Klassifikationsbäume in Echtzeit

Martin Leucker, Moritz Löser, Dirk Nowotka
Universität Stuttgart

Joachim Rischen
Bosch Engineering GmbH

1 Überblick

Im Rahmen des äquivalenzklassenbasierten Testens werden Eingabegrößen eines zu entwickelnden Systems in Äquivalenzklassen eingeteilt. Da in der Praxis erwartet werden kann, dass sich ein System bei äquivalenten Eingaben gleich verhält, reduziert sich der Aufwand des Testens eines Systems auf das Testen von Vertretern der einzelnen Äquivalenzklassen.

Die Klassifikationsbaum-Methode [4] ist eine Methodik zur interativen und strukturierten Definition und Beschreibung einzelner Äquivalenzklassen, die überlicherweise aus einem Anforderungsdokument gewonnen werden. Für die Methodik gibt es mit dem Classification-tree Editor (CTE) [1] auch eine adäquate Werkzeugunterstützung.

Für die resultierenden Äquivalenzklassen können dann überlichweise automtisch geeignete Testfälle generiert und auf dem zu testenden System ausgeführt werden. Der manuelle Aufwand des Testens wird somit auf die Erstellung der einzelnen Äquivalenzklassen reduziert.

Im automotive Umfeld findet die Äquivalenzklassenmethode häufig insbesondere bei OEM Einsatz, da in diesem Umfeld umfangreiche Anforderungsdokumente vorliegen, die, wenngleich mit viel Aufwand, zur Definition von geeigneten Äquivalenzklassen herangezogen werden können.

Ein Problem der Äquivalenzklassenmethode stellt die Definition von Echtzeitanforderungen und insbesondere die sich anschliessende Generierung geeigneter Testfälle dar. Somit lassen sich auch komplexe Signalverläufe nur unzureichend in geeignete Äquivalenzklassen zerlegen, die zeitliche Aspekte berücksichtigen.

In diesem Abstract wird eine Erweiterung der Klassifikationsbaummethode zur Definition von Echtzeitbedinungen vorgestellt, die eine automatische Generierung von Testfällen für die resultierenden Äquivalenzklassen erlaubt. Aufbauend auf dieser Methode wurde ein Testprozess definiert, der seit kurzem bei Bosch Engineering eingesetzt wird. Durch die Verwendung des Testprozesses wurde in geeigneten Testprojekten bereits eine wesentliche Kostenersparnis erzielt.

2 Contribution

Die vorgestellte Arbeit erweitert die Klassifikationsbaummethode zur Spezifikation von Testfällen um Echtzeitbedingungen unter der Bedingung das in der Testfallgenerierung diejenigen Äquivalenzklassen, die nicht in den Anforderungen genannt werden aber trotzdem gültig sind, in jeder möglichen Kombination getestet werden. Genau diese Kombination stellt herkömmliche Verfahren vor große Probleme.

Bisherige Ansätze zur Erweiterung der Klassifikationsbaummethode um das Zeitverhalten von Tests, wie in [2] und [3] beschrieben, eignen sich nur bedingt für unsere Zwecke. In [2] wird ein Testfall in ein Sequenz von Testfällen unterteilt, um die zeitliche Abfolge von Äquivalenzklassen zu beschreiben. Dieser Ansatz hat zum Nachteil, dass er erstens das Zeitverhalten vom Äquivalenzklassenbaum trennt und zweitens die unübersichtliche Darstellung von Sequenzen mit vielen Schritten. Der zweite Nachteil wurde zwar in der Arbeit [3] adressiert, aber die prinzipiellen Einwände gegen die Trennung des Zeitverlaufs vom Äquivalenzklassenbaum bleiben bestehen. Diese Trennung heben wir mit der vorgestellten Darstellung auf, was sich außerdem positiv auf die Lösung des Problems der Testfallgenerierung mit erschöpfender Kombination von Äquivalenzklassen, die nicht in den Anforderungen aufgeführt werden aber trotzdem gültig sind, aus. Außerdem werden in unserem Ansatz Ausgabewerte zusammen mit den Eingaben modelliert. Dies erlaubt die automatische Generierung nicht nur von Eingaben für Testfälle sondern gleichzeitig auch die erwartete Ausgabe, so dass die generierten Testfälle automatisch auf dem zu untersuchenden System sowohl abgelaufen und das System hinsichtlich Korrektheit überprüft werden kann.

Die vorgestellte Erweiterung des Äquivalenzklassentests wird semantisch (intern) mit Hilfe von Zeitautomaten modelliert. Zeitautomaten bilden einen ausdrucksstarken Formalismus, der sich besonders bei der Erzeugung der Testfälle auszahlt. Weitere Einzelheiten sind in [5] beschrieben.

Abbildung 1 dient als laufendes Beispiel zur Illustration unseres Ansatzes.

Die zwei beschriebenen Testfälle stehen für die folgenden Anforderungen: (1) „Wenn TargetStateBBF von enable auf finishing wechselt und vVeh_Status den Wert invalid besitzt und vVeh für mindestens $0,4$s unter 3km/h ist dann setzt BBF ReqType auf den Wert RelReq_BBF und bActive auf den Wert false.“ und (2) „Wenn vVeh_Status den Wert invalid besitzt dann setzt BBF ReqType auf den Wert RelReq_BBF und bActive auf den Wert false.“

Der erste Testfall verlangt, dass die beiden Forde-

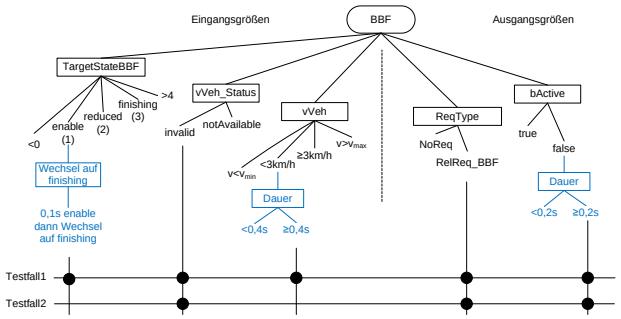

Abbildung 1: BBF-Beispiel

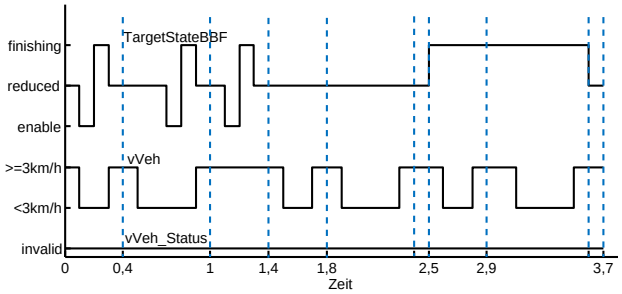

Abbildung 3: Kombination

rungen „0, 1s enable und dann Wechsel auf finishing“ für TargetStateBBF und „unter 3km/h für mindestens 0, 4s“ für vVeh im Testfall zum selben Zeitpunkt realisiert werden müssen. Eine Kunjunktion von Stellen der entsprechenden Zeitautomaten zusammen mit einem geeigneten Füllwert setzt diese Anforderung um (siehe Abb. 2). Zur Realisierung des zweiten Testfalls ist die sequenzielle Kombination von insgesamt neun Kombinationen (inklusive der Kombination von zeitbehafteten Klassen) notwendig. Im vorgestellten Ansatz entsprechen diese Kombinationen eine Kombination der Zeitautomaten für jede Äquivalenzklasse derart, dass im Fall festgelegter Anforderungen bestimmte Stellen und im Fall offener Anforderungen alle Stellen durchlaufen werden müssen.

Abbildung 3 illustriert das Resultat einer solchen Kombination für den zweiten Testfall. Gerade diese Möglichkeit der automatischen Testfallgenerierung in einer mit Echtzeitbedingungen erweiterten Klassifikationsbaummethode und einer vollständigen Kombination nicht mit Anforderungen belegten Äquivalenzklassen stellt eine Besonderheit des vorgestellten Ansatzes dar.

3 Evaluation

Das hier vorgestellte Verfahren wird seid kurzem im Modultest bei Bosch Engineering produktiv eingesetzt. Die ersten Ergebnisse sind sehr vielversprechend. Wie erwartet konnte insbesondere für Tests von Echtzeitsystemen sowohl die Testeffizienz als auch die Testeffektivität erhöht werden. Durch die

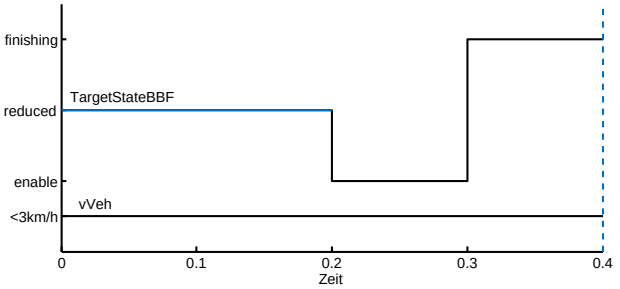

Abbildung 2: Füllwert

Einführung der Klassifikationsbaummethode wurde zudem die Qualität der Dokumentation verbessert. Damit einhergehend ist eine Verbesserung der Tests bezüglich ihrer Überprüfbarkeit und Anpassbarkeit.

Ein wesentlicher Nachteil der Klassifikationsbaum-Methode, nämlich der hohe zeitliche Aufwand zur Erstellung der Klassifikationsbäume, bleibt weiterhin erhalten. Insbesondere die für die Prozessautomatisierung nötige Attributierung der Klassifikationsbäume ist sehr zeitintensiv. Zum einen die hohe Anzahl der Attribute aber auch einige Mängel in der Benutzerschnittstelle des CTE sind die Ursachen dafür.

Weiterhin besteht die Problematik, dass im Rahmen der Klassifikationsbaumethode genau eine oder keine Klasse pro Klassifikation gewählt werden muss, so dass ungleich- und oder-Bedingungen nicht direkt darstellbar sind. Um im obigen Sinn vollständig zu Testen muss für jede Möglichkeit solche Bedingungen zu erfüllen, ein Testfall angelegt werden. Als Ausweg könnte zukünftig der in [3] vorgestellte Ansatz dienen, in dem Klassen durch verschiedene Symbole mit entsprechender Semantik markiert werden können. Eine Integration der in [3] mit den hier vorgestellten Erweiterung um Realzeit-Bedingung scheint jedoch problemlos möglich und ist Teil zukünftiger Arbeit, die Modifikationen am CTE nach sich ziehen.

Literatur

[1] Classification Tree Editor.

[2] M. Conrad. Systematic testing of embedded automotive software - the classification-tree method for embedded systems (ctm/es). In E. Brinksma, W. Grieskamp, and J. Tretmans, editors, *Perspectives of Model-Based Testing*, number 04371 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2005.

[3] M. Conrad and A. Krupp. An extension of the classification-tree method for embedded systems for the description of events. *Electronic Notes in Theoretical Computer Science*, 164(4):3–11, 2006.

[4] K. Grimm and M. Grochtmann. Classification trees for partition testing. *Software Testing, Verification and Reliability*, 3(2):63–82, 1993.

[5] M. Löser. äquivalenzklassen-basiertes testen. Master's thesis, Universtität Stuttgart, 2008.