Ein Werkzeug zum automatischen Test ereignisgesteuerter Java Programme

G. Fischer[1], T. Renner[2], J. Wolff v. Gudenberg[1]
[1] Institut für Informatik, Universität Würzburg
fischer,wolff@informatik.uni-wuerzburg.de
[2] 3soft GmbH, Erlangen
thomas.renner@3soft.de

25. April 2005

1 Varianten des Testaufbaus

Im Rahmen des Java Online Praktikums [1] wurden Programme zum automatischen Test von abgegebenen studentischen Lösungen entwickelt. In einem besonders aussagekräftigen Test wurden dabei die Teilnehmerlösung und die Musterlösung intelligent verglichen. Für ereignisgesteuerte Programme greifen diese Tests nicht in vollem Umfang, weil wesentliche Teile der Funktionalität durch Interaktion hervorgerufen werden. In dieser Arbeit wird nun ein Werkzeug vorgestellt, das den Benutzer bei der Erstellung von Struktur und Funktionstests für ereignisgesteuerte Java Programme unterstützt und deren Durchführung kontrolliert.

Eine Applikation kann sowohl auf die Erfüllung einer Spezifikation wie auch im Vergleich zur Musterlösung getestet werden. Die erstellten Testszenarien sind während der Entwicklungsphase schrittweise ausführbar. Dabei wird der aktuelle Zustand der Ausführung und der einzelnen Elemente grafisch dargestellt. Für automatisiertes Testen ist der Batchbetrieb möglich, bei dem die grafische Darstellung deaktiviert und dadurch die Ausführungsgeschwindigkeit verbessert wird.

2 Fernbedienung

Um ein ereignisgesteuertes Programm fernsteuern zu können, müssen ihm die entsprechenden auslösenden Ereignisse zugestellt werden. Hierzu sind zwei Ansätz möglich: Man kann Native System Events in die System Event Queue einfügen. Damit werden einfachste Systemereignisse wie das Drücken oder Loslassen einer Taste oder Mausknopfes simuliert. Alternativ lassen sich auch Java Events direkt erzeugen. Diese stellen komplexere Aktionen wie z.B. das Tippen einer Zeichenkette zur Verfügung. Damit lassen sich Simulationen erheblich einfacher und zuverlässiger erzeugen, jedoch verwendet man bereits eine Abstraktionsschicht über dem, was bei der realen Bedienung passiert. Es kann daher zu Abweichungen kommen.

Weiterhin kann man auch durch direkten (reflexiven) Zugriff auf die Java-Komponenten Einfluss auf das Programm nehmen. Dabei werden aber keinerlei Ereignisse mehr erzeugt. Dies ist also nur z.B. zur Vorbereitung von weiteren Test geeignet, wenn man die ferngestuerten Komponenten bereits über Events getestet hat.

3 Testserver

Um den Testserver von dem auszuführenden Programm abzuschotten, und um die simultane Ausführung mehrerer Anwendungen zu ermöglichen, erzeugt der entwickelte Testserver für jede zu startende Anwendung jeweils eine eigene Virtual Machine. Dort wird der Testclient gestartet. Dieser trägt sich zunächst bei der AWT-Event-Queue als Listener ein, um das Überwachen der Events zu ermöglichen. Dann wird eine RMI-Kommunikation zum Testserver aufgebaut, so dass dieser auf die weitere Ausführung Einfluss nehmen kann. Erst nach erfolgreichem Aufbau der RMI-

Kommunikation wird die main-Methode der zu testenden Applikation gestartet.

4 Identifikation von Komponenten

Bei Tests der Benutzeroberfläche müssen zunächst die Komponenten identifiziert werden. Zum einen wird damit ihre Existenz sichergestellt, zum anderen ist dies nötig, damit ihnen später die Events zugestellt werden können.

Die Komponenten werden mit Hilfe von rekursiven Schablonen z.B. über ihren Typ, Namen oder auch die Panelhierarchie identifiziert. Dazu wird die im Programm aufgebaute Komponentenhierarchie durchlaufen und versucht diese mit den Schablonen zu unifizieren.

5 Testszenarien

Testszenarien sind Hierarchien von Java-Klassen, welche vom Testserver beim Testlauf zur Ausführung gebracht werden.

Statements stellen die ausführbaren Instanzen der Testszenarien dar. Jede Aktion in einem Test erfolgt im Rahmen eines Statements. Die Aktion wird durch die Implementierung der execute-Methode bereitgestellt bzw. durch deren Aufruf ausgeführt. Statements arbeiten mit Werten, Referenzen und Ausnahmen.

Die Elemente können sich selbst zu Java-Programmcode serialisieren, und so nach der Bearbeitung im Testfall-Editor gespeichert und vom Testserver verwendet werden oder später wieder geladen werden.

Darüber hinaus lassen sich auch leicht neue Statements und Hilfselemente definieren, die sich dann direkt in das bestehene Framework eingliedern.

6 Editor

Testfälle lassen sich mit Hilfe eines graphischen Editors zusammenstellen und konfigurieren. Weiterhin unterstützt er die Erstellung von Schablonen zur Identifikation von Komponenten. Dies kann insbesondere durch Selektion von Komponenten in einer Applikation erfolgen, oder durch Auswahl in der Komponentenhierarchie.

Außerdem können natürlich auch die Testfälle ausgeführt, und der Zustand einer Applikation während der Ausführung inspiziert werden.

7 Zusammenfassung

Der vorliegende Artikel stellt ein Framework zur Erstellung und Durchführung von Tests an Java Programmen mit grafischer Benutzeroberfläche vor, bei dessen Entwicklung besonders die Erweiterbarkeit im Vordergrund stand.

Durch die flexible Architektur ist es nicht nur möglich Testfälle aufzuzeichnen, zu modifizieren und wiederzugeben, sondern auch eine gegebene (nicht notwendigerweise detailgetreue) Spezifikation zu überprüfen und einen Vergleich mit einer Musterlösung durchzuführen.

Die Testfälle können sowohl interaktiv in einem komfortablen Editor zusammengestellt und durchgeführt werden, als auch automatisch im Batchbetrieb ablaufen.

Literatur

[1] H. Eichelberger et al., Programmierausbildung Online, DeLFI 2003, A.Bode,J.Desel,S.Rathmayer,M.Wessner (eds), Lecture Notes in Informatics, GI, p.134-143

[2] T. Renner, Jatest - Korrektheitstests für ereignisgesteuerte Java Programme, Diplomarbeit, Univ. Würzburg, Oktober 2002