Entwicklungsbegleitendes Testen mittels UML Sequenzdiagrammen

Falk Fraikin

Promotion: 3.12.2003, TU Darmstadt, Fachbereich Informatik
Gutachter: Prof. Dr. Wolfgang Henhapl, Prof. Dr. Mira Mezini
Publikation: http://elib.tu-darmstadt.de/diss/000396/

Abstract

UML Sequenzdiagramme sind ein wesentlicher Bestandteil des Designs bei der Anwendung von einer Reihe von verbreiteten Softwareentwicklungsprozessen wie zum Beispiel *Rational Unified Process* oder *Feature Driven Development*. Außerdem finden sie Verwendung bei der Beschreibung von Design Patterns und generell bei der Beschreibung von Schnittstellen. Da die Implementierung konform zu diesen Diagrammen erfolgen soll, stellt sich die Frage, inwieweit sich die Diagramme als Basis für dynamische Tests nutzen lassen. Diese Fragestellung ist insofern nahe liegend, da UML Sequenzdiagramme sich mit ihren Objekten und Nachrichten intuitiv auf die Instanzen und Methodenaufrufe eines objektorientierten Programms abbilden lassen und damit einen logischen Testfall darstellen. Ergänzt man die Methodenaufrufe im Sequenzdiagramm um Aufrufparameter und Rückgabewerte, so erhält man einen konkreten Testfall für die Kommunikation zwischen den abgebildeten Instanzen der zu testenden Klassen.

Im ersten Teil dieser Arbeit wird der Begriff der „Testbaren Sequenzdiagramme“ eingeführt, der die Anforderungen an ein UML Sequenzdiagramm definiert, die sicherstellen, dass das Diagramm einen konkreten, d.h. ausführbaren Testfall beschreibt. Einzelne testbare Sequenzdiagramme lassen sich zu kombinierten testbaren Sequenzdiagrammen zusammenfügen. Dies reduziert sowohl die Anzahl als auch die Komplexität der benötigten einzelnen testbaren Sequenzdiagramme.

Da das in einem testbaren Sequenzdiagramm beschriebene Verhalten nicht nur als Testeingabe dient, sondern auch gleichzeitig das erwartete Ergebnis beschreibt, bietet es sich beim Testen auf der Basis von Sequenzdiagrammen an, die Testergebnisse ebenfalls als Sequenzdiagramme darzustellen. Um eine effektive Evaluation der Testergebnisse zu ermöglichen, wird für testbare Sequenzdiagramme ein Verfeinerungskonzept entwickelt, das eine Informationshierarchie für testbare Sequenzdiagramme definiert.

Im zweiten Teil der Arbeit wird das Testwerkzeug SeDiTeC vorgestellt, das die im ersten Teil der Arbeit vorgestellten Konzepte implementiert. Dabei werden weitere Anforderungen definiert, die an ein solches Testwerkzeug gestellt werden, und es wird beschrieben, wie diese in SeDiTeC realisiert wurden. Hierzu gehört u.a. die Möglichkeit der automatischen Generierung von „intelligenten“ Teststubs.

Improving testability of object-oriented systems

Stefan Jungmayr

Promotion: 18.12.2003, FernUniversität in Hagen, Fachbereich Informatik
Gutachter: Prof. Dr. Hans-Werner Six, Ao.Prof. Dr. Gerald Futschek (TU-Wien)
Publikation: dissertation.de, 2004, ISBN 3-89825-781-9, http://www.dissertation.de

Abstract

Software kontrolliert in zunehmendem Maße sicherheits- und geschäftskritische Systeme. Zur Sicherstellung der Qualität und Fehlerfreiheit dieser Software werden überwiegend Tests eingesetzt. Der Testvorgang selbst ist dabei mit einem großen Ressourcenaufwand verbunden. Bisher wurden zahlreiche Ansätze zur Reduktion des Testaufwandes entwickelt, die auf eine Verbesserung der eingesetzten Testmethoden und Testwerkzeuge fokussieren. Nicht zuletzt hängt der Testaufwand und Testerfolg aber auch wesentlich von den Eigenschaften des zu testenden Produktes, d.h. seiner Testbarkeit, ab.

Diese Arbeit beschreibt einen methodischen Ansatz zur Verbesserung der Testbarkeit von objekt-orientierter Software. Der Schwerpunkt liegt dabei auf der Kontrolle der Abhängigkeiten zwischen den verschiedenen Teilen des Softwaresystems.

Für jede Entwicklungsaktivität (beginnend mit der Anforderungsermittlung bis zum Test) werden Schritte angegeben die notwendig sind, um die Auswirkung der Abhängigkeiten auf den Testaufwand unter Kontrolle zu halten. Diese Schritte beinhalten die Identifikation, Bewertung, Vermeidung sowie Restrukturierung von Abhängigkeiten innerhalb von Anwendungsfall-diagrammen, Klassendiagrammen, und Quellcode. Ebenso werden jene Klassen identifiziert und bewertet, die bzgl. der Abhängigkeiten eine zentrale Rolle spielen.

Die Bewertung der Abhängigkeiten basiert auf Metriken und der Differenzierung von u.a. indirekten, zyklischen und fest-verdrahteten Abhängigkeiten. Ein zentrales Bewertungskriterium ist die Auswirkung der Abhängigkeit auf die Möglichkeit, eine Klasse isoliert zu testen. Weitere Elemente des Ansatzes sind konkrete Entwurfsrichtlinien, sowie ein neu entwickeltes, in eine kommerzielle Softwareentwicklungsumgebung (IDE) integriertes Analysewerkzeug, welches neben der Metrikberechnung auch die Untersuchung der Ursachen von Abhängigkeiten unterstützt.

Wesentliche Teile des Ansatzes zur Verbesserung der Testbarkeit werden anhand von Fallbeispielen demonstriert und validiert.