Time Partition Testing: Systematischer Test eingebetteter Systeme

Eckard Lehmann, DaimlerChrysler AG, Forschung und Technologie

Zusammenfassung

Das Time Partition Testing (TPT) ist ein spezialisiertes Testverfahren für den automatischen Funktionstest des kontinuierlichen Verhaltens eingebetteter Systeme, das insbesondere auch reaktive Tests unterstützt. Charakteristisch für TPT ist die gemeinsame Modellierung *aller* relevanten Testfälle für ein Testobjekt, bei der alle testrelevanten Aspekte systematisch betrachtet werden.

Mit TPT modellierte Testfälle sind dank grafischer Notation intuitiv lesbar; hinter jedem Testfall verbirgt sich dennoch eine formal präzise Beschreibung. Da diese Beschreibung plattform-unabhängig ist, sind die Testfälle in verschiedenen Umgebungen vollautomatisch ausführbar. Durch eine integrierte automatisierte Testauswertung ist TPT eine durchgängige Lösung von der Testmodellierung bis zur Auswertung.

Keywords. Eingebettete Systeme, systematischer Test, automatischer Test, kontinuierliches Verhalten, hybride Systeme, Klassifikationsbaum-Methode.

1 Motivation.

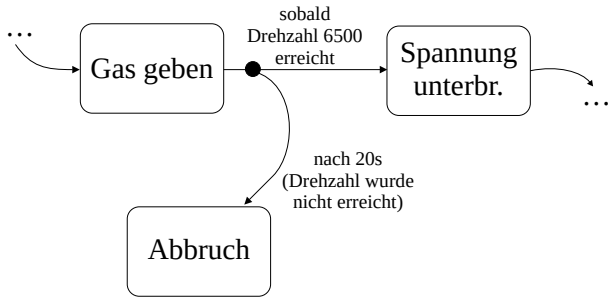
Eingebette Systeme – insbesondere eingebettete Regelungssysteme – weisen häufig kontinuierliches Verhalten auf. Kontinuierliches Verhalten unterscheidet sich von sequentiellem Verhalten darin, dass nicht nur zeitdiskrete Ereignisse und Daten (wie z.B. bei Kommunikationsprotokollen), sondern auch kontinuierliche Abläufe, wie sie bei Regelungssystemen üblich sind, betrachtet werden.

Testfälle für kontinuierliches Verhalten haben oft einen reaktiven Charakter, d.h., Tests sollen auf das Systemverhalten „reagieren“. Ein einfaches Beispiel aus dem Fahrzeugbereich ist der Test des Verhaltens einer Motorsteuerung, wenn bei einer Dehzahl von 6500min^{-1} plötzlich die Versorgungsspannung ausfällt. Der Testfall muss hierbei die Gaspedalstellung beeinflussen, um die gewünschte Drehzahl anzusteuern. Erst wenn diese Drehzahl erreicht ist, soll der Testfall die Versorgungsspannung unterbrechen. Je nach Motorisierung und anderen Faktoren kann dieser Zeitpunkt früher oder später sein.

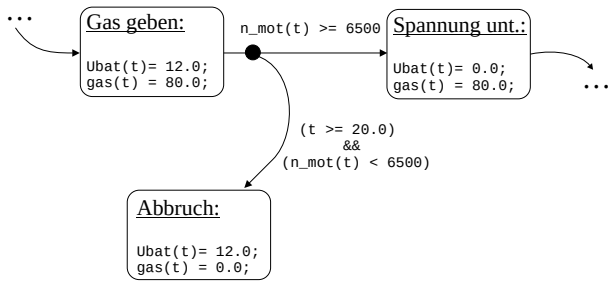
Das Time Partition Testing wurde entwickelt, um diese Klasse von Tests systematisch, intuitiv lesbar und dennoch automatisierbar testen zu können.

2 Modellierung von Testfällen

Mit TPT wird jeder Testfall mit Hilfe von hierachischen hybriden Automaten modelliert [2]. Hierbei wird ein Testfall in eine Folge von einzelnen Zeitphasen zerlegt. Der Zeitpunkt des Übergangs zwischen den Phasen wird an Transitionsbedingungen geknüpft, die reaktive Tests möglich machen. Gehen von einem Zustand mehrere Transitionen ab, sind auch Verzweigungen möglich, so dass der Test je nach Systemverhalten verschieden ablaufen kann. Für das oben beschriebene Beispiel könnte ein Teil des Ablaufs folgendermaßen modelliert werden:



Hinter jedem Zustand verbirgt sich auf unterster Ebene eine formale Beschreibung der kontinuierlichen Verläufe der zu stimulierenden Größen: Jede Größe wird dabei punktweise als Funktion über der Zeit definiert. Entsprechend ist jeder Transition eine formale Bedingung zugeordnet. In dem einfachen Beispiel werden die Größen `Ubat` (Versorgungsspannung in Volt) und `gas` (Gaspedalstellung in Prozent) phasenweise konstant modelliert. Der Phasenübergang hängt vom Wert der Systemgröße `n_mot` (Motordrehzahl in min^{-1}) ab. In der folgenden Darstellung sind die formalen Definitionen mit eingeblendet:



Durch die Möglichkeit, Hierarchien von Automaten zu bilden, lassen sich auf diese Weise selbst komplexe reaktive Testszenarien übersichtlich modellieren. Durch die grafische Darstellung sowie die Verwendung natürlichsprachlicher Annotationen sind Testfälle auch für Nicht-Programmierer lesbar.

TPT unterstützt die Modellierung weiterhin durch Techniken wie Actions an Transitionen, nicht-flüchtige Größen, finale Transitionen, user-defined functions u.v.m. Eine ausführliche Darstellung aller Details ist im Rahmen dieses Papiers nicht möglich.

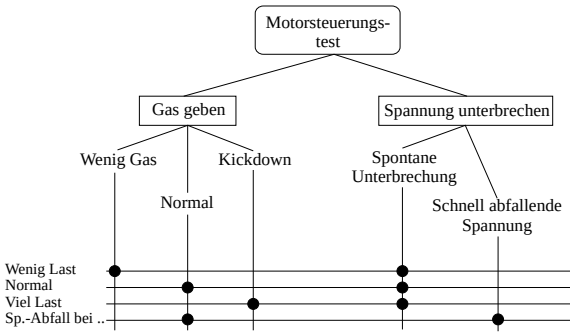
3 Systematische Strukturierung von Testfällen

Bislang ist TPT damit „nur“ eine Sprache zur Modellierung von reaktiven, kontinuierlichen Testfällen.

Die volle Mächtigkeit von TPT kommt erst dann zum Tragen, wenn zu einem Testobjekt die Menge *aller* relevanten Testfälle modelliert werden soll.

Es lässt sich beobachten, dass sich Testfälle zu einem Testobjekt häufig nur in wenigen spezifischen, klar lokalisierbaren Details unterscheiden, während sich die Grobstruktur der Fälle ständig wiederholt. Bezogen auf TPT bedeutet dies, dass der grundsätzliche Ablauf einzelner Phasen des Tests (Gas geben → Spannung unterbrechen → ...) für alle Testfälle gleich ist. Das Verhalten innerhalb einer Phase unterliegt jedoch Variationen. Im obigen Beispiel sind für die Phase „Gas geben“ unterschiedliche testrelevante Varianten denkbar: 1. „wenig Gas geben“, 2. „normal“, 3. „Kickdown“. Zusätzlich könnten unterschiedliche Varianten zur Phase „Spannung unterbrechen“ testrelevant sein: 1. „spontane Unterbrechung“, 2. „schnell abfallende Spannung“.

Hinter einem Zustand des Automaten verbirgt sich demnach nicht nur *eine* formale Beschreibung, sondern beliebig viele, die jeweils unterschiedliche Varianten für diese Phase bilden. Will man nunmehr einen konkreten Gesamttestfall definieren, muss zu jedem Zustand individuell festgelegt werden, welche Variante in diesem Gesamttestfall verwendet werden soll. Anders ausgedrückt: Sind die Varianten der Zustände definiert, müssen diese Varianten zu Gesamttestfällen sinnvoll kombiniert werden. Hierfür eignet sich die Klassifikationsbaum-Methode hervorragend, die die systematische Kombination von Varianten verschiedlicher Aspekte unterstützt [1, 3].



In dem Beispiel gibt es $2 \cdot 3 = 6$ mögliche Kombinationen, wie Gesamttestfälle gebildet werden können. Von diesen 6 Möglichkeiten wurden 4 Kombinationen als testrelevant erachtet: Jede Zeile der obigen Kombinationstabelle beschreibt einen dieser vier Fälle.

Jeder Gesamttestfall ist durch die Auswahl der zu verwendenden Varianten eindeutig und formal präzise beschrieben, so dass die so definierten Testfälle bereits ausführbar sind. Die Klassifikationsbaum-Methode ermöglicht durch die grafisch anschauliche Darstellung der Kombinatorik, den Überblick über die betrachteten Testfälle und die testrelevanten Kombinationen zu wahren.

4 Testdurchführung

Die mit TPT modellierten Testfälle haben eine präzise Semantik, die unabhängig von der konkreten Testumgebung oder der Plattform des Testobjekts ist. Dadurch kann TPT in unterschiedlichsten Bereichen zum Einsatz kommen. Für die automatisierte Testdurchführung ist es lediglich notwendig, einen plattform-spezifischen Testtreiber zu entwickeln, der TPT-Testfälle ausführen kann, falls ein solcher Treiber noch nicht existiert.

Weiterhin ist die Plattform-Unabhängigkeit äußerst nützlich, wenn ein und dasselbe System in unterschiedlichen Testumgebungen getestet werden soll, wie es bei eingebetteten Systemen üblich ist. In diesem Fall können Testfälle unverändert in verschiedenen Umgebungen verwendet werden.

Zur Zeit existieren bei DaimlerChrysler zwei TPT-Testtreiber: eine Referenz-Implementierung auf Java-Basis sowie ein Treiber für eine Getriebe-Simulationsumgebung.

5 Testauswertung

TPT beinhaltet auch Techniken für die automatische Testauswertung. Hierzu werden den einzelnen Phasen sogenannte *assessments* zugeordnet, die die Resultate der Durchführung hinsichtlich des Sollverhaltens bewerten. Hierfür stehen komplexe Hilfsmittel wie beispielsweise der Vergleich mit Referenzmessungen, Analysen der Dauer bestimmter Testphasen usw. zur Verfügung. Die Testauswertung ist von der Durchführung entkoppelt – findet also „offline“ mit Hilfe des TPT-Auswertungstools statt.

6 Praktischer Einsatz

Für TPT existiert eine bei DaimlerChrysler entwickelte Werkzeugumgebung, die die Modellierung, Durchführung (in spezifischen Testumgebungen) und Auswertung von Tests unterstützt. TPT wird bei DaimlerChrysler bereits im Rahmen der Entwicklung einer Getriebesteuerung aktiv und erfolgreich eingesetzt. Weitere Projekte sind für 2002 geplant.

Literatur

[1] Matthias Grochtmann. Test Case Design Using Classification Trees. In *Proceedings of STAR'94*, pages 93–117, Washington, DC, 1994.

[2] Thomas A. Henzinger. The Theory of Hybrid Automata. In *11th IEEE Symposium on Logics of Computer Science*, pages 287–293, 1996.

[3] Eckard Lehmann and Joachim Wegener. Test Case Design by Means of the CTE XL. In *8th European International Conference on Software Testing, Analysis & Review (EuroSTAR 2000)*, Copenhagen, Denmark, 2000.