# Ein Werkzeug zur Unterstützung zustandsbasierter Testverfahren für Java-Klassen

Dehla Sokenou
Forschungsschwerpunkt für Netzwerktechnologien und Multimedia-Anwendungen
FSP-PV/PRZ, Sekr. MA073, TU Berlin,
Straße des 17. Juni 136, D-10623 Berlin

14.01.1999

## 1 Einleitung

Da Objekte in der Regel einen Zustand besitzen, nehmen zustandsbasierte Testverfahren beim Test objektorientierter Programme eine wichtige Stellung ein.

Dabei wird ein besonderer Aspekt objektorientierter Software, die Vererbung, meist außer Acht gelassen. Besonders die typkonforme Vererbung , die die Sicht von Objekten als Implementation abstrakter Datentypen unterstützt, wird bisher beim Test wenig genutzt, obwohl sich der Testaufwand dadurch reduzieren würde.

Im folgenden soll ein Werkzeug vorgestellt werden, daß zustandsbasierte Testverfahren durch die automatische Auswertung von Zuständen, Zustandsübergängen und Überdeckungsmaßen unterstützt und zudem die typkonforme Vererbung berücksichtigt (siehe auch [7]).

Das System ist in Java implementiert und kann zum Test von Java-Klassen verwendet werden.

## 2 Verwendung des Testwerkzeugs

Das Werkzeug vergleicht den Ist-Zustand eines Objekts der zu testenden Klasse mit seinem Soll-Zustand und meldet Differenzen an den Benutzer. Dazu muß dieser eine Spezifikation in Form eines Zustandsautomaten eingeben, aus der das System eine Automatenklasse in Java generiert. Diese wird an die zu testende Klasse mit Hilfe eines Instrumentierers [5] angebunden. Objekte der zu testenden Klassen werden von Objekten der Automatenklasse überwacht. Alle Zustände, Zustandsübergänge und Fehler werden protokolliert und die anfallenden Daten automatisch ausgewertet (siehe Abschnitt 4).

Der Zustand eines Objekts wird bestimmt durch Äquivalenzklassen seiner Attribute [3, 4, 6]. Ereignisse sind Nachrichten, die an das Objekt geschickt und durch Methoden implementiert werden. Man unterscheidet bei den Methoden zwischen solchen, die zustandsändernd sind und solchen, die den Wert von Attributen zurückgeben (Observer) [1]. Als Zustand wird der nach außen sichtbare Zustand eines Objekts definiert [4].

Der Benutzer muß demnach seine Definition so aufbauen, daß er die Observer nutzt, um den Zustand zu definieren, die anderen Methoden können die Zustandsübergänge definieren.

Die definierbaren Zustandsautomaten beruhen auf einer Teilmenge der Statecharts von Harel [2].

## 3 Unterstützte Vererbungsformen

Um typkonforme Vererbung zu realisieren und für den Test nutzen zu können, muß die typkonforme Vererbung von Zustandsautomaten, wie sie in [4] vorgestellt wird, unterstützt werden.

McGregor und Dyer gehen davon aus, daß es fünf Möglichkeiten gibt, wie ein Zustandsautomat einer Unterklasse sich in Bezug auf den Zustandsautomaten der Oberklasse ändern kann, damit die typkonforme Vererbung gewahrt bleibt [4]. Abbildung 1 verdeutlicht dieses. Es gibt die Möglichkeit, eine neue Transition einzuführen (1), einen Zustand zu teilen (2), ein nebenläufiges, unabhängiges Diagramm zu definieren (3) oder einen Zustand (4) oder eine Transition (5) nicht signifikant zu verändern.

Das Werkzeug unterstützt die ersten drei Varianten eines Unterklassenzustandsautomaten. Darüber hinaus ist es möglich, den Zustandsautomaten der Oberklasse unverändert für den Test zu nutzen. In allen unterstützten Varianten ist eine Definition nur auf der Basis des Oberklassenautomaten möglich.

## 4 Unterstützte Überdeckungsmaße

Analog zu den Überdeckungsmaßen der traditionellen kontrollflußorientierten Testverfahren kann auch ein Zustandsautomat als Graph verstanden und eine Überdeckungsmessung durchgeführt werden.

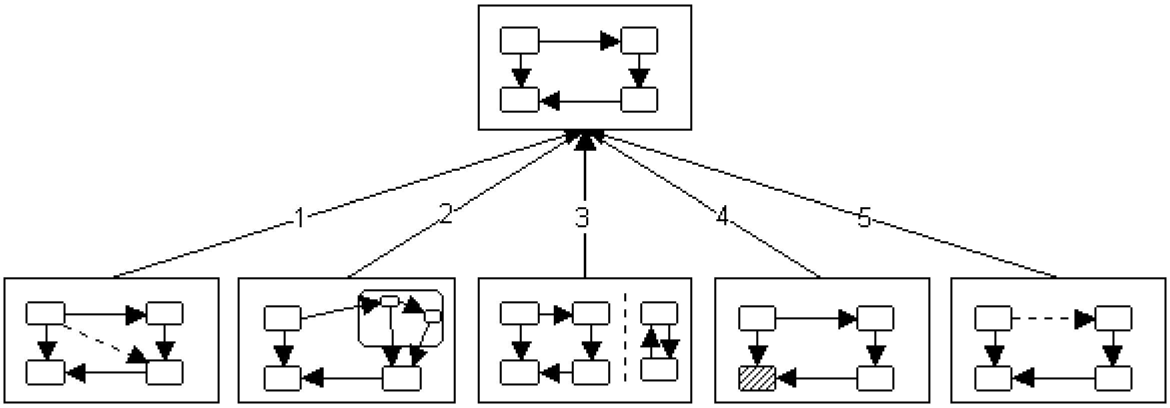


Abbildung 1: Vererbung von Zustandsautomaten (nach [4])

Die Überdeckung aller Knoten wäre gleichbedeutend mit der Überdeckung der Zustände, die Überdeckung der Zweige wäre eine Überdeckung der Zustandsübergänge. Eine vollständige Pfadüberdeckung ist auch bei Zustandsautomaten in den seltesten Fällen möglich, da sie meist zyklisch sind. Das System implementiert deshalb nur eine Messung der Überdeckung der Zustände und Zustandsübergänge des Zustandsautomaten.

Dabei kann sowohl die Überdeckung des Zustandsautomaten eines einzelnen Objekts der zu testenden Klasse betrachtet werden als auch die Summe aller Zustandsautomaten einer Klasse, die während des Test erzeugt wurden.

## 5 Zusammenfassung und Ausblick

Mit dem System wurde ein Ansatz zur programmtechnischen Umsetzung zustandsbasierter Testverfahren für objektorientierte Software vorgestellt. Insbesondere die Unterstützung der Vererbung berücksichtigt die besonderen Eigenschaften objektorientierter Software. Der dabei auf typkonforme Vererbung gelegte Schwerpunkt unterstreicht die Besonderheit dieser Vererbungsform, da sie die Sicht von Objekten als Implementation abstrakter Datentypen berücksichtigt und den Testaufwand durch Benutzung und Erweiterung der Zustandsautomaten der Oberklasse reduziert. Zudem kann sie darüber hinaus dazu dienen, Unterklassen anhand ihrer Spezifikation auf Typkonformität zu testen.

Das vorliegende System kann noch erweitert werden, zum Beispiel durch die Einführung einer graphischen Oberfläche zur Definition des Zustandsautomaten oder durch Einbindung in ein größeres System, das auch bei der Testfallermittlung Unterstützung bietet.

## Literatur


[1] Derek Coleman, Fiona Hayes, and Stephen Bear. Introducing Objectcharts or How to Use Statecharts in Object-Oriented Design. *IEEE Transactions on Software Engineering*, 18(1):9–18, Jan. 1992.

[2] David Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, (8):231–274, 1987.

[3] Brian Marick. *The Craft of Software Testing (Subsystem Testing).* Prentice Hall, 1995.

[4] John D. McGregor and Douglas M. Dyer. A Note on Inheritance and State Machines. *Software Engineering Notes*, 18(4):61–69, Okt. 1993.

[5] Frank Posadny. Entwicklung eines Systems von Protokollierungsklassen für den objektorientierten Intergrationstest in der Programmiersprache JAVA. Diplomarbeit, Technische Universität Berlin, Fachbereich 13 - Informatik, Institut für Kommunikations- und Softwaretechnik, Fachgebiet Softwaretechnik, Berlin, Nov. 1997.

[6] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. *Object-Oriented Modeling and Design.* Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[7] Dehla Sokenou. Entwicklung eines Werkzeug zur Unterstützung zustandsbasierter Testverfahren für JAVA-Klassen. Diplomarbeit, Technische Universität Berlin, Fachbereich 13 - Informatik, Institut für Kommunikations- und Softwaretechnik, Fachgebiet Softwaretechnik, Berlin, Juni 1998.