das Diagramm wird die Erkennung fehlerhafter Übergänge oder Pfade im Modell unterstützt. Ebenso bietet die Anzeige der nicht überdeckten Modellelemente eine Hilfestellung beim Finden von Modellfehlern aufgrund nicht ausführbarer Modellteile.

4 Visualisierung überdeckter und zu überdeckender Modellentitäten

In Abhängigkeit von dem vorgegebenen Testüberdeckungskriterium lassen sich die betrachteten Modellentitäten wie folgt in Kategorien einteilen:

1. Atomare Entitäten eines Graphen sind seine elementaren Bestandteile, d.h. seine Knoten und seine Kanten. Übertragen auf Zustandsautomaten handelt es sind um Zustände und Transitionen.

2. Zusammengesetzte Entitäten eines Graphen bestehen aus mehreren atomaren Entitäten. In diesem Fall reicht die Überdeckung der einzelnen Bestandteile nicht aus, diese müssen vielmehr in einem vorgegebenen Zusammenhang bzw. in vordefinierten Kombinationen überdeckt werden. Beispiele für solche Entitäten sind Transitionspaare, sowie sogenannte Mappings und Mapping-Gruppen, die im Folgenden eingeführt werden.

Ein Mapping ist ein Paar (t, t'), bestehend aus zwei Transitionen jeweils unterschiedlicher Zustandsmaschinen mit der Eigenschaft, dass als Effekt der Transition t in einer aufrufenden Komponente die Transition t' in einer aufgerufenen Komponente getriggert werden kann. Als Beispiel betrachte man das Mapping m1=(tA6, tB4) in Abb. 1, wobei Effekt(tA6)=Trigger(tB4). Wie am selben Beispiel deutlich wird, kann eine Transition Bestandteil unterschiedlicher Mappings sein, die Transition tA6 z.B. gehört zu den beiden Mappings m1 und m2.

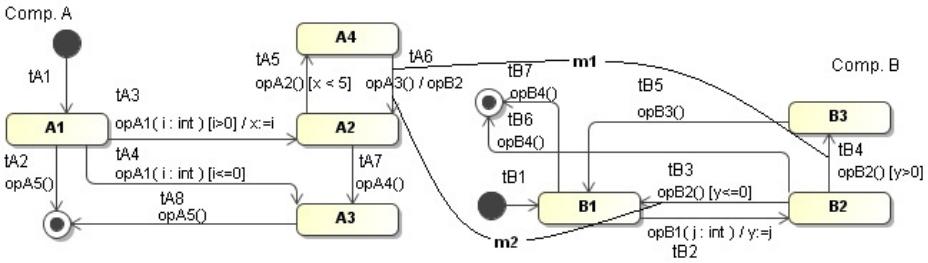

Abbildung 1: Beispiele für 2 Mappings m1=(tA6, tB4) und m2=(tA6, tB3)

Aufbauend auf diesem Konzept wurden in [SOP07] neue Überdeckungskriterien zusammen mit einer Aufwandsschätzung zu deren Erfüllung vorgestellt. Speziell die dort eingeführten zustandsbasierten Überdeckungskriterien partitionieren die Gesamtmenge der Mappings in sogenannte Mapping-Gruppen, d.h. Gruppen zueinander alternativer Mappings. Entsprechende Testkriterien fordern, dass aus jeder Mapping-Gruppe mindestens ein Mapping durch Testfälle überdeckt wird.

Zur Veranschaulichung betrachte man beispielhaft die in Abb. 2 illustrierte Komponente. Zur Erfüllung des Testkriteriums „invoking / invoked transitions", das die Überdeckung aller Mappings verlangt, müssen in diesem Fall die beiden Mappings m1 und m2 durch Testfälle zur Ausführung kommen. Zum etwas bescheideneren Testkriterium „invoking transitions on pre-states", das die Überdeckung aller Transitionen aufrufender Komponenten in allen möglichen Zuständen aufgerufener Komponenten fordert, würde hier hingegen nur die Überdeckung eines der beiden Mappings ausreichen. Dargestellt wird dies wie in Abb. 2 illustriert: die zu einer Mapping-Gruppe gehörenden, alternativen Mappings werden durch zeilenweise angeordnete, durch Komma getrennte Paare dargestellt (s. Abb. 2 (b)), während die unterschiedlichen Mapping-Gruppen untereinander (s. Abb. 2 (a)) aufgelistet werden.

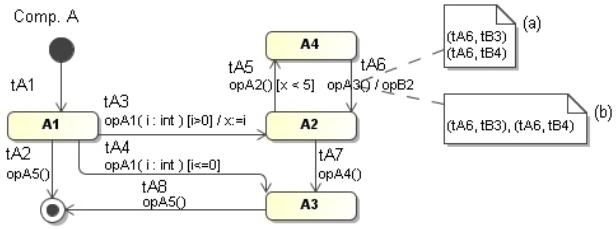

Abbildung 2: Darstellung der zu überdeckenden Mappings für die Kriterien (a) „invoking / invoked transitions“ und (b) „invoking transitions on pre-states“

4.1 Visualisierung überdeckter und noch zu überdeckender atomarer Entitäten

Im Falle atomarer Entitäten wird die Visualisierung durch Färbung realisiert, indem die einzelnen Transitionen und Zustände wie folgt eingefärbt werden:

- Grün, wenn sie von mindestens einem Testfall der Testfallmenge überdeckt werden.
- Rot, wenn sie noch nicht überdeckt wurden, aber zu überdecken sind.
- Gar nicht markiert, wenn sie noch nicht überdeckt wurden und nicht unbedingt zu überdecken sind (wie z.B. nicht durchlaufene Transitionen im Falle einer angestrebten Zustandsüberdeckung).

Wie schon in Kapitel 3 erwähnt, ist die Visualisierung der von einem einzelnen Testfall überdeckten Entitäten auch hilfreich. Deshalb wird zur Markierung der Entitäten, die von einem bestimmten Testfall überdeckt werden, die Farbe Blau zusätzlich verwendet.

4.2 Visualisierung überdeckter und noch zu überdeckender zusammengesetzter Entitäten

Zur Markierung der zusammengesetzten Entitäten reicht die Färbung einzelner Transitionen nicht mehr aus, weil die Überdeckung einer Transition hier nur in Zusammenhang mit einer anderen Transition relevant ist. Zu diesem Zweck werden in diesem Fall die unterschiedlichen Mappings, zu denen eine Transition gehört, mittels eines Notiztextfeldes festgehalten (s. Abb. 2). Die Visualisierung überdeckter Mappings wird anhand der farblichen Markierung der Mappings in den Notiztextfeldern folgendermaßen realisiert:

- Grün, wenn das Mapping von mindestens einem Testfall überdeckt wird.
- Rot, wenn es noch nicht überdeckt wurde.

Zusätzlich zur Färbung der Notiztexte werden auch die Transitionen eingefärbt. Somit kann man auch an der Farbe der Transition erkennen, inwieweit die Mappings, zu denen diese Transition gehört, überdeckt wurden. Gerade im Falle sehr informationsdichter Notiztextfelder kann sich die Transitionsfärbung als anschauliche Abstraktion erweisen.

Die Färbung der Transition erfolgt folgendermaßen:

- Grün, wenn jede Gruppe, zu der die Transition gehört, durch mindestens einen Testfall überdeckt wurde.
- Rot, wenn nicht alle Gruppen, zu der die Transition gehört, von der Testfallmenge überdeckt wurden.

Die von einem einzigen Testfall überdeckten zusammengesetzten Entitäten werden blau koloriert. Ein blaues Mapping bedeutet also, dass dieses von einem ausgezeichneten Testfall überdeckt wurde. Die blaue Färbung der Transition bedeutet andererseits, dass alle Mapping-Gruppen, zu der sie gehört, durch den betrachteten Testfall überdeckt wurden.

5 Umsetzung der Visualisierung als MagicDraw Plugin

Die in Kapitel 4 beschriebenen Konzepte wurden als Plugin für das Modellierungswerkzeug MagicDraw umgesetzt. Die Benutzung dieses Plugin erlaubt die Anzeige der Testfallüberdeckung im Bezug auf ein bestimmtes Überdeckungskriterium. Zur Auswahl der Testfallmenge und des Kriteriums stehen als Steuerungselemente 2 DropDowns zur Verfügung (s. Abb. 3). Falls im DropDown “Testfall“ kein bestimmter Testfall ausgewählt ist, wird die Überdeckung angezeigt, die von der gesamten Testfallmenge erreicht wird. Das DropDown “Kriterium“ ermöglicht andererseits eine Auswahl unter den Testkriterien der Zustandsüberdeckung, der Transitionsüberdeckung und der in [SOP07] eingeführten zustandsbasierten Überdeckungskriterien.

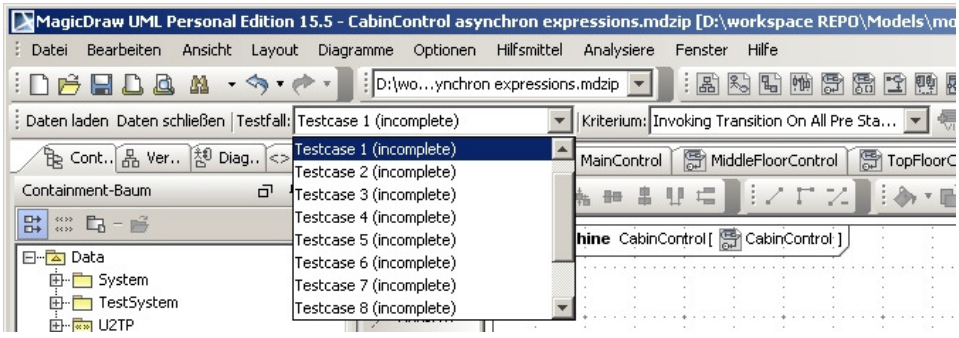

Abbildung 3: Menüleiste zur Visualisierung

Nachdem die Testfallmenge und das Kriterium ausgewählt wurden, erfolgt die Färbung, wie in Abb. 4 beispielhaft illustriert. Hier wurde das Testkriterium „invoking transitions on pre-states“ ausgewählt. Den Transitionen c8, c9 und c10 sind jeweils 4, 3 und 2 Mapping-Gruppen zugeordnet. Insgesamt wurden zum Zeitpunkt des gezeigten Snapshot 5 Mappings noch nicht überdeckt. Die noch nicht erfolgte Überdeckung von 2 Mappings, zu denen die Transitionen c9 bzw. c10 gehören, hat keine Auswirkung auf die Erfüllung des Überdeckungskriteriums, weil diese Mappings Teil bereits überdeckter Mapping-Gruppen sind. Die Tatsache, dass die restlichen 3 Mappings (zu denen die Transition c8 gehört) noch nicht überdeckt wurden, wirkt sich aber durchaus auf die Erfüllung des Überdeckungskriteriums aus, da diese jeweils eine Mapping-Gruppe bilden und deshalb notwendigerweise einzeln zu überdecken sind.

Die ursprüngliche Anzeige wurde mittels eines Bildbearbeitungsprogramms dadurch ergänzt, dass nicht überdeckte Mappings zusätzlich eingerahmt werden; dadurch können selbst bei einem Schwarz-Weiß-Druck der vorliegenden Publikation bereits überdeckte Mappings von noch nicht überdeckten Mappings deutlich unterschieden werden.

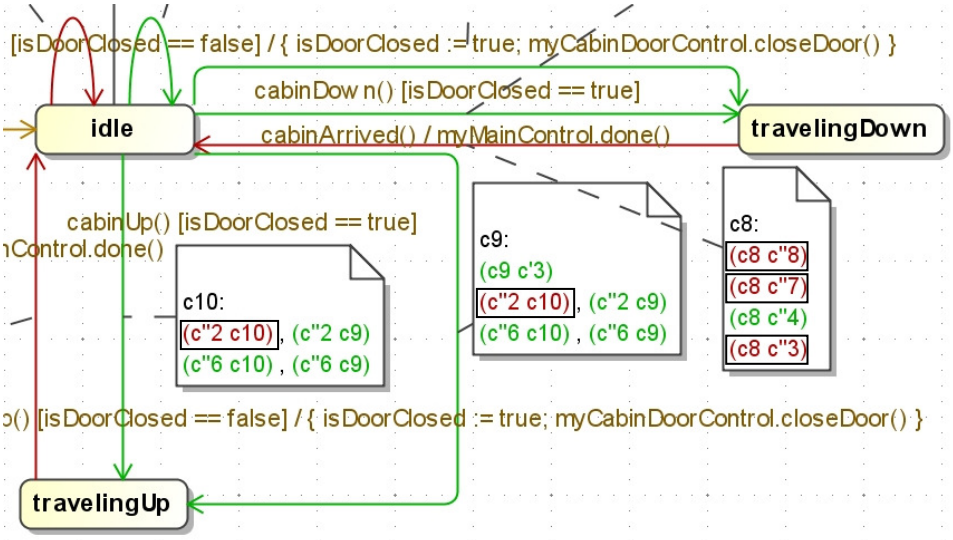

Abbildung 4: Teil eines Zustandsdiagramm in dem die zu überdeckenden und überdeckten Mappings farblich markiert sind

6 Zusammenfassung

Dieser Artikel beschreibt einen Ansatz zur Visualisierung der von einer Testfallmenge erreichten Modellüberdeckung in Bezug auf verschiedene Testüberdeckungskriterien, einschließlich zustandsbasierter Integrationsteststrategien. Zum einen erlaubt diese Visualisierung eine schnelle Bewertung der erreichten strukturellen Überdeckung, zum anderen unterstützt sie die Fehlerfindung bereits in den frühen Phasen des Software Lebenszyklus.

Literaturverzeichnis

[LI07] Lefticaru, R.; Ipate, F.: Automatic State-Based Test Generation Using Genetic Algorithms. In Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2007), IEEE Computer Society, 2007.

[Ne08] Neubauer, A.: Visualisierung überdeckter sowie zu überdeckender Modellelemente im modellbasierten Test. Diplomarbeit, Lehrstuhl für Software Engineering, Universität Erlangen-Nürnberg, 2008.

[Os07] Oster, N. et. al.: Automatische, modellbasierte Testdatengenerierung durch Einsatz evolutionärer Verfahren. In Informatik 2007 - Informatik trifft Logistik, Lecture Notes in Informatics, Vol. 110, Gesellschaft für Informatik, 2007.

[Pi08] Pinte, F. et.al.: Automatische Generierung optimaler modellbasierter Regressionstests. In Informatik 2008 - Beherrschbare Systeme dank Informatik (Band 1), Lecture Notes in Informatics, Vol. 133, Gesellschaft für Informatik, 2008.

[PSO08] Pinte, F.; Saglietti, F.; Oster, N.: Automatic Generation of Optimized Integration Test Data by Genetic Algorithms. In Software Engineering 2008 - Workshopband, Lecture Notes in Informatics, Vol. 122, Gesellschaft für Informatik, 2008.

[SOP07] Saglietti, F.; Oster, N.; Pinte, F.: Interface Coverage Criteria Supporting Model-Based Integration Testing. In M. Platzner, K.-E. Großpietsch, C. Hochberger, A. Koch (Eds.): ARCS '07 Workshop Proceedings, VDE Verlag, 2007.

[We08] Weißleder, S.: Partition-Oriented Test Generation. In Informatik 2008 - Beherrschbare Systeme dank Informatik (Band 1), Lecture Notes in Informatics, Vol. 133, Gesellschaft für Informatik, 2008.