SeDiTeC - Testen auf der Basis von Sequenzdiagrammen

Falk Fraikin

Technische Universität Darmstadt, E-Mail: fraikin@informatik.tu-darmstadt.de

Inzwischen gibt es eine Reihe von Arbeiten, die sich mit dem Testen von objektorientierten Programmen auseinandersetzen. Trotz der zahlreichen dabei gewonnenen Erkenntnisse, wird jedoch auch heute noch in vielen Bereichen der objektorientierten Anwendungsentwicklung unzureichend oder ineffizient getestet.

Die Gründe hierfür sind sicherlich vielfältig, sollen jedoch nicht Thema dieses Beitrags sein, der sich vielmehr mit einer der Auswirkungen, nämlich der nach wie vor mangelnden Integration des Testens in den Entwicklungsprozess zu einem frühen Zeitpunkt, beschäftigen wird. Dabei ist diese mangelnde Integration weniger ein theoretisches Problem als vielmehr ein praktisches. Zwar sehen viele Vorgehensmodelle in der Softwareentwicklung (Rational Unified Process, Extreme Programming etc.) Testen bereits zu frühen Zeitpunkten vor, es fehlen jedoch die notwendigen Werkzeuge, um das Testen effizient in einem frühen Stadium in den Softwareentwicklungsprozess zu integrieren.

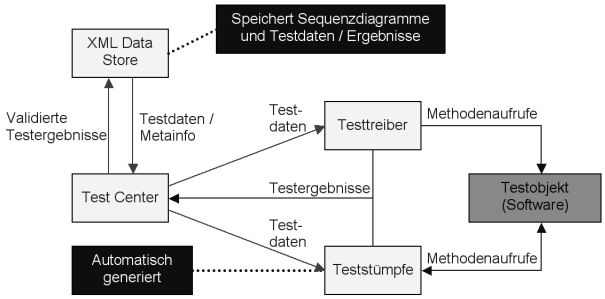
Aus diesen Überlegungen heraus ist SeDiTeC entstanden, ein Werkzeug zum entwicklungsbegleitenden Testen von Java-Programmen. Die Eingabe für die Ausführung von Tests durch SeDiTeC besteht aus UML Sequenzdiagrammen, die um Daten für Argumente und Rückgabewerte der Methodenaufrufe erweitert werden und somit ausführbar sind.

Der Vorteil dieser Art der Testspezifikation liegt darin begründet, dass Sequenzdiagramme in (auf der UML beruhenden) Entwicklungsprozessen häufig bereits während der Analyse- und der Designphase erstellt werden und nur um entsprechende Testdaten zu erweitern sind. Aus diesem Grund kann die Testspezifikation daher in vielen Bereichen bereits vorgenommen werden, bevor die Implementierung überhaupt begonnen hat. Es müssen hierfür allerdings nicht unerheblich mehr Sequenzdiagramme erzeugt werden, als das bei momentanen Vorgehensweisen üblicherweise der Fall ist. Da es sich bei Sequenzdiagrammen um ein leicht verständliches und weitverbreitetes Mittel zur Kommunikation handelt, fällt dies jedoch vergleichsweise leicht. Insgesamt ist so ein entwicklungsbegleitendes Testen ohne großen zusätzlichen Aufwand gleich von Beginn der Implementierungsphase an möglich.

Das Systemkonzept SeDiTeC benötigt drei Arten von Informationen für die Ausführung eines Tests:

- Informationen über zu testende Sequenzdiagramme (beteiligte Instanzen, Methodenaufrufe inklusive deren Reihenfolge)
- Typinformationen über Klassen und zugehörige Methoden, die in den zu testenden Sequenzdiagrammen vorkommen
- Argumente und Rückgabewerte der in den zu testenden Sequenzdiagrammen vorkommenden Methoden (Testfälle)

Die beiden erstgenannten Informationsarten können aus einem beliebigen UML CASE-Tool extrahiert werden. Derzeit besteht hierfür eine Schnittstelle zu Together®. Die relevanten Testfalldaten (Argumente und Rückgabewerte) werden über eine grafische, tabellarisch aufgebaute Benutzeroberfläche eingegeben und verwaltet.



Komponenten von SeDiTeC

Bei einem Testlauf ruft der Testtreiber dann eine oder mehrere Methoden auf dem Testobjekt auf. Problematisch ist hierbei, dass das Testobjekt in vielen Fällen seinerseits weitere Objekte oder Komponenten verwendet, die u.U. noch nicht zur Verfügung stehen. Daher werden für durch Klassen- und Sequenzdiagramme spezifizierte aber noch nicht implementierte Klassen automatisch Teststümpfe generiert (s. Abbildung). Diese beziehen ihr spezifiziertes Verhalten vollständig dynamisch zur Laufzeit von SeDiTeC, so dass sie - einmal generiert - für beliebige Sequenzdiagramme lauffähig sind. Ein selektives Entwickeln und Testen (Unit Test) einzelner Klassen und Komponenten eines großen Systems ist somit möglich. Der Aufwand für Integrationstests geht deutlich zurück.

Perspektiven

- Verkürzung der Entwicklungszeit und geringere Entwicklungskosten, da Fehler früher erkannt und beseitigt werden können
- Die Möglichkeit, selbst eine einzelne Klasse unabhängig vom Rest des Systems zu entwickeln und zu testen
- Mehr Vertrauen in das Gesamtsystem, da es jederzeit vollständig gegen die Spezifikation getestet werden kann

Zukünftige Entwicklung In heutigen Entwicklungsprozessen ist das Erstellen von Sequenzdiagrammen zwar häufig vorgesehen, aber es wird nur in seltenen Fällen in ausreichendem Maße betrieben, um ein umfassendes Testen der betroffenen Klassen zu ermöglichen. Daher erscheint es angebracht, nach geeigneten Methoden zu suchen, die das Erstellen von Sequenzdiagrammen erleichtern bzw. teilweise automatisieren. Denkbar sind hier Capture & Replay Werkzeuge, die eine entsprechende Abbildung auf Sequenzdiagramme vornehmen oder auch Statische Analyse Werkzeuge, die bereits bestehenden Code analysieren können. Beide Ansätze werden derzeit in weiteren Entwicklungen an der TU Darmstadt verfolgt.