Automatische Erzeugung von Unit Tests für Klassen mit Umgebungs-Abhängigkeiten

Andrea Arcuri
Certus Software V&V Center
Simula Research Laboratory
Lysaker, Norway
arcuri@simula.no

Gordon Fraser
University of Sheffield
Dep. of Computer Science
Sheffield, UK
Gordon.Fraser@sheffield.ac.uk

Juan Pablo Galeotti
Saarland University
Dep. of Computer Science
Saarbrücken, Germany
galeotti@cs.uni-saarland.de

Abstract: Die automatische Erzeugung von Tests für objektorientierte Software besteht typischerweise aus der Erstellung von Sequenzen von Methodenaufrufen um hohe Codeabdeckung zu erzielen. In der Praxis könnte der Erfolg dieses Prozesses eingeschränkt sein, wenn Klassen mit ihrer Umgebung, wie zum Beispiel dem Dateisystem, dem Netzwerk oder Nutzern interagieren. Das fürt zu zwei großen Problemen. Erstens kann Code, der von der Umgebung abhängt, manchmal nicht voll abgedeckt werden indem einfach nur Sequenzen von Methodenaufrufen generiert werden. Das ist zum Beispiel der Fall, wenn die Ausführung eines Zweigs von dem Inhalt einer Datei abhängt. Zweitens können, auch wenn Code, der von der Umgebung abhängt, abgedeckt wird, die generierten Tests *unstabil* sein, d.h. sie wüden durchlaufen wenn sie generiert werden aber missglücken wenn sie zu einer anderen Zeit ausgeführt werden.

In dieser Arbeit nutzen wir Bytecodeinstrumentierung um Code automatisch von seinen Umgebungsabhänigkeiten zu trennen, und erweitern das EvoSuite Javatesterzeugungswerkzeug so, daß es als Teil der generierten Aufrufsequenzen explizit den Zustand der Umgebung modifizieren kann. Indem wir eine Prototypimplementierung nutzen, die eine große Anzahl an Umgebungsinteraktionen handhaben, wie mit dem Dateisystem, Konsoleneingaben und vielen nicht-deterministischen Funktionen der Java Virtual Machine (JVM), haben wir Experimente mit 100 zufällig ausgewählten Java Projekten von SourceForge durchgeführt (dem SF100 corpus). Die Ergebnisse zeigen signifikant bessere Codeabdeckung — in manchen Fällen sogar bis zu +80% / +90%. Darüber hinaus hat unsere Methode die Anzahl unstabiler Tests um mehr als 50% reduziert.