Strukturelle Testabdeckung funktionaler Spezifikationen

Mario Friske
Fraunhofer FIRST, Kekuléstr. 7, D-12489 Berlin
mario.friske@first.fraunhofer.de

Abstract: In diesem Papier wird dargestellt, wie strukturelle Abdeckungskriterien auf mithilfe von Metamodellen formalisierte funktionale Spezifikationen übertragen werden können. Bewährte kontroll- und datenflussbasierte Kriterien, die üblicherweise die Testvollständigkeit in Bezug auf den Programmcode beschreiben, lassen sich so auch zur Überdeckungsmessung spezifizierter Interaktionsflüsse nutzen.

1 Einleitung

In der Literatur wird streng zwischen funktionsorientierten und strukturorientierten Testverfahren unterschieden [Lig02]. Während die funktionsorientierten Testtechniken von der Spezifikation ausgehen, wird bei den strukturorientierten Testtechniken die Vollständigkeit der Tests anhand der Abdeckung des Programmcodes beurteilt. Für den strukturorientierten Test ist eine Vielzahl von Abdeckungskriterien definiert und deren Leistungsfähigkeit ausgiebig dargestellt. Relationen zwischen Kriterien sind in Subsumptionshierarchien beschrieben, z. B. in [Lig02]. Demzufolge können Tester leicht eine für eine Testaufgabe angemessene Kombination struktureller Abdeckungskriterien bestimmen. Schwieriger ist es, angemessene Abdeckungskriterien für den funktionsorientierten Test auszuwählen:

1. Es wird eine große Bandbreite von Notationsstilen unterschiedlichen Formalisierungsgrades für Anforderungen verwendet: von informellen Beschreibungen über strukturierten Text bis hin zu formalen Modellen.
2. Die Anwendung klassischer funktionsorientierter Testverfahren erfordert die vorherige Ableitung weitergehender Datenstrukturen aus der Spezifikation, wie z. B. Ursache-Wirkungs-Graphen, welche nicht trivial ist. Auf diesen Strukturen definierte Abdeckungskriterien lassen sich nicht 1:1 in der Spezifikation wiedererkennen.
3. Typische funktionsorientierte Abdeckungskriterien erreichen bei Weitem nicht die Komplexität und Leistungsfähigkeit strukturorientierter Kriterien. Stellenweise werden nur Trivialkriterien angewandt, wie mindestens einmalige Referenz jeder Anforderung durch einen Testfall.
4. Aufwand, Leistungsfähigkeit und die Subsumptionsrelationen funktionsorientierter Kriterien sind nicht so ausführlich untersucht worden, wie es bei den strukturorientierten Kriterien der Fall ist. Insbesondere das Fehlen von Subsumptionshierarchien funktionsorientierter Kriterien erschwert die Auswahl durch den Tester deutlich.

Aufgrund obiger Punkte fällt es Testern in der Regel äußerst schwer, für die weit verbreiteten textuellen Spezifikationen eine angemessene Abdeckung zu definieren und die mit der auf deren Grundlage erstellten Testsuite tatsächlich erzielte Abdeckung präzise anzugeben.

In diesem Papier wird dargestellt, wie strukturelle Abdeckungskriterien auf funktionale Spezifikationen mit entsprechendem Formalisierungsgrad übertragen werden können. Voraussetzung ist, dass in einem vorhergehenden Formalisierungsschritt die der funktionalen Spezifikation zugrunde liegende Struktur in Form eines durch ein Metamodell definierten Modells expliziert wird. In den im Folgenden als Ausgangspunkt genutzten formalisierten Anwendungsfallbeschreibungen ist der spezifizierten Interaktionsfolgen zugrunde liegende Kontroll- und Datenfluss explizit gemacht worden, sodass die in [Lig02] beschriebenen kontroll- und datenflussbasierten strukturorientierten Kriterien unmittelbar übertragen werden können.

Der verbleibende Teil dieses Papiers ist wie folgt gegliedert: In Abschnitt 2 wird die Bedeutung von Abdeckungskriterien erklärt und auf typische Kriterien für den spezifikationsbasierten Test eingegangen. In Abschnitt 3 werden unter Verwendung eines Metamodells für Anwendungsfallbeschreibungen strukturelle Abdeckungskriterien für den funktionsorientierten Test definiert. Schließlich werden in Abschnitt 4 die Ergebnisse und weitergehende Fragestellungen diskutiert.

2 Funktions- und strukturorientierte Abdeckungskriterien

Zu den wesentlichen Artefakten in qualitätsgesicherten Softwareentwicklungsprozessen zählen *Spezifikation*, *Implementierung* und *Testsuite*. Sowohl Implementierung als auch Testsuite werden auf Grundlage der Spezifikation erstellt. Zwischen Spezifikation und Implementierung besteht eine Verfeinerungsbeziehung, in welcher idealerweise die Implementierung sämtliche spezifizierten Anforderungen realisiert. Mithilfe der Testsuite wird die Umsetzung der Anforderungen in der Implementierung durch Ausführung überprüft. Da bei realen Systemen ein vollständiger Test nicht möglich ist, gilt es, eine geeignete Auswahl an Testfällen zu treffen.

Der Grad der Vollständigkeit der Tests kann durch *Abdeckungskriterien* qualifiziert werden. Die Vollständigkeit der Testsuite in Bezug auf die Spezifikation wird mithilfe *funktionsorientierter Testtechniken* und in Bezug auf die Implementierung mittels *strukturorientierter Testtechniken* beurteilt. Die Beurteilung erfolgt bei den einzelnen Testtechniken anhand korrespondierender *funktionsorientierter* bzw. *strukturorientierter Abdeckungskriterien*.

Zur Notation funktionaler Anforderungen werden in der Praxis verschiedenartige Spezifikationsstile eingesetzt, siehe [Lau02], z. B. anwendungsfallbasierte Spezifikationen. Für die einzelnen Spezifikationsstile stellt sich die Situation hinsichtlich Testmethoden und Abdeckungskriterien unterschiedlich dar: Die Abdeckung, die mit Testmethoden für anwendungsfallbasierte Spezifikationen erzielt werden kann, bezieht sich oft nicht unmittelbar auf die Interaktionsabläufe. So bezieht sich beispielsweise die CRUD-Abdeckung

[Bin99] auf Klassen des Datenmodells. Für Feature-basierte Spezifikationen wird nicht selten in der Praxis das Kriterium „Mindestens ein Testfall pro Feature“ verwendet. Für zustandsbasierte Spezifikationen existieren in der wissenschaftlichen Literatur umfangreiche Abdeckungskriterienkataloge, siehe z. B. [Bin99]. Die mit kommerziellen Generatoren, wie beispielsweise ATG [IL04] erzielbare Abdeckung ist jedoch beschränkt (MCDC auf dem generierten Code) und statt auf den funktionsorientierten Test eher auf den modellbasierten Whitebox-Test ausgerichtet. Für den funktionsorientierten Test existiert kein allgemein akzeptiertes Minimalkriterium [Lig02].

Die strukturorientierten Kriterien lassen sich in die drei Kategorien Pfadüberdeckungskriterien, Bedingungsüberdeckungskriterien und Datenflussüberdeckungskriterien einteilen. Pfadüberdeckungskriterien beschreiben, in welchem Umfang Programmschleifen getestet werden. Bedingungsüberdeckungskriterien betrachten die logische Struktur zusammengesetzter komplexer Entscheidungen und gewährleisten den Test komplizierter Verarbeitungslogik. Datenflussorientierte Kriterien bewerten die Testvollständigkeit im Hinblick auf Datenzugriffe. Beim Einordnen in eine gemeinsame Subsumptionshierarchie bilden die beiden Kriterien Pfadüberdeckung und Zweigüberdeckung die Vereinigungspunkte zwischen kontrollflussbasierten und datenflussbasierten Kriterien, siehe Abbildung 1. Pfadüberdeckung ist in der Regel nicht erreichbar. Zweigüberdeckung gilt als allgemein akzeptiertes Minimalkriterium [Lig02].

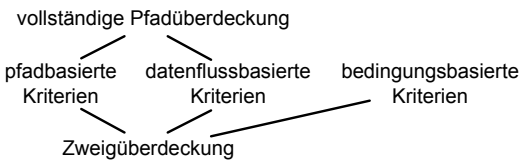

Abbildung 1: Beziehungen zwischen strukturorientierten Abdeckungskriterien

3 Übertragung strukturorientierter Abdeckungskriterien auf funktionale Spezifikationen

In diesem Abschnitt wird am Beispiel anwendungsfallbasierter Spezifikationen skizziert, wie strukturorientierte Testabdeckungskriterien auf funktionale Spezifikationen übertragen werden können. Dazu wird das in Abbildung 2 dargestellte Anwendungsfallmetamodell genutzt. Es ermöglicht die Formalisierung von Interaktionsflüssen, welche sowohl die Interaktionsformalisierung (links unten) als auch die Kontrollflussformalisierung (rechts) beinhaltet. Bei der Interaktionsformalisierung werden allen in einer Textzeile (`TextLine`) enthaltenen Schritten (`Step`) die beschriebenen Interaktionen (`Interaction`) zugeordnet. Die Kontrollflussformalisierung umfasst u. a. Sequenzen (`Sequence`), Schleifen (`Loop`) und Verzweigungen (`Switch`), jeweils einschließlich Bedingungen (`Condition`). Detaillierte Beschreibungen des Metamodells und des zugehörigen Formalisierungsverfahrens sind in [FP05, FS05] zu finden.

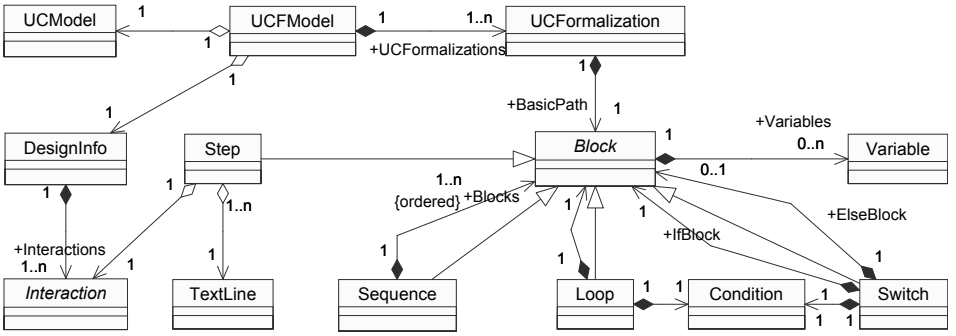

Abbildung 2: Metamodell einer formalisierten Anwendungsfallbeschreibung

Interaktionsflussformalisierungen in Form von Instanzen des Anwendungsfallmetamodells beinhalten sämtliche Informationen, um aus ihnen einen Kontrollflussgraphen zu generieren. Die Generierung kann mittels einer Transformation erfolgen, in welcher Blöcke aus dem Metamodell in Knoten des Kontrollflussgraphen und Verbindungen zwischen den Blöcken in Kanten überführt werden. Somit lassen sich strukturelle Abdeckungskriterien, die auf dem Kontrollflussgraphen definiert sind, auf den formalisierten Interaktionsfluss übertragen. Als Hilfsmittel wird dazu das in Abbildung 3 dargestellte Analogierad [EK04] verwendet. In ihm werden ausgehend von Abdeckungsmerkmalen eines Kontrollflussgraphen systematisch Analogien in formalisierten Anwendungsfallbeschreibungen ermittelt. Es zeigt, dass zu sämtlichen Konzepten des Kontrollflussgraphen Äquivalenzen im formalisierten Anwendungsfallmodell existieren und somit die strukturorientierten Kriterien auf dieses übertragen werden können.

Zur Definition struktureller Abdeckungskriterien und zugehöriger Testziele wird die OCL [Obj06] genutzt, siehe auch [FSW08]. Einer der Vorteile ist, dass Abdeckungskriterien in Form von OCL-Abfragen bei Anwendung auf ein spezifisches Modell die Menge sämtlicher zu erfüllender Testziele liefern können. Ein einzelnes Testziel in OCL beschreibt dabei eine spezifische Kombination von Modellelementen eines konkreten Modells.

Für die Definition mittels OCL ergeben sich zwei Möglichkeiten: (1.) Kriterien werden direkt unter Verwendung des Anwendungsfallmetamodells in Abbildung 2 definiert. Vorteil ist die Unmittelbarkeit der Definition – nachteilig ist das Fehlen einer expliziten Repräsentation des Konzepts einer Kontrollflussgraphkante, woraus komplexe OCL-Ausdrücke resultieren. (2.) Aus der Formalisierung wird zunächst der Kontrollflussgraph erzeugt, welcher durch ein explizites Kontrollflussgraphmetamodell definiert wird. Auf diesem werden die Abdeckungskriterien definiert. Vorteilhaft ist hierbei die Einfachheit der Definition – nachteilig ist, durch die weitere Indirektionsstufe bedingt, das Fehlen der Unmittelbarkeit der Definition.

Es folgt als einfaches Beispiel die OCL-basierte Definition des Kriteriums *allBlocks*, welches der Anweisungsüberdeckung entspricht, unmittelbar auf dem Metamodell:

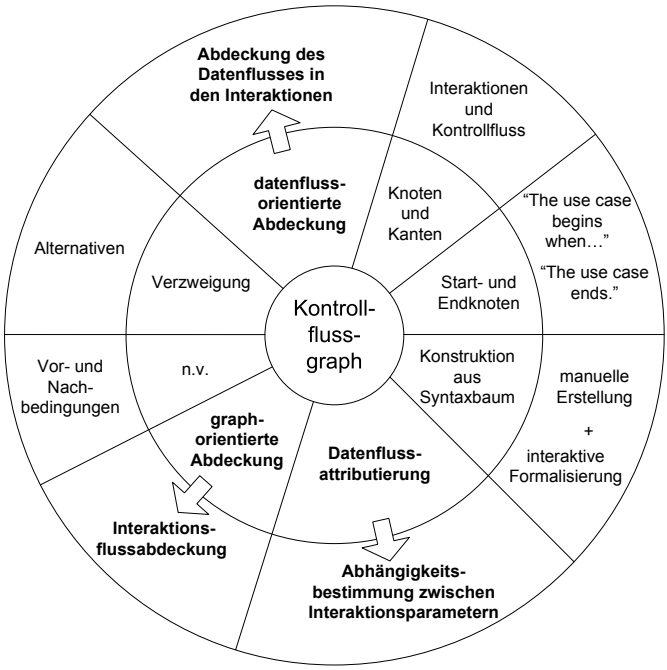

Abbildung 3: Übertragung struktureller Merkmale von Kontrollflussgraphen auf funktionale Spezifikationen mithilfe eines Analogierades

```
context UCFormalization
def: allBlocks: set(Block) = self.BasicPath.getAllBlocks
```

Obige Definition nutzt eine rekursiv definierte Hilfsfunktion `getAllBlocks`, die zu einem Block sämtliche enthaltene elementare Schritte liefert. Das der Zweigüberdeckung entsprechende Kriterium *allEdges* kann in OCL durch die Abfrage sämtlicher enthaltener Zweiersequenzen von Blöcken definiert werden:

```
context UCFormalization
def: allEdges : Set(Sequence(Block)) = ...
```

Auf diese Art und Weise lassen sich beliebig komplexe strukturelle Abdeckungskriterien auf funktionale Interaktionsflussbeschreibungen anwenden.

4 Diskussion und Ausblick

Wie oben skizziert, ist es möglich, strukturorientierte Testabdeckungskriterien, die üblicherweise zur Beurteilung der Testvollständigkeit im Hinblick auf den Programmcode verwendet werden, ebenso auf formalisierte funktionale Ablaufbeschreibungen anzuwen-

den. Der Tester kann so eine für eine Testaufgabe angemessene Kriterienkombination aus Pfad-, Bedingungs- und Datenflussüberdeckungskriterien wählen [FSW08]. Mit dem Ansatz können sowohl die häufig verwendeten einfachen, auf mindestens einmaliger Referenz basierenden Kriterien als auch die komplexeren in [Lig02] katalogisierten Kriterien umgesetzt werden.

Durch die Testsuite soll möglichst die vollständige Spezifikationstreue nachgewiesen werden. Es ist davon auszugehen, dass sämtliche in Interaktionsbeschreibungen spezifizierten Interaktionssequenzen und Schleifen, einschließlich Bedingungen und Datenfluss von Relevanz sind, ansonsten wären sie nicht explizit spezifiziert worden. Präzise Angaben zur Vollständigkeit des Tests dieser Konstrukte werden mithilfe von strukturorientierten Abdeckungskriterien möglich.

Interessante weiterführende Fragestellungen ergeben sich an dieser Stelle bezüglich der Korrespondenz zwischen Spezifikationsabdeckung, Implementierungsabdeckung und der jeweiligen Fehlerabdeckung. Wie in [Pos96] dargestellt, ist idealerweise der spezifikationsbasierte Test mit einer Abdeckungsmessung zu kombinieren, da nur so in der Implementierung vorhandene zusätzliche Funktionalitäten bzw. Auslassungen in der Spezifikation entdeckt werden können. Aufgrund der Partialität funktionaler Spezifikationen, d. h., nur wichtige Aspekte sind festgehalten, wird es in den meisten Fällen keine 1:1-Korrespondenz bei der Abdeckung von Spezifikation und Implementierung geben. Insbesondere in einem solchen Fall stellt sich die Frage nach der erzielbaren Fehlerabdeckung.

Literatur

[Bin99] Robert V. Binder. *Testing Object-Oriented Systems: Models, Patterns, and Tools*. Object Technology Series. Addison-Wesley, 1999.

[EK04] Helga Esselborn-Krumbiegel. *Von der Idee zum Text. Eine Anleitung zum wissenschaftlichen Schreiben*. Verlag Ferdinand Schöningh, 2004.

[FP05] Mario Friske und Holger Pirk. Werkzeuggestützte interaktive Formalisierung textueller Anwendungsfallbeschreibungen für den Systemtest. In *Beiträge der 35. Jahrestagung der Gesellschaft für Informatik (Band 2)*, Jgg. P-68 of *LNI*, September 2005.

[FS05] Mario Friske und Holger Schlingloff. Von Use Cases zu Test Cases: Eine systematische Vorgehensweise. In *Tagungsband des Dagstuhl-Workshops MBEES I*, Januar 2005.

[FSW08] Mario Friske, Holger Schlingloff und Stephan Weißleder. Composition of Model-based Test Coverage Criteria. In *Tagungsband des Dagstuhl-Workshops MBEES IV*, April 2008.

[IL04] I-Logix. *Rhapsody Automatic Test Generator, Release 2.3, User Guide*, 2004.

[Lau02] Soren Lauesen. *Software Requirements - Styles and Techniques*. Addison-Wesley, 2002.

[Lig02] Peter Liggesmeyer. *Software-Qualität: Testen, Analysieren und Verifizieren von Software*. Spektrum Akadamischer Verlag, 2002.

[Obj06] Object Management Group. OCL 2.0 Specification, version 2.0 (formal/06-05-01), 2006.

[Pos96] R. M. Poston. *Automating Specification-Based Software Testing*. IEEE Computer Society, Los Alamitos, 1.˙ Auflage, 1996.