Rollenbasierter Test objektorientierter Kollaborationen

Moritz Schnizler
Lehr- & Forschungsgebiet Informatik III
RWTH Aachen, D-52056 Aachen
Moritz.Schnizler@informatik.rwth-aachen.de

Ein häufiges Argument für die objektorientierte Programmierung ist der höhere Grad der Wiederverwendung und damit Produktivität, der mit ihr erreicht werden kann. Vor allem objektorientierte Rahmenwerke als abstrakter Entwurf [Johnson & Foote 88] einer Familie von Anwendungen sind ein sehr leistungsfähiges Mittel, um bereits geleistete Entwicklungsarbeit wiederzuverwenden. Ein Rahmenwerk definiert nicht nur eine bestimmte Klassenstruktur, sondern schreibt zudem ein bestimmtes Zusammenspiel der künftigen Objekte vor. Diese Objektkollaborationen können dann benutzt und erweitert werden, um eine neue Anwendung zu implementieren.

Der hohe Stellenwert der Wiederverwendung wirkt sich auch auf den Test objektorientierter Anwendungen aus. So erfordert der iterative Prozess bei der Entwicklung objektorientierter Software häufige Regressionstests. Dabei erfordern nicht nur neue oder veränderte Operationen einen erneuten Test, sondern beispielsweise auch die neue Konfiguration existierender Objekte in einem Rahmenwerk. Es muss immer wieder überprüft werden, ob die Objekte noch wie ursprünglich gedacht zusammenspielen.

Betrachtet man die heute gängigen Testverfahren für objektorientierte Software, siehe beispielsweise [Binder 99], so zeigt sich, dass der Regressionstest bei den meisten Verfahren nur eine untergeordnete Rolle spielt. Tatsächlich gibt es kein tragfähiges Verfahren, um Testfälle so implementierungs-unabhängig zu formulieren, dass mit ihnen Anwendungen getestet werden können, die auf ein- und demselben Rahmenwerk aufbauen. Was fehlt, ist ein Verfahren für den objektorientierten Integrationstest, mit dem die korrekte Kollaboration mehrerer Objekte gezielt geprüft werden kann.

Zwar gibt es eine ganze Reihe von Verfahren für den Klassentest und die üblichen Verfahren für den Systemtest, wie sie auch bei imperativen Anwendungen zum Einsatz kommen, können relativ leicht auf objektorientierte Systeme angewandt werden. Jedoch das Zusammenspiel mehrerer Objekte wird, wenn überhaupt, meist mit Klassentestverfahren geprüft. Dieses Vorgehen ist meist mit hohem Aufwand verbunden und führt zu einer engen Kopplung der Testfälle an die zugrundeliegenden Klassen. Wird dann im Nachhinein die Konfiguration der beteiligten Objekte verändert, beispielweise durch Erweitern einer beteiligten Klasse, werden die entwickelten Testfälle oft unbrauchbar.

Ein Grund für die Probleme beim Integrationstest ist nicht zuletzt die Vorgehensweise beim objektorientierten Entwurf. Ein vorliegendes Problem wird so in seine Schlüsselabstraktionen zerlegt, dass sich die entstehende Struktur gut mit einem Klassendiagramm beschreiben lässt. Gleichzeitig wird das Verhalten der Anwendung, das letztendlich getestet wird, über alle Klassen verstreut. Um das Zusammenspiel der Objekte zu prüfen, ist deshalb eine Spezifikation des korrekten Zusammenspiels erforderlich.

Die Erfahrung zeigt jedoch, dass in der Praxis die Kollaboration mehrerer Objekte nur selten dokumentiert wird. Auf der einen Seite ist es aufwendig eine Objektkollaboration zum Beispiel mit Hilfe eines UML Interaktionsdiagramms zu spezifizieren. Zusätzlich wird die Modellierung objektorientierter Kollaborationen auch durch das Konzept der Klasse erschwert.

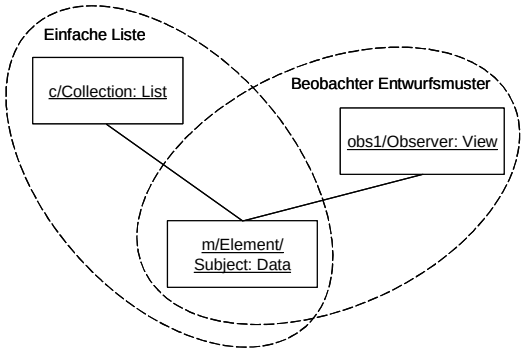

Abb. 1: Kollaborierende Objekte

Ein Interaktionsdiagramm wird traditionell auf Basis der tatsächlich beteiligten Objekte oder deren Klassen definiert. Die Klasse ist jedoch ungeeignet, um die Beteiligten einer Kollaboration implementierungsunabhängig und dadurch wiederverwendbar zu beschreiben. Während ein Objekt meist Teil mehrerer Kollaborationen ist, Abb. 1, ist immer nur ein Teil der Klasse für die jeweilige Kollaboration von Bedeutung.

Bei der Rollenmodellierung, wie sie beispielsweise die OORAM Methode [Reenskaug 96] propagiert, werden Kollaborationen statt dessen auf Basis der beteiligten Rollen spezifiziert. Anstatt des vollständigen Objekts oder dessen Klasse werden nur jene Attribute und Nachrichten berücksichtigt, die im Kontext der Kollaboration relevant sind. Ebenso wird die Semantik auf die entsprechende Rolle beschränkt. Sie ist somit ein Platzhalter für ein beliebiges Objekt, das die geforderten Voraussetzungen erfüllt, um an der Kollaboration teilzunehmen. So ist es möglich, das Verhalten einer Objektkollaboration weitgehend unabhängig von einer konkreten Implementierung zu spezifizieren.

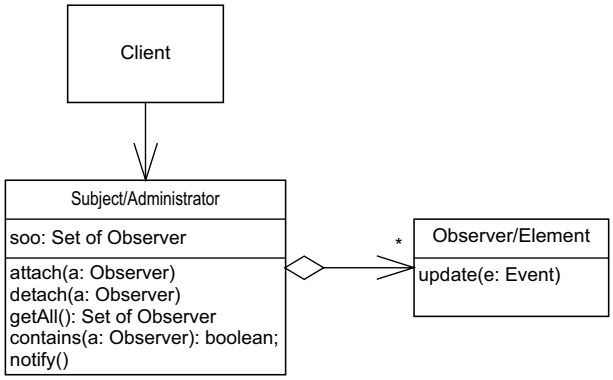

Abb. 2: Beobachter Rollendiagramm

Das Rollenmodell einer Objektkollaboration beschreibt dann, unabhängig von einer konkreten Instantiierung einzelner Klassen, wie die Objekte zusammenspielen. Es dient als Grundlage für den rollenbasierten Test. Die Beziehungen zwischen den Objekten, das heißt die Struktur der Kollaboration, wird dazu mit einem Rollendiagramm spezifiziert. So zeigt beispielsweise Abb. 2 das Rollendiagramm für die Kollaboration zweier Objekte gemäß dem Beobachter-Entwurfsmuster [Gamma et al. 95]. Die Semantik der Kollaboration wird zusätzlich natürlich-sprachlich beschrieben. Neben dem allgemeinen Kollaborationsziel werden für jede beteiligte Rolle die Verantwortlichkeiten sowie die notwendigen Attribute und Nachrichten notiert, die das Objekt erfüllen oder mitbringen muss.

Um Testfälle für die Kollaboration zu gewinnen, werden anschließend gemäß der Methode von [Ostrand & Balcer 88] Kategorien hergeleitet, die über den Ablauf der Kollaboration entscheiden. Dazu wird in erster Linie der Einfluss verschiedener Werte für die Attribute und Parameter der beteiligten Rollen und ihrer Nachrichten untersucht. Für das Beispiel in Abb. 2 bilden beispielsweise die Anzahl der registrierten Beobachter sowie der Wert des Parameters `a` der Nachricht `attach` jeweils eine eigene Kategorie. Jede Kategorie wird anschließend in Wertebereiche zerlegt oder besser gesagt partitioniert, die theoretisch zu unterschiedlichem Verhalten der beteiligten Objekte führen. Aus jedem dieser Wertebereich wird wenigstens ein Vertreter als Teil eines Testfalls ausgewählt.

Die verschiedenen Kombinationen von Werten für die ermittelten Kategorien führen zu einer Tabelle, die ergänzt um einen erwarteten Rückgabewert oder ein Sequenzdiagramm der erwarteten Nachrichtenfolge im Kontext des Rollenmodells die Testfälle für eine beliebige Implementierung des Rollenmodells darstellen. Bei der Definition der Testfälle, insbesondere bei der Bildung von Kategorien, können Beobachtungen und Fehler früherer Implementierungen der Kollaboration nützlich sein. Auf diese Weise können rollenmodellspezifische Testfälle gesammelt werden, die sich für jede Implementierung der Kollaboration eignen.

Rollenmodelle können mittels Synthese auch kombiniert werden, um das Modell einer komplexeren Kollaboration zu erhalten. Beispielsweise trifft man häufig auf die Situation, dass mehrere Objekte in einer Kette angeordnet sind, wobei jedes Objekt mit seinem Vorgängerobjekt gemäß dem Beobachter-Rollenmodell kollaboriert. Diese Kette von Objekten kann ebenfalls durch ein Rollenmodell spezifiziert werden, das zweimal das Beobachter-Rollenmodell kombiniert. Abhängig von der Semantik, die der Ereignispropagierung entlang der Objektkette zugrunde liegt, können dann Testfälle formuliert werden, die prüfen, ob dieses Protokoll von allen Objekten korrekt implementiert wird.

Der vorgestellte Ansatz erlaubt es, eine implementierungs-unabhängige Sicht und besser wiederverwendbare Testfälle für Objektkollaborationen zu erhalten. Der relativ hohe Aufwand für die Rollenmodellbildung scheint dabei aufgrund der Wiederverwendbarkeit der entstehenden Testfälle gerechtfertigt. Dabei darf auch der Zusatznutzen nicht übersehen werden, wenn komplexe Kollaborationen durch Rollenmodelle dokumentiert werden.

Das hier vorgestellte Verfahren soll als Grundlage auf dem Weg zu einem Prüfstand für rahmenwerksbasierte Anwendungen dienen, wie er in [Schnizler & Lichter 00] vorgestellt wird. In einem nächsten Schritt soll die Implementierung der rollenbasierten Testfälle in einem Testrahmenwerk, wie beispielsweise JUnit, untersucht werden, was dem Prüfstandsgedanken schon ziemlich nahe kommt.

[Binder 99] Binder, R.: *Testing Object-Oriented Systems: Models, Patterns, and Tools*, Addison-Wesley, 1999.

[Gamma et al. 95] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns - Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[Johnson & Foote 88] Johnson, R. E., Foote, B.: *Designing Reusable Classes*, JOOP, vol. 1, no. 2, pp. 20 - 30; 35, Juni/Juli, 1988.

[Ostrand & Balcer 88] Ostrand, T.J., Balcer, M.J.: *The Category-Partition Method for Specifying and Generating Functional Tests*, Communications of the ACM, vol. 31, no. 6, pp. 676 – 686, Juni, 1988.

[Reenskaug 96] Reenskaug, T., Wold, P., Lehne, O. A.: *Working With Objects*, Manning Publications Co., 1996.

[Schnizler & Lichter 00] Schnizler M., Lichter H.: *Test Automation for Object-Oriented Frameworks*, In Proc. First International Workshop on Program Analysis, Testing and Verification, Limerick, Ireland, 4-5 Juni, 2000.