Sequenzgenerierung aus Klassifikationsbäumen

Peter M. Kruse

Berner & Mattner Systemtechnik GmbH
Berlin
Email: peter.kruse@berner-mattner.com

Joachim Wegener

Berner & Mattner Systemtechnik GmbH
Berlin
Email: joachim.wegener@berner-mattner.com

I. Einleitung

Die weit verbreitete Klassifikationsbaum-Methode [1] erlaubt, unter Verwendung des CTE XL Professional [2] oder des CTE XL [3], die automatische Generierung von Testfallmengen für underschiedliche Testobjekte. Die in der Praxis ebenso wichtige Erstellung von Testsequenzen wird von den genannten Werkzeugen bislang nur manuell unterstützt. Berner & Mattner arbeitet daher an Generierungsmechanismen für die automatische Erzeugung sinnvoller, fehlersensitiver Testsequenzen. Erste Ergebnisse dieser Arbeiten und die sich daraus für die Klassifikationsbaum-Methode ergebenden Erweiterungen werden in diesem Beitrag vorgestellt.

II. Stand von Wissenschaft und Technik

Neben der Erstellung von Testfällen zählt die Entwicklung von Testsequenzen zu den wichtigsten Aktivitäten in der Testfallermittlung. Für die systematische Prüfung zustandsbasierter Systeme müssen Testsequenzen erzeugt werden, die unterschiedliche Zustandswechsel und Transitionsfolgen umfassen.

Heimdahl et al. beschreiben einige Ansätze zur Testsequenzgenerierung mittels Techniken des Modell-Checkings [4]. Der gemeinsame Ansatz ist es, die Gegenbeispiel-Generierung des Modell-Checkers zur Erzeugung relevanter Testsequenzen zu verwenden.

Zwei temporale Logiken, *Linear Temporal Logic* (LTL) und *Computation Tree Logic* (CTL), werden in [5] verglichen.

Krupp and Mueller führen eine interessante Anwendnung von CCTL Logik zur Verifikation von Klassifikationsbaum-Testsequenzen ein [6]. Mittels eines Echtzeit-Modell-Checkers werden Testsequenzen und ihre Transitionen verifiziert unter Zuhilfenahme einer Kombination von E/A-Schritt-Beschreibungen und CCTL Ausdrücken.

Wimmel et al. schlagen eine Methode zur Testsequenzgenerierung unter Verwendung von Aussagenlogik vor [7].

Bernard et al. haben sich intensiv mit der Generierung von Testsequenz mit der Spezifikationssprache *B* befasst [8]. Ein System von Äquivalenz-Abhängigkeiten wird dann aus einer Spezifikation abgeleitet, um mit einem Constraint-Solver die Grenzzustände und Testfälle zu berechnen.

Binder listet Orakelmuster auf, die zum Softwaretest verwendet werden können [9]. Das *Simluation Oracle Pattern* simuliert ein System basierend auf einem vereinfachten Systemmodell.

Ural beschreibt formale Methoden zur Generierung von Testsequenzen basierend auf der Beschreibung endlicher Zustandsautomaten (FSM) [10] und fragt, ob die gegebene Implementierung eines System dem FSM-Modell dieses Systems genügt. Testsequenzen werden dabei aus dem FSM-Modell des Systems abgeleitet, um die Ausagen des Modells mit denen der Implementierung zu verglichen.

III. Generierung von Testsequenzen mit der Klassifikationsbaum-Methode

Existierende Werkzeuge, wie der CTE XL Professional, zur Unterstützung der Klassifikationsbaum-Methode ermöglichen die regelbasierte Generierung von Testfallmengen für das kombinatorische Testen. Die Generierung beruht auf dem Klassikationsbaum, sowie Abhängigkeitsregeln und Generierungsvorschriften. Über den Klassifikationsbaum werden die Äquivalenzklassen erster Ordnung beschrieben. Mittels Abhängigkeitsregeln lassen sich Abhängigkeiten und Wechselwirkungen zwischen den eingeführten Äquivalenzklassen definieren. Die definierten Abhängigkeitsregeln gelten dabei für jeden einzelnen Testfall gleichermaßen. Die Generierungsvorschriften zur Erzeugung der Testfälle beziehen sich hingegen auf die Gesamtmenge der zu generierenden Testfälle. Die Testfallmenge als Ganzes entspricht einer einzelnen Generierungsvorschrift. Sowohl automatisch generierte Testfälle als auch durch den Benutzer spezifizierte Testfälle lassen sich mit den Abhängigkeitsregeln auf Gültigkeit überprüfen.

Diese drei bewährten Komponenten sollen beibehalten werden, um Anwendern der Klassifikationsbaum-Methode einen möglichst einfachen Einstieg in die Testsequenzgenerierung zu ermöglichen. Um die Klassifikationsbaum-Methode um die Möglichkeit der Sequenzgenerierung zu erweitern, werden die bisher vorhandenen Sprachmittel der Methodik um neue Ausdrucksmöglichkeiten ergänzt. Während die Semantik des Klassifikationsbaums weitgehend beibehalten wird, werden neue

Sprachmittel für die Definition von Abhängigkeiten zwischen den Testschritten einer Testsequenz eingeführt. Jede Abhängigkeitsregel soll wiederum für jede Testsequenz gelten und jede einzelne Testsequenz darf keine Anhängigkeitsregel verletzen. Mit den neuen Abhängigkeitsregeln sollen ebenfalls manuell erzeugte als auch automatische generierte Testsequenzen auf Gültigkeit geprüft werden können. Erweitere Generierungsvorschriften dienen der Erzeugung einer Menge von Testsequenzen.

A. Neue Abhängigkeitsregeln

Mit den erweiterten Ausdrucksmöglichkeiten für die Beschreibung von Abhängigkeiten in Testsequenzen ist es möglich, Abhängigkeiten für die in den Testschritten einer Testsequenz gewählten Klassen zu definieren. Es werden folgenden zusätzliche Sprachmittel eingeführt (mit $i, j, k, n, o \in \mathbf{N}; m \in \mathbf{Z}$):

- Wenn Klasse c_i aus Klassifikation C im Testschritt t_n ausgewählt ist, dann muss Klasse c_j aus Klassifikation C im darauffolgenden Testschritt t_{n+1} der Testsequenz ausgewählt sein.
- Wenn Klasse c_i aus Klassifikation C im Testschritt t_n ausgewählt ist, dann muss Klasse c_j aus Klassifikation C in einem späteren Testschritt t_{n+m} der Testsequenz ausgewählt sein.
- Wenn Klasse c_i aus Klassifikation C im Testschritt t_n ausgewählt ist, dann muss Klasse c_j aus Klassifikation C in allen Testschritten t_{n+1} bis t_{n+m} der Testsequenz ausgewählt sein.
- Wenn Klasse c_i aus Klassifikation C im Testschritt t_n ausgewählt ist, dann muss Klasse c_j aus Klassifikation C in allen Testschritten t_{n+m} bis t_{n+o} der Testsequenz ausgewählt sein.
- Zusätzlich sind Kompositionen mit den bereits aus den bisherigen Abhängigkeitsregeln bekannten logischen Operatoren möglich (*AND*, *OR*, *NOT*, *NAND*, *NOR*, *XOR*, ...) so dass sich auch komplexere Abhängigkeitsregeln für Testsequenzen definieren lassen.

Ein Beispiel: Wenn in Testschritt t_n Klasse *ja* zur Klassifikation *Vollbremsung* ausgewählt ist *ODER* wenn Klasse *ja* zur Klassifikation *Notbremsung* ausgewählt ist, *dann NOT* Klasse $>= 100km/h$ für Klassifikation *Fahrzeuggeschwindigkeit* im späteren Testschritt t_{n+m} (siehe Abb. 1).

Die existierenden Abhängigkeitsregeln bilden eine Untermenge der neuen Abhängigkeitsregeln mit $m = 0$ für beliebige t_n und t_{n+m}.

B. Neue Generierungsvorschriften

Die neuen Generierungsvorschriften lassen sich über die folgenden Parameter steuern:

- Gewünschter Abdeckungsgrad.
- (Unter-)Menge aller Kombination von Klassen des Klassifikationsbaumes.
- Minimale und maximale Anzahl von Testschritten pro Testsequenz.
- Definierte Menge von zugelassenen Klassenkombinationen für den ersten und den letzten Testschritt von Testsequenzen.
- Maximale Anzahl von Testschritt-Wiederholungen (sowohl unmittelbar aufeinander folgende als auch innerhalb der gesamten Testsequenz).
- Maximale Anzahl von Klassen-Wiederholungen (sowohl unmittelbar aufeinander folgende als auch innerhalb der gesamten Testsequenz).

C. Prototypische Umsetzung im CTE XL Professional

Als interne Repräsentationen für die automatische Testsequenzgenerierung werden Entscheidungsbäume und Zustandsautomaten verwendet.

Die neuen Abhängigkeitsregeln basieren auf der Verwendung von *Linearer temporale Logik* (LTL) mit den Operatoren (X, G, F, U, R). LTL-Formeln passen gut zu unseren Anforderungen, die sich aus der Definition von Abhängigkeiten zwischen den Testschritten einer Testsequenz ergeben.

Für die Generierungsvorschriften wird auf *Computation Tree Logic* (CTL) zurückgegriffen. CTL Regeln gelten nicht pro erzeugter Testsequenz sondern beziehen sich auf die Gesamtheit aller generierten Testsequenzen.

Die beschriebenen Erweiterungen der Abhängigkeits- und Generierungsregeln werden derzeit im CTE XL Professional prototypisch umgesetzt und in ersten Projekten pilotiert.

IV. Zusammenfassung

In dieser Arbeit haben wir einen Ansatz für die automatische Testsequenzgenerierung aus Klassifikationsbäumen vorgestellt. Wir haben dafür neue Abhängigkeitsregeln und neue Generierungsvorschriften definiert. Die neuen Abhängigkeitsregeln bilden eine Obermenge der bestehenden Abhängigkeitsregeln, die um Ausdrucksmöglichkeiten für Zusammenhänge zwischen mehreren Testschritten erweitert worden sind. Die neuen Generierungsvorschriften enthalten Parameter zur Definition des Testumfangs, sie erlauben die Spezifikation Abdeckungsgraden und Wiederholungen. Damit ist eine systematische Generierung von Testsequenzen zu Klassifikationsbäumen möglich.

Mit Abschluss der prototypischen Implementierung der Testsequenzgenerierung im CTE XL Professional werden sich künftige Arbeiten zunächst auf die Evaluierung des vorgestellten Ansatzes konzentrieren. Insbesondere die Fragen nach Skalierbarkeit und Performance werden dabei im Fordergrund stehen. Danach soll eine reguläre Umsetzung der Testsequenzgenerierung im CTE XL Professional erfolgen.

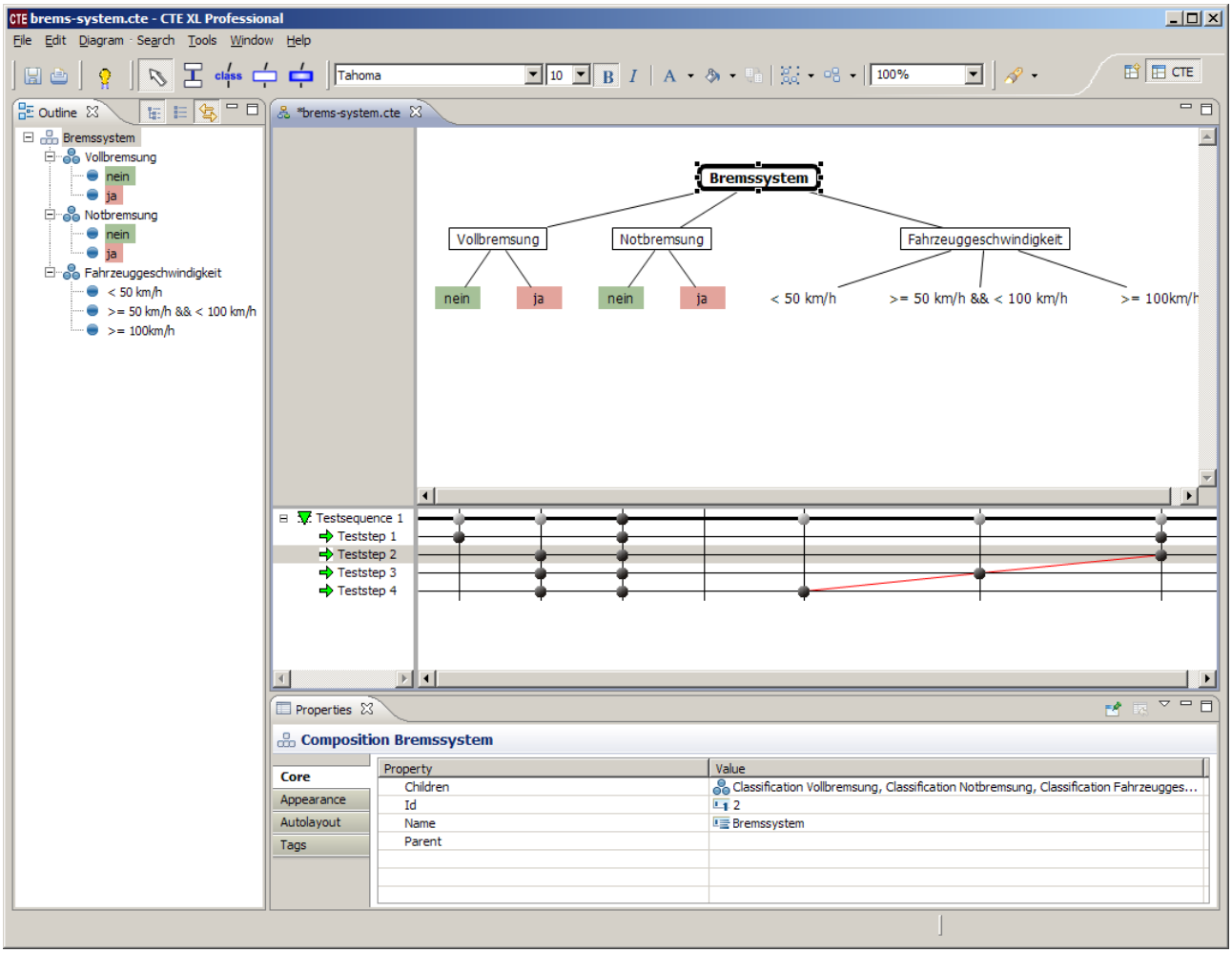

Fig. 1. CTE XL Professional Beispiel

References

[1] M. Grochtmann and K. Grimm, "Classification trees for partition testing," *Softw. Test., Verif. Reliab.*, vol. 3, no. 2, pp. 63–82, 1993.

[2] "Cte xl professional," 2010. [Online]. Available: http://www.berner-mattner.com/de/berner-mattner-home/produkte/cte/cte-xl-professional/

[3] E. Lehmann and J. Wegener, "Test case design by means of the CTE XL," *Proceedings of the 8th European International Conference on Software Testing, Analysis and Review (EuroSTAR 2000), Kopenhagen, Denmark, December*, 2000.

[4] M. P. Heimdahl, S. Rayadurgam, W. Visser, G. Devaraj, and J. Gao, "Auto-generating test sequences using model checkers: A case study," in *3rd International Worshop on Formal Approaches to Testing of Software (FATES 2003)*, 2003.

[5] M. Y. Vardi, "Branching vs. linear time: Final showdown," in *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS 2001, 2001, pp. 1–22.

[6] A. Krupp and W. Müller, "Modelchecking von klassifikationsbaum-testsequenzen," 1 Apr. 2005, gI/ITG/GMM Workshop "Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen", München.

[7] G. Wimmel, H. Loetzbeyer, A. Pretschner, and O. Slotosch, "Specification based test sequence generation with propositional logic," 2000.

[8] E. Bernard, B. Legeard, X. Luck, and F. Peureux, "Generation of test sequences from formal specifications: Gsm 11-11 standard case study," *Softw. Pract. Exper.*, vol. 34, pp. 915–948, August 2004.

[9] R. V. Binder, *Testing object-oriented systems: models, patterns, and tools.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[10] H. Ural, "Formal methods for test sequence generation," *Comput. Commun.*, vol. 15, pp. 311–325, June 1992.