

Interaktive Aufbereitung von Anforderungen für den modellbasierten Test

Mario Friske
Fraunhofer FIRST, Berlin
mario.friske@first.fhg.de

Zusammenfassung. Es wird am Beispiel von Anwendungsfallbeschreibungen dargestellt, wie informelle Anforderungsspezifikationen in testbare Modelle überführt werden können. Die Anforderungen werden interaktiv angereichert und Mehrdeutigkeiten werden aufgelöst, indem Entwurfsinformation berücksichtigt wird. Der metamodellbasierte Ansatz ermöglicht es, das resultierende Modell mit MDA-basierten Testfallgenerierungsverfahren unmittelbar weiterzuverarbeiten.

Motivation. Das Testen zählt zu den wichtigsten qualitätssichernden Maßnahmen in der Softwareentwicklung. Durch den Softwaretest soll stichprobenartig und risikobasiert die Konsistenz zwischen Anforderungsspezifikation und Implementierung geprüft werden. Da eine manuelle Ableitung der Testfälle aus den Anforderungen aufwendig und fehlerträchtig ist und oft intuitiv und unsystematisch geschieht, ist eine automatisierte Testfallerstellung wünschenswert.

Als problematisch erweist sich jedoch, dass die Anforderungsspezifikationen in der Regel nicht formal genug sind, um aus ihnen unmittelbar Testfälle abzuleiten. Mit dem modellbasierten Testen liegt ein allgemeiner Ansatz zur automatisierten Testfallableitung aus Modellen vor. Dieser setzt ein formalisiertes Modell als Ausgangspunkt voraus. Es bleibt die Frage, wie man von den informellen Anforderungen zum testbaren Modell kommt.

Nutzung von Entwurfsinformation zur Anforderungsaufbereitung. Das zu erstellende Modell soll als Ausgangspunkt für die automatische Testfallgenerierung genutzt werden. Da ein Testfallgenerator ein eindeutig zu interpretierendes Modell erfordert, sind bei dessen Erstellung aus den Anforderungen darin enthaltene Mehrdeutigkeiten aufzulösen.

Anforderungsspezifikationen sollten so geschrieben sein, dass sie noch jede mögliche Systemrealisierung zulassen, welche die Anforderungen des Auftraggebers erfüllt. Im Systemtest soll die vorhandene Implementierung gegen die gestellten Anforderungen geprüft werden, d.h. es muss nur die tatsächliche Realisierung der Spezifikation betrachtet werden. Die gefallenen Entwurfsentscheidungen lassen sich als Entwurfsinformation für die Formalisierung nutzen [3].

Beispiel: Aufbereitung von Anwendungsfallbeschreibungen. Eine häufig genutzte Spezifikationsform für funktionale Anforderungen an Softwaresysteme sind textuelle Anwendungsfallbeschreibungen (Use

Cases). Diese beschreiben typische Nutzer-System-Interaktionen, deren Ausführung einen bestimmten Nutzen für den Anwender des Systems hat.

Bei der Anforderungsanalyse großer Softwaresysteme werden in der Regel Teams von Experten aus den unterschiedlichsten Fachrichtungen eingesetzt, die parallel und verteilt einzelne Use Cases erstellen und zur Menge der Anforderungen hinzufügen. Daraus ergeben sich oft Anwendungsfallbeschreibungen, die zwar durchaus als Grundlage für die Systementwicklung genutzt werden können, aus denen sich jedoch aufgrund von Inkonsistenzen und Mehrdeutigkeiten nicht unmittelbar automatisch Testfälle generieren lassen.

Bei der Aufbereitung von Anwendungsfallbeschreibungen sind zwei Probleme zu lösen: Einerseits ist der den Interaktionen zugrunde liegende Kontrollfluss zu bestimmen und andererseits sind die einzelnen Interaktionsschritte zu formalisieren. Der Kontrollfluss ist sowohl explizit in der Struktur des Textes enthalten, als auch implizit in den Interaktionsschritten in Textform beschrieben. Zum Beispiel wird die sequentielle Abfolge von Interaktionsschritten in der Regel explizit durch die Textstruktur festgelegt, während jedoch Abbruchbedingungen meist implizit im Text beschrieben werden. Beide Formen sollen formalisiert und in dem zu erstellenden Modell repräsentiert werden. Die Interaktionsschritte beschreiben die vom Akteur aufgerufenen Systemfunktionen und die Systemreaktionen, wobei die Interaktionen oftmals an verschiedenen Stellen in der Anforderungsspezifikation auf unterschiedliche Art und Weise textuell repräsentiert sind. In dem zu erstellenden Modell sollen diese Mehrdeutigkeiten aufgelöst sein.

Mit dem bei Fraunhofer FIRST entwickelten *Use Case Validator* (UCV) lässt sich diese Aufbereitung werkzeuggestützt durchführen. Dazu werden zunächst die textuellen Anwendungsfallbeschreibungen eingelesen und die einzelnen Abschnitte der Beschreibungen interaktiv typisiert. Anschließend wird die Entwurfsinformation importiert, d.h. Angaben über Akteure, Systemfunktionen und -reaktionen einschließlich deren Parameter. Die Entwurfsinformation kann ggf. interaktiv erweitert und modifiziert werden. Es ist ebenfalls möglich, die Entwurfsinformation ohne Import vollständig mit dem UCV zu erstellen.

Nun kann der eigentliche Formalisierungsvorgang erfolgen. Der Kontrollfluss wird durch Markieren von Blöcken und Zuordnen von Kontrollflusselementen formalisiert. Den einzelnen Schritten werden die Sys-

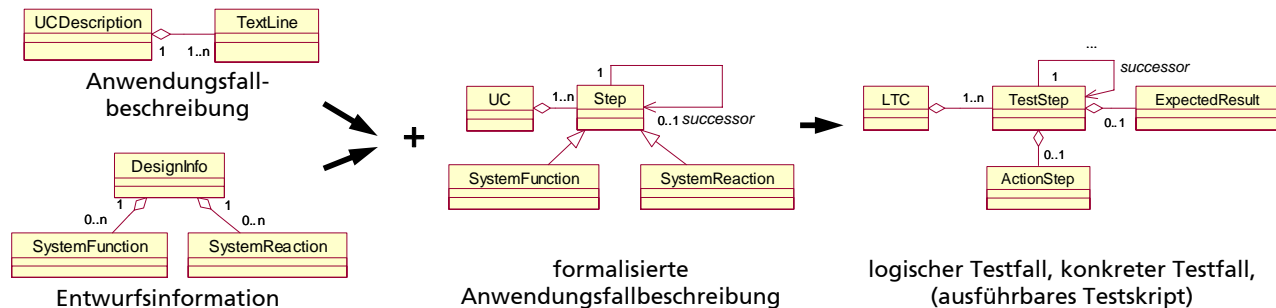


Abbildung 1: Interaktives Verknüpfen von Anforderungen und Entwurfswissen sowie automatisierte Testfallgenerierung mithilfe von Metamodellen (vereinfachte Darstellung)

temfunktionen und -reaktionen aus der Entwurfswissen zugeordnet. So wird schrittweise ein formales Modell der textuellen Anwendungsfallbeschreibung erstellt, in welchem die Mehrdeutigkeiten der textuellen Repräsentation interaktiv beseitigt worden sind. Dieses Modell ist durch das in [2] beschriebene Metamodell definiert, welches im Folgenden skizziert wird.

Metamodellbasierter Ansatz. Eine unserer Zielsetzungen bei der Entwicklung des UCV war es, für jegliche Artefakte und Zwischenstufen Metamodelle zu nutzen, d.h. sowohl für die Aufbereitung der Anforderungen, als auch für die Testfallgenerierung. Weiterhin sollten die textuellen Anwendungsfallbeschreibungen und die während der Formalisierung hinzugefügten Informationen getrennt gehalten werden. Eine grobe Skizze der aus diesen Forderungen resultierenden Metamodelle ist in Abbildung 1 dargestellt.

Im ersten Schritt, der interaktiven Aufbereitung, wird die aus unspezifizierten Textzeilen bestehende Anwendungsfallbeschreibung mit der Entwurfswissen, d.h. Systemfunktionen und -reaktionen, verknüpft und weitere Informationen zur Präzisierung des Kontrollflusses hinzugefügt. Folglich werden die Instanzen der zugehörigen Metamodelle verknüpft und angereichert und daraus eine Instanz des Metamodells der formalisierten Anwendungsfallbeschreibung erstellt. In dieser sind den einzelnen Schritten jeweils Systemfunktionen und -reaktionen zugeordnet.

In einem zweiten Schritt lassen sich anschließend aus den formalisierten Anwendungsfallbeschreibungen Testfälle durch automatisierte Modelltransformation generieren. Dazu können die Techniken und Werkzeuge der *Model Driven Architecture* (MDA) verwendet werden [4, 1]. Wie im rechten Teil von Abbildung 1 skizziert, lassen sich aus dem Modell logische plattformunabhängige Testfälle erzeugen, welche in konkrete plattformspezifische Testfälle transformiert werden können. Die plattformspezifischen Testfälle können anschließend in ausführbare Testskripte überführt werden. Die erforderlichen Transformationen sind abhängig von den gewünschten Teststrategien und Abdeckungskriterien zu erstellen.

Fazit und Ausblick. Der skizzierte metamodellebasierte Ansatz zur interaktiven Anforderungsaufbereitung hat sich bei der Realisierung des *Use Case Validator* als praktikabel erwiesen. Momentan werden darin die wichtigsten aber noch nicht alle in Anwendungsfällen auftretenden Kontrollflusskonstrukte unterstützt. Das durch interaktive Aufbereitung erstellte Modell ist formal genug, um Ausgangspunkt für die automatische Testfallgenerierung zu sein. Eine Vielzahl von Erweiterungen ist denkbar. Stellvertretend sei hier die automatisierte Extraktion der Entwurfswissen aus Entwurfsdokumenten genannt.

Literatur

- [1] Mario Friske: *Testfallerzeugung aus Use-Case-Beschreibungen*. Softwaretechnik-Trends, Band 24, Heft 3, 2004.
- [2] Mario Friske und Holger Pirk: *Werkzeuggestützte interaktive Formalisierung textueller Anwendungsfallbeschreibungen für den Systemtest*. In: A. B. Cremers, R. Manthey, P. Martini, und V. Steinhage (Hrsg.): *INFORMATIK 2005 - Informatik LIVE! Band 2, Beiträge der 35. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, Bd. 68 d. Reihe LNI. GI, September 2005.
- [3] Mario Friske und Holger Schlingloff: *Von Use Cases zu Test Cases: Eine systematische Vorgehensweise*. In: T. Klein, B. Rumpe, und B. Schätz (Hrsg.): *Tagungsband des Dagstuhl Workshops "Modellbasierte Entwicklung eingebetteter Systeme" (MBEES)*. Technische Universität Braunschweig, Januar 2005.
- [4] Object Management Group: *Model Driven Architecture (MDA)*. <http://www.omg.org/mda/>, 2003.