Test dynamisch gebundener Operationsaufrufe

Beitrag für den 16-ten TAV-Workshop "Test, Analyse und Verifikation von Software"
15.-16. Februar 2001, Nordakademie, Elmshorn bei Hamburg

Mario Winter, Praktische Informatik III, FernUniversität Hagen, D-58084 Hagen
eMail: Mario.Winter@FernUni-Hagen.de
http://www.informatik.fernuni-hagen.de/import/pi3/mario

Während zu Beginn der 80er Jahre die ersten industriell verwendeten Implementierungen objektorientierter Programmiersprachen verfügbar waren und sich ab 1990 Ansätze zur objektorientierten Softwareentwicklung durchzusetzen begannen, blieb der Test objektorientierter Software lange Zeit unbeachtet. Glaubte man anfangs, dass objektorientierte Software den Aufwand für ihre Prüfung reduzieren würde und bekannte Testtechniken unmodifiziert übernommen werden könnten, so wissen wir heute, dass diese Hoffnungen sich nicht erfüllt haben. In diesem Beitrag wird — aufbauend auf [1] — die beim Test objektorientierter Software immer wieder auftretende Frage beleuchtet, wie dynamisch gebundene Operationsaufrufe getestet werden.

1 Problemstellung

Instanzvariablen, Parameter und Rückgabewert einer Operation sowie die “Pseudovariable“ this sind Referenzen auf Objekte. Diese können nicht nur Objekte des angegebenen Typs, sondern auch aller Untertypen referenzieren. Zusätzlich weisen die Typsysteme fast aller Sprachen Schwächen auf oder können durch bestimmte Sprachkonstrukte (z.B. Casts in Java und C++) aufgeweicht werden. Berücksichtigt man nun noch die mit der Generalisierung einhergehende Zerfaserung der Klassendefinitionen, so werden statische Analysen und damit auch die Erstellung von strukturellen Testfällen erheblich erschwert [2]. Dynamisch gebundene Operationsaufrufe zwingen den Tester, alle potenziellen Aufrufziele zu simulieren. Somit ist der Klassen- und Klassenintegrationstest für objektorientierte Software komplizierter als für prozedurale Software [3].

Beispiel 1 Das in Abb. 1 dargestellte Klassendiagramm beinhaltet die beiden Oberklassen A und B und ihre Unterklassen A1 und A2 sowie B1 und B2. Innerhalb der in der Klasse A1 definierten Operation m1() wird die in der Klasse B definierte Operation m2(einB: B) mit dem aktuellen Parameterobjekt einB vom Typ B aufgerufen. Mögliche Typen des die Operation m1() ausführenden Objekts sind also A, A1 und A2. Die aufgerufene Operation m2() kann von einem Objekt der Klassen B, B1 und B2 ausgeführt werden.

2 Ziele und Vorgehensweise

Das Ziel des Tests dynamisch gebundener Operationsaufrufe besteht darin, Fehler bei dem Aufruf überschriebener oder in einem neuen Kontext ausgeführter geerbter Operationen aufzudecken. Zusätzlich werden auch aufgrund nicht korrekt berücksichtigter Zustände verursachte Fehler gefunden. Prüfgegenstand ist die umgebende Operation (Operation unter Test, OpUT), die derartige Operationsaufrufe enthält.

Die Voraussetzung für den Test dynamisch gebundener Operationsaufrufe ist das Vorliegen des Klassendiagramms, des Quellcodes der OpUT, die den zu testenden dynamisch gebundenen Operationsaufruf enthält, und der Signatur und Spezifikation der aufgerufenen Operation. Sollen Zustände betrachtet werden, so werden auch Zustandsdiagramme der beteiligten Klassen benötigt.

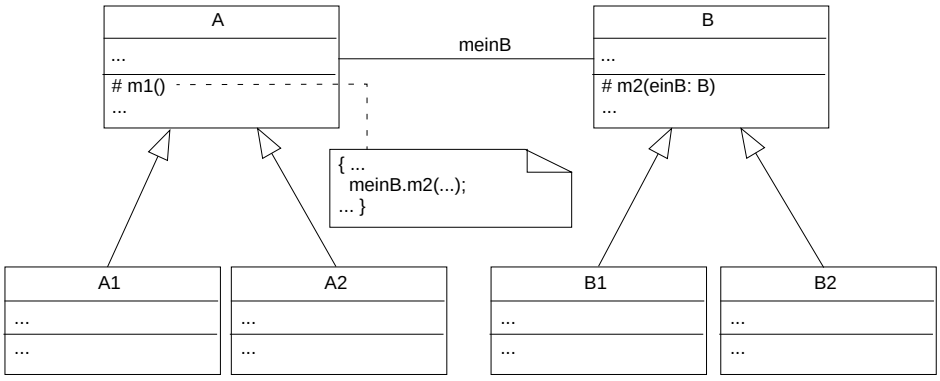

Abbildung 1 Operationsaufruf mit dynamischer Bindung

Die Arbeitsschritte des Tests dynamisch gebundener Operationsaufrufe sind:

1. Auflisten der beteiligten Objekte und deren Typen.
2. Eingrenzen der zu testenden Kombinationen von Klassen und Zuständen.
3. Erstellen und Ausführen parametrisierter Testfälle.

Schritt 1: Auflisten der beteiligten Objekte und deren Typen. In diesem Schritt werden alle am dynamisch gebundenen Aufruf beteiligten Objekte und deren Typen anhand der Generalisierungsbeziehungen und ggf. der realisierten Interfaces ermittelt. Für jede Klasse werden zusätzlich die für den Operationsaufruf relevanten Zustände der beteiligten Instanzen aus den Zustandsdiagrammen ermittelt. Dies sind alle Zustände, von denen im Zustandsdiagramm eine ausgehende Kante mit dem entsprechenden Operationsaufruf verknüpft ist.

Ergebnis des ersten Arbeitsschrittes ist die Menge von Klassen und ihren Unterklassen, deren Instanzen in den dynamisch gebundenen Operationsaufruf involviert sein können, sowie die Menge ihrer für diesen Aufruf relevanten Zustände.

Beispiel 2 Für das in Abb. 1 gezeigte Klassendiagramm nehmen wir an, dass Instanzen der Klassen A, A1 und A2 die zwei Zustände zustandA1 und zustandA2 und Instanzen der Klassen B, B1 und B2 die drei Zustände zustandB1, zustandB2 und zustandB3 einnehmen können.

Schritt 2: Eingrenzen der zu testenden Kombinationen von Klassen und Zuständen. Wie das folgende Beispiel zeigt, führt die Betrachtung aller Kombinationen von Klassen und Zuständen des den Operationsaufruf ausführenden Objekts und der aufgerufenen Objekte zu einer exponentiell anwachsenden Anzahl von zu testenden Fällen. In den meisten Fällen muss diese Anzahl systematisch reduziert werden.

Beispiel 3 Bezeichnet man mit *k* die jeweilige Anzahl möglicher (Unter-)Klassen und mit *z* die der Zustände, so ergeben sich für das die OpUT ausführende Objekt (Dienstnutzer), das Parameterobjekt und das aufgerufene Objekt (Dienstleister) insgesamt (3 *k* * 2 *z*) * (3 *k* * 3 *z*) * (3 *k* * 3 *z*) = 486 verschiedene zu testende Kombinationen von Klassen und Zuständen.

Eine systematische Möglichkeit zur Einschränkung der Kombinationen verwendet die aus dem Entwurf statistischer Experimente bekannte Idee, nicht alle möglichen Kombinationen, sondern nur alle *paarweise verschiedenen* Kombinationen zu testen. Bei der vorliegenden Testaufgabe wählt man gemäß dieser Idee die Testfälle so aus, dass nur alle möglichen Kombinationen von je zwei Objekten bzw. Zuständen von Objekten geprüft werden.

Für jedes betrachtete Objekt bezeichnet man jeweils seine Klasse und seinen Zustand als *Faktor* des Tests. Die Anzahl der jeweils möglichen Klassen bzw. Zustandswerte wird *Wertigkeit* (des Faktors) genannt. Eine untere Grenze für die Anzahl n_{min} der notwendigen Testfälle ergibt sich bei k Faktoren und deren Wertigkeiten W_i (i=1,...,k) nach der Formel (vgl. [4])

$$n_{min} = \sum_{i=1}^{k} (w_i - 1) + 1.$$

Beispiel 4 Bezüglich des dynamisch gebundenen Operationsaufrufes m2(einB: B) in Abb. 1 sind 6 Faktoren zu berücksichtigen: das aufrufende Objekt, das aufgerufene Objekt, das Parameterobjekt und jeweils deren Zustände. Die Wertigkeit aller drei "Klassenfaktoren" ist 3 (Klassen A, A1 und A2 bzw. B, B1 und B2), die der "Zustandsfaktoren" ist 2 (zustandA1 und zustandA2) bzw. 3 (zustandB1, ZustandB2 und zustandB3). Es ergibt sich für die Anzahl der Testfälle die untere Grenze

$$n_{min} = 3*(3-1) +(2-1) + 2*(3-1) + 1 = 12.$$

Die zu testenden Kombinationen werden in einer Tabelle aufgeführt, in der jedem Faktor eine Spalte zugeordnet wird. Jede Zeile der Tabelle entspricht einer konkret zu prüfenden Kombination von Klassen und Zuständen. Um sicherzustellen, dass wirklich alle paarweise verschiedenen Kombinationen von (Klassen- und Zustands-)Faktoren mit einer möglichst geringen Zahl nicht-redundanter Testfälle berücksichtigt werden ("orthogonale Testfälle"), kann man die aus dem statistischen Entwurf von Experimenten bekannten orthogonalen Felder (OATS, Orthogonal Array Test Specification) als "generische Testfallschemata" verwenden (vgl. [4]).

Beispiel 5 Im obigen Beispiel ist eine Tabelle mit 6 Spalten und mindestens 12 Zeilen zu wählen. Die Klassen und Zustände werden nach folgendem Schema nummeriert:

A = 1, A1 = 2, A2 = 3, zustandA1 = 1, zustandA2 = 2
und
B = 1, B1 = 2, B2 = 3, zustandB1 = 1, zustandB2 = 2, zustandB3 = 3.

In der ersten Spalte nummeriert man die Kombinationen und ordnet den weiteren Spalten abwechselnd die Klassen und Zustände der beteiligten Objekte zu. Dabei ist zu beachten, dass die dritte Spalte der Tabelle ("Dienstnutzerzustand") die den zwei Zuständen des aufrufenden Objekts entsprechende Wertigkeit 2 und die anderen Spalten die Wertigkeit 3 aufweisen. Unter Benutzung des in [4] aufgeführten orthogonale Feldes OA18(21x37) (18 Zeilen, 9 Spalten mit den Wertigkeiten 2 und 3) ergeben sich die in Tab. 2 angegebenen 18 Kombinationen aus Klassen und Zuständen für den Test des dynamisch gebundenen Aufrufs der Operation m2(). Nach der gewählten Nummerierung entspricht z.B. die erste Zeile von Abb. 2 der folgenden zu testenden Kombination aus Klassen und Zuständen: das aufrufende Objekt ist eine Instanz der Klasse A im Zustand zustandA1, welches die Operation m2() mit einem Parameterobjekt der Klasse B im Zustand

Nr	Dienstnutzer-klasse	Dienstnutzer-zustand	Parameter-klasse	Parameter-zustand	Dienstleister klasse	Dienstleister zustand
1	1	1	1	1	1	1
2	1	1	2	2	2	2
3	1	1	3	3	3	3
4	2	1	1	1	2	2
5	2	1	2	2	3	3
6	2	1	3	3	1	1
7	3	1	1	2	1	3
8	3	1	2	3	2	1
9	3	1	3	1	3	2
10	1	2	1	3	3	2
11	1	2	2	1	1	3
12	1	2	3	2	2	1
13	2	2	1	2	3	1
14	2	2	2	3	1	2
15	2	2	3	1	2	3
16	3	2	1	3	2	3
17	3	2	2	1	3	1
18	3	2	3	2	1	2

Abbildung 2 Zu testende Kombinationen von Klassen und Zuständen

zustandB1 in einem Objekt der Klasse B aufruft, das sich ebenfalls im Zustand zustandB1 befindet. Diese Konstellation ist als Kollaborationsdiagramm in Abb. 3 dargestellt.

Schritt 3: Erstellen und Ausführen der Testfälle. Nach der Ermittlung der zu testenden Kombinationen von Klassen und Zuständen sind Testfälle und entsprechende Testskripte zu erstellen und auszuführen. Für jeden Testfall ist die zu testende Konstellation durch entsprechende Operationsaufrufe zu erstellen (Test-Setup). Danach wird durch einen Aufruf der OpUT mit entsprechenden Parametern die dynamisch gebundene Operationsausführung erzwungen. Das erwartete Ergebnis wird aus den Spezifikationen der OpUT und der aufgerufenen Operation ermittelt.

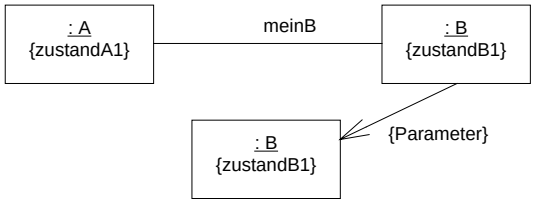

Abbildung 3 Test-Kombination bei Aufruf der Operation B::m2(einB:B)

3 Endekriterien

Endekriterien für den Test eines dynamisch gebundenen Operationsaufrufes sind:

- Der Operationsaufruf erfolgte von Instanzen der Klasse, welche die OpUT definiert, und von Instanzen aller ihrer Unterklassen in allen Zuständen.
- Die aufgerufene Operation wurde von Instanzen der sie definierenden Klasse und von Instanzen aller ihrer Unterklassen in allen möglichen Zuständen ausgeführt.
- Die Parameter des Operationsaufrufes waren Instanzen der in der Signatur der aufgerufenen Operation angegebenen Klassen und aller ihrer Unterklassen in allen möglichen Zuständen.

4 Literatur

[1] John D. McGregor: *Interactions.* Journal of Object-Oriented Programming, Oktober 1999, S. 16-20

[2] M. Winter: *Ein interaktionsbasiertes Modell für den objektorientierten Integrations- und Regressionstest.* Informatik Forschung und Entwicklung, Vol. 15, Nr. 3, 2000, S. 121-132

[3] H.-W. Six, M. Winter: *Objektorientierte Softwareentwicklung.* Kurs 1794, FernUniversität Hagen, 2000

[4] Madhav S. Phadke: *Quality Engineering using Robust Design.* Prentice Hall, Englewood Cliffs, N.J., 1989