

Strategien zur Testfallgenerierung aus SysML Modellen

Sebastian Schlecht
Technische Universität Darmstadt
Fachgebiet Echtzeitsysteme
Merckstrasse 25
D-64283 Darmstadt
schlecht@es.tu-darmstadt.de

Oliver Alt
Robert-Bosch GmbH
CM-DI/EAA3
Daimlerstr. 6
D-71229 Leonberg
oliver.alt2@de.bosch.com

Abstract: Um die zunehmende Komplexität moderner Telematik- und Infotainmentsysteme in den Griff zu bekommen und eine gleichbleibende Qualität der Produkte zu sichern, werden im Hause der Robert-Bosch/Blaupunkt GmbH neue Testkonzepte für den Systemtest solcher Systeme entwickelt. Dabei werden Teile der bislang informellen Spezifikation durch formale Modelle auf Basis der Systems Modeling Language (SysML) ersetzt. Eines der wesentlichen Ziele dabei ist, das Modell im Rahmen eines Modell-basierten Entwicklungsprozesses zu nutzen, um daraus möglichst automatisiert Testfälle abzuleiten (Modell-basiertes Testen). Im Rahmen dieses Papiers wird beschrieben, welche Strategien zur Testfallgenerierung aus solchen Modellen untersucht, letztendlich ausgewählt, erweitert und umgesetzt wurden um dies zu erreichen.

1 Einleitung

Mit der stetig steigenden Anzahl an neuen Funktionen und der damit einhergehenden zunehmenden Komplexität moderner Telematik- und Infotainmentsysteme spielt der Softwaretest eine tragende Rolle in der Qualitätssicherung. Eine Erstellung der Testfälle per Hand führt auf Grund von Budget- und Entwicklungszeitbegrenzungen nur zu einer unbefriedigenden Testabdeckung. Ein möglicher Ausweg aus dieser Situation ist die automatisierte Gewinnung von Testdaten aus einem Systemmodell, welches die bisherigen informellen Systemspezifikationen durch eine formale Darstellung ersetzt (Modell-basierter Test). In diesem Papier wird ein Ansatz präsentiert, der es erlaubt aus einem in der Systems Modeling Language (SysML) erstellten Systemmodell automatisiert Testfälle abzuleiten. Den Hauptbestandteil bilden dabei dynamische Modelle in Form von Aktivitätsdiagrammen.

Das weitere Papier gliedert sich wie folgt. In Abschnitt 2 werden verschiedene Strategien zur Testfallgenerierung betrachtet und bewertet. Die Beschreibung des gewählten Lösungsansatzes findet sich in Abschnitt 3. Eine kurze Einführung in das verwendete CD-Player Beispiel gibt Abschnitt 4. Abschnitt 5 beschreibt das Modellierungskonzept, welches Verwendung findet um daraus mit Hilfe des in Abschnitt 6 geschilderten Konzeptes Aktivitätssequenzen und letztendlich Testfälle ableiten zu können. Eine Zusammenfassung und einen Ausblick gibt Abschnitt 7.

2 Betrachtung verschiedener Testfallgenerierungsstrategien

Neben verschiedenen statistischen Ansätzen (z.B. [WT94], [WPT95]) sind auch deterministische Methoden zum Erzeugen von Testfällen aus zustandsbasierten Modellen weit verbreitet. Hierbei kommen meist Formen von Überdeckungsalgorithmen zum Einsatz. In [Bin00] werden mögliche Überdeckungskriterien für Statecharts umfassend behandelt.

Briand und Labiche präsentieren in [BL02] eine Methodik um aus verschiedenen UML-Modellartefakten Systemtestfälle zu generieren. Aktivitätsdiagramme werden benutzt um sequentielle Abhängigkeiten zwischen Use-Cases darzustellen. Sequenzdiagramme beschreiben Abläufe und Systemzustandsänderungen als Folge der ausgeführten Use-Cases.

In den Arbeiten [BNBM05] und [BB05] werden Ereignis-Sequenzgraphen (*Event Sequence Graphs*) für die Modellierung interaktiver Systeme verwendet. Hier handelt es sich wie bei Briand und Labiche um einen Ansatz der auf eine explizite Modellierung der Zustände des Systems verzichtet. Vielmehr wird das System durch eine Menge von Graphen beschrieben, die alle möglichen Kombinationen von externen Ereignissen und den dazugehörigen Systemantworten beschreiben. Ein Vergleich der beiden Herangehensweisen in Form einer Fallstudie findet sich in [BBS06].

3 Lösungsansatz

In dem hier beschriebenen Ansatz wird, wie in [BL02] und den Arbeiten von Belli et al., eine explizite Definition von Zuständen vermieden. Da zum Zeitpunkt der Modellierung oftmals nichts über die interne Implementierung und die resultierenden internen Zustände des zu testenden Systems bekannt ist (Blackbox-Test), würde ein zustandsbasiertes Modell vor allem auf Annahmen des Testingenieurs beruhen, bzw. nur dazu dienen das nach außen sichtbare Verhalten abzubilden. Zudem beschreiben Spezifikationen, z.B. in Form von Anwendungsfällen, eher zu realisierende Funktionen als Zustände.

Ein zu erzeugender Testfall wird hier als eine Sequenz von externen Triggern (Benutzeraktionen) angesehen. Dem Testfallgenerator kommt dann die Aufgabe zu aus allen möglichen Benutzeraktionen sinnvolle Sequenzen von Benutzerinteraktionen zu erstellen. Um eine Validierung der Systemreaktion durchzuführen muss anschließend das Verhalten des zu testenden Systems mit den in der Spezifikation enthaltenen Angaben verglichen werden.

Anstatt wie in [BNBM05] Ereignissequenzgraphen von Hand zu Erstellen, wird hier ein Weg vorgeschlagen über das dynamische Verhalten des Systems eine logische Folge von möglichen Ereignissen abzuleiten. Hierfür wird die Systemspezifikation des dynamischen Verhaltens in Form von Aktivitätsdiagrammen umgesetzt. Im Rahmen eines Modell-basierenden Entwicklungsprozesses könnten solche Modelle außerdem bereits als Ergebnis der Anforderungsermittlung vorliegen.

Als Modellierungssprache wurde die Systems Modeling Language (SysML) [OMG06] ausgewählt. Sie bietet gegenüber der Unified Modeling Language (UML) unter anderem einige Erweiterungen, die die Verhaltensmodellierung durch Aktivitätsdiagramme verein-

fachen, wie z.B. die Beeinflussung des Laufzeitzustandes von Aktivitäten durch den sogenannten *control operator*.

3.1 Modellierung auf funktionaler Ebene

Um eine Wiederverwendbarkeit sowohl von Modellen als auch von erzeugten Testfällen zu gewährleisten, wurde in [Alt06] ein Konzept vorgestellt um funktionale und produkt-spezifische Informationen im Modell voneinander zu trennen. Damit wird es ermöglicht Systemverhalten und Testfälle auf einer abstrakten, funktionalen Ebene zu spezifizieren und zum Testen verschiedener Produkte der selben Domäne verwenden zu können. Hierzu werden Benutzer- und Systemaktionen definiert, die für Geräte einer Domäne Gültigkeit haben. Testfälle zum Test des Systems sind dann Sequenzen aus Benutzeraktionen (z.B. Wiedergabe starten, etc.) und zusätzlichen Validierungsaktionen. Wie solche funktionalen Testfälle dann für konkrete Produkte automatisiert ausführbar gemacht werden können wurde ebenfalls bereits in [Alt06] beschrieben und ist nicht Teil dieses Papiers.

Um die Vorteile der funktionalen Beschreibung zu nutzen, wird daher auch das dynamische Systemverhalten auf dieser Ebene modelliert, um daraus mit dem hier vorgestellten Ansatz funktionale Testfälle abzuleiten.

4 Beispiel

Als Beispielsystem dient ein CD-Player mit Grundfunktionalität. Da für die Erstellung von Testsequenzen auf funktionaler Ebene kein Wissen über die technische Realisierung nötig ist, kann hier auf eine detailliertere technische Beschreibung verzichtet werden. Für das Modell werden lediglich die nach Außen sichtbaren Funktionen benötigt, die von jedem handelsüblichen CD-Player unterstützt werden. Namentlich sind das Funktionen wie CD Abspielen, Pause, Stopp, Nächster Titel, direkte Titelwahl über eine numerische Tastatur, Zufallsmodus, etc.

5 Konzeption des Modells

Das dynamische Systemmodell besteht im wesentlichen aus Datenobjekten (*Objects*), Benutzeraktionen (*User Activities*) und Systemaktivitäten (*System Activities*). Hinzu kommen Start- und Endknoten, Kontroll- und Datenflüsse, sowie Entscheidungs- und Zusammenführungsknoten (*Decision and Merge node*).

Datenobjekte

Mit Datenobjekten werden dem Systemmodell Parameter übergeben, die für den Testablauf benötigt werden. Im Beispielmmodell sind dies beispielsweise Informationen über die im Test verwendete CD. Weiterhin bilden Datenobjekte auch implizit den Zustandsraum des Systems (z.B. aktueller Titel, etc.).

Systemaktivitäten

Systemaktivitäten bilden das Gegenstück zu den Benutzeraktivitäten (externe Sicht) und beschreiben das Systemverhalten aus der Sicht des Systems (interne Sicht). Die innere Struktur der Systemaktivitäten beschreibt mittels SysML Aktivitätsdiagrammen das dynamische Verhalten (vgl. Abbildung 1).

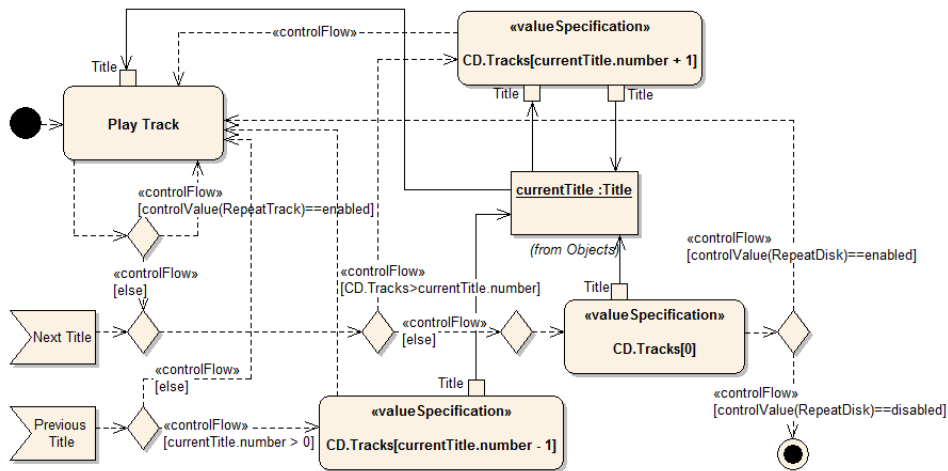


Abbildung 1: Modell der Systemaktivität Wiedergabe (*Playback*)

Benutzeraktionen

Um entscheiden zu können welche Benutzeraktionen zu einem Zeitpunkt sinnvoll sind, gibt es zu jeder Benutzeraktion eine Reihe von Vorbedingungen. Sinnvoll aus Sicht der Testfallgenerierung sind hier alle Aktionen, die eine Änderung innerhalb des Systems zur Folge haben. Für das zu testende System bedeutet das auch, dass nicht sinnvolle Benutzeraktionen zu keiner Systemreaktion führen dürfen und insbesondere auch keinen Systemabsturz zur Folge haben. Um Vorbedingungen zu spezifizieren, können alle Eigenschaften des Systems (Datenobjekte, derzeitige Laufzeitzustände der Systemaktivitäten) abgefragt werden. Auch zur Spezifikation der Vorbedingungen wird die Aktivitätsdiagrammsprache verwendet. Ein Beispiel einer solchen Spezifikation zeigt Abbildung 2 für die Benutzeraktion Wiedergabe starten.

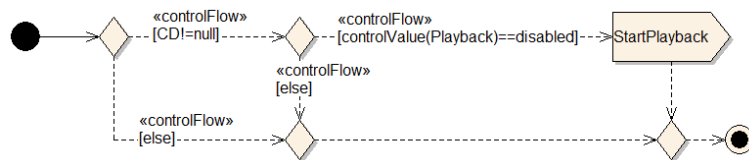


Abbildung 2: Vorbedingung zur Benutzeraktion Start playback (Wiedergabe starten)

5.1 Verknüpfung von Benutzer- und Systemaktivitäten

Um nun die Abhängigkeiten zwischen Benutzer- und Systemaktivitäten darzustellen, werden Instanzen beider Aktivitäten verknüpft. Es ergeben sich vier Möglichkeiten wie Benutzeraktivitäten auf das System einwirken können. Eine Benutzeraktion kann entweder keine Änderung des Systemzustandes, nur eine Änderung des Laufzeitzustandes einer Systemaktivität, nur eine Änderung eines Objektzustandes oder eine Änderung von beidem hervorrufen.

Der in SysML neu hinzugekommene *control operator* wird verwendet um die Systemaktivitäten zu steuern. Abbildung 3 zeigt die Anwendung des control operator am Beispiel der Systemaktivität Playback (Wiedergabe).

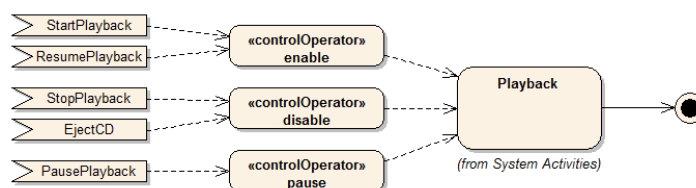


Abbildung 3: Zuordnung möglicher control values zu einer Systemaktivität

6 Erzeugung der Testsequenzen

Mit Hilfe der Simulation des dynamischen Verhaltens können nun Sequenzen von gültigen Testabläufen generiert werden. Um bekannte Algorithmen zur Testfallüberdeckung einsetzen zu können, wird nun aus erstellten Ereignissequenzen ein - den Ereignissequenzgraphen ähnlicher - Graph erzeugt.

Konkret müssen folgende Schritte durchgeführt werden um einen solchen Graphen zu generieren:

1. Initialisieren der Datenobjekte (z.B. Erstellen eines Objektes das die im Test benutzte CD abbildet) und Initialisieren der control values der Systemaktivitäten.
2. Aus der Menge der Benutzeraktionen werden nun diejenigen dem Graphen hinzugefügt und mit dem vorherigen Knoten verbunden, die in dem derzeitigen Systemzustand eine Änderung im System hervorrufen.
3. Schritt 2 wird nun solange wiederholt bis vorher gewählte Überdeckungskriterien erfüllt sind.

Über die Wahl der Überdeckungskriterien kann die mögliche Anzahl und Komplexität der Testsequenzen gesteuert werden. Denkbar ist hier beispielsweise eine Überdeckung aller Entscheidungspfade der Vorbedingungen. Weitere Kriterien zur Steuerung des Algorithmus werden zur Zeit noch untersucht. Eine Generierung einzelner Testfälle kann dann mit Hilfe von Standard-Graphenalgorithmen erfolgen.

7 Zusammenfassung und Ausblick

In diesem Papier wurde ein Ansatz vorgestellt um aus einem SysML-Aktivitätsmodell automatisiert Testsequenzen abzuleiten. Dabei wird das Systemverhalten durch Aktivitätsdiagramme aus der Benutzer- und Systemsicht modelliert. Durch den Einsatz des in SysML spezifizierten *control operator*, sowie spezieller Aktionen zum Ändern von Objektzuständen lassen sich beide Modellteile verknüpfen. Die Ableitung der Testsequenzen erfolgt dann durch eine Simulation des dynamischen Systemmodells und den sukzessiven Aufbau gültiger Folgen von Benutzeraktionen.

Der Einsatz der Aktivitätsdiagramme an Stelle von zustandsbasierten Modellen hat darüber hinaus den Vorteil, dass man weithin eingesetzte Verfahren zur Spezifikation von Systemen, wie beispielsweise Anwendungsfälle, leicht in solche Aktivitätsdiagrammmodelle umsetzen kann.

Um den beschriebenen Ansatz in der Praxis zu evaluieren, wird zur Zeit an der Fertigstellung eines solchen Modellsimulators für das UML Werkzeug Enterprise Architect gearbeitet. Weiterhin wird untersucht, inwieweit sich das Modell eignet um verschiedene Testüberdeckungskriterien bei der Generierung von Testfällen zu erfüllen.

Literatur

- [Alt06] Oliver Alt. Generierung von Systemtestfällen für Car Multimedia Systeme aus domänenspezifischen UML Modellen. In Christian Hochberger und Rüdiger Liskowsky, Hrsg., *INFORMATIK 2006 Informatik für Menschen Band 2*, 10 2006.
- [BB05] Fevzi Belli und Christof J. Budnik. Towards Minimization of Test Sets for Coverage Testing of Interactive Systems. In *Software Engineering (SE) Essen, Germany*, 2005.
- [BBLs06] Fevzi Belli, Christof J. Budnik, Michael Linschulte und Ina Schieferdecker. Testen Web-basierter Systeme mittels strukturierter, graphischer Modelle - Vergleich anhand einer Fallstudie. In Christian Hochberger und Rüdiger Liskowsky, Hrsg., *INFORMATIK 2006 Informatik für Menschen Band 2*, 10 2006.
- [Bin00] Robert V. Binder. *Testing Object-Oriented Systems: Models, Patterns, and Tools*. Object Technology Series. Addison-Wesley, 2000.
- [BL02] Lionel Briand und Yvan Labiche. *A UML-Based Approach to System Testing*. Software Quality Engineering Laboratory, Carleton University, 2002.
- [BNBM05] Fevzi Belli, Nimal Nissanke, Christof J. Budnik und Aditya Mathur. Test Generation Using Event Sequence Graphs. Technical reports and working papers, Universität Paderborn, <http://adt.upb.de>, 9 2005.
- [OMG06] OMG. OMG Systems Modeling Language (OMG SysML™) - Final Adopted Specification. ptc/06-05-04, Object Management Group, 5 2006.
- [WPT95] Gwendolyn H. Walton, J.H. Poore und Carmen J. Trammell. Statistical Testing of Software Based on a Usage Model. *Software - Practice and Experience*, 25(1):97–108, 1995.
- [WT94] James A. Whittaker und Michael G. Thomason. A Markov Chain Model for Statistical Software Testing. *IEEE Transactions on Software Engineering*, 20(10), 1994.