

Automatisiertes Testen asynchroner nichtdeterministischer Systeme mit Daten

vorgelegt von
Diplom-Informatiker
Dirk Seifert

von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Sahin Albayrak
Berichtende: Prof. Dr. Stefan Jähnichen
Prof. Dr. Ina Schieferdecker
PD Dr. Thomas Santen

Tag der wissenschaftlichen Aussprache: 20. Juli 2007

Berlin 2007

D 83

Berichte aus der Informatik

Dirk Seifert

**Automatisiertes Testen asynchroner
nichtdeterministischer Systeme
mit Daten**

D 83 (Diss. TU Berlin)

Shaker Verlag
Aachen 2007

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Zugl.: Berlin, Techn. Univ., Diss., 2007

Copyright Shaker Verlag 2007

Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN 978-3-8322-6579-3

ISSN 0945-0807

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen

Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9

Internet: www.shaker.de • E-Mail: info@shaker.de

Zusammenfassung

Eingebettete Systeme setzen sich aus Hardware- und Softwarekomponenten zusammen, die asynchron miteinander kommunizieren und nichtdeterministisches Verhalten aufweisen können. Dies macht eine umfassende und systematische Prüfung mit manuellen Techniken nahezu unmöglich. Auch automatisierte Prüftechniken können mit Asynchronität und Nichtdeterminismus bislang nur unzureichend umgehen. Der breite Einsatz eingebetteter Systeme im täglichen Leben und das blinde Vertrauen in deren Funktionsfähigkeit erfordern jedoch eine zuverlässige und umfassende Qualitätssicherung.

Die vorliegende Arbeit beschäftigt sich mit dem automatisierten Testen asynchroner nichtdeterministischer Systeme mit Daten. Zur Beschreibung des reaktiven Verhaltens von Komponenten technischer Systeme werden Zustandsmaschinen der *Unified Modeling Language* in reduzierter Form verwendet. Ausgehend von einer solchen Zustandsmaschine werden automatisiert Testfälle für einen Konformitätstest abgeleitet und gegen das zu testende System ausgeführt. Das korrekte, mögliche Verhalten wird für ausgewählte Eingaben a priori berechnet und in einem Testfall gespeichert. Die Speicherung ermöglicht eine wiederholte Ausführung der Testfälle nach Änderungen in der Entwicklung. Der Hauptbeitrag der Arbeit liegt einerseits in der vollständigen Formalisierung der reduzierten Zustandsmaschinen, die die wesentlichen syntaktischen Ausdrucksmittel und komplex strukturierte Daten umfassen, und andererseits in der Entwicklung eines praktikablen Ansatzes zum automatisierten Testen reaktiver technischer Systeme, deren Verhalten durch solche Zustandsmaschinen beschrieben werden kann. Das Testen solcher Systeme ist aus zwei Gründen eine besondere Herausforderung. Einerseits ist die Anzahl möglicher Eingabesequenzen und damit die Anzahl der möglichen Testfälle unendlich groß. Andererseits führen die asynchrone Kommunikation und der inhärente Nichtdeterminismus in Zustandsmaschinen für jede einzelne Eingabesequenz zu einem sehr komplexen Verhalten. Die präzise und widerspruchsfreie Interpretation dieses komplexen Verhaltens ist notwendige Voraussetzung für jegliche Art von Automatisierung, erfordert jedoch einen enorm hohen Berechnungsaufwand und ist damit nicht immer praktikabel. Um die Anzahl der Testfälle sinnvoll zu beschränken werden Testspezifikationen verwendet. In Testspezifikationen werden Art und Umfang der betrachteten Eingaben festgelegt und relevante Eingabesequenzen durch Nutzungsprofile beschrieben. Einfache Nutzungsprofile erlauben die probabilistische Auswahl von Eingaben oder die Vorgabe fester Eingabesequenzen. Komplexere Nutzungsprofile beschreiben das Verhalten der Umgebung explizit in Form von Zustandsmaschinen mit probabilistischen Zustandsübergängen und ermöglichen damit eine systematische und automatisierte Auswahl relevanter Eingaben. Durch ein Nutzungsprofil kann zum Beispiel die technische Umgebung, in die das System später eingebettet werden soll, explizit modelliert werden.

Die Bestimmung des möglichen korrekten Verhaltens für eine gegebene Eingabesequenz erfolgt im vorgestellten Testansatz in einer schrittweisen Ausführung der Zustandsmaschine. Da der Berechnungsaufwand mit zunehmender Länge der betrachteten Eingaben exponentiell ansteigt und deshalb eine Berechnung für typische Sequenzlängen unmöglich ist, können kürzere Eingabesequenzen mehrerer Testfälle kombiniert werden und für diese jeweils das korrekte

mögliche Verhalten bestimmt werden. Die Einfügung von Beobachtungspunkten nach jeder Eingabesequenz führt dabei zu einer erheblichen Reduzierung des Berechnungsaufwands für die folgenden Eingaben, wobei das mögliche korrekte Verhalten approximiert wird. Der Berechnungsaufwand steigt dabei linear mit der Anzahl der kombinierten Testfälle. Durch eine geeignete Wahl der Anzahl an Teilsequenzen, aus denen eine Eingabesequenz zusammengesetzt wird, kann hinsichtlich des Aufwands für die Testfallerzeugung und der Erkennungsrate der Testfälle ein guter Kompromiss erzielt werden.

Praktisches Ergebnis der Arbeit ist die prototypische Werkzeugumgebung TEAGER. Diese setzt sich auf der einen Seite aus einer Umgebung zur Testfallerzeugung und -ausführung und auf der anderen Seite aus einer Umgebung zur Simulation von Zustandsmaschinen zusammen. Letztere ermöglicht eine realitätsnahe Ausführung von Zustandsmaschinen, d. h. insbesondere, dass nichtdeterministisches und zeitlich variables (probabilistisches) Ausführungsverhalten simuliert werden kann. Zum einen kann damit eine Zustandsmaschine, z. B. die Spezifikation, schon in einer frühen Entwicklungsphase analysiert werden und zum anderen konnte der von mir entwickelte Ansatz zum Testen eingebetteter Systeme selbst validiert werden. Mit der Werkzeugumgebung TEAGER habe ich die Anwendbarkeit des Ansatzes anhand kleinerer Beispiele und zweier Fallstudien demonstriert. Die durchgeführten Untersuchungen zeigen, dass durch die Verhaltensapproximation mit der Komplexität und der Zustandsexplosion innerhalb von Zustandsmaschinen umgegangen, eine akzeptable Erkennungsrate erzielt und die Ablehnung korrekter Systeme vermieden werden kann.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Einleitung und Motivation	1
1.2	Zielsetzung der Arbeit	3
1.3	Aufbau der Arbeit	6
I	Theoretische Grundlagen	9
2	Grundlagen	11
2.1	Softwareentwicklung	11
2.2	Qualitätssicherung	14
2.3	Testen	16
2.4	Unified Modeling Language	23
2.5	Problembereich	25
3	Zustandsmaschinen mit Daten	27
3.1	Einführung	28
3.2	Beispiel: Musikanlage	29
3.3	Abstrakte Syntax	36
3.4	Semantik	43
3.5	Mögliche Erweiterungen	52
3.6	Zusammenfassung	54

4	Testen auf Basis von Zustandsmaschinen	57
4.1	Testhypothese	58
4.2	Implementierungsrelation	60
4.3	Spezifikation von Testfällen	69
4.4	Ableitung von Testfällen	76
4.5	Kombination von Testfällen	87
4.6	Aufwandsbetrachtung	91
4.7	Zusammenfassung	94
5	Weiterführende Themen	99
5.1	Überdeckungsmessung zur Testbewertung	99
5.2	Testfallableitung mit Daten	103
5.3	Anforderungstest	105
5.4	Zusammenfassung	109
II	Praktische Evaluierung	111
6	Werkzeugumgebung TEAGER	113
6.1	Architektur der Werkzeugumgebung TEAGER	114
6.2	Konfigurationsansicht	116
6.3	Spezifikationsansicht	122
6.4	Testansicht	123
6.5	Simulator für Zustandsmaschinen	125
6.6	Zusammenfassung	127
7	Fallstudien	129
7.1	Musikanlage	131
7.2	Jalousiesteuerung	141
7.3	Zusammenfassung	154

III Diskussion	155
8 Fazit	157
8.1 Verwandte Arbeiten	157
8.2 Ausblick	162
8.3 Zusammenfassung und Ergebnisse	165
 Literaturverzeichnis	 171
 IV Anhänge	 181
A Semantischer Schritt	183
A.1 Implementierung	183
A.2 Beispiel	186
 B Grammatiken	 189
B.1 Zustandsmaschinen	189
B.2 Testfälle	198
 C Modelle der Jalousiesteuerung	 201
 D Deutsch-Englische Begriffswelt	 207

Abbildungsverzeichnis

1.1	Schematische Darstellung eines eingebetteten technischen Systems.	2
1.2	Zusammenhänge zwischen den Teilzielen.	6
3.1	Vereinfachtes Bedienfeld der Musikanlage.	29
3.2	Datenlose Spezifikation der Musikanlage.	31
3.3	Baumstruktur der Zustandsmaschine der Musikanlage.	33
3.4	Datenbehaftete Spezifikation der Musikanlage	35
3.5	Zusammensetzung der Mengen von Ereignisinstanzen	51
4.1	Ausgangssituation für die Durchführung von Tests.	58
4.2	Zusammenhänge, die sich aus der <i>Testhypothese</i> ergeben.	59
4.3	Abstrakte Testarchitektur.	62
4.4	Zusammenhänge zwischen den Testschnittstellen.	68
4.5	Umgebungsmodell des Windes der Jalousiesteuerung.	72
4.6	Ablaufschema der Testableitung in der Werkzeugumgebung TEAGER.	77
1	Algorithmus: Vollständige Verhaltensbestimmung (Kontrollstruktur).	80
4.7	Schrittweise Ausführung einer Zustandsmaschine für $[a, b, c]$	81
2	Algorithmus: Vollständige Verhaltensbestimmung (semantischer Schritt).	83
4.8	Hülle mit den berechneten Beobachtungssequenzen.	84
4.9	Nicht-terminierende Ausführung einer Zustandsmaschine.	85
4.10	Akzeptanzgraph zur Eingabesequenz $[a, b, c]$	86
4.11	Akzeptanzgraph mit <i>inconclusive</i> Testurteil.	87
4.12	Struktur eines kombinierten Testfalls.	89

3	Algorithmus: Anpassung für die Ableitung kombinierter Testfälle.	90
4.13	Linearisierung des exponentiellen Algorithmus.	93
5.1	Zusammenhänge zwischen den Teilzielen.	100
5.2	Ablauf eines Anforderungstests.	106
5.3	Muster zur Modellierung von Anforderungen mit Zustandsmaschinen.	107
5.4	Testfallgenerierung bei gleichzeitiger Überprüfung der Anforderungen.	108
6.1	Abstrakte Architektur der Werkzeugumgebung TEAGER.	114
6.2	Konfigurationsansicht des TCGD.	119
6.3	Spezifikationsansicht des TCGD.	123
6.4	Testansicht des TCGD.	124
6.5	Ansicht des State Machine Executors.	126
7.1	Messergebnisse des ersten Experiments für P_1 (Musikanlage).	133
7.2	Messergebnisse des ersten Experiments für P_2 (Musikanlage).	134
7.3	Messergebnisse des zweiten Experiments (Musikanlage).	137
7.4	Messergebnisse des dritten Experiments für P_1 und P_2 (Musikanlage).	140
7.5	Messergebnisse des ersten Experiments für P_1 (Jalousiesteuerung).	148
7.6	Messergebnisse des ersten Experiments für P_2 (Jalousiesteuerung).	149
7.7	Messergebnisse des zweiten Experiments (Jalousiesteuerung).	151
7.8	Messergebnisse des dritten Experiments für P_1 und P_2 (Jalousiesteuerung).	153
A.1	Zustandsmaschine zum Programmausdruck A.2	187
C.1	Systemarchitektur der Jalousiesteuerung.	201
C.2	Softwarearchitektur der Jalousiesteuerung.	202
C.3	Zustandsmaschine des Jalousie-Controllers.	203

Definitionsverzeichnis

Definition 1	Datenraum	36
Definition 2	Knotenmenge	37
Definition 3	Typfunktion für Knoten	37
Definition 4	Mengen der Unterknoten	38
Definition 5	Wohlgeformte Knotenhierarchie	39
Definition 6	Wurzelknoten einer Knotenhierarchie	39
Definition 7	Containerfunktion	40
Definition 8	Menge der Ereignisinstanzen	40
Definition 9	Menge der Übergangsbedingungen	41
Definition 10	Menge der Aktionen	42
Definition 11	Menge der Transitionen	42
Definition 12	Wohlgeformte Transitionsmenge	42
Definition 13	Struktur einer Zustandsmaschine	43
Definition 14	Menge der Zustandskonfigurationen	44
Definition 15	Ereignisspeicher	44
Definition 16	Semantischer Zustand	45
Definition 17	Aktivierte Transition	46
Definition 18	Wirkungsbereich einer Transition	46
Definition 19	Hauptquell- und Hauptzielzustand einer Transition	47
Definition 20	Menge der Zustände, die durch eine Transition verlassen werden	47
Definition 21	Konfliktrelation	47
Definition 22	Prioritätsrelation	48

Definition 23	Menge gemeinsam schaltender Transitionen	48
Definition 24	Menge der Zustände, die durch eine Transition betreten werden	49
Definition 25	Semantischer Schritt	51
Definition 26	Systemlauf	62
Definition 27	Beobachtbarer Systemlauf	63
Definition 28	Notationen für Systemläufe	63
Definition 29	Menge der beobachtbaren Systemläufe	64
Definition 30	Menge der Ausgaben zu einer gegebenen Eingabe	66
Definition 31	Implementierungsrelation für Zustandsmaschinen	67
Definition 32	Untätiger semantischer Zustand	84
Definition 33	Erreichbare semantische Zustände	88