Kombinierte Validierung von Use Cases und Klassenmodellen

Georg Kösters
ISOWARE GmbH
Hagen
Georg.Koesters@isoware.de

Bernd-Uwe Pagel
Praktische Informatik III
FernUniversität Hagen
Bernd-Uwe.Pagel@FernUni-Hagen.de

Mario Winter
Praktische Informatik III
FernUniversität Hagen
Mario.Winter@FernUni-Hagen.de

Positionspapier zum Workshop „Modellierung 98", 12./13. März 1998, Münster

Abstrakt: Ausgangspunkt dieses Beitrages sind die aus den unterschiedlichen Sichtweisen von Benutzern und Entwicklern resultierenden Probleme bei der objektorientierten Analyse. Aufbauend auf der Kopplung von Use Cases und Klassenmodellen wird eine qualitätszentrierte Methode zur Ermittlung und Validierung von Anforderungen vorgestellt. Der Beitrag endet mit einem Ausblick auf einen unterstützenden Werkzeugprototyp sowie bisher gewonnene Erfahrungen beim Einsatz der Methode.

Keywords: Requirements Engineering, Objektorientierte Analyse, Use Case, Szenario, Validierung, Review, Modellierungsprozeß

1 Einleitung

Zufriedene Benutzer sind der bestimmende Faktor für den Erfolg kommerzieller Software geworden. Der beste Weg, Benutzer zufriedenzustellen, ist ihnen ein Produkt zu liefern, das sie tatsächlich benötigen, d.h. ihre Erwartungen möglichst vollständig zu erfüllen. Diese werden bei der Entwicklung neuer Software als beobachtbare Eigenschaften des Produkts — sogenannte operationale Anforderungen — in einem Pflichtenheft spezifiziert.

Vornehmlich aus Management-Gründen werden viele größere Softwareprojekte in einem eher traditionellen und phasenorientierten als iterativen und inkrementellen Prozeß durchgeführt — auch in Kombination mit objektorientierten Techniken. Dies impliziert, daß die Benutzer hauptsächlich zu Beginn und ganz zum Schluß des Projektes eingebunden sind, also bei der Erstellung des Pflichtenheftes und bei den Abnahmetests.

Da das Auffinden und Korrigieren von fehlenden, falschen oder mehrdeutigen Anforderungen ohne Einbeziehung des Benutzers so gut wie unmöglich ist, werden fehlerhafte Anforderungen bei Projektende im Rahmen der Abnahmetests oder sogar erst im Produktionsbetrieb festgestellt. Korrekturen zu diesem Zeitpunkt sind extrem schwierig und teuer. Es ist daher empfehlenswert, bei der Anforderungsermittlung neben der Verwendung adäquater Techniken und Methoden alle Dokumente vor ihrer Freigabe rigorosen Tests zu unterwerfen. Gerrard behauptet [Gerrard94]:

> *Testing of requirements is potentially the most valuable testing we can do.*

Das „richtige Produkt" kann also nur entwickelt werden, wenn seine zukünftigen Benutzer die Anforderungsdokumente validieren können. Dazu müssen diese informell, intuitiv und auch für „Computerlaien" leicht verständlich sein. Zur „richtigen Produktentwicklung" werden jedoch möglichst formale Anforderungsdokumente benötigt, die oft nur für Spezialisten lesbar sind.

In der objektorientierten Praxis finden sich deshalb oft im Pflichtenheft sowohl Use Cases [Jaco92] als auch Klassenmodelle. Use Cases werden mehr oder weniger strukturiert umgangssprachlich notiert und vom Benutzer „durch Gegenlesen" validiert. Die z.B. in [Winter97] zusammengestellten Review-Techniken sind hierbei allerdings nur bedingt anwendbar, da Quelldokumente, gegen die validiert werden kann, entweder nicht existieren oder aber zu informell sind. Im Gegensatz dazu können Klassenmodelle zusammen mit dynamischen Spezifikationen wie z.B. State Charts oder Interaktionsdiagrammen sehr gut mit diesen Techniken validiert werden. Wünschenswert ist eine Kombination informeller und formaler Anforderungsdokumente, bei deren Validierung Analytiker und Benutzer erfolgreich zusammenarbeiten.

Dieser Beitrag konzentriert sich auf den besonders kritischsen Teil des Überganges von den benutzerorientiert formulierten Anforderungen zu den entwicklerorientierten Modellen; nämlich die Abbildung von Use Cases auf Klassen und Operationen inklusive der Validierung des dadurch entstehenden integrierten Anforderungsdokumentes.

Zu diesem Zweck wird im nächsten Abschnitt die Kopplung von Use Cases und Klassenmodellen präzisiert [KöPaWi97]. Zur Validierung der dynamischen Systemaspekte werden dann in Abschnitt 3 „Test-"Szenarien eingeführt. Aufbauend auf diesen Konzepten wird im vierten Abschnitt im Rahmen der in [KöPaRiWi97] vorgestellten Methode zur Anforderungsermittlung der Einsatz verschiedener Formen von Reviews mit und ohne Benutzerbeteiligung umrissen. Im vorletzten Abschnitt skizzieren wir einen unterstützenden Werkzeugprototyp. Der Beitrag endet mit einem Ausblick auf bisher gewonnenen Erfahrungen beim Einsatz der Methode.

2 Use Cases und Klassenmodelle

Use Cases (zu dt. „Nutzungsfälle“ oder „Anwendungsfälle“) beschreiben die operationalen Anforderungen an ein System innerhalb eines Anwendungsbereiches. Zur Abgrenzung des Systems von seiner Umgebung dienen *Aktoren*, die jeweils eine bestimmte Rolle repräsentieren, die ein Benutzer oder externes System spielt (z.B. „Kunde“, „Operator“ oder „angeschlossenes Abrechnungssystem“). Ein *Use Case* beschreibt auf relativ abstrakter Ebene eine bestimmte, abgrenzbare Aufgabe bzw. Verantwortlichkeit eines Aktors innerhalb des betrachteten Anwendungsbereiches. Wir charakterisieren jeden Use Case durch einen im Anwendungskontext eindeutigen Namen, eine informelle Beschreibung der Aufgabe und den sogenannten *primären Aktor*, der die Aufgabe unter Benutzung des Systems bearbeitet.

Im Normalfall können wir die Bearbeitung einer Aufgabe (funktional) in einzelne Teilschritte zerlegen, die der primären Aktor ggf. unter Zuhilfenahme sog. *sekundärer Aktoren* oder des Systems auszuführt. Ein Schritt im Use Case ist ein

1. *Interaktionsschritt,* dessen (Teil-) Aufgabe direkt vom System unterstützt wird. Ein Interaktionsschritt beginnt mit einem auslösenden Ereignis und endet, wenn das System die (Teil-) Aufgabe erledigt oder als nicht bearbeitbar erkannt hat.
2. *Kontextschritt*, der eine durch einen Menschen oder ein anderes System ausgeführte Tätigkeit, also Teile des Geschäftsprozesses beschreibt.
3. *Referenzschritt*, der dem „Aufruf“ eines weiteren Use Cases entspricht und eine hierarchische, redundanzvermeidende Strukturierung von Use Cases ermöglicht.

Jeder Schritt ist durch einen im Kontext des Use Cases eindeutigen Namen, eine informelle Beschreibung seiner Aufgabe, ggf. benötigte sekundäre Aktoren und eine Menge möglicher Folgeschritte gekennzeichnet.

Wir betrachten nun die sich aufgrund der Angabe möglicher Folgeschritte ergebenden Abhängigkeiten zwischen Use Case Schritten und fordern, daß jeder Use Case mindestens einen Schritt beinhalten muß, von dem aus alle anderen Schritte über die Folgeschrittrelation erreichbar sind. Ein solcher Schritt heißt *Wurzelschritt* des Use Case. Wir markieren genau einen Wurzelschritt als *Startschritt* des Use Case und einen oder mehrere weitere Schritte als sogenannte *Abschlußschritte*. Interpretieren wir diese Abhängigkeiten als gerichteten Wurzelgraph, so spiegeln die mit dem Startschritt beginnenden und in einem Abschlußschritt endenden Pfade durch den Graphen mögliche Abläufe oder *Szenarien* wider, die bei der Bearbeitung der dem Use Case zugeordneten Aufgabe auftreten können.

Wir verknüpfen nun Use Cases mit dem korrespondierenden Klassenmodell und identifizieren Interaktionsschritte als Verbindungspunkte.

1. Die Durchführung eines Interaktionsschrittes entspricht der Ausführung von Operationen, die im Klassenmodell spezifiziert sind.
2. Die erfolgreiche Ausführung dieser Operationen ist nur auf bestimmten, durch die Vor- und Nachbedingungen und die Aufgaben der Interaktionsschritte eingeschränkten Objektkonstellationen möglich, die zum Klassenmodell konsistent sein müssen.

Dementsprechend bestimmen wir für Use Cases und Schritte aus der (Teil-) Aufgabenbeschreibung zwei als Vor- und Nachbereich bezeichnete Mengen von Klassen. Der *Vorbereich* enthält alle Klassen, von denen Instanzen von der (Teil-) Aufgabe betroffen sind, d.h. die erzeugt, gelöscht oder aber in ihrem Zustand geändert werden. Der *Nachbereich* enthält alle Klassen, von denen Instanzen zur Auswahl und Ausführung eines Folgeschrittes benötigt werden. Zusätzlich bilden wir jeden Interaktionsschritt auf eine bestimmte Operation im Klassenmodell ab. Diese Operation bzw. die Klasse, in der sie definiert ist, wird *Wurzeloperation* bzw. *Wurzelklasse* des Interaktionsschrittes genannt.

3 Test-Szenarien

Die sich bis jetzt ergebende Kombination aus Use Cases und Klassenmodell ermöglicht eine eher statische Sicht auf das projektierte System. Ein vielversprechender Ansatz zur Validierung der dynamischen Systemaspekte besteht in dem Durchspielen relevanter Szenarien: Unter Berücksichtigung eines bestimmten Systemzustandes wird ein Use Case simuliert und der sich ergebende Ablauf mit dem erwarteten Ablauf verglichen. Szenarien werden also zu Testfällen der Anforderungsspezifikation. Jedes *Test-Szenario* besteht aus einer Folge von Szenarioschritten, die jeweils einem Schritt im Use Case zugeordnet sind. Der Vor- und Nachbereich der Interaktionsschritte wird für die Szenarienschritte übernommen und um Kardinalitätsangaben für die erwartete Menge von Instanzen erweitert.

Darüberhinaus können wir für jeden Szenarioschritt, der einem Interaktionsschritt im Use Case entspricht, eine sogenannte *Episode* angeben, die einen (Aufruf-) Baum von Operationen im Klassenmodell modelliert. Episoden spiegeln die Ausführung der Wurzeloperation im angenommenen Systemzustand wider. Hierbei entscheidet der Entwickler, ob er die Episode bis hin zu elementaren Attributzugriffen oder aber nur bis hin zu wohlverstandenen Operationen verfolgt bzw. protokolliert. Zu beachten ist, daß dieselbe Wurzeloperation bei verschiedenen Systemzuständen zu unterschiedlichen Episoden führen kann.

4 Qualitätszentrierte Anforderungsermittlung

Aufbauend auf der Kombination von Use Cases und Klassenmodell schlagen wir vor, bei der Anforderungsermittlung vier Aktivitätsebenen zu unterscheiden, die in einem iterativen Prozeß mehrfach überarbeitet werden. Für jede Ebene geben wir die betrachteten Teile des kombinierten Anforderungsdokumentes sowie eine Reviewtechnik zusammen mit passenden Abbruch- bzw. Überdeckungskriterien an.

Funktionalität: Analytiker und Benutzer erstellen gemeinsam grundlegende Szenarien und sammeln Begriffe und Definitionen. Die (textuell spezifizierten) Szenarien werden dann zu Use Case-Graphen zusammengefaßt. Parallel dazu können erste, sozusagen „ER-diagrammartige“ Skizzen des Klassenmodelles angefertigt werden.

Bei der Validierung der Use Cases ist vor allem auf die komplexe Schnittstellenproblematik zu achten, da Geschäftsprozesse angerissen werden und neben den Analytikern verschiedene Benutzergruppen als primäre und sekundäre Aktoren betroffen und an der Validierung beteiligt sind. Erfahrungsgemäß führt in solchen Situationen eine Reihe von Walk-Throughs zu den besten Ergebnissen: anhand eines konkreten Szenarios wird der Use Case-Graph durchwandert und auf korrekt widergespiegelte Funktionalität und richtige Resultate überprüft. Falsche, fehlende und mehrdeutige Schritte werden zusammen mit dem Pfad und Kommentaren protokolliert, korrigiert und erneut validiert. Sofern nicht schon geschehen, wird das Szenario festgehalten. Bezüglich der Frage, wann ein Use Case ausreichend „getestet“ ist, machen wir Anleihen bei den strukturellen Testverfahren und wenden Kriterien wie z.B. die Knoten-, Zweig- und „Boundary-Interior-“ Überdeckung auf den Use Case-Graphen an. Alle Zweige sollten mindestens einmal durchlaufen werden.

Struktur: Sind die betrachteten Use Cases stabil, konzentrieren sich die Analytiker auf die strukturellen Teile des Klassenmodells, d.h. fügen fehlende Klassen hinzu, vervollständigen Vererbungshierarchien und Objektbeziehungen, ergänzen Attribute und weisen ggf. einigen Interaktionsschritten Wurzeloperationen zu. Danach werden Vor- und Nachbereiche aus den informellen Beschreibungen der Use Cases und Interaktionsschritte abgeleitet. Die Analytiker spielen die bekannten Szenarien erneut durch, verfeinern sie und fügen weitere hinzu. Diese Aktivitäten werden iteriert, bis das Klassenmodell stabil ist.

Die Analytiker können nun aus dem Klassenmodell eine Reihe von Objektmodellen erzeugen, die in Reviews gemeinsam mit den Benutzern auf ihre Übereinstimmung mit realen Konstellationen in der Problemwelt hin validiert werden. Hauptaugenmerk sind fehlende, unvollständige oder überflüssige Klassen und Beziehungen, Mehrdeutigkeiten und die Adäquatheit der vorgeschlagenen Strukturen hinsichtlich der funktionalen Anforderungen. Jedes instanziierbare Element des Klassenmodelles sollte wenigstens einmal in den Reviews vorkommen. Bei Beziehungen sind in Abhängigkeit von den Kardinalitätsbeschränkungen Objektmodelle ggf. mit der minimal und maximal möglichen und mit „typischen“ Anzahlen beteiligter Objekte zu erzeugen.

„Externes“ Verhalten: Sofern noch nicht geschehen, werden die Interaktionsschritte der Use Cases durch die Angabe von Vor- und Nachbereichen und Wurzeloperationen an das Klassenmodell gekoppelt sowie die in den Reviews der Struktur gewonnenen Informationen eingearbeitet. Hierzu werden die Verantwortlichkeiten der Interaktionsschritte mit vorhandenen Klassen und Operationen verglichen und passende Operationen zugeteilt bzw. im Klassenmodell ergänzt. Besondere Berücksichtigung finden Szenarien, die Sonderfälle behandeln.

Die mit dem Klassenmodell gekoppelten Use Cases werden mit kombinierten Walk-Throughs unter Beteiligung von Analytikern und Benutzern validiert, wobei die Vor- und Nachbereiche die Fokussierung auf relevante Teile des Klassenmodells ermöglichen. Im Mittelpunkt stehen die nun auch im Klassenmodell verfolgbare Funktionalität und die Balancierung der Modelle — z.B., ob Operationen und Attribute jeweils sinnvolle, kohärente Abstraktionen des Problembereiches verkapseln und wirklich „objektorientiert“, also nicht etwa in zu enger Anlehnung an komplette Use Cases funktional zerlegt sind. In Einzelfällen ist es sinnvoll, den Ablauf eines Szenarios anhand von Objektmodellen zu verdeutlichen. Auf Seiten des Klassenmodells sollte jedes Element mindestens in einem Walk-Through erzeugt, modifiziert und ggf. gelöscht worden sein. Für jeden Use Case sind das Hauptszenario und mindestens ein Ausnahmeszenario zu durchlaufen.

„Internes“ Verhalten: Operationen und Episoden werden weiter ausspezifiziert. Es folgt eine Reihe von Inspektionen bezüglich der Vollständigkeit und Konsistenz des kombinierten Anforderungsdokumentes nach funktionalen, strukturellen und allgemeinen Kriterien wie z.B.

- Existieren Zyklen aufgrund von Referenzschritten in den Use Cases?
- Sind alle möglichen Resultate jedes Referenzschrittes hinreichend in den referenzierenden Use Cases beachtet?
- Ist jeder Interaktionsschritt im Klassenmodell verankert?
- Existiert zu jeder Operation mindestens eine Episode?
- Sind die Spezifikation jeder Operation, d.h. ihre Signatur und die Vor- und Nachbedingungen, sowie die Klasseninvarianten in den Episoden beachtet?

Da die Inspektionen kein tieferes Modellverständnis erfordern, können sie größtenteils von „Nachwuchs-Analytikern“ durchgeführt, automatisiert oder direkt konstruktiv berücksichtigt werden.

In abschließenden Walk-Throughs kritischer Szenarien evaluieren die Analytiker das kombinierte Modell vertikal über alle Abstraktionsstufen vom Geschäftsprozeß bis hin zur einzelnen Operation.

5 Werkzeugunterstützung

Die Wartung der vorgestellten integrierten Anforderungsdokumente ist ohne den Einsatz von Werkzeugen für auch nur einigermaßen realitätsnahe Probleme schlichtweg undurchführbar. Zur Validierung der Methode haben wir daher ein vorhandenes Werkzeug zur objektorientierten Analyse [KöPaSi96] um Komponenten zur Eingabe und Editierung von Use Cases, Test-Szenarien und Episoden erweitert. Viele Konsistenzregeln werden bereits bei der Konstruktion der entsprechenden Anforderungsdokumente überprüft. Zudem bietet das Werkzeug sowohl integrierte Modellsichten zu Unterstützung von Reviews und Walk-Throughs (Abbildung 1) als auch verschiedene komfortable Browser und Kreuzreferenzierungsmöglichkeiten für einzelne Modellsichten an.

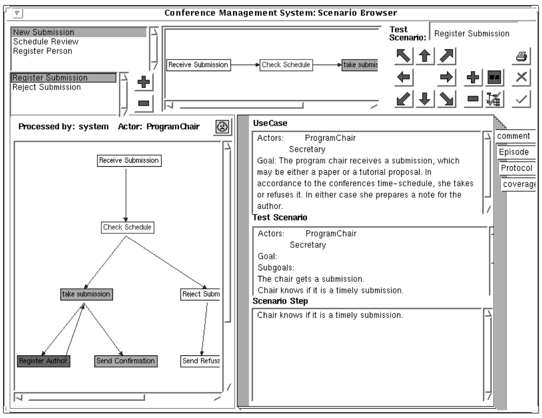

Abbildung 1 Walkthrough Browser

6 Verwandte Arbeiten

Use Cases als rein funktionale Sicht auf ein System wurden von Jacobson [Jaco92] eingeführt. Die unklare bzw. fehlende Semantik insbesondere der von Jacobson vorgeschlagenen extends- und uses-Relationen zwischen Use Cases führte allerdings zu nicht unerheblicher Konfusion sowohl bei Benutzern als auch Analytikern. Cockburn [Cockburn97] berichtet aus seiner Beratungspraxis von mehr als 18 verschiedenen Interpretationen bzw. Arten von Use Case Modellierungen, kurz *„...people are using whatever they think of as a use case to good effect“*[Cockburn97], und schlägt ein hierarchisches Use Case Modell strukturiert nach den mit dem Systemeinsatz verfolgten Zielen vor. Regnell et al. entwickeln ein Use Case Modell mit drei Abstraktionsebenen [ReAnBe96]. Die Umgebungsebene (environmental level) entspricht Jacobsons Use Cases Modell, eine Zwischenebene (intermediate level) definiert zusammenhängende Teilabläufe eines Use Case, genannt Episoden, deren Semantik über Vor- und Nachbedingungen spezifiziert wird. Episoden sind funktional hierarchisch strukturiert und werden zeitlich über Konstrukte verknüpft, die an Message Sequence Charts (MSC, [ITU93]) angelehnt sind. Auf der Ereignisebene (event level) wird der Nachrichtenaustausch eines Aktors mit dem System (betrachtet als Black Box) in zeitlicher Reihenfolge mittels MSC modelliert.

Während die rein textuelle Formulierung der Use Cases bei Cockburn die Validierung erschwert, ist die von Regnell et al. schon auf relativ hoher Abstraktionsebene verwendete MSC-Notation zu sehr auf Anwendungsgebiete der Telekommunikation beschränkt. Beide Arbeiten vernachlässigen die strukturellen Anteile der Anforderungsdokumente.

7 Erfahrungen

Wir haben die vorgestellten Techniken bei der Re-Modellierung einiger existierender Anwendungen mit gutem Erfolg eingesetzt. Die Rolle der Benutzer wurde sowohl von Mitarbeitern als auch von Personen ohne Modellierungserfahrung übernommen.

Großen Anklang fanden die hierarchische Modellierung von Use Cases sowie die Validierung von Klassenmodellen mit dedizierten, anwendungsbezogenen Objektmodellen. Kombinierte Walk-Throughs der Use Cases und Klassenmodelle haben insbesondere bei Personen ohne Modellierungserfahrung sehr zum Modellverständnis beigetragen und bei der präzisen Formulierung von Kommentaren und Änderungswünschen geholfen. In den abschließenden Walk-Throughs sollten aufgrund der ständig variierenden Abstraktionsstufen keine Personen ohne Modellierungserfahrung beteiligt werden, da diese erfahrungsgemäß Probleme bei der Fokussierung auf die relevanten Details der jeweiligen Abstraktionsstufe haben und den Erfolg der Walk-Throughs in Frage stellen. Auf Seiten der Analytiker wurde die überprüfbare Funktionalität des Analysemodells am meisten begrüßt. Geplant sind weitere Validierungen der Methode auch im industriellen Umfeld.

Literatur

[Cockburn97] Alistair Cockburn *Goals and Use Cases* JOOP Vol. 8 No. 6/7 Sept./Nov. 1997

[Gerrard94] Paul Gerrard *Testing Requirements* Proc. 2nd euroSTAR, 10-13 Okt., Brüssel, Belgien, 1994 http://www.ftech.net/~evolutif/TestReqs/TESTREQS.html

[ITU93] ITU Telecommunications Standard Sector *Z.120 Message Sequence Charts* ITU, 1993

[Jaco92] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, Gunnar Övergaard *Object-Oriented Software Engineering* Addison-Wesley/acm Press, Reading, Massachusetts, 1992

[KöPaSi96] Georg Kösters, Bernd-Uwe Pagel, Hans-Werner Six *GeoOOA: Object-Oriented Analysis for Geographic Information Systems* Proc. of the 2nd International Conference on Requirements Engineering, Colorado Springs, 1996

[KöPaWi97] Georg Kösters, Bernd-Uwe Pagel, Mario Winter *Coupling Use Cases and Class Models* In: Andy Evans, Kevin Lano (Eds.) Proc. of BCS FACS/ EROS Workshop on Making Object-Oriented Methods more Rigorous, 24. Juni, London, GB, 1997

[KöPaRiWi97] Georg Kösters, Bernd-Uwe Pagel, Thomas de Ridder, Mario Winter *Animated Requirements Walkthroughs Based on Business Scenarios* Erscheint in: Proc. 5th euroSTAR 97, 24.-28. Nov., Edinburgh, GB, 1997

[ReAnBe96] Björn Regnell, Michael Andersson, Johan Bergstrand *A Hierarchical Use Case Model with Graphical Representation* Proc. ECBS'96, IEEE Intnl. Workshop on Engineering of Computer-Based Systems, IEEE Press, März 1996

[Winter97] Mario Winter *Reviews in der objekt-orientierten Softwareentwicklung* TAV 10 6/7.03.1997 Böblingen in: GI Softwaretechnik-Trends Band 17 Heft 2, Mai 1997