SCORES: Ein UML-Profil für die objektorientierte Anforderungsanalyse

Beitrag für den 7-ten GROOM-Workshop "UML - Erweiterungen (Profile) und Konzepte der Metamodellierung"

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

Die Spezifikation von Softwaresystemen umfasst im wesentlichen die Beschreibung funktionaler, verhaltens- und ablauforientierter sowie struktureller Systemeigenschaften. In der UML-basierten objektorientierten Anforderungsanalyse werden funktionale Anforderungen mit Hilfe von Anwendungsfällen (Use Cases) formuliert und strukturelle Anforderungen durch Klassenmodelle beschrieben. Die UML empfiehlt Zustands- bzw. Interaktionsdiagramme zur Modellierung von verhaltens- bzw. ablauforientierten Aspekten.

Für Anwendungsfälle ist der ablauforientierte Aspekt, d.h. der „Kontrollfluss", meist wichtiger als das ereignisgesteuerte Verhalten, so dass wir uns darauf konzentrieren. Zur Modellierung dieser Dynamik sieht die UML Interaktions- und Aktivitätsdiagramme vor. Ein Interaktionsdiagramm kann allerdings nicht einen gesamten Anwendungsfall beschreiben, sondern lediglich eine einzelne spezifische Aktionsfolge (Szenario) im Rahmen des Anwendungsfalls. Ansätze, einen Anwendungsfall durch eine Menge von Interaktionsdiagrammen zu beschreiben, führen nicht zum Ziel. Auch Aktivitätsdiagramme sind nicht frei von Problemen. Ein Anwendungsfall-Modell sollte drei Informationsarten unterscheiden können: die systeminterne Information, die sich auf den Systemkern bezieht, die Interaktionsinformation, die auf die direkte Interaktionen des Systems mit seiner Umgebung fokussiert, und die Kontextinformation, welche die Systemumgebung beschreibt. Ein Aktivitätsdiagramm repräsentiert aber nur ein einzelnes Modellelement, z.B. eine Operation, eine Klasse oder ein Gesamtsystem, und kann z.B. nicht die Interaktion mehrerer Elemente beschreiben.

Schließlich fehlen in der UML-Spezifikation Ansätze, die wichtigen «include»- und «extend»-Assoziationen zwischen Anwendungsfällen auf entsprechende Aktivitätsdiagramme zu übertragen. Die «include»-Assoziation beispielsweise unterstützt Modularität, da durch sie das redundante Modellieren eines Anwendungsfalls vermieden wird, der Teil verschiedener übergeordneter Anwendungsfälle ist.

Um diese Nachteile zu überwinden, erweitert und präzisiert das SCORES (Systematic Coupling Of REquirements Specifications [1]) UML-Profil Aktivitätsdiagramme im Hinblick auf die Modellierung des dynamischen Verhaltens von Anwendungsfällen. Das Profil enthält Stereotype für die UML-Metaklassen ActionState, SubmachineState und ObjectFlowState sowie entsprechende TaggedValues und Constraints (Tab. 1). Die beiden Stereotypen «Kontextaktion» und «Interaktion» für die UML-Metaklasse ActionState sowie einige TaggedValues für die UML-Metaklassen Actor und UseCase reflektieren Kontext- und Interaktionsinformation. Systeminterne Information wird ausgedrückt durch den TaggedValue Klassenbereich für die UML-Metaklasse ActionState sowie durch Folgen von Operationsausführungen im Klassenmodell, die mit Interaktionsdiagrammen modelliert werden. Der Stereotyp «Makroaktion» für die UML-Metaklasse SubmachineState dient zusammen mit entsprechenden Constraints als Vehikel, um die «include»- und «extend»-Assoziationen zwischen Anwendungsfällen auf Aktivitätsdiagramme zu übertragen. Entsprechend vermeidet «Makroaktion» das redundante Modellieren von Aktivitätsdiagrammen, die in mehreren anderen Aktivitätsdiagrammen vorkommen.

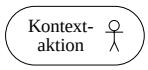
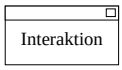
Stereotyp	*Beschreibung*	*Darstellung*
«Kontextaktion»	Ein ActionState mit dem Stereotyp «Kontextaktion» kennzeichnet eine Aktion vom Typ Kontext, die ohne Unterstützung des Systems allein von den Akteuren bearbeitet wird.	
«Interaktion»	Ein ActionState mit dem Stereotyp «Interaktion» beschreibt eine Aktion vom Typ Interaktion, die von den Akteuren mit Unterstützung des Systems, d.h. interaktiv, bearbeitet wird.	
«Makroaktion»	Ein SubmachineState mit dem Stereotyp «Makroaktion» steht für eine Aktion vom Typ Makro, die einen anderen Anwendungsfall (genauer: ein anderes Aktivitätsdiagramm) „aufruft".	
«AktorInAktion»	Eine Instanz der UML-Metaklasse ObjectFlowState mit dem Stereotyp «*AktorInAktion*» verbindet einen Akteur mit Aktionen.	

Tabelle 1. SCORES-Stereotype für Aktivitätsdiagramme

Das den Anwendungsfall *Geld Abheben* eines Bankautomaten verfeinernde Aktivitätsdiagramm zeigt Abb. 1. Die Startaktion ist durch den Übergang vom Startzustand ersichtlich, Endaktionen durch Übergänge zu Endzustän-

den. Die Startaktion stellt eine Makroaktion dar, die den Anwendungsfall bzw. das Aktivitätsdiagramm *Anmelden* „aufruft“. Die Aktion *Geld Entnehmen* stellt eine Kontextaktion dar. Die restlichen Aktionen sind Interaktionen. Da bestimmte Szenarien des Anwendungsfalls Geld Abheben mit einer fehlgeschlagenen Anmeldung enden können, ist die Startaktion gleichzeitig eine Endaktion.

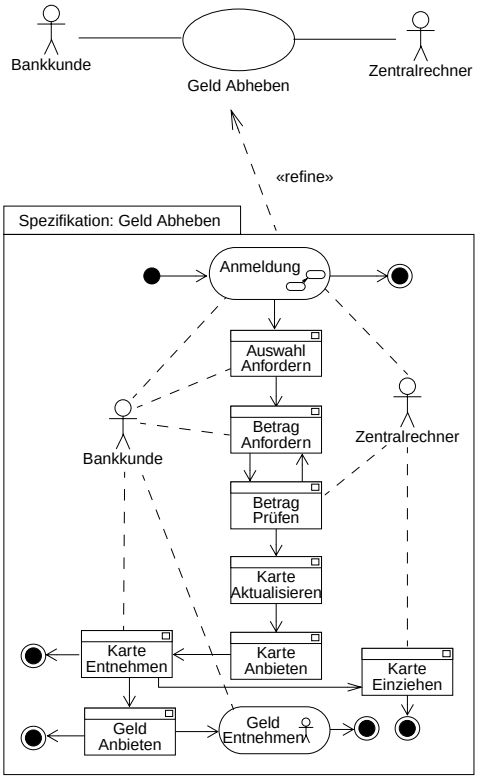


Abbildung 1 Anwendungsfall *Geld Abheben*, Aktivitätsdiagramm

Bei der zunehmenden Komplexität heutiger Anwendungen ist eine effektive Softwareentwicklung ohne adäquate Werkzeugunterstützung unmöglich, so dass die Anwendbarkeit und Akzeptanz (neuer) Methoden zur Softwareentwicklung entscheidend von der Verfügbarkeit entsprechender Werkzeuge abhängen. Der Beitrag schließt dementsprechend mit ersten Erfahrungen bei der Implementation des SCORES-Profils in Rational Rose (SCORESTOOL) und beim Einsatz des Profils.

Bis jetzt haben wir die Methode und das Werkzeug hauptsächlich in eigenen Re-Engineering Projekten erprobt, in denen wir die Anforderungen an einige kleinere bis mittelgroße z.T. kommerzielle Anwendungen mit SCORES spezifiziert und geprüft haben. Die Ergebnisse dieser Aktivitäten führten zu einer wiederholten Überarbeitung bzw. Präzisierung des methodischen Vorgehens, des SCORES-Metamodells und des SCORESTOOLs.

Soweit Auftraggeber und Benutzer von SCORES betroffen sind, erwarten wir weiterhin das gleiche hohe Maß an Zustimmung wie wir es bereits bei unseren Re-Engineeringprojekten erhalten haben. Ausschlaggebend sind die zur Verfügung gestellten benutzerfreundlichen Sichten und die Betonung des Ablaufaspektes: In der Regel waren die Anwender offen für den Ansatz und empfanden die erstellten SCORES-Anforderungsspezifikationen als zweckmäßiger und aussagekräftiger als die bisherigen textuellen Beschreibungen der Anwendungsfälle. Kombinierte Walk-Throughs der Aktivitätsdiagramme (vgl. [2]) mit konkreten (benutzerfreundlich dargestellten) Objektkonstellationen und „Mockups“ der Benutzungsoberfläche (in Abb. 2 rechts zu erkennen) haben insbesondere bei Personen ohne Modellierungserfahrung sehr zum Verständnis der strukturellen Aspekte der Anforderungsspezifikation beigetragen und bei der präzisen Formulierung von Kommentaren und Änderungswünschen geholfen.

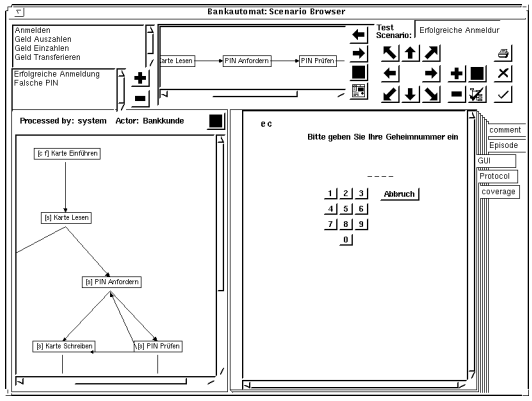


Abbildung 2 ScoresTool SzenarioBrowser

Auch auf Seiten der Analytiker und Tester hatten wir bisher keine Probleme, diese für SCORES zu begeistern. Analytiker begrüßten am meisten die hierarchische, redundanzvermeidende Modellierung von Anwendungsfällen sowie die erstmals anhand der funktionalen Anforderungen überprüfbare „Funktionalität“ des Klassenmodells [3]. Tester sind erstmals in der Lage, auch bei der Qualitätssicherung in der Anforderungsermittlung systematische, mit entsprechenden Endekriterien untermauerte Prüfverfahren anwenden zu können [1]. Alles in allem wurden Werkzeuge wie der SCORESTOOL-Szenario-Browser (Abb. 2) als adäquate Unterstützung der entsprechenden Tätigkeiten angesehen.

[1] M. Winter: *Qualitätssicherung für objektorientierte Software: Anforderungsermittlung und Test gegen die Anforderungsspezifikation*. Verlag dissertation.de, Berlin 2000, ISBN 3-89825-031-8. Zugl. Dissertation, FernUniversität Hagen, 1999

[2] Georg Kösters, Bernd-Uwe Pagel, Thomas de Ridder, Mario Winter: *Animated Requirements Walkthroughs Based on Business Scenarios.* Proc. 5th euroSTAR 97, Edinburgh, Scotland, Nov. 1997 (CD-ROM)

[3] Georg Kösters, Bernd-Uwe Pagel, Mario Winter: *Kombinierte Validierung von Use Cases und Klassenmodellen.* Proc. Modellierung’98, Münster, März 1998, S. 117-121