Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

InIT Institut für angewandte
Informationstechnologie

Nie mehr Unit-Tests schreiben! [Software Engineering]

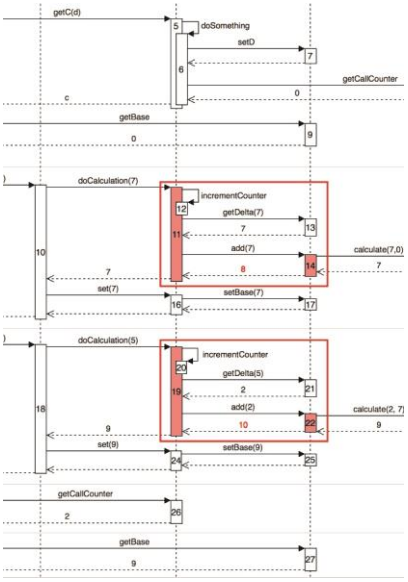
Diplomand
Thomas Moser

Dozierende
Mark Cieliebak
Walter Eich

Bei der Überarbeitung von Legacy-Software ist es aufgrund fehlender Dokumentation und spärlicher Kommentierung des Codes oft schwierig zu ermitteln, ob die bearbeiteten Stellen noch das gewünschte Verhalten aufweisen. In der Regel existieren auch keine Unit-Tests, was diese Validierung zusätzlich erschwert. Es wäre deshalb wünschenswert, die Analyse dieses Verhaltens zu automatisieren.

Das Ziel dieser Arbeit ist es, das Programmverhalten vor dem Refactoring aufzuzeichnen. Mit diesen aufgezeichneten Daten können Tests erzeugt und auf die überarbeitete Software angewendet werden. Auf diese Weise kann ermittelt werden, ob in der Software geändertes Verhalten auftritt und, falls dies der Fall ist, wie weit sich dieses auf die Software auswirkt.

Um dieses Ziel für Java-Programme zu erreichen, wurden mit Hilfe von Bytecode-Instrumentierung zur Laufzeit des Programmes alle Konstruktor- und Methodenaufrufe inklusive der Parameter und Rückgabewerte aufgezeichnet. Diese Daten wurden anschliessend verwendet, um die geänderte Software zu validieren. Dazu wurden die bekannten Objekte einzeln instanziiert. Wurden auf diesen Instanzen die bekannten Methoden ausgeführt, konnte ihr Verhalten validiert werden (Rückgabewerte, Aufrufe andere Methoden). Im ersten Schritt wurden die Objekte durch Mocking aller Abhängigkeiten isoliert geprüft. Bei festgestellten Änderungen wurde schrittweise die Anzahl gleichzeitig instanziierter Objekte erhöht, um so die Ausbreitung einer Änderung feststellen zu können.



Änderungen im Programmverhalten lokalisieren und Ausbreitung ermitteln

Im Rahmen dieser Arbeit konnte ein Prototyp entwickelt werden, welcher für eine einfache Beispiel-Applikation erfolgreich Verhaltensänderungen finden und deren Ausbreitung feststellen kann. Für den Praxiseinsatz ist dieser Prototyp jedoch noch nicht geeignet, da er vielen Einschränkungen unterliegt. Beispielsweise können nur primitive Datentypen oder selbst definierte Objekte als Parameter verwendet werden.

Für eine Weiterentwicklung muss die Frage geklärt werden, ob und wie mit Aufrufen von Funktionen aus der Java-Klassenbibliothek oder von Third-Party-Libraries umgegangen werden kann. Ein weiterer zu untersuchender Punkt ist, wie das Programmverhalten vor und nach dem Refactoring verglichen werden kann, wenn sich Methodensignaturen der verwendeten Klassen ändern.