

Diplomarbeit

Modellierung und Architektur mit der UML - Ein Musterbeispiel

zur Erlangung des akademischen Grades
Diplom-Informatiker (FH)
vorgelegt dem

Fachbereich Mathematik, Naturwissenschaften und Informatik
der Fachhochschule Gießen-Friedberg
von

Rüdiger Anlauf

Gießen, August 2002

Referent: Prof. Dr. Burkhardt Renz
Korreferent: Prof. Dr. Manfred Scheer

Inhaltsverzeichnis

I. Einleitung und Grundlagen	7
1. Über diese Arbeit	9
2. Überblick über den ICONIX-Prozess	13
2.1. Grundgedanken	13
2.2. Phasen und Elemente des ICONIX-Prozesses	14
2.2.1. Anforderungsanalyse	14
2.2.2. Analyse und vorläufiges Design	17
2.2.3. Design	17
2.2.4. Implementierung	18
2.3. Meilensteine	18
II. Ein Anwendungsbeispiel	21
3. Aufgabenstellung	23
4. Informationsmodell	25
4.1. Allgemeines	25
4.2. Auffinden des Informationsmodells	25
4.2.1. Allgemeines zum Informationsmodell	25
4.2.2. Identifizieren der Klassen	27
4.2.3. Identifizieren der Beziehungen	28
4.2.4. Identifizieren weiterer Klassen	30
4.2.5. Erstellen eines Glossars	31
4.2.6. Anmerkungen zum Informationsmodell	31
4.3. Vergleich mit dem Original	34
5. Anwendungsfälle	37
5.1. Erste Anwendungsfälle	38
5.1.1. Katalog ansehen	39

Inhaltsverzeichnis

5.1.2.	Buch suchen	39
5.1.3.	Warenkorb verwalten	39
5.1.4.	Buch bestellen	39
5.1.5.	Bestellung verfolgen	39
5.2.	GUI Prototyp und Benutzerhandbuch	39
5.2.1.	Prototyp	39
5.2.2.	Benutzerhandbuch	41
5.3.	Identifizieren der Anwendungsfälle	41
5.4.	Beziehungen zwischen den Anwendungsfällen	41
5.5.	Zuordnen der funktionalen Anforderungen	46
5.6.	Beschreibung der Anwendungsfälle	46
5.6.1.	Buchbestellung	47
5.6.2.	Buchrezension und Bewertung	51
5.6.3.	Produktübersichten	54
5.7.	Anwendungsfallpakete	55
5.8.	Vergleich mit dem Rosenberg-Modell	58
6.	Stabilitätsanalyse	61
6.1.	Buchkatalog ansehen	64
6.2.	Warenkorb editieren	67
6.3.	Buch bestellen	70
6.4.	Lieferdaten bestimmen	73
6.5.	Vergleich mit der Lösung von Rosenberg und Scott	73
7.	Überlegungen zur Softwarearchitektur	77
7.1.	Allgemeine Überlegungen	77
7.2.	Spezielle Überlegungen	77
7.3.	J2EE und MVC	80
7.4.	Architekturfragment des Internet-Shops	81
8.	Interaktionsmodellierung	85
8.1.	Warenkorb editieren	86
8.1.1.	Entwicklung des Detailentwurfs aus dem Stabilitätsdiagramm	86
8.1.2.	Ablauf des Anwendungsfalls	88
8.2.	Verwendung von Entwurfsmustern	90
8.3.	Buchkatalog ansehen	93
8.3.1.	Entwicklung des Detailentwurfs aus dem Stabilitätsdiagramm	93
8.3.2.	Ablauf des Anwendungsfalls	95
8.4.	Vergleich mit der Lösung von Rosenberg und Scott	97

9. Überarbeitetes Klassendiagramm	101
9.1. Kategorielliste und Buchkatalog	101
9.2. Warenkorb	103
9.3. Vergleich mit der Lösung von Rosenberg und Scott	105
 III. Schluss	 109
10. Diskussion der Ergebnisse	111
11. Bewertung des ICONIX Prozesses	115
A. Prototyp und Benutzerhandbuch	121
A.1. Startseite (Azamon)	121
A.2. Katalog	122
A.3. Büchersuche	126
A.4. Warenkorb	127
A.5. Bestellen	127
A.6. Historie	133
A.7. Kontakt	133
B. Anwendungsfälle vor der Stabilitätsanalyse	135
B.1. Paket Buchbestellung	135
B.1.1. Artikel in den Warenkorb legen	135
B.1.2. Buch bestellen	135
B.1.3. Bestellung ansehen	136
B.1.4. Bestellung stornieren	136
B.1.5. Fehlerhafte Dateneingabe behandeln	136
B.1.6. Lieferdaten bestimmen	137
B.1.7. Warenkorb editieren	137
B.1.8. Zahlungsdaten bestimmen	138
B.2. Paket Buchrezension	139
B.2.1. Buch bewerten	139
B.2.2. Buch rezensieren	139
B.2.3. Rezension vorbereiten	139
B.2.4. Rezension überarbeiten	139
B.3. Paket Systemanmeldung und Kontaktinformationen	140
B.3.1. Anmelden	140
B.3.2. Kontaktinformationen ansehen	140
B.3.3. Neuen Account eröffnen	140
B.4. Paket Buchkatalog	141

Inhaltsverzeichnis

B.4.1. Buchkatalog ansehen	141
B.4.2. Buchkatalog durchsuchen	141
B.4.3. Kurzinfo anzeigen	142
B.4.4. Neuheiten und Empfehlungen ansehen	142
B.4.5. Produktinformationen ansehen	142

C. Eidesstattliche Erklärung	145
-------------------------------------	------------

Teil I.

Einleitung und Grundlagen

1. Über diese Arbeit

Vorgeschichte Diese Arbeit basiert auf dem Buch von Doug Rosenberg und Kendall Scott: „Applying Use Case Driven Object Modeling With UML“. Dieses Buch wurde in der Fachpresse (Dr. Dobbs Journal) als ein Werk gelobt, das Entwurfsentscheidungen transparent macht und an dem man gut nachvollziehen kann, wie ein Softwaresystem nach und nach zustande kommt. Als solches soll das Buch in der Literatur keine Entsprechung finden.

Die Autoren selbst verstehen das Buch als ein Arbeitsbuch, das es dem Leser ermöglichen soll, das Vorgehensmodell „ICONIX Process“ am praktischen Beispiel eines Internet-Buchshops nachzuvollziehen. Dabei wird in jedem Kapitel eine Phase des Prozesses kurz und prägnant vorgestellt und anschließend werden die zehn häufigsten Fehler beschrieben, die von Lernenden in der jeweiligen Phase gerne gemacht werden. Diese Fehler demonstrieren die Autoren in den verschiedenen Entwicklungsphasen jeweils an Beispielen aus der Entwicklung des Buchshops und der Leser kann - basierend auf der fehlerhaften Lösung - darüber nachdenken, wie er es besser gemacht hätte. Anschließend geben die Autoren einen eigenen Lösungsvorschlag.

Eine erster Versuch, die Beispiele des Buches in einer Lehrveranstaltung zu verwenden, hat gezeigt, dass die in das Buch gesetzten Erwartungen nur zum Teil erfüllt werden. Es scheint einige Schwächen aufzuweisen, die den Einsatz des präsentierten Beispiels in Lehrveranstaltungen sehr schwierig machen. Insbesondere beschreiben die Autoren zwar für jeden Analyseschritt die am häufigsten gemachten Fehler, begründen jedoch nicht, aufgrund welcher Erwägungen man zu den von ihnen dargelegten Teilen des Modells gelangt. An dieser Stelle setzt diese Arbeit an.

Zielsetzung Das Thema dieser Arbeit ist es, am von Rosenberg und Scott verwendeten Beispiel den ICONIX-Prozess durchzuführen und zu einem nachvollziehbaren Teilmodell zu gelangen. Dabei wird in jedem Kapitel geschildert, welche Schritte der ICONIX-Prozess vorsieht, um die jeweilige Phase abzuschließen. Der Autor folgt dem vorgeschlagenen Weg und beschreibt für jede Phase des Prozesses genau, wie man mit der dargestellten Methode zu den jeweiligen Entwurfsentscheidungen findet. Soweit auf diesem Weg Probleme auftreten - etwa weil mit dem ICONIX-Prozess bestimmte Probleme nicht befriedigend gelöst werden können - wird nach Möglichkeiten gesucht, diese Lücken zu füllen. Dabei legt der Autor besonderen Wert darauf, darzustellen, wie man von einem Schritt des Prozesses zum nächsten gelangt. Abschließend wird für jede Phase ein Vergleich mit der von Rosenberg und Scott gefundenen Lösung gezogen.

Im Detail erörtert diese Arbeit folgende Fragestellungen:

- Kapitel 4 beschreibt, wie man mit der auch als „Methode von Abbot“ bekannten grammatikalischen Analyse der Problembeschreibung zu einem statischen Modell gelangt.
- Kapitel 5 beschreibt, wie man aus einer anfänglichen Sammlung von Geschäftsvorfällen einen GUI-Prototypen und ein Benutzerhandbuch gewinnt, und wie man aus dem Benutzerhandbuch Anwendungsfälle extrahiert.

1. Über diese Arbeit

- Kapitel 6 beschreibt, wie man mit Hilfe der Stabilitätsanalyse die Anwendungsfälle auf Vollständigkeit und Durchführbarkeit überprüft und mit dem Informationsmodell abgleicht, und wie man die Objekte und Attribute findet, die noch fehlen, um diese Anwendungsfälle zu implementieren. Außerdem legt es dar, durch welche Überlegungen man in der Stabilitätsanalyse zur Architekturfindung motiviert werden kann.
- Kapitel 7 beschreibt, aufgrund welcher Überlegungen man zu einer problemgerechten Softwarearchitektur gelangt und stellt einen Teil dieser Architektur vor.
- Kapitel 8 beschreibt, wie man die in Kapitel 6 gefundenen Control-Objekte in Operationen überführt, die man auf die verschiedenen Objekte verteilt, und wie man Entwurfsmuster einsetzen kann, um zu einem besseren Entwurf zu gelangen.
- Kapitel 9 zeigt, wie sich das Informationsmodell durch die vorangehenden Kapitel zu einem vollständigen Klassenmodell entwickelt.
- Der Schlussteil schließlich enthält eine abschließende Bewertung des ICONIX-Prozesses hinsichtlich seiner Praxistauglichkeit, fasst noch einmal die Stärken und Schwächen des Prozesses zusammen und beinhaltet eine Erörterung der hier gefundenen Ergebnisse.

Abgrenzung Um den Umfang der Arbeit von vornherein beherrschbar zu halten, ist es wichtig, die Aufgabenstellung so abzugrenzen, dass der Rahmen einer Diplomarbeit zeitlich und inhaltlich eingehalten wird. Daher werden bei dieser Arbeit folgende Rahmenbedingungen gesetzt:

- Es wird nur der Teil des Systems modelliert, der die direkte Beziehung des Kunden mit dem Buchshop beschreibt. Die internen Vorgänge im Shop - wie etwa die Rechnungsstellung oder die Abwicklung der Lieferungen und die Lagerhaltung - bleiben unberücksichtigt, genauso wie die Beziehungen des Shops zu Dritten - wie etwa das Beschaffen der Bücher von Verlagen und Großhändlern oder die Zahlungsabwicklung über Banken und Kreditkartenfirmen.
- Die Arbeit legt dar, wie man mit den im ICONIX-Prozess geschilderten Methoden arbeitet. Es ist nicht das Ziel, ein implementierbares System zu entwerfen, sondern nachvollziehbar darzustellen, wie man mit den beschriebenen Methoden (und ein wenig darüber hinaus) zu bestimmten Entwurfsentscheidungen gelangt. Die Arbeit enthält eine vollständige Analyse der Anwendungsfälle und eine Teilarchitektur. Auch werden bestimmte Sequenzdiagramme vorgestellt. Ein vollständiges, programmierbares System entwickeln zu wollen, würde aber - selbst mit der genannten inhaltlichen Abgrenzung - den Rahmen dieser Arbeit komplett sprengen.

Vorbemerkung Bevor der geneigte Leser in die Lektüre dieses Werkes einsteigt, seien noch zwei Hinweise erlaubt:

In dieser Arbeit kommen an mehreren Stellen wörtliche Zitate vor. Für manche der zitierten Bücher gibt es keine deutschen Übersetzungen. Dies gilt vor allem für die beiden Werke von Rosenberg und Scott und für das Buch von Singh u.a. „Designing Enterprise Applications with the J2EE Platform“. Für die Bequemlichkeit des Lesers hat sich der Autor dieser Arbeit erlaubt, die entsprechenden Passagen selbst ins Deutsche zu übersetzen.

Außerdem enthält diese Arbeit für jede Prozessphase einen Vergleich mit der Lösung von Rosenberg und Scott. Die Beispiele dieser Autoren entstammen [TFWM]. Es handelt sich dabei um eine elektronische Form (als Rational Rose Modell) der Beispiele aus [RoSc01], die auf der Website der Firma

ICONIX zum Download erhältlich ist. Die Verwendung dieser elektronischen Vorlage anstelle von eingescannten Grafiken aus dem Buch erhöht die optische Qualität der Darstellung. Das Layout kann aber geringfügig von dem der Beispiele im Buch abweichen. Inhaltliche Unterschiede sind jedoch - außer in Kapitel 9 nicht vorhanden. Dort wurde zur besseren Vergleichbarkeit beider Modelle der Teil des Rosenberg-Modells entfernt, der sich mit den Shop-Interneta befasst. Darauf wird aber an der entsprechenden Stelle noch einmal verwiesen.

1. Über diese Arbeit

2. Überblick über den ICONIX-Prozess¹

2.1. Grundgedanken

Der ICONIX-Prozess positioniert sich irgendwo zwischen dem sehr umfangreichen Rational Unified Process (RUP) und dem sehr schlanken Extreme Programming (XP). Ähnlich wie bei XP wird auf einen *schlanken Ansatz* Wert gelegt, bei dem der Benutzer des Prozesses sich vorwiegend auf seine eigentlichen Aufgaben konzentrieren kann, anstatt seine Energien in die Erzeugung von Artefakten² zu investieren, deren Nutzen fragwürdig erscheinen mag. Jedoch wird - anders als bei XP - Wert auf ein gutes Design gelegt, bevor man in die Implementierungsphase einsteigt. Auch erzwingt der ICONIX-Prozess keine Reihenfolge der Abarbeitung. Vielmehr wird Wert auf Flexibilität gelegt. Manches Projekt ist schon gescheitert, weil der zugrunde liegende Prozess zu restriktiv und bevormundend war und ein bestimmtes Vorgehen an Stellen nicht erlaubte, an denen es opportun gewesen wäre.

Der ICONIX-Prozess baut auf der Arbeit von Jacobson, Rumbaugh und Booch (ab 1990) auf, den Erfindern der UML. Rosenberg und Scott haben dabei eine minimale, aber hinreichende Untermenge der UML (und der Modellierung im Allgemeinen) identifiziert, die notwendig ist, um ein Softwareprojekt erfolgreich durchzuführen. Bei Bedarf können jedoch auch andere Elemente der UML hinzugenommen werden. Es handelt sich also um einen minimalistischen, zielgerichteten³ Ansatz, der das Gebiet zwischen Anwendungsfällen und Code behandelt: Wie kommt man von der anfänglichen Problembeschreibung zu einem guten Design? Bei der Beantwortung dieser Frage bedient man sich dreier Perspektiven, die parallel benutzt werden (nach [RoSc01, S. 12]):

- outside-in:
Man arbeitet sich von den Anforderungen der Benutzer (Sicht von außen auf das System) zur inneren Funktionsweise und zum technischen Aufbau des Systems vor.
- inside-out:
Man arbeitet sich von den Hauptabstraktionen des Problembereichs nach außen zu den technisch motivierten Klassen und Details vor.
- top-down:
Man beginnt mit einer groben Vorstellung vom System in Form der ungefähren Anforderungen und gelangt am Ende zu einem detaillierten Design.

¹Die hier geschilderte Beschreibung des Prozesses ist in weiten Teilen eine sinngemäße Wiedergabe des ersten Kapitels von [RoSc99].

²Der Begriff „Artefakt“ stammt aus dem Sprachgebrauch des Rational Unified Process. Man versteht darunter die Ergebnisse von Arbeitsschritten, die in greifbarer Form vorliegen, wie etwa Modelle, Teile von Modellen, Dokumente oder die Software selbst.

³engl. streamlined

2. Überblick über den ICONIX-Prozess

Außerdem ist der Prozess *anwendungsfallgetrieben*. Dabei wird nach den von Jacobson begründeten Konzepten auf konkrete, leicht verständliche Anwendungsfälle Wert gelegt, die man gut für die Entwicklung des Systems nutzen kann. Darin unterscheidet sich der ICONIX-Prozess von anderen Prozessen, bei denen die Anwendungsfälle eher als mehr oder minder abstraktes Mittel betrachtet werden, die Anforderungen des Kunden zu eruieren. Einer der Grundgedanken des Begriffs „anwendungsfallgetrieben“ ist es, in der schnellstmöglichen Zeit von den Anwendungsfällen zum Code zu gelangen.

Ein weiteres Merkmal des ICONIX-Prozesses ist, dass er *iterativ* und *inkrementell* vorgeht. Dies bezieht sich einerseits auf die Anwendungsfälle und andererseits auf das Klassenmodell des zu bauenden Systems. Beide werden im Verlauf des Prozesses mehrfach überarbeitet und vervollständigt.

Schließlich bietet der Prozess ein hohes Maß an *Rückverfolgbarkeit*. Über alle Phasen des Prozesses werden die anfänglich definierten Anforderungen im Auge behalten, meist in der Form der Anwendungsfälle. Diese sind sogar noch Teil der Sequenzdiagramme bei der Interaktionsmodellierung.

2.2. Phasen und Elemente des ICONIX-Prozesses

Wie jedes Vorgehensmodell ist der ICONIX-Prozess in Phasen eingeteilt, die nacheinander durchlaufen werden und aus denen schließlich das komplette, ausführbare System hervorgeht. Konkret unterscheidet man vier verschiedene Prozessphasen:

1. Anforderungsanalyse,
2. Analyse und vorläufiges Design,
3. Design und
4. Implementierung.

In jeder Phase werden bestimmte Anstrengungen unternommen, die man als die Elemente des Prozesses bezeichnen kann. Die Elemente lassen sich nach bestimmten inhaltlichen Zusammenhängen und Abläufen ordnen, doch sind sie nicht immer deckungsgleich mit den Phasen des Prozesses. Nimmt man zum Beispiel die Anwendungsfallanalyse, so besteht diese aus einer Reihe von Tätigkeiten, die zusammenhängend durchgeführt werden, bis hin zum Schreiben der Anwendungsfalltexte. Während die anfänglichen Tätigkeiten - wie Finden von Anwendungsfällen, Kategorisieren und Zuordnen der funktionalen Anforderungen zu den Anwendungsfällen - aber zur Phase „Anforderungsanalyse“ gehören, gehört das Schreiben der Texte zur Phase „Analyse und vorläufiges Design“.

In den folgenden Abschnitten werden die wichtigsten Aspekte der verschiedenen Prozesselemente kurz dargestellt.

2.2.1. Anforderungsanalyse

Der Prozess beginnt mit der Anforderungsanalyse. Im Wesentlichen werden hier drei Tätigkeiten ausgeführt:

- Es wird ein Informationsmodell erstellt,
- es wird ein GUI-Prototyp erstellt,
- es werden Anwendungsfälle identifiziert und kategorisiert.

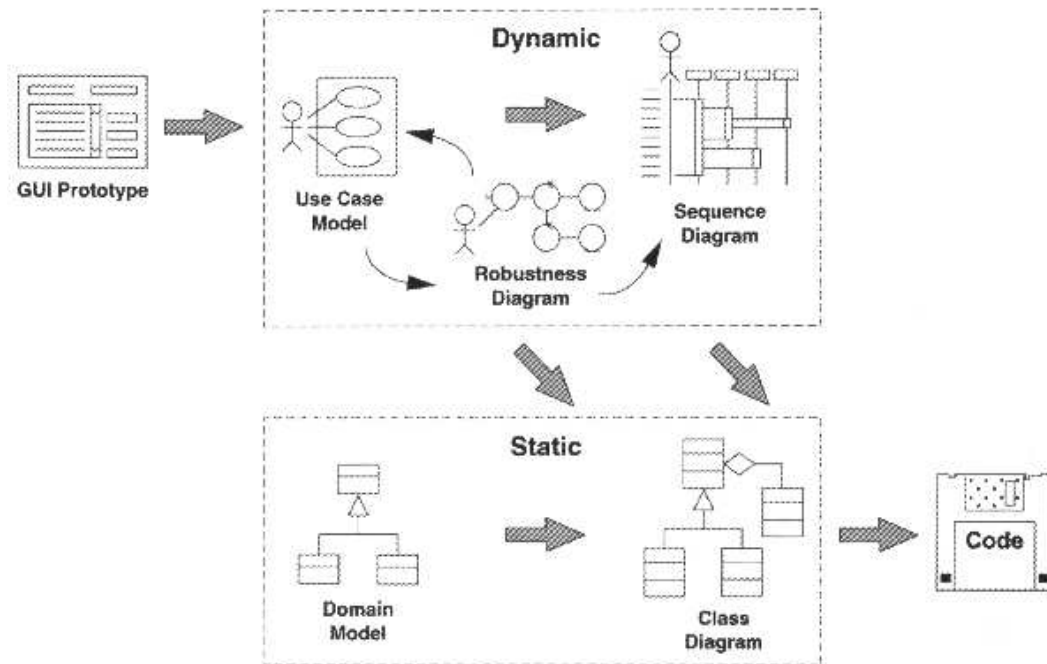


Abbildung 2.1.: Die Phasen des ICONIX-Prozesses (aus [RoSc99, S. 1])

Erstellen des Informationsmodells Das Informationsmodell wird als sehr grobkörniges Klassendiagramm dargestellt. Es beschreibt die Kernabstraktionen (Klassen) des Problembereichs und dient als Startpunkt für die detaillierten Klassendiagramme, die am Ende der Entwurfsphase stehen. Gleichzeitig dient es als Glossar für diese Kernabstraktionen, das allen an der Anforderungsanalyse beteiligten Parteien als Referenz zur Begriffserklärung zur Verfügung steht.

Das Informationsmodell ist die Ausgangsbasis für die gesamte Entwicklung, denn man modelliert die Anwendungsfälle im Kontext des Informationsmodells und man entwickelt es während der Stabilitätsanalyse und der Interaktionsmodellierung zu detaillierten Klassendiagrammen weiter.

Zunächst benutzt man das Informationsmodell allerdings im Sinne einer ersten Annäherung an den Problembereich. Dabei analysiert man die zur Verfügung stehenden textlichen Beschreibungen des Problembereichs nach der von Rumbaugh eingeführten „Object Modeling Technique“ und übernimmt die wichtigsten der dabei gefundenen Substantive als Objekte (Domain Objects) in das Informationsmodell. Außerdem sucht man die strukturellen Beziehungen zwischen den Objekten mit einem besonderen Schwerpunkt auf Generalisierungen und Aggregationen.

Im Hinblick auf die Wiederverwendbarkeit des Informationsmodells in späteren Projekten wird zunächst ausschließlich der Problembereich als ein Abbild der realen Welt modelliert. Das ist auch deshalb wichtig, weil sich das wirkliche Leben nicht so schnell ändert wie die Anforderungen des Kunden. Deshalb ist ein Informationsmodell in aller Regel ein recht stabiles Modell.

Das konkret zu bauende System wird an dieser Stelle des Prozesses noch nicht berücksichtigt. Da es sich nur um eine erste Annäherung an das Problem handelt, werden auch die meisten in Klassendiagrammen enthaltenen Details weggelassen, insbesondere die Attribute und Operationen. Auch andere Entscheidungen, wie etwa, ob es sich bei Beziehungen um Aggregationen oder Kompositionen handelt, werden an dieser Stelle bewusst nicht getroffen, da man für die meisten Entscheidungen

2. Überblick über den ICONIX-Prozess

noch nicht genügend Informationen hat, und man in erster Linie das Ziel verfolgt, schnell zu einer ersten Visualisierung zu kommen. Man will sich also nicht unnötig durch Diskussionen über Details aufhalten lassen.

Erstellen von GUI-Prototypen Bevor man mit der Entwicklung des dynamischen Modells beginnt, muss sichergestellt sein, dass hinreichende Informationen für erste Anwendungsfälle oder Benutzungsszenarios vorhanden sind. Auch sollte eine einigermaßen klare Idee bezüglich der Benutzerschnittstelle vorhanden sein, weil diese eine wichtige Hilfe für die Diskussion mit dem Kunden über das gewünschte Systemverhalten darstellt. Dafür kann man sehr gut GUI-Prototypen einsetzen. Ein erster Schritt sollte also sein, ohne großen Aufwand einige Prototypen zu erstellen, die notfalls auch gezeichnet werden können. Man ist jedoch nicht immer in der Situation, ein völlig neues System entwickeln zu wollen. Oft wird ein altes System abgelöst, weil sich die technische Infrastruktur geändert hat oder weil neue, zusätzliche Funktionen gewünscht sind. Das bedeutet aber nicht unbedingt, dass das alte System schlecht war. In so einem Fall kann man auch so viele Informationen wie möglich über das alte System in Erfahrung bringen, und seine Analyse darauf aufbauen.

Anwendungsfallanalyse Anders als andere Prozesse wird im ICONIX-Prozess großer Wert darauf gelegt, keine Artefakte zu erzeugen, die später nicht mehr von Nutzen sind. Insbesondere benutzen manche anderen Prozesse die Anwendungsfallanalyse als ein mehr oder weniger abstraktes Mittel, um die Anforderungen an das System klar zu ziehen. Der ICONIX-Prozess hingegen beschreibt die Anwendungsfälle als Hilfe für das Design des Systems, mit einem besonderen Gewicht auf der Rückverfolgbarkeit zwischen den Teilen des Modells und den Anwendungsfällen, aus denen sie entstanden sind. Zum Beispiel wird der Text der Anwendungsfälle in die Sequenzdiagramme mit aufgenommen, so dass vollkommen transparent wird, wie die Abläufe mit dem Anwendungsfalltext in Zusammenhang stehen. Um also von Anfang an zu erleichtern, dass man die Anwendungsfälle für den Entwurf des Systems benutzen kann, beschreiben diese die Anwendung des Systems durch den Benutzer im Kontext des Objektmodells. Das bedeutet, dass der Analytiker bereits beim Erstellen von Anwendungsfällen die beteiligten Objekte des Informationsmodells konkret benennt. Auch die Boundaries des Systems, wie zum Beispiel die Bildschirmseiten, bekommen in den Anwendungsfällen bereits Namen.

Zur Identifizierung von Anwendungsfällen nutzt man die Prototypen, um herauszufinden, welche Arbeitsprozesse der Benutzer des Systems durchführen will. Die so gefundenen Anwendungsfälle zeichnet man in die Anwendungsfalldiagramme ein. Außerdem schreibt man Texte für alle Anwendungsfälle, die die Standard- und Alternativabläufe beinhalten, aber keine darüber hinausgehenden Informationen. Diese wären nur Ballast und würden den Blick von den wichtigen Aspekten ablenken, nämlich wie der Benutzer und das System interagieren.

Nachdem die Anwendungsfälle identifiziert sind, werden sie in Anwendungsfalldiagramme eingetragen und Paketen zugeordnet. Abschließend werden die funktionalen Anforderungen den Anwendungsfällen und den damit verbundenen Objekten zugeordnet. Zur Anwendungsfallanalyse gehört auch das Schreiben der Anwendungsfalltexte. Allerdings ist genau hier die Grenze zur nächsten Phase des Prozesses.

Die Anforderungsanalyse schließt mit dem Meilenstein „Anforderungsreview“ ab.

2.2.2. Analyse und vorläufiges Design

In dieser Phase werden die Texte der Anwendungsfälle geschrieben. Das beinhaltet sowohl die Standard- als auch die Alternativabläufe. Dabei sollte besonders darauf geachtet werden, dass die Alternativabläufe vollständig sind, weil viele Systemfunktionen - vor allem die nicht offensichtlichen - in den Alternativabläufen stecken. Daher würde das Vergessen von Alternativabläufen zur Unvollständigkeit der Spezifikation führen.

Stabilitätsanalyse⁴ Die Stabilitätsanalyse ist der Kern des ICONIX-Prozesses. Als Mittel zur Darstellung dienen dabei Stabilitätsdiagramme. Dabei handelt es sich um stereotypierte Klassendiagramme, wobei die Klassen in die Stereotypen Boundary, Entity und Control aufgeteilt werden.

Es handelt sich bei der Stabilitätsanalyse um eine erste Annäherung an das *dynamische* Modell des Systems, einen „ersten Versuch“. Das Ziel dabei ist es, die Lücke zwischen den Anforderungen in Form von Anwendungsfällen und dem detaillierten Design in Form von Sequenzdiagrammen zu füllen. Rosenberg und Scott zufolge ist es erfahrungsgemäß sehr schwierig, diesen Schritt zu vollziehen, wenn man die Stabilitätsanalyse nicht vornimmt. Anwendungsfälle beschreiben nämlich eine anforderungsorientierte Sicht der Dinge, während Sequenzdiagramme eine spezielle, designorientierte Sicht schildern. ICONIX versucht hier die Lücke zwischen dem „Was“ und dem „Wie“ zu füllen.

Bei der Stabilitätsanalyse führt man ein nochmaliges Review der Anwendungsfalltexte durch und macht sich erstmalig Gedanken über die Art, wie man die Anwendungsfälle umsetzen könnte. Dabei versucht man, Objekte zu entdecken, die im Informationsmodell vergessen wurden, und man fügt den Objekten Attribute hinzu, während man den Datenfluss analysiert. Auch trägt man in die Stabilitätsdiagramme die Boundary-Objekte des Systems (z.b. die verschiedenen Bildschirme) ein. Deren Namen sollten zumeist auch schon in den Anwendungsfällen vorkommen. Die neu gefundenen Objekte und Attribute trägt man ins Klassendiagramm ein.

Außerdem verfeinert man die Texte der Anwendungsfälle. Sie sollen sich stabilisieren und konkretisieren, bevor mit Hilfe der Sequenzdiagramme das Detaildesign gemacht wird.

Diese Phase schließt mit dem Meilenstein „Vorläufiges Designreview“ ab.

2.2.3. Design

Die vorletzte Phase des Prozesses befasst sich mit dem Detailentwurf des Systems. Während man in der Stabilitätsanalyse eine erste Idee von der Art entwickelt hat, wie die Objekte miteinander interagieren, um die Anwendungsfälle zu realisieren, überdenkt und detailliert man diese Ideen jetzt und setzt sie in ein Interaktionsmodell um.

Interaktionsmodellierung Mittel zur Darstellung der Ergebnisse sind bei diesem Entwurfsschritt die Sequenzdiagramme. Sie zeigen ein detailliertes Modell der Laufzeitaspekte des Systems. Insbesondere zeigen sie, wie zur Laufzeit vorhandene Instanzen von Klassen durch Nachrichten miteinander kommunizieren. Daher sind sie gut geeignet, um die Verantwortlichkeiten auf die verschiedenen Klassen zu verteilen. Um die Rückverfolgbarkeit von Sequenzdiagrammen zu den Anwendungsfällen, die sie darstellen, zu gewährleisten, zeichnet man ein Diagramm pro Anwendungsfall, den man implementieren will.

⁴Rosenberg und Scott sprechen hier von „Robustness Analysis“. Gemeint ist, dass die Anwendungsfälle auf ihre Stabilität oder Robustheit gegen spätere Änderungen geprüft werden. Insbesondere sollen die Anwendungsfälle in der Designphase nicht mehr geändert werden müssen.

2. Überblick über den ICONIX-Prozess

Beim Entwerfen von Sequenzdiagrammen identifiziert man sowohl die Nachrichten, die zwischen den Objekten ausgetauscht werden müssen, als auch die Objekte selbst und die Methoden, die aufgerufen werden müssen. (Die Nachrichten in den Sequenzdiagrammen sind ja nichts anderes als Methodenaufrufe.)

Um die direkte Zuordnung der Anwendungsfalltexte zu den Abläufen im Diagramm zu gewährleisten, trägt man den Text des Anwendungsfalls auf der linken Seite der Diagramme ein und die Designinformationen auf der rechten Seite.

Optional ist es an dieser Stelle auch noch möglich, ergänzende Spezifikationen in Kollaborationsdiagrammen (für die Schlüsseltransaktionen) oder Zustandsdiagrammen (für die Darstellung des Echtzeitverhaltens) niederzulegen.

Wie im Folgenden erläutert wird, wird abschließend in dieser Phase das in den Klassendiagrammen enthaltene statische Modell fertig gestellt.

Klassendiagramme Die Klassendiagramme entstehen aus dem ursprünglichen Informationsmodell und dienen als Schablone für den Code. Man gewinnt sie aus dem Informationsmodell, indem man dieses im Laufe des Entwurfs nach und nach ergänzt und verfeinert bzw. berichtigt. Dies geschieht im Verlauf der Stabilitätsanalyse und der Interaktionsmodellierung. In der Stabilitätsanalyse ergänzt man das Modell mit den dort gefundenen Operationen und Attributen und während der Interaktionsmodellierung fügt man die Operationen hinzu.

Anschließend werden noch die Kardinalitäten, Sichtbarkeiten, „static“- und „const“-Attribute, abstrakte Klassen und so weiter eingetragen, so dass die so gewonnenen Klassendiagramme nun wirklich alle Details des Entwurfs zeigen und man diese Schablonen lediglich noch ausprogrammieren muss.

Die Designphase endet mit dem Meilenstein „Endgültiges Designreview“.

2.2.4. Implementierung

In der letzten Phase schließlich wird die Software produziert und getestet. Hier wird also endlich Code erzeugt. Soweit nötig, zeichnet man vorher Komponenten- und Verteilungsdiagramme, die bei der Implementierung helfen können.

Zu einer handwerklich guten Umsetzung gehören natürlich auch Modul- und Integrationstests, die nach der Fertigstellung größerer und kleinerer Einheiten der Software durchgeführt werden müssen. Außerdem werden System- und Benutzerakzeptanztests durchgeführt, wobei wieder die Anwendungsfälle als Testszenarien benutzt werden.

Der letzte Meilenstein vor dem Abschluss des Projekts ist demnach die Auslieferung.

2.3. Meilensteine

Obwohl der Weg zum Ziel - abhängig von den Vorlieben und Fähigkeiten der Beteiligten - sehr verschieden sein kann, sollten doch bestimmte Meilensteine immer passiert werden. Sorgt man dafür, dass diese Meilensteine passiert werden, garantiert das zwar nicht den Projekterfolg, aber es erhöht die Erfolgsaussichten deutlich. Die meisten Meilensteine des Prozesses stellen Reviews dar. In diesen wird sichergestellt, dass die oben erwähnten Regeln der einzelnen Phasen auch eingehalten wurden.

Allgemein gesprochen, sollten nach Fertigstellung des Systems folgende Arbeitsschritte durchgeführt worden sein:

- „Das Team hat alle Anwendungsfälle identifiziert und beschrieben, die das System implementieren soll.
- Das Team hat sorgfältig über wiederverwendbare Abstraktionen nachgedacht (Klassen), die in verschiedenen Szenarios vorkommen.
- Das Team hat den Problembereich beschrieben und die Abstraktionen, die in diesem vorkommen (Wiederverwendbarkeit über dieses System hinaus).
- Das Team hat im Design alle funktionalen Anforderungen des Systems bedacht.
- Das Team hat sorgfältig erwogen, wie das geforderte Systemverhalten auf die Klassen verteilt wird. Dabei sind die Prinzipien guten Softwaredesigns berücksichtigt worden.“ (nach [RoSc01])

2. Überblick über den ICONIX-Prozess

Teil II.

Ein Anwendungsbeispiel

3. Aufgabenstellung

In [1] werden die Anforderungen an eine Internet-Buchhandlung beschrieben. Diese Arbeit folgt diesem Beispiel und beschreibt den Weg, auf dem man in Zusammenarbeit mit dem Kunden zu einem fertig modellierten System gelangen kann. Als Startpunkt auf diesem Weg soll die folgende grobe textliche Beschreibung einer solchen Internet-Buchhandlung dienen, die hier die Funktion der Problembeschreibung des Kunden innehat¹:

Bei dem zu entwerfenden System handelt es sich um eine Internet-Buchhandlung, die für bis zu 1.000.000 verschiedener Kunden ausgelegt sein soll. Jeder Kunde soll einen Account besitzen, den er benötigt, um bestimmte Aktionen (wie etwa das Bestellen von Büchern) auszuführen. Für andere Aktionen ist kein Account notwendig. Zur sicheren Zuordnung der Kunden zu ihren Accounts sollen diese mit Passwörtern geschützt sein.

Die Hauptfunktionen der Buchhandlung sind es, Kunden den Buchkatalog zu präsentieren sowie ihnen die Bestellung und Bezahlung von Büchern direkt über das Internet zu ermöglichen.

Um sich die verfügbaren Artikel anzusehen, sollen die Kunden entweder den Buchkatalog durchstöbern können oder dort gezielt nach Büchern suchen können. Dabei soll eine Suche nach Autor, Titel, ISBN-Nummer und Schlüsselwörtern möglich sein. Während der Kunde den Katalog durchsieht, soll er online Informationen über die Verfügbarkeit von Büchern erhalten können. Jedem Buch soll außerdem ein bestimmter Preisrahmen zugeordnet sein, aus dem der Buchshop je nach Marktlage den aktuellen Preis bestimmen kann. Der aktuelle Preis soll natürlich ebenfalls im Katalog angezeigt werden.

Jedem Kunden soll während einer „Einkaufssitzung“ ein Warenkorb zugeordnet werden, in dem er Bücher platzieren kann, die er später zu bestellen gedenkt. Wenn der Kunde seine Auswahl getroffen hat, soll er die Bestellung aufgeben können. Die Bezahlung soll entweder per Kreditkarte oder per Rechnung erfolgen können. Kunden sollen den Status von Bestellungen online verfolgen können. Auch sollen bereits gelieferte Bestellungen noch nachträglich nachvollziehbar sein.

Außerdem soll es jedem Kunden möglich sein, Rezensionen für Bücher zu schreiben, die dann im Buchshop eingesehen werden können. Bestandteil einer Rezension kann auch eine Punktbewertung sein. Diese kann aber auch allein abgegeben werden. Andere (auch potentielle) Kunden sollen die Rezensionen und Bewertungen einsehen können.

¹Grundlage für diesen Text ist die stichpunktartige Anforderungsbeschreibung auf Seite 16 des Buches von Rosenberg und Scott [RoSc01].

3. *Aufgabenstellung*

4. Informationsmodell

4.1. Allgemeines

Die Grundlage für das statische Modell bildet im ICONIX Prozess das Informationsmodell. Dabei handelt es sich um ein sehr grob gehaltenes Klassendiagramm, das die Objekte des Problembereichs enthält. Die Grundidee dabei ist, ein möglichst wahrheitsgetreues Modell des Problembereichs zu erhalten. Mit diesem Ansatz erreicht man eine relativ hohe Stabilität des resultierenden Klassenmodells, da sich die Gegebenheiten der realen Welt normalerweise wesentlich langsamer ändern als die Anforderungen des Kunden. Außerdem erreicht man ein hohes Maß an Wiederverwendbarkeit des Informationsmodells, insbesondere wenn man plant, noch weitere Software für dieses Problemfeld zu schreiben. Hier kann man dann seine weiteren Analyse- und Entwurfsschritte auf dem gleichen Informationsmodell aufbauen. Das Informationsmodell ist jedoch nur der Startpunkt für das weitere Vorgehen. Im Laufe des fortschreitenden Modellierungsprozesses wird es den sich verdichtenden Erkenntnissen immer weiter angepasst, so dass man am Ende zu ein sehr detailliertes Klassenmodell erhält, das als Schablone für die Codierung ohne weiteres geeignet ist.

4.2. Auffinden des Informationsmodells

Wie kommt man nun zu einem Informationsmodell? Die grundlegende Methodik findet sich in der „Object Modeling Technique“ (siehe dazu auch [Rum94]). Dabei handelt es sich im Prinzip um eine grammatikalische Analyse der zur Verfügung stehenden Texte. Dies sind in erster Linie die grobe Problembeschreibung des Kunden und weitere, detailliertere Anforderungsdefinitionen. Darüber hinaus kann auch weiterführende Fachliteratur und Expertenwissen zur Analyse herangezogen werden. In diesen Quellen wird nun nach Substantiven und Verben gesucht. Die aufgefundenen Substantive sind Kandidaten für Objekte und Attribute. Besitzanzeigende Satzteile deuten in diesem Kontext darauf hin, dass die betreffenden Substantive eher Attribute denn Objekte sind. Verben hingegen sind Kandidaten für Assoziationen und Operationen¹.

4.2.1. Allgemeines zum Informationsmodell

Attribute oder Objekte Beim Versuch, im Informationsmodell Objekte zu finden, stellt sich immer wieder die Frage, ob bestimmte „Dinge“ im Problembereich als Objekte modelliert werden müssen, oder ob es sich um Attribute handelt, die somit nicht dargestellt werden. Grundsätzlich ist es möglich, etwa mehrteilige Attribute als eigene Objekte mit eigenen Operationen zu modellieren (zumindest was die Zugriffsmethoden angeht) und als Teil-Ganzes-Beziehung mit übergeordneten Objekten darzustellen. Alternativ dazu kann man aber die Attribute der aggregierten Objekte auch in das

¹Diese Vorgehensweise ist auch als die „Methode von Abbot“ bekannt.

4. Informationsmodell

übergeordnete Objekt aufnehmen. In diesem Fall würden sie im Informationsmodell nicht dargestellt, da in diesem Teil des Prozesses Attribute noch keine Rolle spielen.

In [BoRuJa99, S. 206f.] wird der Objektbegriff folgendermaßen definiert:

„Eine Abstraktion besitzt in ihrem *Objekt* eine konkrete Ausprägung, auf die man eine Menge von Operationen anwenden kann und die einen Zustand besitzen kann, in dem die Auswirkungen der Operationen gespeichert sind. Instanz und Objekt sind weitgehend synonym.“

und weiterhin:

„Generell ist ein Objekt etwas, das in der wirklichen oder konzeptionellen Welt Raum einnimmt, und mit dem man etwas machen kann. So ist z.B. das Objekt eines Knotens üblicherweise ein Computer, der sich physisch in einem Raum befindet, ein Objekt einer Komponente nimmt Raum in einem Dateisystem ein und das Objekt eines Kundendatensatzes verbraucht etwas vom physischen Speicherplatz. Auch die Instanz einer Einflugschneise für ein Flugzeug ist etwas, das man mathematisch manipulieren kann.“

In diesem Sinne sind zunächst alle möglichen Dinge Objekte. Das gilt auch für einfache Variablen wie Zahlen, denn sie haben einen Zustand - ihren Wert - und es lassen sich Operationen auf ihnen ausführen, nämlich Rechenoperationen. Dieser Zustand ist aber *im Problembereich* nicht immer unmittelbar bedeutsam, genauso wenig wie die auszuführenden Operationen, so dass es sich je nach Kontext eben doch nicht um Objekte handelt, da sie keinen Raum im zu modellierenden Teil der Welt einnehmen. Letztendlich ist eben doch die Bedeutsamkeit des Objekt-Kandidaten für den jeweiligen Problembereich das entscheidende Kriterium. Wenn man ein objektorientiertes System baut, das Arithmetik ausführt, so sind die Werte als Zustände der Zahlenobjekte ebenso bedeutsam wie die Rechenoperationen. Im anderen Systemen sind solche einfachen Attribute jedoch in der Regel Mittel zum Zweck - denn sie tragen einen Teil des Objektzustands - und sind somit keine Objekte.

Beziehungsklassen Eine Assoziation dient zum Verbinden von „Paaren“ von Objekten. Eine Beziehungsklasse ist eine spezielle Form der Assoziation, die eigene Eigenschaften besitzt. Diese Eigenschaften gelten für genau eine Kombination von Objekten einer Klasse [BoRuJa99, S. 167f.]. Sie zeichnet sich also dadurch aus, dass sie nicht existieren kann, wenn eines der beiden Objekte fehlt. Da die Assoziation über eigene Attribute verfügt, stellt man sie als Klasse dar.

Beziehungen und Aktionen Rein temporäre Zusammenhänge in Form von Aktionen bleiben im statischen Modell unberücksichtigt; sie sind Teil des dynamischen Modells. Wichtig sind hier vielmehr diejenigen Sachverhalte, die über den betreffenden Zeitpunkt hinaus noch von Bedeutung sind, die also die strukturellen Beziehungen zwischen den Objekten ausmachen.

Hier einige Beispiele für Sätze, die Aktionen beschreiben:

- Ein Kunde bezahlt ein Buch.
- Kunden durchstöbern den Buchkatalog.
- Der Kunde verfolgt den Status der Bestellung.

Diese Sätze enthalten keinerlei Informationen über die *statischen* Beziehungen zwischen den betroffenen Objekten. Wenn ein Kunde z.B. den Buchkatalog durchstöbert, bleibt dies zunächst völlig ohne Folgen. Weder der Buchkatalog noch der Kunde erfährt dadurch eine Zustandsänderung. Das Gleiche gilt für die Verfolgung des Bestellstatus. Anders ist dies etwa beim Bestellen eines Buches. Hier ist es auch nach der Bestellung noch wichtig zu wissen, wer die Bestellung aufgegeben hat, um hinterher die Rechnung richtig adressieren zu können, die Bestellung in die richtige Liste einzuordnen und so weiter. Daraus folgt eine „stammt von“ Beziehung zwischen Bestellung und Kunde.

4.2.2. Identifizieren der Klassen

Im vorliegenden Fall steht als Quelle zunächst lediglich die Anforderungsdefinition des Kunden zur Verfügung (siehe Kapitel 3). Diese wird im Folgenden noch einmal stichpunktartig wiederholt. Dabei werden die Substantive fett dargestellt und die Verben kursiv, um die grammatikalische Analyse zu vereinfachen:

1. Der **Buchshop** soll **Bestellungen** aus dem **Internet** *entgegennehmen* können.
2. Er soll bis zu 1.000.000 **Kunden** *verwalten* können.
3. **Kunden** sollen **Accounts** *besitzen*, die durch **Passwörter** *geschützt* sein sollen.
4. **Kunden** sollen **Bücher** online *bestellen* können.
5. **Kunden** sollen **Bücher** in einem **Warenkorb** zur **Bestellung** *vormerken* können.
6. **Kunden** sollen **Informationen** über die **Verfügbarkeit** von **Büchern** *erhalten* können.
7. Der **Shop** soll eine sichere **Möglichkeit** bieten, per **Kreditkarte** *zu bezahlen*.
8. Der **Shop** soll eine sichere **Möglichkeit** bieten, per **Rechnung** *zu bezahlen*.
9. **Kunden** sollen den **Buchkatalog** *durchsuchen* können. Dabei soll eine **Suche** nach **Autor**, **Titel**, **ISBN-Nummer** und **Schlüsselwörtern** möglich sein.
10. **Kunden** sollen den **Status** von **Bestellungen** online *verfolgen* können. Auch bereits gelieferte **Bestellungen** sollen *nachvollziehbar* sein.
11. Es soll jedem **Kunden** möglich sein, **Rezensionen** für **Bücher** *zu schreiben*, die dann im **Buchshop** *eingesehen* werden können.
12. **Kunden** sollen **Bücher** *bewerten* dürfen (in der Art von **Punkten** oder ähnlichem). Die **Bewertungen** sollen von anderen **Kunden** *eingesehen* werden können.

Alle in diesem Text vorkommenden Substantive sind *Kandidaten* für Objekte², jedoch reicht die Tatsache, in einem Text ein Substantiv zu finden, noch nicht aus, um daraus eine Klasse zu machen. Um herauszufinden, welche Kandidaten als Klassen geeignet sind und welche nicht, stellt man sich am besten die Frage, welchen Sinn ein Kandidat für den Problembereich hat³. Man eliminiert dann

²Das Informationsmodell ist ein Klassendiagramm, kein Objektdiagramm. Bevor man jedoch Klassen abstrahieren kann, müssen zunächst die Objekte gefunden werden, aus denen sich diese Klassen herleiten.

³Nach der OMT ist alles das ein Objekt, was für die betreffende Anwendung einen Sinn ergibt. [Rum94]

4. Informationsmodell

schwammige Begriffe, Attribute, Synonyme, Irrelevantes und Implementierungsdetails bzw. technische Begriffe. Übrig bleiben die geeigneten Klassen, mit denen man das statische Modell darstellen kann.

Die mit dieser Methode gefundenen Klassen sind in Tabelle 4.1 dargestellt.

4.2.3. Identifizieren der Beziehungen

Um von diesem Klassendiagramm zum Informationsmodell zu gelangen, fehlen aber noch die Assoziationen. Auch diese lassen sich durch die grammatikalische Analyse der zugrunde liegenden Texte finden. Dabei ist allerdings zu beachten, dass das Informationsmodell nur die statischen Zusammenhänge zwischen den Objekten des Problembereichs darstellt (siehe auch Abschnitt 4.2.1).

Die obige Problembeschreibung des Kunden bietet hier nicht genügend Informationen, um daraus die Beziehungen zwischen den Objekten herleiten zu können, da sie überwiegend Aktionen beschreibt, die im Informationsmodell ja eben nicht zum Tragen kommen. Es ist daher nötig, diejenigen Sachverhalte zu ergänzen, die implizit in den Angaben enthalten sind, oder die so selbstverständlich sind, dass sie gar nicht genannt werden⁴. Die Beziehungen, die bereits in der Anforderungsbeschreibung enthalten waren, sind im folgenden Text fett kursiv dargestellt, die impliziten (also ergänzten) Beziehungen sind kursiv dargestellt:

Bei dem zu entwerfenden System handelt es sich um eine Internet-Buchhandlung, die für bis zu 1.000.000 verschiedener Kunden ausgelegt sein soll. **Jeder Kunde soll einen Account besitzen**, den er benötigt, um bestimmte Aktionen (wie etwa das Bestellen von Büchern) auszuführen. Für andere Aktionen ist kein Account notwendig. Zur sicheren Zuordnung der Kunden zu ihren Accounts sollen diese mit Passwörtern geschützt sein.

Der Shop führt (hat) eine Kundenliste.

Die Kundenliste enthält die Kunden(-Daten).

Die Hauptfunktionen der Buchhandlung sind es, Kunden den Buchkatalog zu präsentieren sowie ihnen die Bestellung und Bezahlung von Büchern direkt über das Internet zu ermöglichen.

Der Buchkatalog enthält Bücher.

Um sich die verfügbaren Artikel anzusehen, sollen die Kunden entweder den Buchkatalog durchstöbern können oder dort gezielt nach Büchern suchen können. Dabei soll eine Suche nach Autor, Titel, ISBN-Nummer und Schlüsselwörtern möglich sein.

Der Shop präsentiert das Suchergebnis.

Das Suchergebnis enthält Bücher.

Während der Kunde den Katalog durchsieht, soll er online Informationen über die Verfügbarkeit von Büchern erhalten können. **Jedem Buch soll außerdem ein bestimmter Preisrahmen zugeordnet sein**, aus dem der Buchshop je nach Marktlage den aktuellen Preis bestimmen kann. Der aktuelle Preis soll natürlich ebenfalls im Katalog angezeigt werden.

Jedem Kunden soll während einer „Einkaufssitzung“ ein Warenkorb zugeordnet werden, in dem er Bücher platzieren kann, die er später zu bestellen gedenkt. Wenn der Kunde seine Auswahl getroffen hat, soll er die Bestellung aufgeben können.

⁴Dies ist ohnehin empfehlenswert, um ein möglichst vollständiges Bild des Problembereichs zu erhalten.

Kandidat	Objekt	Attribut von	irrelevant	Synonym für	Implementierung	zu vage
Account				Kunde		
Autor		Buch				
Bestellung	x					
Bewertung		Buch				
Buch	x					
Buchkatalog	x					
Buchshop			x			
Information						x
Internet					x	
ISBN		Buch				
Kunde	x					
Kreditkarte		Kunde				
(Bezahl-) Möglichkeit						x
Passwort		Kunde				
Punkte				Bewertung		
Preisrahmen	x					
Rechnung	x					
Rezension	x					
Schlüsselwort		Buch				
(Bestell-) Status		Bestellung				
Suche						x
Titel		Buch				
Verfügbarkeit		Buch				
Warenkorb	x					

Tabelle 4.1.: Mit der OMT gefundene Klassen

4. Informationsmodell

Der Warenkorb enthält Positionen, die wiederum Bücher enthalten.

Kunden bestellen Bücher.

Bestellungen bestehen aus Positionen. Positionen enthalten Bücher.

Die Bezahlung soll entweder per Kreditkarte oder per Rechnung erfolgen können.

Eine Bestellung ist eine Rechnungs- oder Kreditkartenbestellung.

Der Shop erstellt Abrechnungen für die Bestellungen.

Eine Abrechnung ist eine Rechnung oder eine Kreditkartenabrechnung.

Eine Abrechnung ist für einen Kunden bestimmt und bezieht sich auf eine Bestellung.

Kunden sollen den Status von Bestellungen online verfolgen können. Auch sollen bereits gelieferte Bestellungen noch nachträglich nachvollziehbar sein.

Der Shop führt Bestelllisten.

Die Bestelllisten enthalten die Bestellungen.

Eine Bestellliste ist einem Kunden zugeordnet.

Außerdem soll es jedem Kunden möglich sein, Rezensionen für Bücher zu schreiben, die dann im Buchshop eingesehen werden können. Bestandteil einer Rezension kann auch eine Punktbewertung sein. Diese kann aber auch allein abgegeben werden. Andere (auch potentielle) Kunden sollen die Rezensionen und Bewertungen einsehen können.

4.2.4. Identifizieren weiterer Klassen

Durch die obige Analyse der Aufgabenstellung treten in den ergänzten Passagen einige neue Klassen zu Tage, die in das Informationsmodell aufgenommen werden müssen:

- die **Kundenliste**, ohne die keine Accounts geführt werden könnten oder irgendwelche Daten von Kunden gespeichert werden könnten,
- die Generalisierung von Rechnung, **Abrechnung**, und eine weitere Spezialisierung, die **Kreditkartenabrechnung**,
- die Spezialisierungen von Bestellung, **Kreditkartenbestellung** und **Rechnungsbestellung**,
- die **Position**, die in Bestellung und Warenkorb enthalten ist (Es reicht nicht aus zu sagen, dass eine Bestellung und ein Warenkorb Bücher enthalten. Vielmehr sind weitere Angaben wie die Menge oder der Gesamtpreis vonnöten. Die Gesamtheit dieser Angaben ist dann eine Position.),
- das **Suchergebnis**, denn die vom Kunden angestoßene Suche muss irgendwie zu sichtbaren Resultaten kommen,
- die **Bestellliste**, ohne die es nicht möglich ist, die Bestellungen eines Kunden zu verwalten.

Nachdem nun die Sammlung der Klassen vervollständigt wurde, kann man daran gehen ein erstes Klassendiagramm ohne Beziehungen zu zeichnen. Dieses Klassendiagramm zeigt Abbildung 4.1. In einem weiteren Arbeitsschritt werden auch die Beziehungen eingetragen, was dann zum kompletten Informationsmodell in Abbildung 4.2 führt.

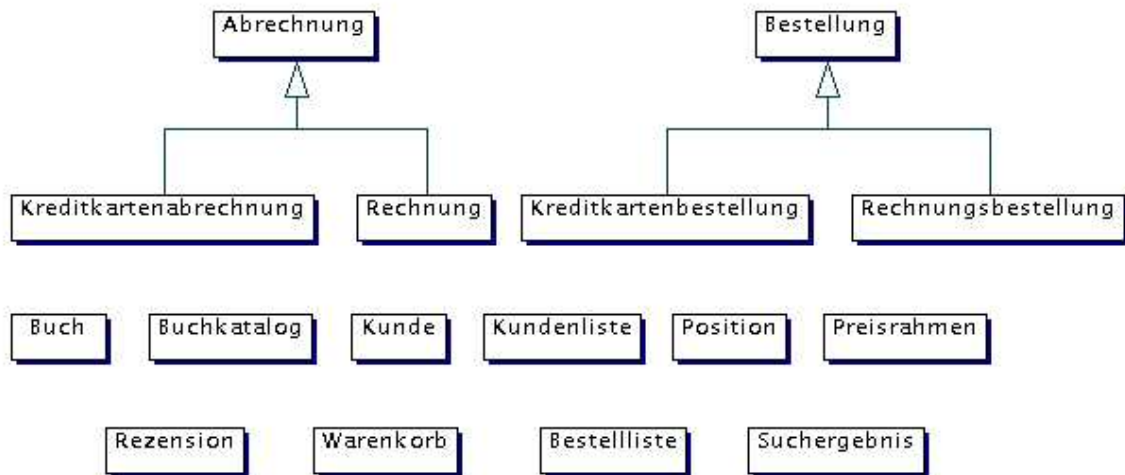


Abbildung 4.1.: Erste Klassen

4.2.5. Erstellen eines Glossars

Das Informationsmodell soll nicht nur der Startpunkt für den Entwurf des Softwaresystems sein, sondern soll darüber hinaus auch als Kommunikationsmittel zwischen dem Auftraggeber und dem Entwicklerteam dienen. Um diese Kommunikation zu erleichtern ist es hilfreich, ein Glossar der Klassen anzufertigen. Dieses beschreibt alle im Modell vorkommenden Abstraktionen eindeutig und in Textform, so dass alle am Projekt beteiligten Parteien stets wissen, was unter den jeweiligen Begriffen zu verstehen ist. Das Glossar für das Informationsmodell zeigt Tabelle 4.2.

4.2.6. Anmerkungen zum Informationsmodell

Nach dem Auffinden aller Objekte und Assoziationen ergibt sich das in Abbildung 4.2 dargestellte Informationsmodell. Hier noch einige abschließende Anmerkungen dazu.

Varianten von Kunde Betrachtet man die Klasse Kunde genauer, so erkennt man verschiedene bedeutsame Eigenschaften. Damit eine Abrechnung erstellt werden kann und Bestellungen ausgeliefert werden können, muss der Kunde zum Beispiel eine Rechnungs- und Lieferanschrift besitzen. Falls er mit Kreditkarte bezahlt, müssen die Kreditkartendaten gespeichert werden. Auf den ersten Blick führt das zu einem Teil des Informationsmodells wie in Abbildung 4.3 dargestellt. Es wäre auch möglich, die Attribute der Adress- und Kreditkartendaten-Klassen in das Kunden-Objekt zu überführen, so dass die Adressen und Kreditkartendaten aus dem Informationsmodell verschwinden. Hier greifen die Überlegungen aus Abschnitt 4.2.1 über die Bedeutsamkeit der betreffenden Objekte: Die Verwendungen, die man sich für Adressen und Kreditkartendaten vorstellen könnte, sind ausnahmslos nur für den Shop bedeutsam, nicht jedoch für die Sicht des Kunden auf den Buchshop. Es handelt sich zum Beispiel um Operationen wie

- das Gruppieren von Bestellungen nach der Postleitzahl der Lieferanschrift, um Bestellungen gemeinsam versenden zu können

4. Informationsmodell

(Klassen-) Name	Beschreibung
Abrechnung	Eine Abrechnung dient dazu, den Kunden zur Zahlung des fälligen Betrages für eine Bestellung zu bewegen. Es handelt sich entweder um eine Rechnung oder um eine Kreditkartenabrechnung, je nach dem Typ der Bestellung.
Bestellliste	Diese Liste enthält alle Bestellungen, die ein Kunde jemals beim Shop in Auftrag gegeben hat. Dies schließt auch die Bestellungen ein, die zwar schon beauftragt, aber noch nicht ausgeliefert wurden.
Bestellung	Eine Bestellung ist ein Auftrag über die Lieferung einer oder mehrerer Positionen von Büchern.
Buch	selbsterklärend
Buchkatalog	Der Buchkatalog enthält alle Bücher, die der Shop im Moment prinzipiell führt, unabhängig vom aktuellen Lagerbestand.
Kreditkartenabrechnung	Eine Kreditkartenabrechnung ist eine Abrechnung für eine Bestellung, die der Kunde mit seiner Kreditkarte bezahlen will. Sie wird direkt mit der Kreditkartenfirma abgerechnet.
Kreditkartenbestellung	Eine Kreditkartenbestellung ist eine Bestellung, die der Kunde mit seiner Kreditkarte bezahlt.
Kunde	Als Kunde zählt jede natürliche oder juristische Person, die in irgendeiner Form mit dem Shop in Kontakt getreten ist und als potentieller Käufer von Büchern in Frage kommt, unabhängig davon, ob ein Account angelegt wurde oder nicht.
Kundenliste	Die Kundenliste ist die Liste aller Kunden, die sich irgendwann einmal beim Buchshop angemeldet haben.
Position	Eine Position enthält im Wesentlichen Bücher, mit Mengenangabe und Gesamtpreis. Sie kann im Warenkorb enthalten sein oder als Teil einer Bestellung auftreten.
Preisrahmen	Der Buchshop gibt die obere und untere Preisgrenze an, zu der ein Buch verkauft werden kann, sowie den aktuellen Preis, der sich der Marktsituation anpasst. Diese drei Parameter bilden zusammen den Preisrahmen.
Rechnung	Eine Rechnung ist eine Abrechnung, die nicht über die Kreditkartenfirma abgewickelt wird, sondern erst nach Erhalt des Artikels vom Kunden direkt bezahlt wird.
Rechnungsbestellung	Eine Rechnungsbestellung ist eine Bestellung, die der Kunde per Rechnung bezahlen darf.
Suchergebnis	Wenn ein Kunde den Buchkatalog durchsucht, sind die Bücher, die den Suchkriterien entsprechen, im Suchergebnis enthalten.
Rezension	Eine Rezension ist eine Beschreibung eines Buches durch einen Kunden. Sie sollte den Inhalt des Buches beschreiben und eine Aussage über dessen Qualität und Nützlichkeit machen. Im Prinzip ist aber jeder Kunde frei zu schreiben, was er möchte.
Warenkorb	Wenn ein Kunde nach Büchern sucht, kann er Bücher im Warenkorb platzieren, die er zu kaufen gedenkt. Wenn er sich später wirklich entschließt, die Bücher zu kaufen, wird daraus eine Bestellung.

4.2. Auffinden des Informationsmodells

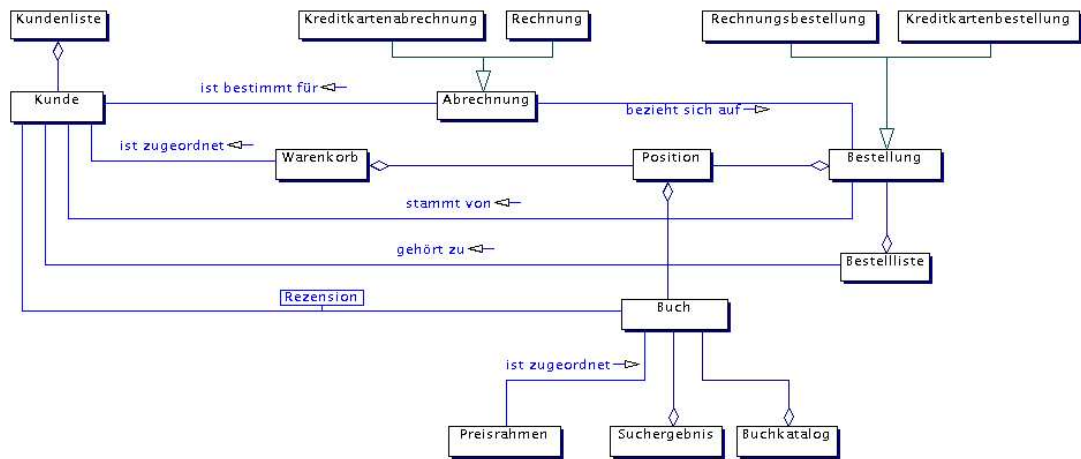


Abbildung 4.2.: Informationsmodell

oder

- das Sortieren von Kunden nach ihrer Kreditkartenfirma, um Kreditkartenbestellungen im Batch-Betrieb abzurechnen.

Da die Interna des Buchshops jedoch in diesem Modell außer Acht gelassen werden, entfallen die betreffenden Objekte im Informationsmodell. Sie werden später zu Attributen von Kunde.

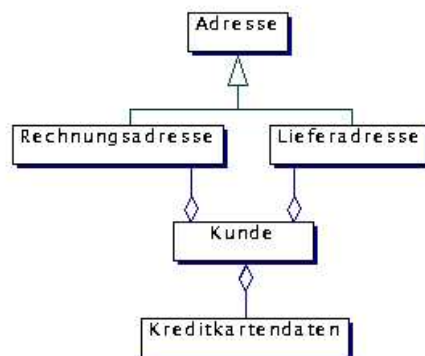


Abbildung 4.3.: Kunde mit Aggregationen

Rezension als Assoziationsklasse Bei der Rezension handelt es sich um eine Paarung von Buch und Kunde. Der Kunde hat dabei die Rolle des Autors der Rezension, während das Buch rezensiert wird. Keines der beiden Objekte kann fehlen, sonst käme keine Rezension zustande. Außerdem

4. Informationsmodell

hat die Rezension aber noch eigene Eigenschaften wie den Text und zum Beispiel eine sich anschließende Punktbewertung oder das Datum der Entstehung, was die Modellierung als Beziehungsklasse angebracht erscheinen lässt.

In diesem Sinne könnte man auch die Abrechnung als Beziehungsklasse modellieren, die den Kunden mit der Bestellung paart, denn ohne eine Bestellung kann es keine Abrechnung geben und man muss wissen, auf welchen Kunden sich die Abrechnung bezieht. Jedoch verfügt auch die Bestellung schon über dieses Wissen, denn sie ist über die „stammt von“-Assoziation bereits mit dem Kunden verbunden. Daher ist hier keine unmittelbare Notwendigkeit gegeben, eine Beziehungsklasse zu modellieren, denn wenn man wissen will, an wen eine Abrechnung zu schicken ist, kann man über den Umweg der Bestellung zum betreffenden Kunden navigieren. Dass es trotzdem eine „ist bestimmt für“-Assoziation zwischen Abrechnung und Kunde gibt, ist zunächst einmal redundant, es erscheint aber durchaus sinnvoll, dass die Abrechnung dem Kunden direkt zugeordnet wird.

4.3. Vergleich mit dem Original

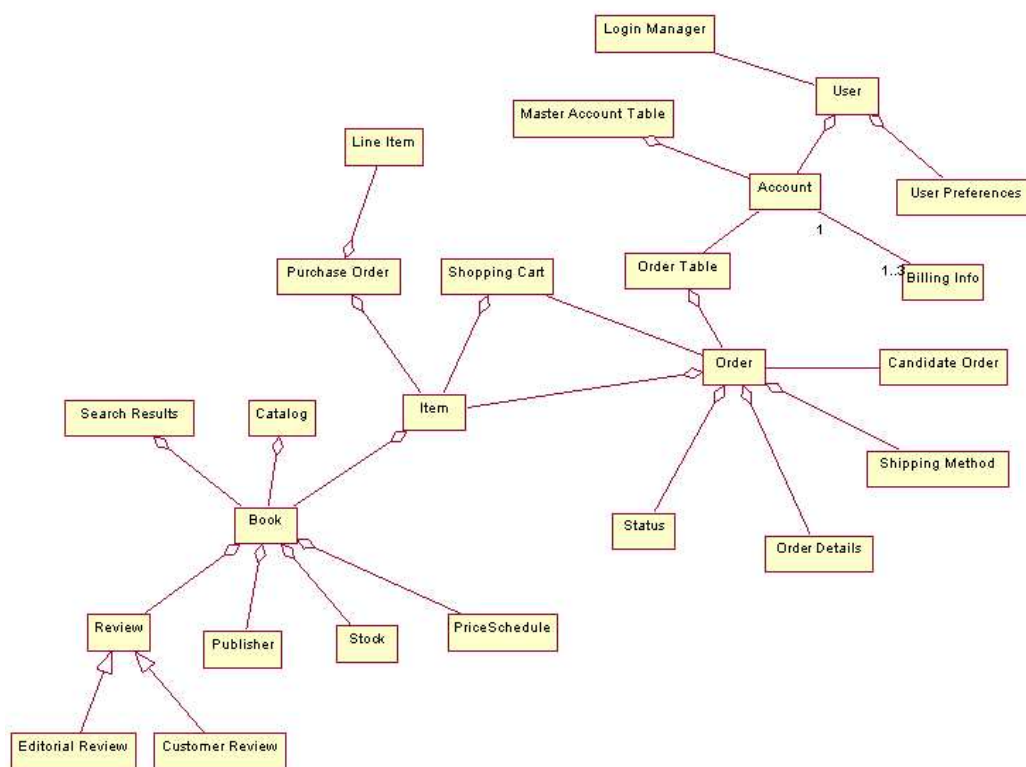


Abbildung 4.4.: Informationsmodell von Rosenberg und Scott

Assoziationen, Aggregationen und Attribute Das in dieser Arbeit entwickelte Informationsmodell und das in [RoSc01] vorgestellte sind in den wesentlichen Abstraktionen identisch. Das ist auch nicht weiter verwunderlich, da es wohl unstrittig ist, dass man Klassen wie Kunde, Buch und

Bestellung benötigt und dass sich daran entsprechende andere Klassen anknüpfen müssen. Der Unterschied liegt hier eher im Detail, nämlich welche der anknüpfenden Klassen man benötigt und wie die Assoziationen zwischen den Klassen aussehen.

Bei der Betrachtung des Informationsmodells von Rosenberg und Scott fällt zunächst einmal die vergleichsweise große Menge an Aggregationen auf. Zwar sind im ICONIX-Prozess die Aggregationen von besonderer Bedeutung, denn auf diese und auf die Generalisierungen wird besonders geachtet, wenn man das Informationsmodell erstellt, doch scheint hier ein wenig zu viel des Guten getan worden zu sein. Insbesondere sind einige Aggregationen enthalten, die nach Meinung des Autors Assoziationen sein sollten und einige Klassen sind eigentlich Attribute:

Ein „Customer Review“ (Rezension) kann keinesfalls Bestandteil eines Buches sein, sondern ist diesem zugeordnet. Das „Editorial Review“ (Waschzettel) ist tatsächlich ein Bestandteil des Buches, jedoch ist nicht einzusehen, dass ein „Customer Review“ und ein „Editorial Review“ beide Spezialisierungen von Review sind, da beide Klassen völlig verschiedene Merkmale haben. Das eine wird nämlich von Kunden nachträglich angefertigt und hat den Charakter einer Bewertung, während das andere eine Beschreibung des Buches ist, die von Anfang an unveränderlich dazugehört.

Auch ein „Price Schedule“ (Preisrahmen) kann kein Bestandteil des Buches sein. Der aktuelle Preis ist wohl ein Attribut des Buches, denn er trägt einen Teil der Zustandsinformation, der Preisrahmen jedoch ist dem Buch nur zugeordnet. Der Lagerbestand wird hier als Attribut modelliert, weil er vom Kunden nur in Form von Verfügbarkeitsinformationen eingesehen wird, für die Sicht des Kunden auf den Shop aber weiter keine Bedeutung hat.

User und Account Diskussionswürdig ist sicherlich auch der obere rechte Teil des Rosenberg-Modells, der sich um User und Account rankt. Nach Meinung des Autors ist der Account in diesem Stadium des Modells nur ein Synonym für User. Beide Klassen unterscheiden sich nur durch ihre Privilegierungsstufe. Bestimmte Aktionen dürfen User nämlich nur ausführen, wenn sie einen Account haben, andere eben nicht.

Weiterhin sind in diesem Teil des Modells Klassen zu finden, die keine fachliche Bedeutung haben, sondern eher technischer Natur sind und deshalb nicht dargestellt werden sollten. Dies sind vor allem die „Master Account Table“ (die aber eine fachliche Entsprechung in der Kundenliste findet) und der „Login Manager“, die eindeutig technische Begriffe sind, sowie die „User Preferences“.

Die Klasse „Billing Info“ spaltet sich im hier vorgeschlagenen Modell wie in Abschnitt 4.2.6 diskutiert in Kreditkartendaten und Rechnungsanschrift auf. Beides sind Attribute von Kunde. Sicherlich kann es an dieser Stelle auch sinnvoll sein, die Beziehung zwischen Kunde und Rechnungsanschrift als Assoziation zu modellieren, da eine Adresse ja von mehreren Personen benutzt werden kann. Für eine Kreditkarte gilt dies aber im Allgemeinen nicht, so dass hier eine Aufspaltung in mehrere Klassen durchaus angebracht erscheint.

Order Die Klasse „Order“ aggregiert bei Rosenberg und Scott „Shipping Method“, „Status“ und „Order Details“. Die beiden ersten sind im hier vorgestellten Modell Attribute von Bestellung, da sie für die Sicht des Kunden auf den Shop keine Bedeutung haben. Zwar hat die Bestellung einen Status, den der Kunde auch einsehen können soll, aber er führt keinerlei Operationen auf dem Bestellstatus aus. Dies ist vielmehr eine Aufgabe des Shop-Betreibers während der internen Abwicklung von Bestellungen. Auch die Versandmethode ändert sich nicht mehr, nachdem die Bestellung aufgegeben ist.

4. Informationsmodell

Unklarheiten im Rosenberg-Modell Darüber hinaus weist das Rosenberg-Modell auch einige Ungereimtheiten auf, die ihre Ursachen unter anderem in der fehlenden Erklärung der Begrifflichkeiten haben. Zum Beispiel ist nicht einsichtig, was eine „Candidate Order“ von einem „Shopping Cart“ unterscheidet. Wenn der Kunde etwas in seinen Warenkorb hineinlegt, so handelt es sich doch dabei um die Vormerkung einer Bestellung. Hier besteht kein Unterschied, was die Klasse „Candidate Order“ überflüssig macht. Auch der Begriff der „Order Details“ ist überhaupt nicht einleuchtend. Diese Klasse entfällt zugunsten von Item (Position). Auch ist der Begriff der „Purchase Order“, die aus „Items“ und „Line Items“ besteht, unklar. Hier handelt es sich wohl um eine besondere Eigenheit von Rosenberg's Modell, die aber ohne weitere Erklärung nicht verständlich ist. Umso wichtiger ist es, an dieser Stelle ein Glossar zu führen. Dies haben die Autoren ja selbst propagiert, unterlassen es jedoch in ihrem Beispiel.

Bestellen von Büchern Eine weitere Abweichung zwischen beiden Modellen besteht in der Art, wie Bestellungen abgewickelt werden. Während nämlich in [RoSc01] der Warenkorb der Bestellung zugeordnet wird und nur über die Bestellliste und den Account herausgefunden werden kann, welcher Kunde sie ausgeführt hat, wird hier der Ansatz verfolgt, die Bestellung und den Warenkorb dem Kunden direkt zuzuordnen. Einem Kunden sind also eine Bestellliste und ein Warenkorb zugeordnet, aber auch die Bestellung selbst verfügt über das Wissen, wer der Auftraggeber ist. Es scheint nur natürlich, dass eine Bestellung zu einem bestimmten Kunden gehören muss.

Abrechnungen Als letzter Punkt ist noch ein für den Shopbetreiber sehr bedauerlicher Sachverhalt zu nennen: Das Originalmodell enthält keine Klasse „Abrechnung“. Während zwar prinzipiell die Möglichkeit gegeben ist, über „Billing Info“ eine Rechnung zu erstellen, ist doch kein Abrechnungsobjekt vorgesehen, so dass es schwer fallen dürfte, das Geld vom Kunden einzuziehen.

5. Anwendungsfälle

Im ICONIX Prozess dienen die Anwendungsfälle nicht nur als Werkzeug zur Anforderungsanalyse, sondern sie dienen auch dazu, den Entwicklungsprozess voran zu treiben. Als Vorstufe zur Architekturfindung ist es das Ziel, alles zu beschreiben, was ein Benutzer mit dem System tun kann. Die Anwendungsfälle beschreiben somit die volle Systemfunktionalität aus Sicht des Benutzers, und zwar im Kontext des Informationsmodells. Es gilt, das Systemverhalten so zu beschreiben, dass man möglichst nahtlos in die Architekturfindung eintreten kann. Die elementare Frage ist nun, wie man die Anwendungsfälle eigentlich findet. Rosenberg macht dazu in [RoSc99, S. 41] folgenden Vorschlag:

„Ich ermutige meine Kunden, so oft wie möglich schnelles Prototyping einzusetzen. Die Idee dabei ist, dass Entwickler und Benutzer sich zusammensetzen und etwas entwerfen, das demonstriert, dass die Konzepte funktionieren.

Im Laufe der Jahre habe ich erkannt, dass der beste Weg, Gruppen von Anwendungsfällen zu finden, der ist, im Zusammenhang mit dem Prototypen ein grobes Benutzerhandbuch zu schreiben, so als ob der Prototyp tatsächlich ein funktionierendes System wäre. [...]

Um noch einen Schritt weiter zu gehen, kann ich sagen, dass es normalerweise ein exzellenter Ansatz ist, die grafische Benutzerschnittstelle (GUI) parallel zum gewünschten Systemverhalten zu erforschen. Das bezieht mit ein, mit den Benutzern zusammen durch die Präsentationsaspekte des Systems zu iterieren und nachdem man sich über einige Bildschirmansichten einig geworden ist, die zugehörigen Anwendungsfälle zu schreiben.“

Bei dem Vorhaben, eine Benutzerschnittstelle zu entwerfen, die das gewünschte Systemverhalten auch wirklich widerspiegelt, stellt sich die Frage, wie man dieses Systemverhalten denn nun eigentlich ergründen soll. Im Grunde genommen ist das ja eben die Leistung beim Schreiben von Anwendungsfällen. Wenn man statt dessen eine Benutzerschnittstelle entwirft, muss man das Systemverhalten doch bereits kennen. Wo also starten?

Eine Möglichkeit, dieses Problem zu lösen, liegt in der Problembeschreibung. Hier ist bereits das gewünschte Systemverhalten zumindest ansatzweise beschrieben. Man kann dort also die wichtigsten Geschäftsvorfälle extrahieren und dafür erste, grobe Anwendungsfälle schreiben. Diese benutzt man dazu, die GUI-Prototypen zu entwerfen, mit denen man die Abläufe, die das System implementieren soll, genauer analysieren (und wie Rosenberg beschreibt mit den Fachleuten besprechen) kann. So findet man dann nach und nach alle Anwendungsfälle, die das Systemverhalten vollständig beschreiben, unter Berücksichtigung der in der Problembeschreibung enthaltenen Anforderungen. Das GUI-Prototyping ist also Teil der Anwendungsfallanalyse. Prototyping und das Niederlegen von Anwendungsfällen gehen Hand in Hand und ergänzen sich gegenseitig. Durch ein paralleles und iteratives Vorgehen an dieser Stelle gewinnt man nach und nach ein komplettes Bild vom gewünschten und erforderlichen Systemverhalten.

Der ICONIX Prozess legt Wert darauf, dass die Anwendungsfälle nicht zu grob gestaltet sind. Es ist vielmehr darauf zu achten, dass sie klar und unzweideutig beschreiben, wie der Benutzer und das

5. Anwendungsfälle

System interagieren. Außerdem soll ein Augenmerk auf ihre Wiederverwendbarkeit gelegt werden, also die Fragestellung beleuchtet werden, welche Anwendungsfälle von welchen anderen aufgerufen werden können. Darüber hinaus gilt es zu beachten, dass man eine Aufgabenteilung zwischen Prototypen und Anwendungsfällen im Auge behält. Die *Details der Präsentation* und weitgehend auch die Details der einzugebenden Daten werden im Prototypen dargestellt. Die *Art und Weise*, wie der Benutzer mit dem System arbeitet, wird hingegen von den Anwendungsfällen beschrieben.

Um schließlich die Anwendungsfälle zu finden, geht man bei Internet-basierten Systemen am Besten so vor, dass man sich die Prototypen Seite für Seite ansieht und daraufhin beleuchtet, wo der Benutzer auf Links oder Schaltflächen klickt und was er dadurch erreichen will. Diese Benutzeraktionen ordnet man dann Anwendungsfällen zu. Manchmal wird durch das Klicken unmittelbar ein Anwendungsfall „aufgerufen“. In anderen Fällen handelt es sich um Aktionen, die lediglich dabei helfen, einen Anwendungsfall abzuarbeiten, oder die zu Alternativabläufen gehören.

5.1. Erste Anwendungsfälle

In der Problembeschreibung in Kapitel 3 lassen sich fünf herausragende Geschäftsvorfälle finden, auf denen man den Prototypen aufbauen kann:

1. Buchkatalog
2. Büchersuche
3. Warenkorb
4. Buchbestellung
5. Bestellungshistorie

Die initialen Anwendungsfälle dazu lesen sich wie folgt:

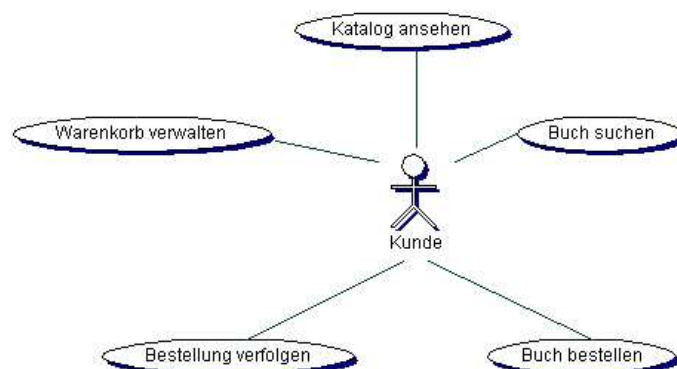


Abbildung 5.1.: Erste Anwendungsfälle

5.1.1. Katalog ansehen

Standardablauf Der Kunde öffnet den Buchkatalog. Das System zeigt eine hierarchische Ansicht der verfügbaren Buch-Kategorien an, durch die der Kunde navigieren kann. Wenn eine Kategorie Bücher enthält, zeigt das System auch eine Kurzzinformation zu diesen Büchern an. Der Kunde hat dann die Möglichkeit, für dieses Buch Produktdetails anzusehen, es in den Warenkorb zu legen, Rezensionen anzusehen und zu schreiben und Informationen über die Verfügbarkeit des Buches zu erhalten.

5.1.2. Buch suchen

Standardablauf Das System präsentiert eine Ansicht zur Büchersuche. Der Kunde wählt dort ein Suchkriterium aus, etwa ISBN, Autor, Titelstichwort oder ähnliches und gibt einen Suchbegriff ein. Das System durchsucht dann den Katalog und zeigt die Kurzzinformationen aller gefundenen Bücher an.

5.1.3. Warenkorb verwalten

Standardablauf Der Kunde klickt auf „Artikel in den Warenkorb legen“. Das System fügt dem Warenkorb das Buch hinzu, berechnet den Gesamtpreis aller Positionen im Warenkorb neu und zeigt die aktuellen Informationen erneut an. Später hat der Kunde die Möglichkeit, Artikel wieder aus dem Warenkorb zu löschen oder die Stückzahl zu ändern.

5.1.4. Buch bestellen

Standardablauf Der Kunde entschließt sich, die Bücher im Warenkorb zu bestellen. Das System öffnet daraufhin die Seite für die Buchbestellung. Der Kunde gibt alle notwendigen Daten für eine Buchbestellung ein, wie Rechnungsanschrift, Kreditkartendaten, Lieferadressen, Versandart und bestätigt die Bestellung. Das System präsentiert noch einmal eine Übersicht der Daten, die auch den Inhalt des Warenkorbs enthält. Der Kunde bestätigt die Bestellung und das System speichert sie ab.

5.1.5. Bestellung verfolgen

Standardablauf Das System bietet auf dieser Seite eine Filtermöglichkeit nach offenen und bereits gelieferten Bestellungen über verschiedene Zeiträume. Der Kunde wählt ein Filterkriterium aus und das System zeigt alle entsprechenden Bestellungen des Kunden an. Der Kunde kann auf dieser Seite auch Bestellungen stornieren, die noch offen sind.

5.2. GUI Prototyp und Benutzerhandbuch

5.2.1. Prototyp

Nachdem man sich nun im Groben über das erforderliche Systemverhalten im Klaren ist, gilt es, dies in einem Prototypen zu verifizieren und zu verfeinern. Dabei wird man sich automatisch über die notwendigen Abläufe klar werden. In mehreren Iterationen wird man versuchen, die Schnittstelle so zu gestalten, dass sie dem Benutzer optimal bei der Ausführung seiner Aufgaben hilft.

5. Anwendungsfälle

Startpunkt zur Entwicklung des Prototypen ist das Hauptmenü. Zunächst werden den genannten fünf Anwendungsfällen dort eigene Positionen zugewiesen. Das Hauptmenü sollte außerdem noch durch zwei weitere Positionen ergänzt werden, die (obwohl in der Problembeschreibung nicht erwähnt) bei einer Internetpräsenz nicht fehlen dürfen: Die Startseite und die Seite mit den Kontaktinformationen. (Erstere ist hier nicht im Hauptmenü enthalten, sondern in dem Hilfsmenü im oberen Rand des Hauptanzeigebereichs. Nichtsdestotrotz kann sie jederzeit von überall aus erreicht werden.) Das Anklicken der entsprechenden Position bringt für jeden der damit verbundenen Anwendungsfälle eine eigene Seite zur Anzeige. Von dieser aus kann der Benutzer dann jeweils über Navigationselemente wie Links oder Schaltflächen auf andere Seiten verzweigen, um Teilaufgaben durchzuführen oder neue Anwendungsfälle aufzurufen. Außerdem enthalten die verschiedenen Bildschirmansichten aber auch Formularelemente, die beim Abarbeiten von Anwendungsfällen helfen. Beim Bestellen eines Buches zum Beispiel muss der Kunde die Versandmodalitäten bestimmen und die Zahlungsdaten angeben und anschließend die Bestellung bestätigen (Abschnitt 5.1.4). Für diese drei Ansichten öffnet man zweckmäßigerweise neue Seiten, um die Buchbestellung nicht zu unübersichtlich werden zu lassen und gleichzeitig eine Gliederung nach Teilaufgaben vorzunehmen. Da diese Teilaufgaben nacheinander bewältigt werden können, kommt man zu einem Wizard-ähnlichen Aufbau des Bestellprozesses, der in Abbildung 5.2 als Zustandsdiagramm dargestellt ist¹. (Die Zustände stellen dabei die verschiedenen Seiten dar, die der Benutzer während der Buchbestellung durchgehen kann und die Events stehen für die Navigationselemente, die er anklicken muss, um von einer Seite zur anderen zu gelangen.) Außerdem finden sich auf den entsprechenden Seiten Formularelemente zum Eingeben der Adressdaten und zum Auswählen der Versandart und der Zahlungsweise. Eine Beschreibung der wichtigsten Bildschirmansichten finden sich im Benutzerhandbuch in Anhang A.

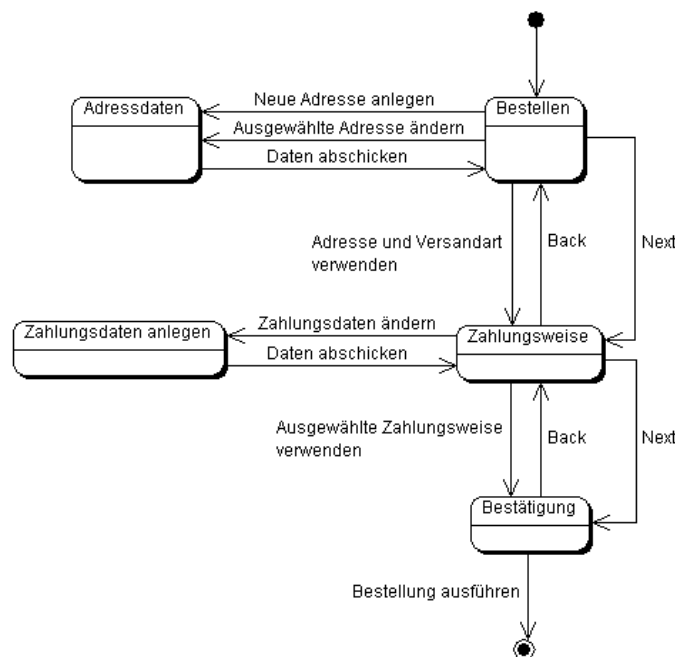


Abbildung 5.2.: Navigation durch den Bestellprozess

¹Diese Art der Darstellung stammt von Meilir Page-Jones und wird in dem Buch „Practical OO with UML“ beschrieben.

Ein Beispiel für den Aufruf eines Anwendungsfalls ohne Navigation ist das Platzieren eines Buches im Warenkorb, das auf der Seite „Produktdetails“ vorgenommen werden kann. Das Klicken auf den Link „In den Warenkorb legen“ öffnet dabei keine neue Seite, sondern fügt lediglich dem Warenkorb den aktuellen Artikel hinzu. Bei diesem Anwendungsfall muss der Benutzer keine weiteren Eingaben vornehmen, sondern er läuft automatisch im Hintergrund ab.

5.2.2. Benutzerhandbuch

Der nächste Schritt zum Anwendungsfallkatalog ist das Verfassen des Benutzerhandbuchs. Dazu werden die Ansichten und Abläufe, die man im Prototypen niedergelegt hat, ganz einfach aus der Sicht des Benutzers dokumentiert. Bei dieser Aktivität nimmt man gleichzeitig eine Gliederung vor, bei der man die verschiedenen Ansichten des Prototypen nach den Aufgabengebieten gruppiert, die man beim Entwickeln schon im Auge hatte.

Die wichtigsten Ansichten des Prototyps werden hier in Anhang A in Form eines Kurzhandbuchs erläutert.

5.3. Identifizieren der Anwendungsfälle

Vom Benutzerhandbuch aus zu einem vollständigen Katalog der Anwendungsfälle zu gelangen, ist nicht weiter schwierig, denn die Identifizierung der Anwendungsfälle ergibt sich direkt aus der Beschreibung der Benutzerschnittstelle. Im Wesentlichen beschreibt dabei ein Abschnitt des Benutzerhandbuchs einen Anwendungsfall. Abweichungen entstehen lediglich durch die Zerlegung von Anwendungsfällen und die naturgemäß eingeschränkte Sicht des Benutzers auf das System. Dies ist zum Beispiel der Fall bei der Rezension von Büchern, wo das Erstellen und das Überarbeiten in verschiedenen Anwendungsfällen unterkommen, obwohl sie im gleichen Abschnitt des Handbuchs beschrieben sind. Man kann nämlich auf der gleichen Seite eine Rezension verfassen oder überarbeiten. Der Unterschied liegt einfach nur im angezeigten Inhalt der Seite. Der Anwendungsfall „Artikel in den Warenkorb legen“ hingegen ist aus dem Abschnitt „Detailansicht“ der Katalogbeschreibung extrahiert worden, denn aus Benutzersicht besteht der Aufruf des Anwendungsfalls nur aus einem einzigen Klick. Im Großen und Ganzen jedoch entspricht eine Seite der Benutzerschnittstelle - bzw. der entsprechende Abschnitt des Kurzhandbuchs - einem Anwendungsfall. Das resultierende Anwendungsfalldiagramm findet sich in Abbildung 5.3. Dies kann jedoch nur ein erster Entwurf sein, denn man wird mit Sicherheit beim Schreiben der Anwendungsfälle noch auf Zerlegungsmöglichkeiten stoßen. Außerdem muss noch das Verhalten der Anwendungsfälle untereinander analysiert werden. Daher sind an dieser Stelle die extend- und include-Beziehungen zwischen den Anwendungsfällen noch nicht enthalten.

5.4. Beziehungen zwischen den Anwendungsfällen

Ein hilfreiches Mittel, um die Beziehungen zwischen den Anwendungsfällen zu modellieren, ist eine Untersuchung des Navigationskonzepts der Benutzerschnittstelle. Ein Ausschnitt davon (nämlich der Unterautomat Bestellung) war bereits in Abbildung 5.2 zu sehen. Abbildung 5.4 zeigt das vollständige Navigationskonzept². Dieses stellt dar, welche Bildschirmansichten von welchen anderen aus

²Die schattierten Zustände sind vom Startzustand aus direkt erreichbar, was aus Übersichtlichkeitsgründen hier aber nicht dargestellt wird. Die Startseite (Zustand „Amazon“) wird vom System automatisch aufgerufen, wenn der Kunde den

5. Anwendungsfälle

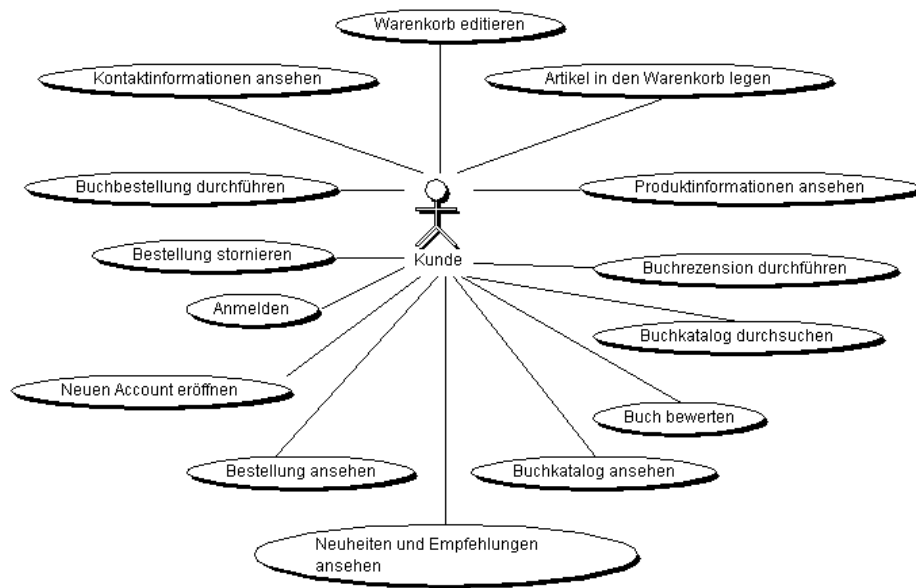


Abbildung 5.3.: Anwendungsfälle vor der Zerlegung

aufgerufen werden können und durch welche Navigationselemente dieses erfolgt. Wenn man außerdem weiß, welche Bildschirmansichten zu welchen Anwendungsfällen gehören, ergeben sich daraus direkt die Abhängigkeiten zwischen den Anwendungsfällen. Als Beispiel sei hier die Anmeldeprozedur genannt. Hier gibt es zwei alternative Wege. Der Kunde kann entweder auf der Startseite auf „Anmelden“ klicken oder er versucht einen Anwendungsfall auszuführen, der erfordert, dass die Kennung des Kunden bekannt ist. In diesem Fall erzwingt das System eine Anmeldung, falls der Kunde noch nicht angemeldet ist.

Die Startseite gehört zum Anwendungsfall „Neuheiten und Empfehlungen ansehen“, während die Seite „Anmeldung“ zum Anwendungsfall „Anmelden“ gehört. Da eine Transition von der Startseite zur Systemanmeldung existiert, gibt es zwischen den zugehörigen Anwendungsfällen ebenfalls eine Beziehung. Der Kunde kann die Transition feuern, indem er den Link „Anmelden“ klickt. Er muss dies aber nicht tun. Daher handelt es sich um ein optionales Aufrufen, also eine extend-Beziehung von „Anmelden“ nach „Neuheiten und Empfehlungen ansehen“. Der alternative Weg kann von drei Seiten aus beschriftet werden:

- vom Warenkorb, wenn der Kunde auf „Bestellen“ klickt,
- von der Seite „Produktdetails“, wenn der Kunde eine Rezension schreiben will
- vom Hauptmenü, wenn der Kunde die Seite „Historie“ aufruft

Wenn der Kunde also z.B. die Schaltfläche „Historie“ im Hauptmenü klickt, ohne angemeldet zu sein, leitet das System ihn zur Anmeldung weiter. Nach erfolgreicher Validierung der Benutzerdaten zeigt

Shop betritt. Sie ist außerdem vom Hilfsmenü oberhalb des Hauptanzeigebereichs immer erreichbar. Die anderen Seiten sind vom Hauptmenü aus zugänglich.

5.4. Beziehungen zwischen den Anwendungsfällen

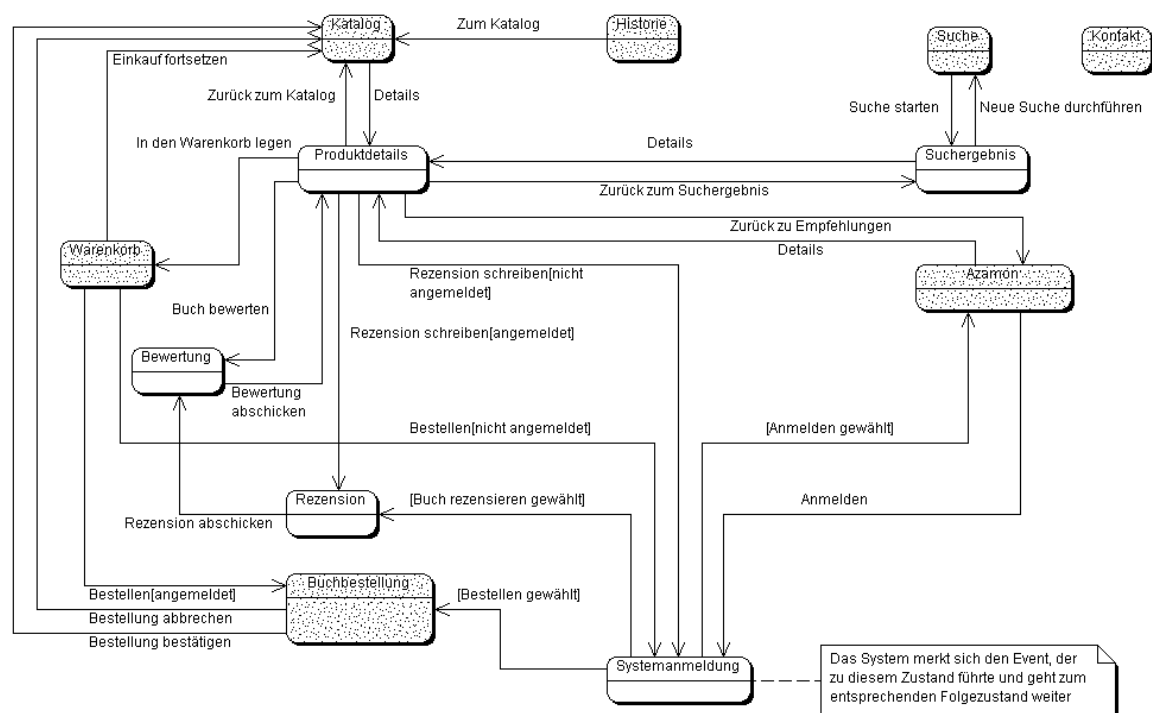


Abbildung 5.4.: Navigationskonzept

5. Anwendungsfälle

das System dann die Seite „Historie“ an. Diese Seite gehört zum Anwendungsfall „Bestellung ansehen“. Da auch hier der Anwendungsfall „Anmelden“ nur aufgerufen wird, wenn der Kunde nicht schon angemeldet war (also optional), erweitert der Anwendungsfall „Anmelden“ den Anwendungsfall „Bestellung ansehen“ über den extend-Mechanismus.

Das Gleiche gilt auch für den Anwendungsfall „Buchbestellung durchführen“. Hier klickt der Kunde auf der Seite „Warenkorb“ auf „Bestellen“. Das System prüft, ob der Kunde angemeldet ist. Wenn nicht, zeigt es zunächst die Seite „Anmeldung“ an und nach erfolgreicher Validierung der Benutzerdaten die Seite „Bestellen“. „Anmelden“ erweitert also „Buchbestellung durchführen“. Außerdem erweitert „Anmelden“ aus demselben Grund den Anwendungsfall „Buchrezension durchführen“. Eine vollständige Liste der Seiten des Prototypen und der zugehörigen Anwendungsfälle zeigt Tabelle 5.1.

Anwendungsfall	Titel der Seite
Neuheiten und Empfehlungen ansehen	Azamon
Anmelden	Anmeldung
	Passworthinweis
Neuen Account anlegen	Account anlegen
Buchkatalog ansehen	Katalog
Produktinformationen ansehen	Produktdetails
Artikel in den Warenkorb legen	Produktdetails
Buchrezension durchführen	Rezension
Buch bewerten	Bewertung
Buchkatalog durchsuchen	Suche
	Suchergebnis
Warenkorb editieren	Warenkorb
Buchbestellung durchführen	Bestellen
	Adressdaten ändern
	Zahlungsweise
	Zahlungsdaten anlegen
	Bestätigung
Bestellung ansehen	Historie
Bestellung stornieren	Historie
Kontaktinformationen ansehen	Kontakt

Tabelle 5.1.: Benutzeraktionen und Anwendungsfälle

Abbildung 5.5 zeigt das vollständige Anwendungsfalldiagramm mit den Beziehungen zwischen den Anwendungsfällen. Man beachte, dass sich die Kommunikationsbeziehungen zwischen dem Kunden und den Anwendungsfällen stark reduziert haben. Dies liegt daran, dass die meisten Anwendungsfälle nicht direkt vom Hauptmenü aus erreichbar sind, sondern nur aufgerufen werden können, wenn der Kunde vorher einen anderen Anwendungsfall ausführt. In dieser Situation geht man davon aus, dass die Kommunikation durch die Kommunikationsbeziehung zwischen Kunde und anfänglich aufgerufenem Anwendungsfall impliziert wird (siehe auch [HiKa99], Seite 109).

5.4. Beziehungen zwischen den Anwendungsfällen

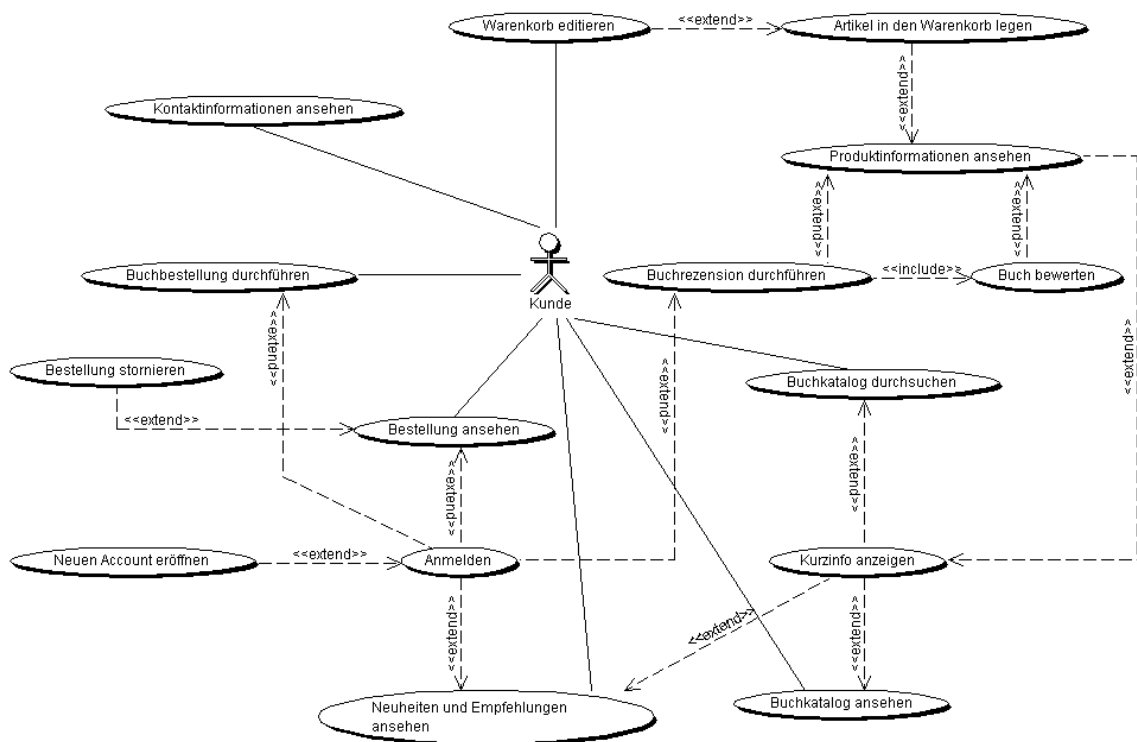


Abbildung 5.5.: Übersicht über die Anwendungsfälle

5. Anwendungsfälle

5.5. Zuordnen der funktionalen Anforderungen

Nachdem man die Anwendungsfälle identifiziert hat, muss unbedingt sichergestellt werden, dass diese alle in der Problemstellung enthaltenen funktionalen Anforderungen abdecken³. Tabelle 5.2 zeigt die Zuordnung der Anforderungen aus der Problemstellung zu den entsprechenden Anwendungsfällen.

Funktionale Anforderung	Abgedeckt durch Anwendungsfall
Passwortgeschützte Accounts	Anmelden; Neuen Account eröffnen
Buchbestellung online	Buchbestellung durchführen
Buch im Warenkorb zur Bestellung vormerken	Artikel in den Warenkorb legen; Warenkorb editieren
Verfügbarkeitsinformationen einsehen	Produktinformationen ansehen
Bezahlung per Kreditkarte	Buch bestellen
Bezahlung per Rechnung	Buch bestellen
Buchkatalog durchsuchen	Buchkatalog durchsuchen
Status offener Bestellungen verfolgen	Bestellung ansehen
Gelieferte Bestellungen nachvollziehen	Bestellung ansehen
Rezensionen schreiben	Buchrezension durchführen
Buch bewerten	Buch bewerten
Rezension einsehen	Produktinformationen ansehen
Bewertung einsehen	Produktinformationen ansehen

Tabelle 5.2.: Anforderungen und Anwendungsfälle

5.6. Beschreibung der Anwendungsfälle

Nachdem nun die Anwendungsfälle identifiziert sind und die funktionalen Anforderungen den Anwendungsfällen zugeordnet wurden, kann man daran gehen, die Texte der Anwendungsfälle zu verfassen. Dazu stützt man sich im Wesentlichen auf den Inhalt des Kurzhandbuchs. Dieses beschreibt jedoch lediglich die Benutzerseite und macht kaum Aussagen über die Antworten des Systems auf bestimmte Benutzeraktionen. Diese müssen also in den Anwendungsfällen ergänzt werden. Auch wird im Benutzerhandbuch der Standardablauf der Anwendungsfälle in den Vordergrund gestellt, und nicht die Alternativabläufe. Daher müssen auch diese vervollständigt werden.

Die Entwicklung von Anwendungsfällen aus dem Text des Benutzerhandbuchs soll hier aus Platzgründen nur an einigen Beispielen erklärt werden. Eine vollständige Liste der Anwendungsfalltexte befindet sich in Anhang B.

³Diese Aktivität wird eigentlich im Rahmen des Anforderungsreviews durchgeführt. Im Rahmen dieser Arbeit machen Reviews allerdings wenig Sinn, so dass wir es bei diesem Arbeitsschritt bewenden lassen.

5.6.1. Buchbestellung

Hier zunächst der Text des Benutzerhandbuchs aus dem Anhang A.5:

Bestellen

Das Bestellen von Büchern kann entweder vom Hauptmenü aus oder von der Seite „Warenkorb“ aus erfolgen. Klicken Sie auf den Link oder die Schaltfläche „Bestellen“ und die Bestellung kann beginnen. Voraussetzung, um eine Bestellung auszuführen, ist allerdings, dass der Warenkorb überhaupt Bücher enthält.

Der Bestellvorgang erfolgt in drei Schritten:

Lieferdaten angeben

Im oberen Teil des Hauptanzeigebereichs finden Sie eine Liste der Adressen, die Sie als mögliche Lieferadressen angegeben haben. Wählen Sie hier eine Adresse aus, indem Sie den entsprechenden Radiobutton markieren. Falls Sie noch keine Lieferadresse hinterlegt haben, klicken Sie auf „Neue Adresse anlegen“, um die Daten für eine neue Adresse einzugeben. Sie können auch falsch eingegebene Adressdaten korrigieren, wenn Sie die Adresse markieren und auf „Ausgewählte Adresse ändern“ klicken. In beiden Fällen öffnet sich dann die Seite „Adressdaten ändern“. Machen Sie dort ihre Eingaben und klicken Sie auf „Daten abschicken“. Die Daten werden dann in die Datenbank übernommen und Sie gelangen hier auf diese Seite zurück. Sie haben hier auch die Möglichkeit, ungültig gewordene Adressdaten zu löschen, indem Sie die Adresse markieren und auf „Ausgewählte Adresse löschen“ klicken.

Anschließend müssen Sie noch die Versandart festlegen. Wählen Sie eine der Optionen aus und klicken Sie danach auf „Adresse und Versandart verwenden“, um zur nächsten Seite zu gelangen.

Zahlungsmodalitäten festlegen

Der zweite Schritt besteht im Angeben der Zahlungsmodalitäten. Wenn Sie noch Neukunde sind, haben Sie zunächst nur die Möglichkeit, per Kreditkarte zu bezahlen. Nach drei erfolgreichen Transaktionen können Sie dann auch auf Rechnung bestellen. Wählen Sie also aus, ob Sie per Rechnung oder per Kreditkarte bestellen wollen und prüfen Sie die Rechnungsanschrift, bzw. die Kreditkartendaten. Falls bestimmte Angaben nicht korrekt sind, klicken Sie auf „Zahlungsdaten ändern“. Sie gelangen dann zur Seite „Zahlungsdaten anlegen“, wo Sie die betreffenden Korrekturen durchführen können. Wenn Sie auf dieser Seite dann auf „Daten ändern“ geklickt haben, gelangen Sie zurück zur Seite „Zahlungsweise“. Nachdem Sie eine Zahlungsweise markiert haben, klicken Sie auf „Ausgewählte Zahlungsweise verwenden“, um zur nächsten Seite zu gelangen.

Daten bestätigen

Als letzten Schritt müssen Sie noch einmal alle Angaben auf ihre Richtigkeit überprüfen. Falls Sie noch einen Fehler entdecken, können Sie mit den „Back“ und „Next“-Symbolen am oberen Rand des Hauptanzeigebereichs durch den gesamten Bestellprozess vor und zurück navigieren und die betreffenden Änderungen durchführen. Sie können den Bestellprozess auch an jeder Stelle abbrechen, wenn Sie sich entscheiden sollten, doch nicht zu bestellen. Sind jedoch alle Daten korrekt, dann klicken Sie auf „Bestellung durchführen“, um die Bestellung in Auftrag zu geben.

5. Anwendungsfälle

Der Text des Benutzerhandbuchs mag ausreichen, um dem Kunden kurz und bündig den Bestellprozess zu beschreiben, doch fehlt hier noch einiges an Details:

- Das System muss die Kundendaten kennen, um die Bestellung dem richtigen Kunden zuzuordnen. Daher muss es sicherstellen, dass der Kunde angemeldet ist und notfalls eine Anmeldung erzwingen.
- Das Systemverhalten bei fehlender Lieferadresse muss geklärt werden. Der Kunde kann ja gar keine Lieferadresse auswählen, wenn er noch keine Adressen angelegt hat.
- Es muss die Frage geklärt werden, wie sich das System vor und nach den drei erfolgreichen Transaktionen verhalten soll. Es wird dem Kunden anfangs nicht erlaubt sein, Rechnungsdaten einzugeben, oder zumindest darf er die Rechnungsbestellung nicht auswählen. Später muss er dann eine Rechnungsbestellung durchführen dürfen.
- Das System sollte die Richtigkeit von Lieferadresse, Rechnungs- und Kreditkartendaten verifiziert, um dem Shop unnötige Kosten durch falsche Lieferungen und Betrug durch falsche Zahlungsdaten zu ersparen. Im dem Zusammenhang muss festgelegt werden, wie sich das System verhält, wenn es Fehler bei der Dateneingabe erkennt.

Darüber hinaus ist es erklärtes Ziel, die Anwendungsfälle im Kontext des Informationsmodells zu beschreiben, denn die Anwendungsfälle sollen als Eingabe für das Detaildesign dienen. Andererseits hat der sehr persönliche Stil, in dem das Benutzerhandbuch gehalten ist, in einem Anwendungsfall keinen Platz. Hier ist eine objektive Beschreibung der Abläufe angebracht.

Der Text des folgenden Anwendungsfalls wird die aufgeworfenen Fragen klären und die Lücken schließen:

Buchbestellung durchführen

Standardablauf Das System stellt zunächst sicher, dass der Kunde angemeldet ist und dass der Warenkorb für diese Einkaufssitzung mindestens eine Position enthält. Dann erzeugt es ein Bestellungs-Objekt und kopiert den Inhalt des Warenkorbs hinein. Anschließend ermittelt das System die Lieferadressen, die für diesen Account gespeichert sind und zeigt diese Adressen auf der Seite „Bestellen“ an. Außerdem ermittelt das System die zur Verfügung stehenden Lieferarten und stellt sie ebenfalls auf dieser Seite dar.

Der Kunde wählt eine Adresse und eine Versandart aus und klickt anschließend auf „Adresse und Versandart verwenden“. Das System stellt nun sicher, dass wirklich eine Adresse und Versandart ausgewählt ist und versieht dann die Bestellung mit der gewünschten Adresse und Versandart. Dann zeigt es wieder die Seite „Zahlungsweise“ an.

Es liest aus dem Kunden-Objekt die Kreditkartendaten und die Rechnungsanschrift aus und zeigt sie auf der Seite „Zahlungsweise“ an. Der Kunde wählt nun die gewünschte Zahlungsweise aus und drückt die Schaltfläche „Ausgewählte Zahlungsweise verwenden“. Das System fügt daraufhin der Bestellung die gewählte Zahlungsweise hinzu. Dann wechselt es zur Seite „Bestätigung“, wo es noch einmal eine Übersicht aller Bestelldaten anzeigt.

Der Kunde versichert sich noch einmal der Richtigkeit der Daten und klickt auf „Bestellung ausführen“. Das System fügt nun die Bestellung in die Bestellliste des Kunden ein und leert den Warenkorb. Dann ruft es den Anwendungsfall „Buchkatalog ansehen“ auf.⁴

Alternativabläufe Wenn der Kunde noch nicht am System angemeldet ist, ruft das System am Anfang dieses Anwendungsfalls den Anwendungsfall „Anmelden“ auf.

Wenn der Inhalt des Warenkorbs leer ist, zeigt das System eine entsprechende Meldung an, bricht diesen Anwendungsfall ab und übergibt die Kontrolle an den Anwendungsfall „Buchkatalog ansehen“.

Wenn das System keine Lieferadresse finden kann oder der Kunde auf der Seite „Bestellen“ auf „Neue Adresse anlegen“ klickt, öffnet das System die Seite „Adressdaten“. Der Kunde gibt dort neue Adressdaten an und bestätigt die Daten durch einen Klick auf „Daten abschicken“. Das System prüft daraufhin so weit wie möglich die Korrektheit der Adressdaten und fügt die Lieferadresse in die Liste der Lieferadressen des Kunden ein. Dann kehrt es zur Seite „Bestellen“ zurück. Falls der Kunde keine Adressdaten angegeben hat, kehrt das System zur Seite „Bestellen“ zurück, ohne Daten zu speichern.

Wenn der Kunde eine Lieferadresse auswählt und anschließend auf „Ausgewählte Adresse ändern“ klickt, öffnet das System die Seite „Adressdaten“. Dort zeigt es die Daten des Adressobjekts, das der Kunde ausgewählt hat, noch einmal im Editiermodus an. Der Kunde ändert die betreffenden Daten und klickt auf „Daten abschicken“. Das System prüft daraufhin so weit wie möglich die Korrektheit der Adressdaten. Dann aktualisiert es das Adress-Objekt mit den geänderten Daten und kehrt zur Seite „Bestellen“ zurück.

Wenn der Kunde eine Adresse markiert und auf die Schaltfläche „Ausgewählte Adresse löschen“ klickt, löscht das System die ausgewählte Lieferadresse im Kunden-Objekt. Dann zeigt es erneut die Seite „Bestellen“ mit den aktuellen Lieferadressen an.

Wenn das System keine Rechnungsanschrift finden kann und der Kunde bereits mindestens dreimal erfolgreich per Kreditkarte bezahlt hat, weist das System den Kunden auf der Seite „Zahlungsweise“ darauf hin, dass er auch per Rechnung bestellen darf und dass er dafür jetzt per „Zahlungsdaten ändern“ Rechnungsdaten anlegen kann.

Wenn das System im Kunden-Objekt keine Kreditkartendaten finden kann, öffnet es die Seite „Zahlungsdaten anlegen“. Es zeigt dort im Abschnitt für die Kreditkartendaten leere Felder an, in denen der Kunde die Daten seiner Kreditkarte einträgt. Der Kunde klickt dann auf „Daten abschicken“. Das System validiert nun die Kreditkartendaten, aktualisiert die entsprechenden Attribute des Kunden-Objekts und speichert es ab. Anschließend wechselt es wieder auf die Seite „Zahlungsweise“.

Wenn der Kunde auf „Zahlungsdaten ändern“ klickt, öffnet das System die Seite „Zahlungsdaten anlegen“. Es zeigt nun die Kreditkartendaten des Kunden im entsprechenden Abschnitt an. Wenn bereits eine Rechnungsanschrift vorliegt, zeigt das System diese ebenfalls im vorgesehenen Abschnitt an. Wenn keine Rechnungsdaten gespeichert sind, der Kunde aber schon mindestens dreimal erfolgreich per Kreditkarte bezahlt hat, stellt das System die Felder für die Rechnungsbestellung leer dar, so dass der Kunde Daten eintragen kann. Der Kunde trägt nun die gewünschten Daten ein oder ändert sie und klickt anschließend auf „Daten abschicken“. Das System prüft dann die Kreditkartendaten und die Rechnungsanschrift soweit wie möglich auf Konsistenz, aktualisiert die entsprechenden Attribute des Kunden-Objekts und speichert es ab.

Wenn das System bei der Validierung der Adress- oder Kreditkartendaten Fehler feststellt, stellt es die betreffende Seite erneut dar, wobei es darauf hinweist, dass es Fehler bei der Dateneingabe gab.

⁴Das System müsste auch noch die Ausführung der Bestellung in Auftrag geben. Die internen Abläufe des Shops bleiben in dieser Arbeit aber unberücksichtigt.

5. Anwendungsfälle

Es markiert die Felder auf der Seite, die die fehlerhaften Daten enthalten, und fordert den Kunden zur Neueingabe der Daten auf.

Wenn das System feststellt, dass der Kunde keine Lieferadresse und Versandart ausgewählt hat, fordert es den Kunden auf, eine entsprechende Auswahl zu treffen.

Wenn das System feststellt, dass der Kunde keine Zahlungsweise ausgewählt hat, fordert es den Kunden auf, eine Auswahl zu treffen.

Wenn der Kunde zu irgendeiner Zeit die Schaltfläche „Bestellung abberechnen“ drückt, verwirft das System das Objekt Bestellung und kehrt zum aufrufenden Anwendungsfall zurück.

Zerlegung des Anwendungsfalls

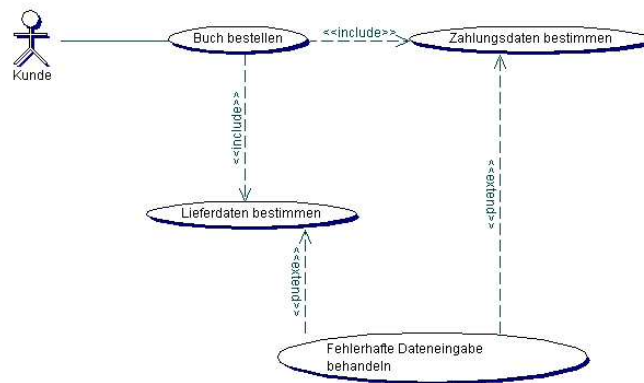


Abbildung 5.6.: Buchbestellung

Der dargelegte Anwendungsfall beschreibt den vollständigen Ablauf einer Bestellung mit den notwendigen Alternativabläufen zum Anlegen von Adress- und Zahlungsdaten. Allerdings erstreckt er sich mit allen Alternativabläufen etwa über eineinhalb Seiten. Ein so umfangreicher Anwendungsfall ist in den nachfolgenden Modellierungsschritten nicht mehr gut handhabbar.

Daher muss an dieser Stelle über eine sinnvolle Zerlegung nachgedacht werden. Die geeignete Schnittstelle für eine Zerlegung sind hier die Orte, an denen das System neue Seiten anzeigt, nämlich das Anlegen von Lieferadressen und Zahlungsdaten. Es werden also zwei neue Anwendungsfälle eingeführt, „Lieferdaten bestimmen“ und „Zahlungsdaten bestimmen“.

Im Anwendungsfall „Buchbestellung durchführen“ sind diese beiden Anwendungsfälle als Alternativabläufe enthalten. Sie werden z.B. aufgerufen, wenn das System keine Lieferadressen finden kann oder der Benutzer die Zahlungsdaten ändern will. Für sich gesehen sind beide Anwendungsfälle aber so umfangreich, dass eine Abspaltung ohne weiteres zu rechtfertigen ist. Da es sich ursprünglich um Alternativabläufe handelte, werden die neuen Anwendungsfälle nicht in jedem Fall aufgerufen. Der anzuwendende Mechanismus ist also „extend“, und zwar erweitern die neuen Anwendungsfälle den ursprünglichen (siehe Abbildung 5.6).

In beiden kommt außerdem die Prüfung von Adressdaten vor. Auch diese könnte man ausgliedern, weil ja beide Anwendungsfälle („Lieferdaten bestimmen“ und „Zahlungsdaten bestimmen“) an die-

ser Stelle ein gemeinsames Verhalten aufweisen. Eine weitere Aufgliederung an dieser Stelle würde jedoch das Modell weiter zerfasern und keinen wirklichen Vorteil bringen. Man könnte in einem Anwendungsfall „Adressdaten validieren“ lediglich genau definieren, welche Art der Prüfung vorgenommen werden soll, und was zu geschehen hat, wenn die Validierung fehlschlägt. Dieser Anwendungsfall wäre aber zu klein. Er hätte wohl im zugehörigen Stabilitätsdiagramm eine Anzahl von Controls, die deutlich unter fünf liegen⁵ würde. Außerdem sind die Anwendungsfälle nur teilweise identisch, da in „Zahlungsdaten bestimmen“ auch die Kreditkartendaten verifiziert werden müssen.

Interessanter ist es, das Systemverhalten bei Fehleingaben auszugliedern (Anwendungsfall „Fehlerhafte Dateneingabe behandeln“). Dieser Anwendungsfall wird von den beiden eben beschriebenen aufgerufen, wenn die Validierung der Adressdaten fehlschlägt. Er zeigt dem Benutzer einen Hinweis auf fehlerhafte Eingaben und fordert ihn zur Neueingabe der Daten auf. Da der Anwendungsfall ein Verhalten beschreibt, was mehreren anderen Anwendungsfällen gemein ist, ist es legitim ihn auszugliedern. Außerdem könnte er im Hinblick auf eine spätere Erweiterung des Modells noch an Wichtigkeit gewinnen. Im Moment sind nicht viele Stellen im System enthalten, an denen der Benutzer größere Eingaben vornimmt. Dies liegt hauptsächlich daran, dass der interne Teil des Buchshops nicht modelliert wird. Wenn man diese Teile des Modells später hinzufügt, wird man diesen Anwendungsfall wieder verwenden wollen.

Abbildung 5.6 zeigt die Zerlegung des Anwendungsfalls „Buchbestellung durchführen“ aus Abbildung 5.5 in die soeben diskutierten neuen Anwendungsfälle.

5.6.2. Buchrezension und Bewertung

Text des Benutzerhandbuchs (siehe Anhang A.2):

Rezensionen schreiben

Wenn Sie in der Detailansicht auf den Link „Rezension schreiben“ klicken, gelangen Sie auf die Seite „Rezension“. Im oberen Teil des Hauptanzeigebereichs wird - zur Erinnerung - noch einmal der Titel des Buches angezeigt, das Sie rezensieren wollen. Im unteren Teil können Sie den Text Ihrer Rezension eingeben. Wenn Sie fertig sind, klicken Sie einfach auf „Rezension abschicken“. Bitte beachten Sie, dass es nicht möglich ist, ein Buch zu rezensieren, ohne es auch zu bewerten. Daher öffnet das System nun die Seite „Bewertung“, auf der Sie das Buch bewerten können (siehe unten). Nachdem Sie die Bewertung durchgeführt haben, zeigt das System wieder die Detailansicht des Buches an, das Sie rezensiert haben. Dort wird unterhalb der Detailinformationen über das Buch die Rezension angezeigt, die Sie soeben verfasst haben. Wenn Ihnen beim Lesen auffällt, dass Sie noch etwas korrigieren möchten, klicken Sie einfach auf den Link „Zurück zur Rezension“ unten links unter dem Rezensionstext und Sie gelangen auf die Seite „Rezension“ zurück, wo Sie ihren Text noch einmal überarbeiten und neu bewerten können.

Bücher bewerten

Zum Bewerten von Büchern steht Ihnen die Seite „Bewertung“ zur Verfügung. Diese öffnen Sie entweder von der Detailansicht des Buches mit dem Link „Buch bewerten“ oder das System präsentiert Ihnen die Seite automatisch, wenn Sie eine Rezension eintragen (s.o.). Geben Sie einfach eine Bewertung ein, indem Sie den Radiobutton anklicken, der

⁵Rosenberg und Scott nennen in [RoSc99, S. 68] eine Anzahl von Controls von durchschnittlich 5 bis 10 für einen Anwendungsfall beherrschbarer Größe.

5. Anwendungsfälle

Ihrer Meinung nach die Qualität des Buches am ehesten beschreibt. Null ist am Schlechtesten, fünf am Besten. Dann bestätigen Sie Ihre Auswahl durch Drücken der Schaltfläche „Absenden“. Das System berechnet nun einen neuen Durchschnitt aus allen Kundenbewertungen und zeigt Ihnen die aktualisierten Informationen in der Detailansicht an. Dort wird der Durchschnitt aller Kundenbewertungen in „Sternen“ ausgedrückt - ebenfalls von 0 bis 5.

Text der Anwendungsfälle:

Buchrezension durchführen

Standardablauf Der Kunde klickt auf der Seite „Produktdetails“ auf „Rezension schreiben“.

Das System stellt sicher, dass der Kunde angemeldet ist. Dann durchsucht es die Liste der Rezensionen zum betreffenden Buch und stellt sicher, dass es noch keine Rezension gibt, die den Kunden als Autor hat. Der Kunde gibt im vorgesehenen Feld für den Rezensionstext einen Text ein. Dann klickt er auf „Rezension abschicken“. Damit der Kunde das Buch auf jeden Fall auch bewertet, ruft das System den Anwendungsfall „Buch bewerten“ auf.

Das System fügt nun der Liste der Rezensionen für dieses Buch ein neues Rezensions-Objekt mit dem soeben eingegebenen Text und der Kennung des Kunden als Autor hinzu. Anschließend kehrt es auf die Seite „Produktdetails“ zurück, wobei es die Rezension des Kunden als erste Rezension anzeigt. Außerdem erzeugt das System auf dieser Seite einen Link, mit dem der Kunde den Anwendungsfall „Rezension überarbeiten“ aufrufen kann.

Alternativabläufe Wenn der Kunde nicht angemeldet ist, ruft das System zunächst den Anwendungsfall „Anmelden“ auf.

Wenn das System in der Liste der Rezensionen zu diesem Buch ein Rezensions-Objekt findet, das den Kunden als Autor hat, wählt es dieses aus und zeigt auf der Seite „Rezension“ dessen Text im Feld für den Rezensionstext an. Der Kunde ändert den Text der Rezension und klickt auf „Rezension abschicken“. Das System ruft nun den Anwendungsfall „Buch bewerten“ auf. Dann aktualisiert es den Rezensionstext im entsprechenden Rezensions-Objekt. Danach kehrt das System zur Seite „Produktdetails“ zurück, wo es diese Rezension als erste Rezension anzeigt. Außerdem zeigt es ein Navigationselement an, mit dem der Kunde den Anwendungsfall „Rezension überarbeiten“ erneut aufrufen kann.

Buch bewerten

Standardablauf Das System öffnet die Seite „Buch bewerten“. Dort wählt der Kunde aus der Liste möglicher Buchbewertungen eine Option aus und klickt auf „Absenden“. Das System stellt daraufhin sicher, dass wirklich eine Bewertung ausgewählt wurde. Anschließend errechnet es aus den bisher abgegebenen Bewertungen für das Buch und der soeben eingegebenen einen neuen Bewertungsdurchschnitt und aktualisiert das Attribut „Bewertung“ im betreffenden Buch-Objekt. Danach kehrt das System zum aufrufenden Anwendungsfall zurück.

Alternativablauf Wenn der Kunde keine Bewertung aus der Liste auswählt, bevor er auf „Absenden“ klickt, fordert das System ihn auf, eine Bewertung abzugeben.

Verfeinerung von „Buchrezension durchführen“

Betrachtet man den Anwendungsfall „Buchrezension durchführen“, so stellt man fest, dass sich der Standardablauf und der Alternativablauf sehr stark ähneln. In beiden prüft das System, ob der Kunde angemeldet ist, der Kunde gibt einen Rezensionstext ein oder ändert ihn und bewertet anschließend das Buch. Der wichtige Unterschied liegt allerdings im Verhalten des Systems. Dieses durchsucht nämlich die Rezensionsliste zum entsprechenden Buch nach einer vorhandenen Rezension des Kunden und entscheidet dann, ob eine neue Rezension angelegt wird oder ob eine vorhandene Rezension überarbeitet wird.

Es handelt sich also eigentlich um zwei verschiedene Anwendungsfälle, nämlich „Rezension erstellen“ und „Rezension überarbeiten“, die nur deshalb im gleichen Anwendungsfall untergekommen sind, weil sie mit derselben Bildschirmansicht bearbeitet werden.

Trennt man diese Anwendungsfälle wieder, muss man aber den einleitenden Teil (Prüfung der Anmeldung, Suchen eines Rezensions-Objektes) verdoppeln. Darum bietet es sich an, diesen Teil in einen dritten Anwendungsfall „Rezension vorbereiten“ auszulagern, der den beiden anderen vorangestellt ist und der - je nachdem was die Suche nach einer vorhandenen Rezension für ein Ergebnis liefert - den einen oder anderen Anwendungsfall automatisch aufruft.

Im Ergebnis führt diese Aufgliederung zu einer extend-Beziehung von den nachgelagerten Anwendungsfällen zum vorbereitenden Anwendungsfall, sowie zu include-Beziehungen von den nachgelagerten Anwendungsfällen zu „Buch bewerten“. Mit letzterer stellt das System sicher, dass die Geschäftslogik „eine Rezension ergänzt immer nur eine Bewertung“ stets eingehalten wird.

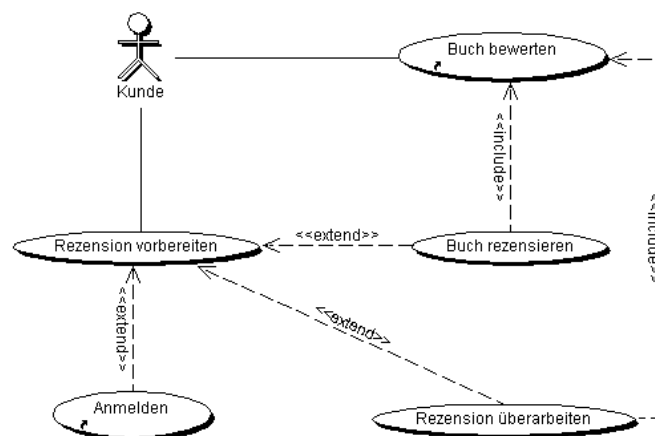


Abbildung 5.7.: Buchrezension

Abbildung 5.7 zeigt die Verfeinerung des Anwendungsfalls „Buchrezension durchführen“ aus Abbildung 5.5 in die neuen Anwendungsfälle „Rezension vorbereiten“, „Buch rezensieren“ und „Rezension überarbeiten“. Zur Verdeutlichung der genauen Zusammenhänge sind auch die Anwendungsfälle „Anmelden“ und „Buch bewerten“ aus Abbildung 5.5 noch einmal mit aufgeführt.

5. Anwendungsfälle

5.6.3. Produktübersichten

Im Prototyp kommen Produktübersichten an drei Stellen vor, nämlich im Produktkatalog im Anwendungsfall „Buchkatalog ansehen“, bei der Präsentation von Suchergebnissen im Anwendungsfall „Buchkatalog durchsuchen“ und auf der Startseite im Anwendungsfall „Neuheiten und Empfehlungen ansehen“. Die Entwicklung und Zerlegung eines Anwendungsfalls soll hier exemplarisch am Beispiel der Startseite demonstriert werden. Auch hier sei auf die Anhänge A und B für eine vollständige Referenz der Abschnitte aus dem Benutzerhandbuch und der zugehörigen Anwendungsfälle verwiesen. Text des Benutzerhandbuchs (siehe Anhang A.1):

Startseite (Azamon)

Wenn Sie den Shop betreten, sehen Sie als erstes diese Seite. Hier können Sie auf einen Blick unsere Neuheiten und Empfehlungen sehen. Falls Sie noch nicht wissen, was Sie kaufen sollen, können Sie sich hier einige Anregungen holen. Klicken Sie auf „Details“ unter jeder Buchbeschreibung, um in die Detailansicht des Katalogs zu wechseln und mehr über diesen Artikel zu erfahren.

Über den Link „Anmelden“ in der Titelzeile der Seite können Sie sich am System anmelden. Dadurch vermeiden Sie, dass Sie sich später „zwischendurch“ anmelden müssen, wenn Sie eine Bestellung durchführen oder ein Buch rezensieren wollen.

Text des Anwendungsfalls:

Neuheiten und Empfehlungen ansehen

Standardablauf Das System zeigt die Startseite an. Es sucht aus der Liste der Neuheiten und aus der Liste der Empfehlungen alle eingetragenen Buch-Objekte heraus und ruft für jedes gefundene Buch die wichtigsten Informationen über das Buch sowie dessen Verfügbarkeit aus und zeigt sie zusammen mit einem Bild des Buches an. Außerdem erzeugt das System für jedes Buch ein Navigationselement „Details“, mit dem der Kunde den Anwendungsfall „Produktinformationen ansehen“ aufrufen kann.

Anschließend übergibt das System die Kontrolle wieder an den aufrufenden Anwendungsfall.

Alternativabläufe Wenn der Kunde auf „Anmelden“ klickt, ruft das System den Anwendungsfall „Anmelden“ auf.

Wenn der Kunde auf „Details“ klickt, übergibt das System die Kontrolle an den Anwendungsfall „Produktinformationen ansehen“.

Wenn das System keine Neuheiten oder Empfehlungen finden kann, unterdrückt es die entsprechende Rubrik. Wenn es in beiden Kategorien keine Bücher finden kann, zeigt es statt dessen einen Standardtext an, der für den Buchshop wirbt.

Ausgliedern gemeinsamen Verhaltens

Interessant an den drei Anwendungsfällen, die die Produktübersichten betreffen, ist der Teil, in dem die Kurzinformationen über das Buch mit einem Link „Details“ und einem Bild angezeigt werden.

Dieser Teil kommt nämlich in allen drei Anwendungsfällen vor. Diese Gleichheit dient dazu, identische Darstellungen der Produktübersichten in allen betroffenen Bildschirmansichten zu haben. Zudem soll es von jeder dieser Ansichten aus möglich sein, durch einen Klick in der Kurzbeschreibung des Buches in die Detailansicht zu wechseln.

Daher wird dieser Teil aus den Anwendungsfällen herausgezogen und im neuen Anwendungsfall „Kurzinformat anzeigen“ festgehalten. Dem Anwendungsfall wird eine Liste der anzuzeigenden Bücher übergeben, deren Kurzinformat dann in die Ansicht, die zum aufrufenden Anwendungsfall gehört, eingefügt wird.

An diesem Beispiel lässt sich gut zeigen, wie man durch geschickte Modellierung ein include in ein extend oder umgekehrt verwandeln kann. Es ist nämlich denkbar, dass überhaupt keine Bücher vorliegen, die angezeigt werden können, die übergebene Bücherliste somit leer ist. In diesem Fall (man merke, dass es sich hier um eine Bedingung handelt) gibt es auch keine Kurzinformat anzuzeigen. Statt dessen wird ein angemessener Alternativtext angezeigt. Nimmt man den Ablauf „Alternativtext anzeigen“ in „Kurzinformat anzeigen“ hinein, muss im Fehlerfall „Kurzinformat anzeigen“ aufgerufen werden und „Kurzinformat anzeigen“ bearbeitet den Fehler selbst. Damit wird der include-Mechanismus angewendet, da in *jedem Fall* „Kurzinformat anzeigen“ aufgerufen wird.

Wird hingegen der Alternativtext im *aufrufenden* Anwendungsfall angezeigt, so muss „Kurzinformat anzeigen“ im Fehlerfall nicht aufgerufen werden. Bei so einem optionalen Aufruf kommt der extend-Mechanismus zur Anwendung.

5.7. Anwendungsfallpakete

Als letzten Schritt dieser Analysephase sieht der ICONIX Prozess die Gruppierung von Anwendungsfällen vor. Diese werden in Pakete aufgeteilt und in Package-Diagrammen dargestellt. Im hier entwickelten Modell lassen sich vier Anwendungsfallpakete unterscheiden:

- Buchbestellung
- Buchkatalog
- Buchrezension
- Systemanmeldung und Kontaktinformationen

In das Paket Buchbestellung (Abbildung 5.8) kommen alle Anwendungsfälle, die im weitesten Sinne mit der Buchbestellung zu tun haben. Zunächst sind dies natürlich die oben hergeleiteten Anwendungsfälle, die den Bestellvorgang beschreiben. Außerdem ist dort das Ansehen und Stornieren von Bestellungen enthalten, sowie die Anwendungsfälle, die mit dem Warenkorb zu tun haben. Das Füllen und Editieren des Warenkorbs bereitet nämlich eine Bestellung vor.

In das Paket Buchkatalog (Abbildung 5.9) gehören alle Anwendungsfälle, die sich um die verschiedenen (unter Umständen gefilterten) Sichten auf den Buchkatalog ranken. Dies sind die drei Anwendungsfälle „Buchkatalog ansehen“, „Buchkatalog durchsuchen“ und „Neuheiten und Empfehlungen ansehen“. Außerdem gehören die Darstellung der Produktübersicht und der Produktdetails dazu.

5. Anwendungsfälle

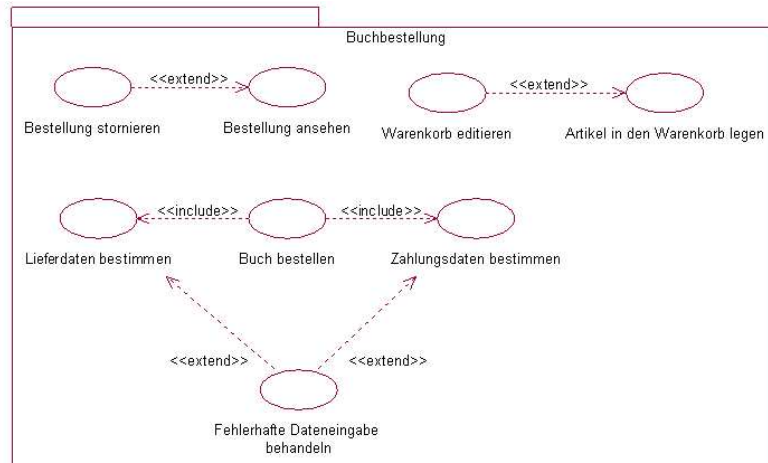


Abbildung 5.8.: Paket Buchbestellung

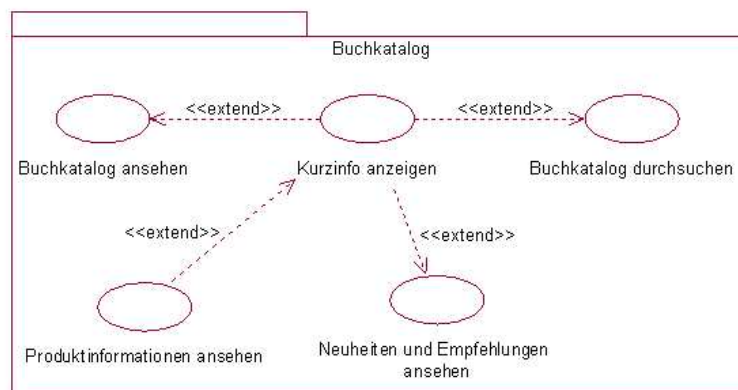


Abbildung 5.9.: Paket Buchkatalog

5.7. Anwendungsfallpakete

Zum Paket Buchrezension (Abbildung 5.10) gehören die oben hergeleiteten Anwendungsfälle, die aus „Buchrezension durchführen“ entstanden sind, sowie die Bewertung von Büchern, da diese immer zu einer Buchrezension dazugehört und dem Bereich „Kommentierung von Büchern“ hinzugerechnet werden kann.

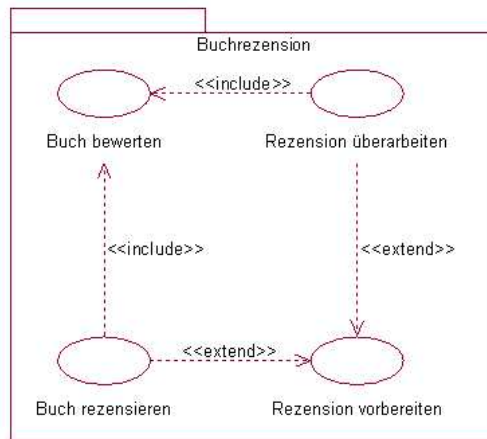


Abbildung 5.10.: Paket Buchrezension

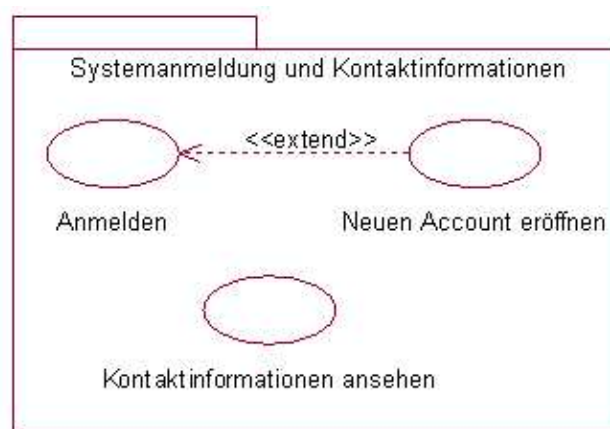


Abbildung 5.11.: Paket Systemanmeldung und Kontaktinformationen

Das letzte Paket (Abbildung 5.11) enthält schließlich die Anwendungsfälle, die zur Systemanmeldung gehören, also „Anmelden“ und „Neuen Account anlegen“. Der letzte Anwendungsfall wird dem Paket einfach hinzugefügt, da es nicht viel Sinn macht, ein Paket mit nur einem Anwendungsfall zu haben. Der Paketname wird entsprechend erweitert.

5.8. Vergleich mit dem Rosenberg-Modell

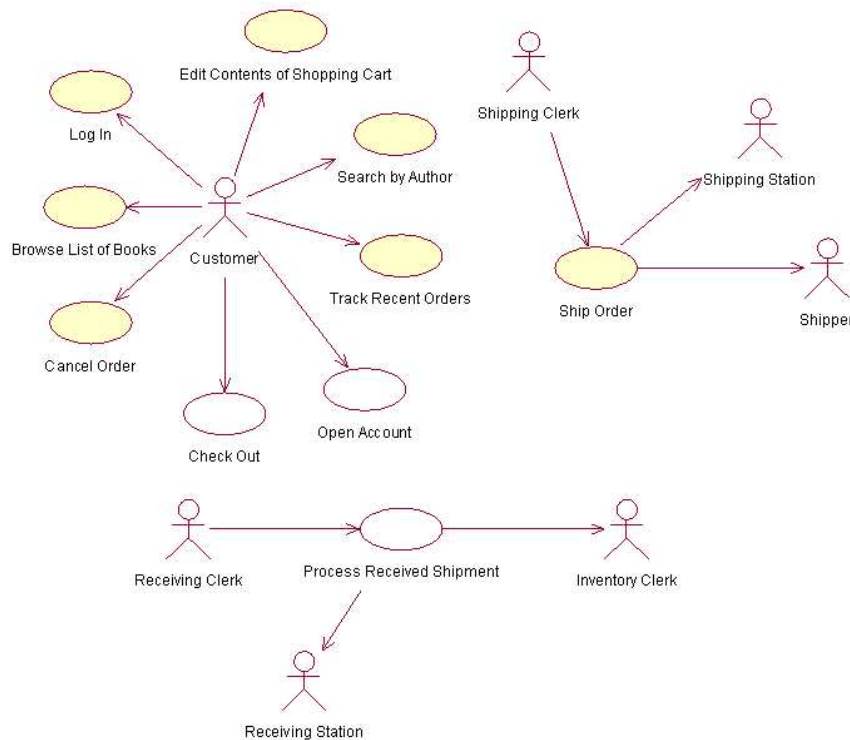


Abbildung 5.12.: Anwendungsfälle von Rosenberg und Scott

Betrachtet man das Anwendungsfallmodell von Rosenberg und Scott, so fällt zunächst auf, dass die Anwendungsfälle auf einer imaginären Benutzerschnittstelle basieren. Aufgrund der besonderen Vorgehensweise des ICONIX Prozesses, die auf der Benutzerschnittstelle aufbaut, kann man das Modell schlicht nicht verstehen, wenn man nicht weiß, wie die zugehörige Benutzerschnittstelle aussieht. Dies verschweigen die Autoren jedoch, was den geeigneten Leser zum Raten nötigt.

Weiterhin betonen die Autoren, dass es bei der Anwendungsfallanalyse wichtig sei, Gemeinsamkeiten im Verhalten von Anwendungsfällen aufzudecken und mit entsprechenden Mechanismen zu modellieren. Im Rosenberg-Modell ist jedoch keine einzige Beziehung zwischen Anwendungsfällen enthalten. Betrachtet man die Anwendungsfälle, die den Kunden betreffen (nur diese werden ja im hier entwickelten Modell betrachtet), so stellt man fest, dass diese alle auch im hier vorgestellten Modell vorhanden sind. Das ist nicht weiter verwunderlich, denn um einigermaßen vergleichbare Ergebnisse zu gewinnen, sah sich der Autor dieser Arbeit genötigt, die Benutzerschnittstelle soweit wie möglich nach den Rosenberg'schen Anwendungsfällen zu modellieren.

Rosenberg deckt allerdings mit seinen Anwendungsfällen nicht alle funktionalen Anforderungen aus der Problemstellung ab. Insbesondere fehlt die Umsetzung folgender Anforderungen:

- Kundenrezensionen von Büchern

- Kundenbewertungen von Büchern
- Anzeigen von Verfügbarkeitsinformationen (Link zur Lagerverwaltung)
- Anzeigen des Lieferstatus (Link zur Versandabteilung)

Hier und da fehlen außerdem Anwendungsfälle, auf die andere Anwendungsfälle Bezug nehmen. Zum Beispiel ist im Anwendungsfall „Browse List Of Books“ die Rede von einem Anwendungsfall „Display Book Details“. Dieser ist jedoch nicht modelliert worden. Im Anwendungsfall „Open Account“ ist die Rede von einer Startseite, zu der der Benutzer zurück geleitet wird. Die Funktion der Startseite ist aber in keinem Anwendungsfall beschrieben.

Schließlich ist noch festzustellen, dass die meisten Anwendungsfälle im Rosenberg-Modell nur sehr oberflächlich gestaltet sind. Sie lassen eine ganze Menge Details vermissen und haben eher den Charakter von Analyse-Anwendungsfällen als von Design-Anwendungsfällen. Man hat hier wohl ganz einfach nicht weit genug gedacht und die Analyse mittendrin abgebrochen. Dadurch ist auch zu erklären, dass im hier entwickelten Modell die Zahl der Anwendungsfälle mehr als doppelt so groß ist wie im Rosenberg-Modell.

Aus den oben genannten Gründen (Reverse Engineering der Benutzerschnittstelle) unterscheiden sich die acht Anwendungsfälle, die in beiden Modellen vorkommen, inhaltlich nicht sehr. Die Anwendungsfälle in diesem Modell sind jedoch genauer ausformuliert und auch - soweit möglich und sinnvoll - zerlegt worden und es sind mehr Alternativabläufe bedacht worden, insbesondere in den Bereichen „Buchbestellung“, „Rezension“ und „Produktdetails“, wie im vorigen Abschnitt ja diskutiert wurde. Darüber hinaus wurde mehr Wert darauf gelegt, die Anwendungsfälle im Kontext des Informationsmodells zu betrachten, so wie Rosenberg und Scott dies in ihren Werken auch empfehlen.

5. Anwendungsfälle

6. Stabilitätsanalyse

Grundsätzliches

Einer der wichtigsten Teile des ICONIX-Prozesses ist die Stabilitätsanalyse. Während sich die vorangehenden Schritte damit befassen, herauszufinden, *welche* Probleme zu lösen sind, und das nachfolgende Detaildesign sich damit befasst, herauszufinden, *wie* die Probleme zu lösen sind, bildet die Stabilitätsanalyse die Brücke zwischen beiden Welten. Zur Durchführung der Stabilitätsanalyse bedient man sich stereotypierter Klassendiagramme, die Stabilitätsdiagramme genannt werden. Diese Diagramme folgen dem Paradigma, dass Präsentation und Logik einer Anwendung stets zu trennen sind. Die Grundlage dieser Stereotypen bildet das in [Gam96] kurz erläuterte und sehr weit verbreitete MVC-Architekturmuster.

Die MVC-Architektur

Eine umfassendere Erläuterung zum MVC-Muster, die auch Web-Applikationen mit berücksichtigt, findet sich in [SiStJa, S. 348ff.]:

„Die Model-View-Controller-Architektur ist ein weit verbreiteter architektonischer Ansatz für interaktive Applikationen. Er verteilt Funktionalität zwischen den Objekten, die mit der Präsentation und Verarbeitung von Daten zu tun haben, um den Grad der Kopplung zwischen den Objekten zu minimieren. Die Architektur übersetzt die klassischen Aufgaben Eingabe, Verarbeitung und Output auf das Modell grafischer Interaktion mit dem Benutzer. Außerdem fügt sie sich gut in den Bereich von internetbasierten Multi-Tier-Enterprise-Applikationen ein.

Die MVC-Architektur unterteilt Applikationen in die drei Schichten Model, View und Controller und entkoppelt deren jeweilige Verantwortlichkeiten. Jede Schicht handhabt bestimmte Aufgaben und hat bestimmte Verantwortlichkeiten bezüglich der anderen Bereiche.

Ein Model repräsentiert die Geschäftsdaten und die Geschäftslogik oder Operationen, die Zugriff und Veränderung der Geschäftsdaten bestimmen. Oft dient das Model als softwaremäßige Näherung an die Geschäftsdaten. Das Model benachrichtigt die Views wenn es sich verändert und stellt den Views die Möglichkeit zur Verfügung, das Model nach seinem Zustand zu befragen. Außerdem bietet es dem Controller die Möglichkeit, auf Applikationsfunktionalitäten zuzugreifen, die im Model gekapselt sind.

Ein View stellt den Inhalt eines Models dar. Es greift auf die Daten des Models zu und spezifiziert, wie die Daten dargestellt werden sollen. Es aktualisiert die Datenpräsentation, wenn das Model sich ändert. Außerdem leitet es die Eingaben an den Controller weiter.

6. Stabilitätsanalyse

Ein Controller definiert das Verhalten der Anwendung. Er verarbeitet Benutzeranfragen und wählt die richtigen Views für die Präsentation aus. Er interpretiert die Benutzereingaben und übersetzt sie in Aktionen, die vom Model ausgeführt werden müssen. In einem autark arbeitenden GUI-Client sind dies zum Beispiel Klicks auf Schaltflächen oder Auswahlen in einem Menü. In einer Internetanwendung handelt es sich dabei um HTTP-GET-Anfragen und HTTP-POST-Anfragen an den Web-Tier. Anhand der Benutzeraktionen und der Ergebnisse der Model-Operationen wählt ein Controller die nächste Ansicht¹ aus, die angezeigt werden soll. Eine Applikation hat üblicherweise einen Controller für einen Satz verwandter Funktionen. Einige Anwendungen benutzen auch separate Controller für verschiedene Typen von Clients, weil sich die Interaktionen mit den Ansichten und Auswahlen oft zwischen den verschiedenen Typen von Clients unterscheiden.“

Wie bereits erwähnt, ist das für die Stabilitätsanalyse eingesetzte Muster Entity-Boundary-Control eng an MVC angelehnt, wobei mehr oder weniger das Entity dem Model, der Controller dem Control und das Boundary-Objekt der View entspricht. Allerdings ist die Verteilung der Verantwortlichkeiten etwas flexibler als es oben für die MVC-Architektur beschrieben wurde. Dennoch wird hier eine Vorabentscheidung für dieses Architekturmuster getroffen, was vielleicht erklärt, warum das ICONIX-Vorgehensmodell keine explizite Architekturfundungsphase enthält. Doch mehr dazu später.

Vorgehensweise

Die Stabilitätsanalyse wird sehr nah am Text der Anwendungsfälle durchgeführt. Man analysiert jeweils den Text eines Anwendungsfalls und findet heraus, was für Objekte man benötigt, um den Anwendungsfall abzarbeiten. Die gefundenen Objekte unterteilt man in Boundary-Objekte, Entity-Objekte und Control-Objekte. Die Boundaries sind diejenigen Objekte, mit denen der Benutzer interagiert, also typischerweise Seiten oder Elemente der Benutzeroberfläche. Die Entity-Objekte sind normalerweise Objekte aus dem Informationsmodell. Häufig müssen diese auch persistiert werden und haben unter Umständen Äquivalente im Datenbankmodell der Anwendung. Die Controller schließlich sind die Objekte, welche die Boundaries mit den Entities verbinden. Sie beschreiben also das Verhalten der Anwendung.

Um sicherzustellen, dass die Präsentation auch wirklich von der Anwendungslogik getrennt ist, gilt es, bei der Stabilitätsanalyse folgende Kommunikationsregeln einzuhalten (nach [RoSc99, S. 69]):

1. Akteure sprechen nur mit Boundaries.
2. Boundaries können mit Controllern und Akteuren sprechen.
3. Entity-Objekte können nur mit Controllern sprechen.
4. Controller können sowohl mit Boundary-Objekten als auch mit Controllern sprechen, aber nicht mit Akteuren.

In der Essenz bedeuten die Kommunikationsregeln folgendes:

- Der Benutzer kann die Anwendung nur über die Oberfläche bedienen.

¹ engl. View. Der Begriff „View“ kommt auch in dieser Bedeutung in der unten entwickelten Architektur als Klasse vor.

- Die Geschäftslogik befindet sich in den Controllern *hinter* der Oberfläche.
- Die Oberfläche hat keinen direkten Zugriff auf die Daten der Anwendung.

Die Einhaltung der Kommunikationsregeln hilft Modellierungsfehler zu vermeiden. Wenn der Designer feststellt, dass Kommunikationsregeln verletzt wurden, ist es sicher, dass die Anwendung nicht funktionieren kann oder dass wichtige Entwurfsregeln verletzt wurden. Daher ist bei der Stabilitätsanalyse auf die Einhaltung dieser Regeln besondere Sorgfalt zu verwenden.

Funktion der Stabilitätsanalyse Die Stabilitätsanalyse erfüllt folgende grundlegende Funktionen:

1. Prüfung der Anwendungsfälle auf Durchführbarkeit und Vollständigkeit,
2. Finden fehlender Objekte und
3. Vorläufiges Design.

Die Abläufe in den Stabilitätsdiagrammen stellen den Ablauf der Anwendungsfälle grafisch dar. Deshalb bietet es sich an dieser Stelle an, noch einmal den Ablauf der Anwendungsfälle zu verifizieren und herauszufinden, ob es mit den zur Verfügung stehenden Objekten möglich ist, die in den Anwendungsfällen geschilderten Abläufe durchzuführen. Gleichzeitig kann eine Überprüfung stattfinden, ob der Anwendungsfall, den man gerade bearbeitet, alle nötigen Alternativabläufe berücksichtigt. Diese Prüfung führt also unter Umständen zur Korrektur von Anwendungsfalltexten.

Da die Anwendungsfälle im Kontext des Informationsmodells geschrieben sein sollen, wird außerdem untersucht, ob alle zur Ausführung nötigen Objekte zur Verfügung stehen und die Objekte über alle nötigen Attribute verfügen. Neu gefundene Objekte und Attribute werden dem Klassenmodell hinzugefügt. Am Ende der Stabilitätsanalyse sollte man nach Aussage von Rosenberg ca. 75-80% der Attribute gefunden haben.

Ausdrücklich ausgeschlossen wird an dieser Stelle jedoch das Hinzufügen von Operationen zu den gefundenen Objekten. Laut Rosenberg sollte man damit nicht beginnen, solange man noch kein vollständiges Bild vom System entwickelt hat. Die Stabilitätsanalyse dient in erster Linie dazu, dieses Bild zu entwickeln. Das Verhalten auf die verschiedenen Objekte zu verteilen ist die Aufgabe, die im Detaildesign gelöst wird. Auch sollte man beachten, dass die Stabilitätsdiagramme „Wegwerf-diagramme“ sind. Hat man erst einmal einen ersten Eindruck gewonnen, wie das Design aussehen könnte, haben diese Diagramme ihren Dienst getan. Sie werden im weiteren Verlauf des Entwurfs nicht weiter gewartet. Rosenberg und Scott schreiben dazu in [RoSc99, S. 76]:

„[...] Die Stabilitätsanalyse ist ein gutes Werkzeug, das hilft, Objekte zu finden, Attribute zuzuweisen und Anwendungsfalltexte auf Richtigkeit und Vollständigkeit zu überprüfen. Wenn aber diese Aufgabe einmal erfüllt ist, macht es keinen Sinn mehr, das Produkt der Arbeit weiter zu warten. Es ist ein Weg zum Ziel, kein Ziel in sich.“

Die folgenden Abschnitte in diesem Kapitel stellen die Stabilitätsanalyse exemplarisch anhand einiger Anwendungsfälle dar, die der Kunde durchlaufen muss, um ein Buch zu bestellen. Die Stabilitätsanalyse gliedert sich in vier Schritte:

6. Stabilitätsanalyse

1. Analyse des Anwendungsfalls auf Vollständigkeit und Durchführbarkeit,
2. Darstellung des Stabilitätsdiagramms,
3. Überarbeiten des Anwendungsfalltextes,
4. Diskussion der Auswirkungen auf das Klassenmodell (neue Objekte und Attribute).

6.1. Buchkatalog ansehen

Analyse

Navigation durch den Produktkatalog Dieser Anwendungsfall beschreibt, wie ein Kunde im Buchkatalog navigieren kann (siehe auch Abschnitt B.4.1). Dabei zeigt das System immer die passenden Unterkategorien der jeweils gewählten Kategorie an. Unklar ist dabei aber noch die Frage, wo das System eigentlich die Subkategorien her nimmt, die es anzeigen soll.

Bevor das System Subkategorien anzeigen kann, muss es sie zuerst suchen. Dazu wird ein neues Objekt „Kategorieliste“ benötigt, das alle Buchkategorien enthält, die im Buchshop eine Rolle spielen. Außerdem muss jede Kategorie Referenzen auf ihre zugehörigen Unterkategorien haben, sowie das Wissen, ob sie eine Top-Level-Kategorie ist.

Die jeweils gerade gültigen Unterkategorien der aktuell gewählten Kategorie müssen in einem Objekt „TmpKategorieliste“ temporär gespeichert werden, damit die Kategorie-Auswahlliste mit dem richtigen Inhalt gefüllt werden kann. Das Objekt kann beim Verlassen der Seite wieder verworfen werden, aber es muss sichergestellt sein, dass beim Mehrbenutzerbetrieb, der bei Internetsystemen die Regel ist, die temporäre Kategorieliste dem richtigen Kunden zugeordnet ist. Dies geschieht am besten über ein Objekt, das den Kunden während seiner ganzen Einkaufssitzung begleitet und ihm eindeutig zugeordnet ist. Ein solches Objekt, das diese und andere Kontextinformationen speichert, könnte zum Beispiel Session genannt werden.

Speichern des aktuellen Auswahlzustands Wenn der Kunde den Anwendungsfall verlässt und später wieder aufruft, stellt das System den Zustand (also die Bildschirmansicht) wieder her, den der Kunde vor dem Verlassen zuletzt gesehen hat. Dieser Vorgang bedarf weiterer Präzisierung. Es müssen dabei folgende Fragen geklärt werden:

- Was charakterisiert eigentlich den zu sichernden Zustand?
- Was muss gespeichert werden?
- Wie soll das Wiederherstellen des Zustands, bei dem der Kunde die Seite verlassen hat, genau funktionieren?

Grundlage für diese Diskussion ist Abbildung A.4. Im Hauptanzeigebereich - das ist der helle innere Bereich der Bildschirmansicht, also rechts vom Hauptmenü und unterhalb der Titelzeile - ist zu sehen, welche Informationen angezeigt werden, wenn der Kunde eine Kategorie mit Büchern ausgewählt hat. Die oberste Zeile enthält die Kategoriehistorie, also alle Kategorien von der Wurzelkategorie „Bücher“ aus, die der Kunde auswählen musste, um zur gerade gezeigten Ansicht zu gelangen. Dort, wo sich auf der Abbildung die Darstellung des Buches befindet, kann alternativ zur Buchansicht die Liste der

möglichen Subkategorien angezeigt werden und ein Hinweistext, dass die gewählte Kategorie keine Bücher enthält. Dies ist dann der Fall, wenn der Kunde sich gerade in einer Kategorie befindet, die keine Bücher enthält, etwa weil es sich nicht um eine Kategorie unterster Stufe handelt.

Um die obere Zeile der Bildschirmansicht wiederherzustellen, muss das System die komplette Kategoriehistorie kennen, für den unteren Teil reicht die Kenntnis der gerade ausgewählten Kategorie.

Ist die Kategorie bekannt, kann das System die Ansicht fast vollständig wiederherstellen, indem es die zugehörigen Subkategorien ermittelt und anzeigt, oder falls es sich um eine Kategorie unterster Stufe handelt, die Bücher, die zu dieser Subkategorie gehören. (Es wird davon ausgegangen, dass nur die Kategorien unterster Stufe Bücher enthalten.) Je nach Implementierung der Kategorie-Hierarchie (einfach oder doppelt verkettete Liste) und der Art, wie die Hierarchie angeordnet ist (z.B. Baum oder Netz) kann es aber zu Problemen kommen, wenn das System versucht, den vollständigen Weg von der Wurzel (Pseudo-Kategorie Bücher) zur gespeicherten Kategorie selbstständig zu finden. Zum Beispiel ist es denkbar, dass der Weg zu einer bestimmten Kategorie nicht eindeutig ist. Also muss im Auswahlzustand die Kategoriehistorie mit gespeichert werden.

Das System muss den Zustand in einem neuen Objekt „Auswahlzustand“ speichern. Konkret handelt es sich dabei um die aktuell ausgewählte Kategorie und eine nach Reihenfolge geordnete Liste der Kategorien, die der Kunde nacheinander auswählen musste, um zur aktuellen Kategorie zu gelangen.

Die genannten Überlegungen führen zu dem in Abbildung 6.1 dargestellten Stabilitätsdiagramm. Der genaue Ablauf ist im überarbeiteten Anwendungsfall nachzulesen, der im folgenden Abschnitt beschrieben wird.

Überarbeiteter Text des Anwendungsfalls

Standardablauf: Das System öffnet die Seite „Katalog“. Es stellt sicher, dass noch kein Auswahlzustand gespeichert ist, durchsucht die Liste der Buchkategorien nach Top-Level-Kategorien und erzeugt ein neues TmpKategorieListe-Objekt. Dies ordnet es der Session des Kunden zu. Dann speichert das System die Top-Level-Kategorien im TmpKategorieListe-Objekt ab und zeigt sie auf der Seite „Katalog“ als Auswahl an.

Der Kunde klickt nun im Buchkatalog auf eine Kategorie. Das System erzeugt ein neues Auswahlzustand-Objekt und speichert den derzeitigen Auswahlzustand (gewählte Kategorie und Pfad von der Wurzel zu dieser Kategorie) darin ab. Dann durchsucht es die Kategorielliste nach den zugehörigen Subkategorien. Danach aktualisiert das System die temporäre Kategorielliste mit den neuen Subkategorien und zeigt diese in der Kategorieauswahl an. Dies wiederholt sich, bis keine weiteren Subkategorien mehr vorhanden sind, so dass das System eine der untersten Kategorien anzeigt. Dann durchsucht das System den Buchkatalog nach den Büchern, die in dieser Kategorie vorhanden sind. Es erzeugt ein Suchergebnis-Objekt, das die Liste der Bücher dieser Kategorie enthält, und ruft für jedes gefundene Buch den Anwendungsfall „Kurzinfo anzeigen“ auf. Die Kurzinfos der Bücher zeigt das System auf der Seite „Katalog“ an.

Alternativabläufe: Wenn das System einen gespeicherten Auswahlzustand findet, versetzt es die Seite „Katalog“ in diesen Zustand zurück. (Dazu liest es die Kategorie und den Pfad zur Kategorie aus, findet die Subkategorien oder Bücher der Kategorie und zeigt alles auf der Seite „Katalog“ an.)

Wenn das System in einer (Sub-)Kategorie keine Bücher finden kann, zeigt es eine entsprechende Fehlermeldung an und fordert den Kunden auf, eine andere Kategorie zu wählen.

6. Stabilitätsanalyse

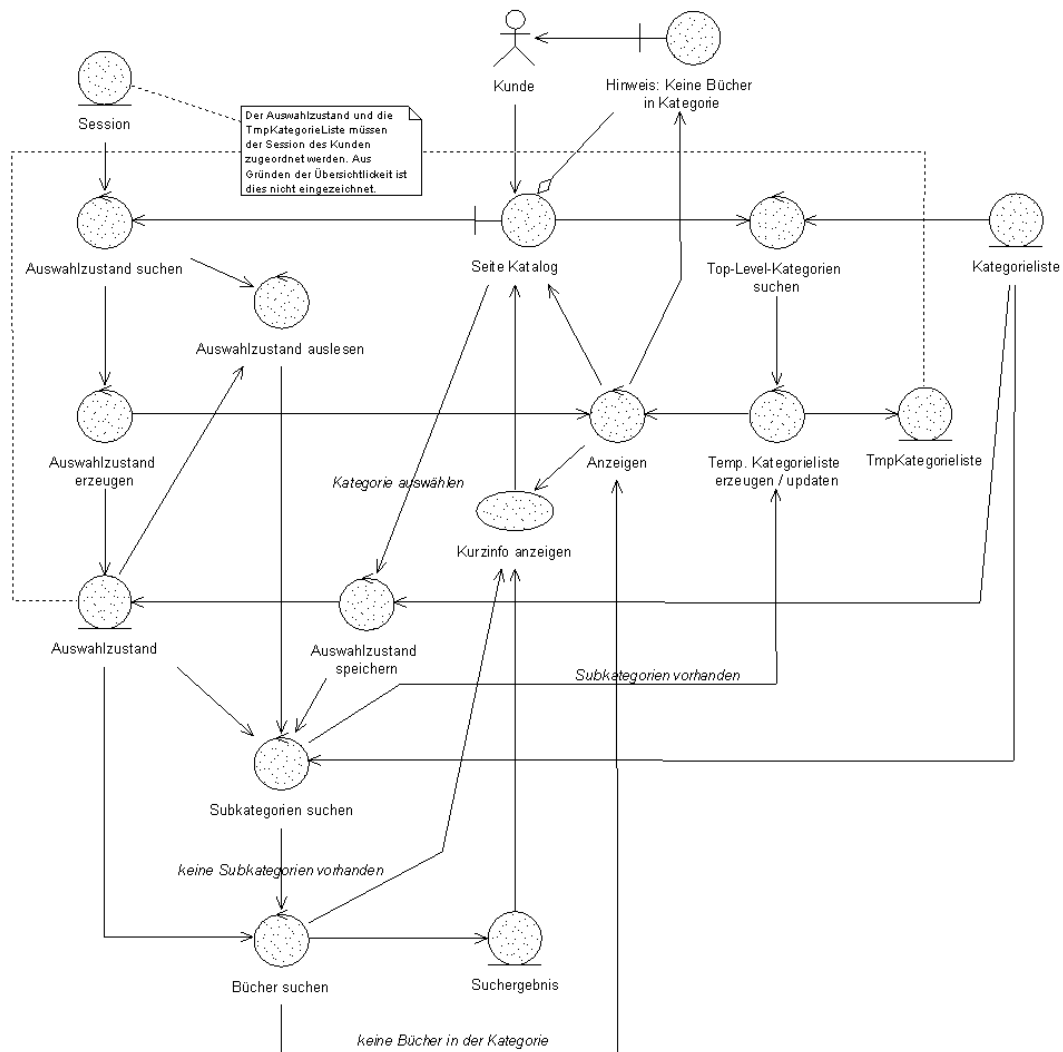


Abbildung 6.1.: Stabilitätsdiagramm Buchkatalog ansehen

Auswirkungen auf das Klassenmodell

- Zum Speichern von Kontextinformationen, die der Kunde benötigt, um sich im Shop zu bewegen, muss ein neues Objekt „Session“ eingeführt werden, in dem diese Informationen dem Kunden eindeutig zugeordnet werden können. Dies schließt Verwechslungen beim Mehrbenutzerbetrieb aus.
- Damit man ein Buch überhaupt einer Kategorie zuordnen kann, benötigt man ein Attribut „Kategorie“ in der Klasse „Buch“.
- Zum Speichern aller Buchkategorien, die im Shop vorkommen und zum Erstellen der temporären Kategorielliste mit den auswählbaren Subkategorien benötigt man eine neue Klasse „Kategorielliste“, sowie eine Klasse „Kategorie“. Diese beiden Klassen bilden einen Baum, wobei die Kategorielliste alle Kategorien enthält und jede Kategorie ihre Unterkategorien kennt und weiß, ob sie eine Top-Level-Kategorie ist.
- Zum Speichern der aktuell ausgewählten Kategorie und des Pfades von der Wurzel der Kategorien zu dieser Kategorie benötigt man eine neue Klasse „Auswahlzustand“, die der Session des Kunden zugeordnet sein muss. Der Auswahlzustand muss die aktuell gewählte Kategorie enthalten und eine geordnete Liste der Kategorien, die nacheinander ausgewählt werden müssen, um zur ausgewählten Kategorie zu gelangen.
- Damit die Auswahlliste für die Kategorien mit Daten gefüllt werden kann, müssen die möglichen Subkategorien der aktuell gewählten Kategorie in einem Objekt „TmpKategorieListe“ gespeichert werden, das eine geordnete Liste von Kategorien darstellt. Dieses wird ebenfalls der Session des Kunden zugeordnet.²

6.2. Warenkorb editieren

Analyse

Dieser Anwendungsfall beschreibt, wie der Kunde den Inhalt des Warenkorbs verändern kann. Im Großen und Ganzen ist der Anwendungsfall korrekt geschildert, er enthält lediglich einige kleine Fehler, die in unklaren Formulierungen und fehlenden Alternativabläufen bestehen. Diese werden im Folgenden dargelegt:

- Das Öffnen des Warenkorbs ist keine relevante Aktion. Was wirklich gemeint ist, ist das Auslesen der Werte aus dem Warenkorb. Der Text muss entsprechend korrigiert werden.
- Das Löschen einer Position des Warenkorbs oder das Setzen einer Stückzahl auf 0 führen zur gleichen Reaktion des Systems. Die Alternativabläufe können also zusammengelegt werden.

²Die Information, wie die Auswahllisten der Bildschirmseiten zu füllen sind, berührt die Präsentationsaspekte des zu bauenden Systems. Die Entscheidung, wo diese Information zu speichern ist (ggf. im für die Präsentation zuständigen Controller) ist aber architektonischer Natur. Sie kann nicht endgültig getroffen werden, solange man noch nicht weiß, welche Technologie für die Präsentationsaspekte des Systems eingesetzt werden soll. Durch die Technologie ergeben sich unter Umständen auch bereits Vorgaben für eine Architektur.

6. Stabilitätsanalyse

- Im Anwendungsfall „Artikel in den Warenkorb legen“ wird das Warenkorb-Objekt erzeugt und in der Session des Kunden abgespeichert. Es ist allerdings denkbar, dass ein Kunde im hier erörterten Anwendungsfall den Warenkorb zu ändern versucht, ohne dass dieser vorhanden ist³. Der Warenkorb wird ja erst erzeugt, wenn der Kunde versucht, Artikel hineinzulegen. Daher muss das System prüfen, ob der Warenkorb vorliegt, wenn der Kunde ihn zu benutzen versucht und ihn ggf. erzeugen. Hierfür muss ein Alternativablauf formuliert werden.
- Ebenso ist es denkbar, dass der Kunde einen leeren Warenkorb zu editieren versucht. Dies kann zum Beispiel passieren, wenn er bereits alle Positionen im Warenkorb gelöscht hat. Auch für diesen Fall muss noch ein Alternativablauf hinzugefügt werden, in dem das System den Kunden darauf hinweist, dass der Warenkorb leer ist.

Die genannten Überlegungen führen zu den im nächsten Abschnitt geschilderten Überarbeitungen des Anwendungsfalltextes. Die Änderungen sind wieder rekursiv markiert. Abbildung 6.2 zeigt das Stabilitätsdiagramm zu diesem Anwendungsfall.

Überarbeiteter Text des Anwendungsfalls

Standardablauf: Das System zeigt die Seite „Warenkorb“ an. *Es stellt zunächst sicher, dass in der Session des Kunden ein Warenkorb-Objekt existiert. Dann liest es aus dem Warenkorb-Objekt für diesen Kunden alle Positionen aus und zeigt sie u.a. mit Stückzahl, Preis, Buchtitel und Bestandsinformation an (siehe dazu GUI-Prototyp). Der Kunde ändert die Stückzahl einer Position im Warenkorb und klickt auf „Aktualisieren“.* Das System speichert die neue Stückzahl der betreffenden Position, berechnet die Kosten für diese Position und den Gesamtpreis über alle Positionen im Warenkorb neu und zeigt die aktuellen Werte an. Der Kunde klickt dann auf „Einkauf fortsetzen“, woraufhin das System die Kontrolle wieder an den Anwendungsfall „Buchkatalog ansehen“ übergibt.

Alternativabläufe: *Wenn das System für diesen Kunden keinen Warenkorb finden kann, erzeugt es ein neues Warenkorb-Objekt und speichert es in der Session des Kunden.*

Wenn der Inhalt des Warenkorbs leer ist, zeigt das System eine entsprechende Meldung auf der Seite „Warenkorb“ an.

Wenn der Kunde die Anzahl eines Artikels auf 0 setzt oder in einer Position auf „Löschen“ klickt, löscht das System die zugehörige Position aus dem Warenkorb und aktualisiert den Gesamtpreis über alle Positionen im Warenkorb. Dann zeigt es die aktuellen Werte neu an.

Wenn der Kunde anstatt auf „Einkauf fortsetzen“ auf „Bestellen“ klickt, ruft das System den Anwendungsfall „Buch bestellen“ auf.

Auswirkungen auf das Klassenmodell

- Damit der Warenkorb einem Kunden eindeutig zugeordnet werden kann, muss er in einem übergeordneten Objekt gespeichert werden, das den Kunden eindeutig identifiziert. Ein solches Objekt ist die Session des Kunden (Attribut „Warenkorb“ in der Session).

³Das Navigationskonzept (Abbildung 5.4) sieht diese Möglichkeit eigentlich nicht vor. Dennoch ist dieser Fall denkbar, wenn man die hier nicht modellierte Navigation über das Hauptmenü mit berücksichtigt.

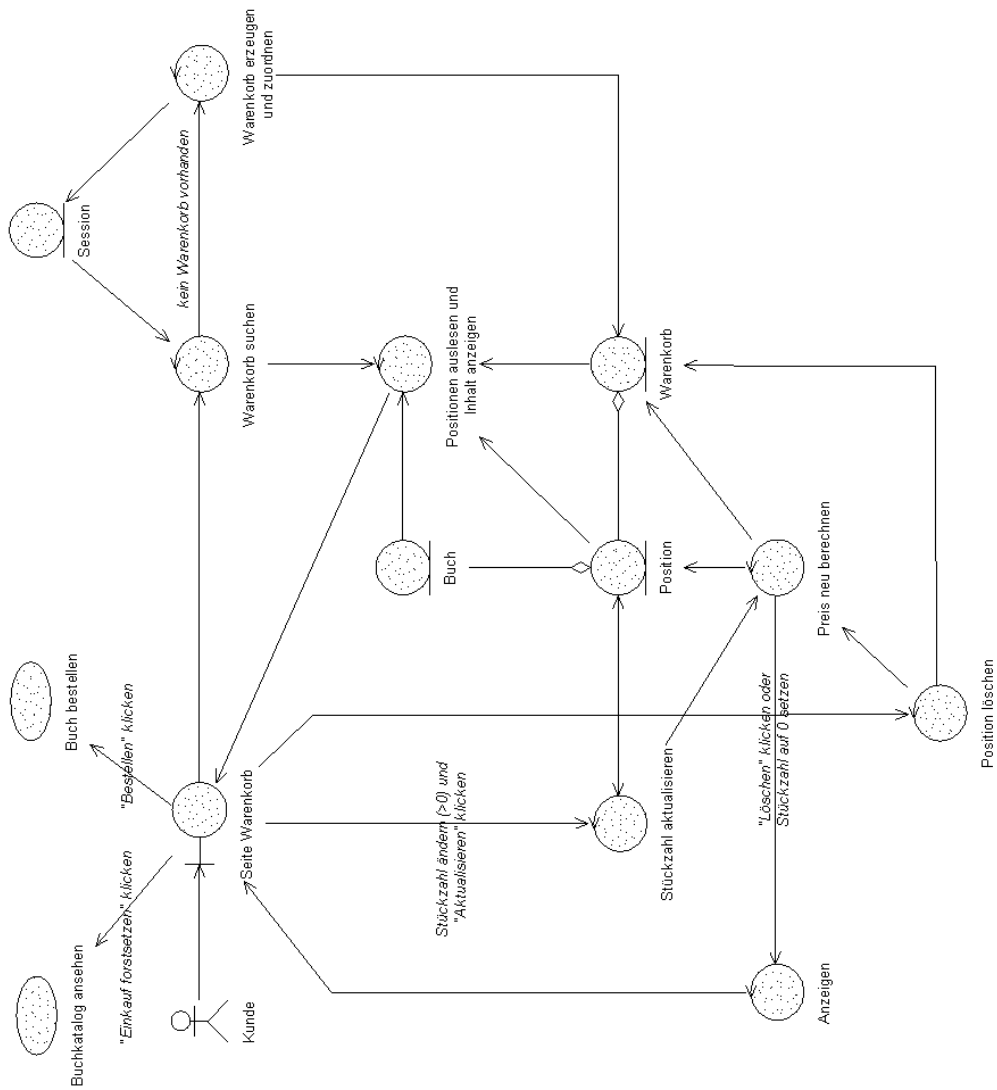


Abbildung 6.2.: Stabilitätsdiagramm Warenkorb editieren

6. Stabilitätsanalyse

- Die Klasse „Position“ muss außer der Stückzahl noch eine Referenz auf das Buch enthalten, das durch diese Position beschrieben wird, sowie den Gesamtpreis der Position. Damit man den Preis der Position überhaupt ermitteln kann, benötigt auch das Buch ein Attribut „Preis“. Der Warenkorb bekommt dieses Attribut ebenfalls, denn er kann als einziges Objekt den Gesamtpreis der Bestellung kennen.

6.3. Buch bestellen

Überlegungen zur Architektur

Dieser Anwendungsfall beschreibt den Kernprozess des Buchshops, die Buchbestellung. In Abschnitt 5.6.1 wurde bereits dargelegt, wie und warum der ursprüngliche Anwendungsfall „Buchbestellung durchführen“ zerlegt wird. Naturgemäß spiegelt sich diese Zerlegung auch im Stabilitätsdiagramm in Abbildung 6.3 wieder. Da sich der Bestellprozess über drei Bildschirmseiten erstreckt, wurde der Anwendungsfall „Buchbestellung durchführen“ in drei Anwendungsfälle zerlegt. Der hier diskutierte Anwendungsfall „Buch bestellen“ kontrolliert dabei die beiden anderen, nämlich „Lieferdaten bestimmen“ und „Zahlungsdaten bestimmen“. Diese Anwendungsfälle werden vom hier dargestellten Anwendungsfall aus aufgerufen. Problematisch ist dabei, dass der Kunde auch in den beiden aufgerufenen Anwendungsfällen die Bestellung abbrechen kann. Es muss also ein Controller eingetragen werden, der den Abbruch prüft.

Interessant ist nun zu erörtern, wie die Kommunikation zwischen den Anwendungsfällen zustande kommen kann und wie der Kontrollfluss der Anwendung abläuft.

Rosenberg und Scott erwähnen in [RoSc99] zwei Entwurfsstrategien zur Implementierung des Kontrollflusses. Dabei handelt es sich um die Strategien „Control in the Screen“ und „Use Case Controller“. Bei der ersten Entwurfsstrategie werden verschiedene Aspekte der Kontrolllogik auf die Boundary-Objekte verteilt. Bei der zweiten Strategie hingegen befindet sich der größte Teil der Logik in einem Kontrollobjekt.

Tatsächlich handelt es sich bei dieser Entscheidung um eine Architekturentscheidung, die das Design des Systems grundlegend bestimmt. Um eine solche Entscheidung qualifiziert treffen zu können, sind aber verschiedene Faktoren zu berücksichtigen. Dazu gehören nicht funktionale Anforderungen wie die Infrastruktur, mit der das zu implementierende System arbeiten soll, die Frage ob ein Framework eingesetzt wird und die Programmiersprache, die man verwenden will. Weiterhin spielen Überlegungen zu Qualitätskriterien wie etwa der Performanz und der Wartbarkeit des Systems eine Rolle. In Kapitel 7 werden diese Fragestellungen näher erörtert.

Die Autoren nehmen in [RoSc01, S. 81] zum Thema Architektur wie folgt Stellung:

„[Der Begriff] technische Architektur [...] bezieht sich auf grundlegende Entscheidungen bezüglich der Technologien, die man bei der Implementierung des Systems zu verwenden beabsichtigt. Diese Entscheidungen schließen sowohl solche Dinge ein wie die Programmiersprache (zum Beispiel Java oder Visual Basic) als auch die Frage, wie man die Komponenten der Software bauen und verteilen will (Wird man eher Enterprise Java Beans [EJB] und Java Server Pages [JSPs] oder die Microsoft-Technologien Distributed Component Object Model [DCOM] und Active Server Pages [ASPs] einsetzen?). Die Entscheidungen, die man bezüglich der technischen Architektur trifft, müssen sich zu einem gewissen Grad in den Stabilitätsdiagrammen widerspiegeln.“

Wenn man zum Beispiel mit einer technischen Architektur arbeiten will, die EJBs und JSPs einschließt, werden die Stabilitätsdiagramme eher in Richtung „control in the screen“ tendieren, als wenn man reine HTML-Seiten erzeugen wollte.“

Einen der modernsten Ansätze für die Entwicklung von Internet-Systemen definiert die Java 2 Enterprise Edition (J2EE). Zwar soll es nicht Ziel dieser Arbeit sein, die Vor- und Nachteile verschiedener Technologien gegeneinander abzuwägen, doch kann als Arbeitshypothese davon ausgegangen werden, dass man das System in Java programmieren wird. Dabei sollen Enterprise Java Beans (EJBs), gepaart mit Servlets oder Java Server Pages (JSPs) zum Einsatz kommen. Aus Gründen, die in Kapitel 7 erörtert werden, wird jedoch in Kapitel 8 die Strategie „use case controller“ verwendet werden und nicht „control in the screen“.

Analyse des Anwendungsfalls

Der erste Abschnitt des Anwendungsfalltextes erweckt den Eindruck, als ob der Anwendungsfall ohne Boundary-Objekte auskommen würde. Dann kann er aber mit dem Kunden nicht interagieren. Es sollte also der Auslöser mit in den Text aufgenommen werden, um deutlich zu machen, wie der Kontrollfluss zwischen den Anwendungsfällen verläuft.

Damit man eine Bestellung eindeutig identifizieren kann, muss eine eindeutige Bestellnummer vergeben werden. Der Anwendungsfall sollte dieses Detail nicht verschweigen.

Am Anfang des Anwendungsfalls überprüft das System die Existenz des Warenkorbs. Für den Fall, dass kein Warenkorb existiert, ist aber kein Alternativablauf vorhanden. Dieser muss ergänzt werden. Anders als in den Anwendungsfällen, die den Warenkorb betreffen, macht es keinen Sinn, an dieser Stelle ein Warenkorb-Objekt zu erzeugen. Vielmehr kann die Bestellung ohne weiteres abgebrochen werden, wenn kein Warenkorb existiert.

Am Ende des Anwendungsfalls wird die Kontrolle an den „aufrufenden Anwendungsfall“ zurückgegeben. Dies ist nicht ganz richtig. Vielmehr soll der Kunde nach der Buchbestellung zum Katalog zurück gelangen. Auch hier muss der Text korrigiert werden.

Im folgenden Abschnitt wird wieder der überarbeitete Text des Anwendungsfalls geschildert, wobei die geänderten Stellen kursiv dargestellt sind. Abbildung 6.3 zeigt das zugehörige Stabilitätsdiagramm.

Überarbeiteter Text des Anwendungsfalls

Standardablauf: *Der Kunde klickt auf der Seite „Warenkorb“ auf „Bestellen“.* Das System stellt nun sicher, dass der Kunde angemeldet ist und dass für den Kunden ein Warenkorb existiert. Dann stellt es sicher, dass der Warenkorb des Kunden mindestens eine Position enthält. Anschließend erzeugt es ein neues Bestellungen-Objekt *mit einer eindeutigen Bestellnummer* für diesen Kunden und kopiert den Inhalt des Warenkorbs hinein. Danach ruft das System nacheinander die Anwendungsfälle „Lieferdaten bestimmen“ und „Zahlungsdaten bestimmen“ auf. Dann stellt das System sicher, dass der Kunde in keinem der aufgerufenen Anwendungsfälle die Bestellung abgebrochen hat.

Anschließend zeigt das System die Seite „Bestätigung“ an. Auf dieser Seite stellt es noch einmal alle Daten der Bestellung dar. Der Kunde überprüft die Richtigkeit der Daten und klickt auf „Bestellung abschicken“. Das System fügt dann der Bestellliste die Bestellung hinzu *und übergibt die Kontrolle an den Anwendungsfall „Buchkatalog ansehen“.*

6. Stabilitätsanalyse

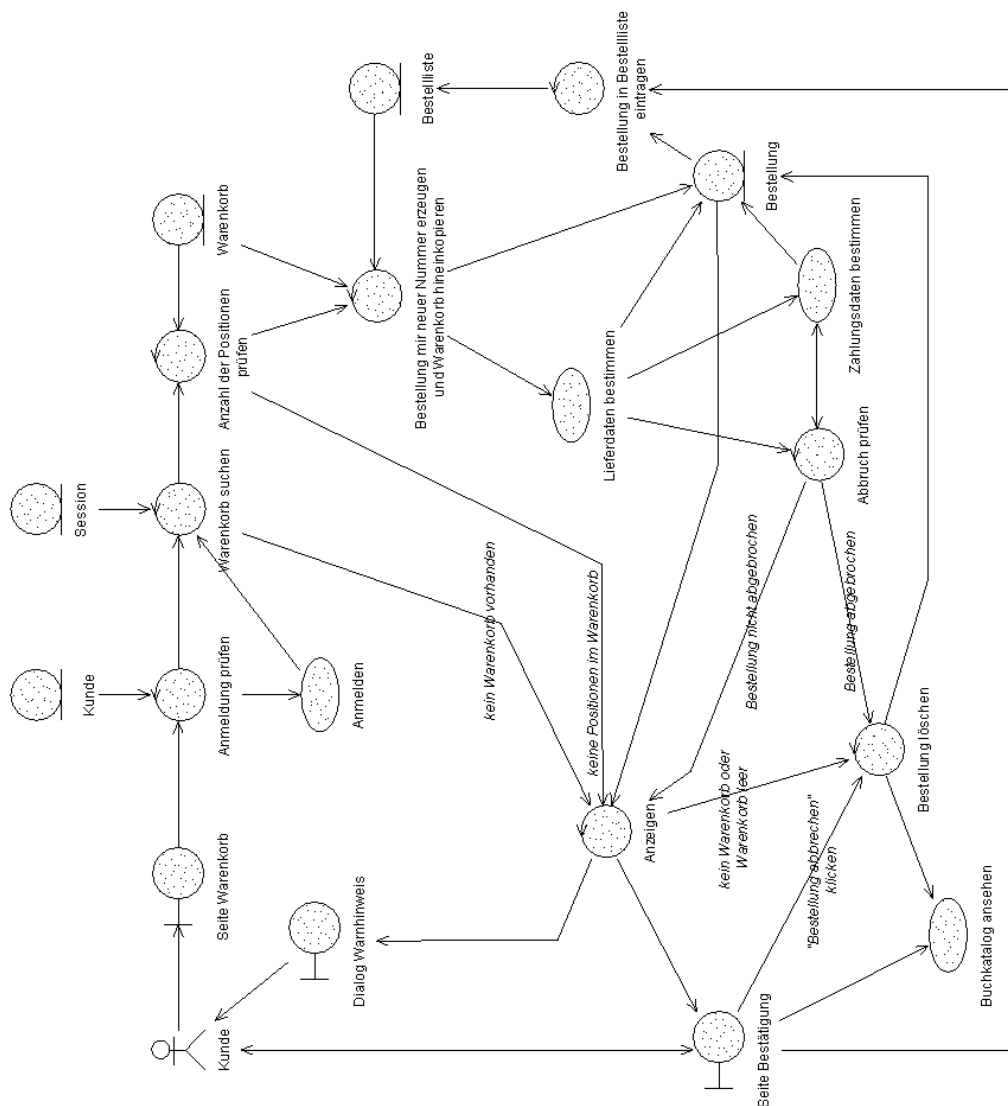


Abbildung 6.3.: Stabilitätsdiagramm Buch bestellen

Alternativabläufe: Wenn der Kunde nicht am System angemeldet ist, ruft das System zunächst den Anwendungsfall „Anmelden“ auf.

Wenn der Warenkorb für diese Einkaufssitzung nicht existiert oder keine Positionen enthält, zeigt das System eine entsprechende Meldung an, bricht diesen Anwendungsfall ab und übergibt die Kontrolle an den Anwendungsfall „Buchkatalog ansehen“.

Wenn der Kunde die Bestellung abbricht, verwirft das System das Bestellungs-Objekt und übergibt die Kontrolle an den Anwendungsfall „Buchkatalog ansehen“.

Auswirkungen auf das Klassenmodell

Die Auswirkungen dieser Stabilitätsanalyse auf das Klassenmodell sind sehr moderat, da der Anwendungsfall hauptsächlich die Bestätigung der Bestellung enthält. Die Erfassung der Bestelldaten wird in den beiden aufgerufenen Anwendungsfällen durchgeführt.

- Da der Inhalt des Warenkorbs komplett in das Bestellungs-Objekt übernommen wird, müssen alle Attribute des Warenkorbs auch in der Bestellung vorkommen. Darüber hinaus hat aber die Bestellung ein Attribut mehr, nämlich die eindeutige Bestellnummer. Es bietet sich daher an, beide Klassen von einer gemeinsamen (abstrakten) Oberklasse abzuleiten. Diese könnte zum Beispiel „BaseBestellung“ heißen, um anzudeuten, dass es sich um eine technische Klasse handelt, von der die Bestellung abgeleitet ist.

6.4. Lieferdaten bestimmen

Analyse

Dieser Anwendungsfall ist korrekt geschrieben. Hier sind keine Änderungen nötig. Abbildung 6.4 stellt die beschriebenen Abläufe grafisch dar.

Auswirkungen auf das Klassenmodell

- Das Kunden-Objekt muss ein Attribut „Lieferadressen“ bekommen, um die möglichen Adressen zu speichern, an die Bestellungen dieses Kunden versendet werden können. Dazu muss auch eine neue Klasse „Adresse“ definiert werden, von deren Typ die Objekte sind, die in der Liste gespeichert werden.
- Um die Liste der möglichen Versandarten darzustellen, benötigt das System eine weitere Liste, die die Versandarten vorhält. Die Versandarten können vom Basistyp „String“ sein.

6.5. Vergleich mit der Lösung von Rosenberg und Scott

Ein Vergleich mit der Lösung von Rosenberg und Scott fällt immer schwerer, je weiter der Entwicklungsprozess voranschreitet. Zwar haben Rosenberg und Scott in [RoSc01] eine Stabilitätsanalyse durchgeführt, jedoch finden sich auch im „Full Workbook Example“ kaum mehr Beispiele als im Buch. Da sich der Autor dieser Arbeit auf die Kernabläufe des Buchshops konzentriert hat, die sich

6. Stabilitätsanalyse

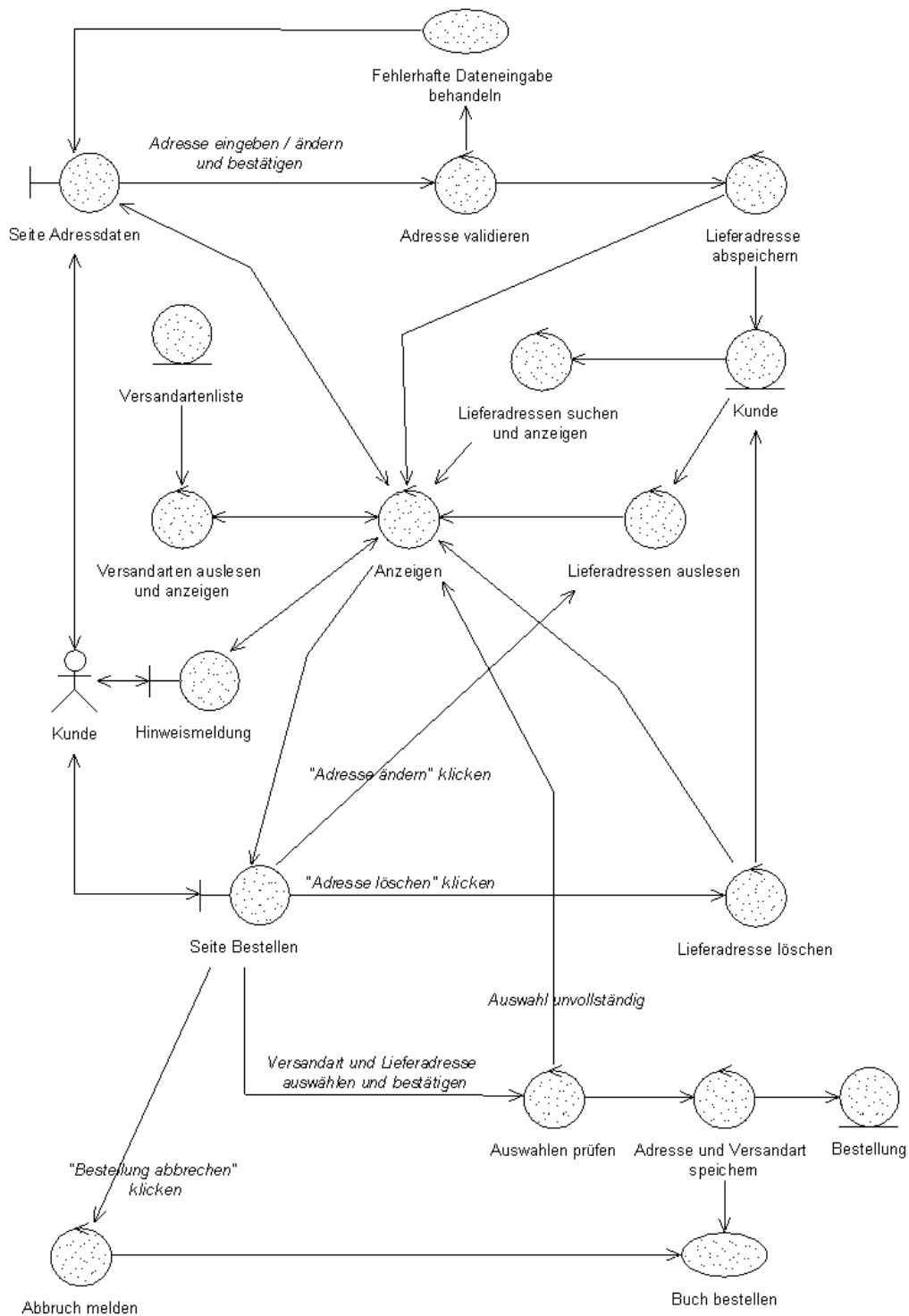


Abbildung 6.4.: Stabilitätsdiagramm Lieferdaten bestimmen

6.5. Vergleich mit der Lösung von Rosenberg und Scott

um den Produktkatalog und das Bestellverfahren ranken, ist ein Vergleich nur bei einem einzigen Diagramm möglich, nämlich „Warenkorb editieren“. Ansonsten haben sich Rosenberg und Scott hauptsächlich mit Aspekten wie dem Login-Verfahren beschäftigt, die nicht unmittelbar zu den Kernprozessen gehören, oder haben Shop-interne Vorgänge wie „Ship Order“ behandelt, die in dieser Arbeit nicht berücksichtigt sind (siehe Kapitel 1). Die einzige Stabilitätsanalyse, die deckungsgleich ist, ist „Warenkorb editieren“. Leider wird gerade dieser Anwendungsfall in der Anwendungsfallanalyse in [RoSc01] nicht besprochen, so dass es nicht möglich ist, herauszufinden, welche Abschnitte des Anwendungsfalls sich in dieser Entwurfsphase gegenüber früheren Phasen geändert haben. Da hier also erneut nicht nachvollziehbar ist, wie die Ergebnisse zustande kommen, muss es bei einem Vergleich der Diagramme bleiben.

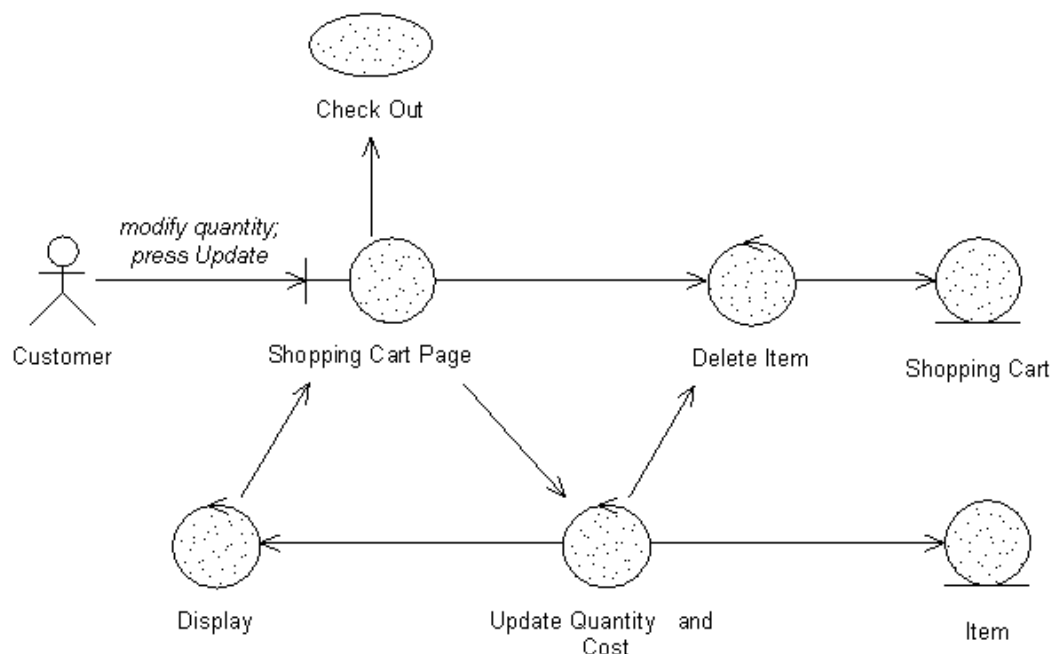


Abbildung 6.5.: Stabilitätsdiagramm „Warenkorb editieren“ von Rosenberg und Scott

Bereits auf den ersten Blick fällt auf, dass das Diagramm von Rosenberg und Scott (Abbildung 6.5) viel weniger Objekte beinhaltet als das in dieser Arbeit entwickelte. Zunächst ruft der Anwendungsfall von Rosenberg nicht den Anwendungsfall „Buch bestellen“ auf. Dies liegt vermutlich an einer anderen Benutzerführung, so dass in dem unbekannten GUI-Prototypen, den die Autoren für ihre Anwendungsfälle benutzen, keine Navigationsmöglichkeit von „Warenkorb editieren“ zu „Buch bestellen“ vorhanden ist. Außerdem enthält Abbildung 6.2 ein Entity-Objekt mehr, nämlich das Session-Objekt. Rosenberg macht sich nämlich in seiner Analyse keinerlei Gedanken über den reinen Wortlaut des Anwendungsfalls hinaus. Es wird nicht berücksichtigt, dass ein Internet-Shop im Mehrbenutzerbetrieb funktioniert und dass ein Warenkorb dem richtigen Kunden zugeordnet werden muss. Außerdem vernachlässigt der Anwendungsfall die Tatsache, dass unter Umständen kein Warenkorb vorhanden ist. Der Alternativablauf für den leeren oder nicht vorhandenen Warenkorb fehlt daher. Auch ist nicht berücksichtigt, dass nach Ändern des Warenkorbs der Preis neu berechnet und die Bildschirmansicht

6. Stabilitätsanalyse

aktualisiert werden sollte.

Weiterhin fehlt ein Controller, der die Werte, die in der Bildschirmansicht gezeigt werden sollen, aus den betroffenen Objekten ausliest. Dies ist von großer Bedeutung, denn nur, wenn man berücksichtigt, welche Entities die Daten enthalten, mit denen man arbeiten muss, kann man den vollständigen Satz von Objekten finden, die man benötigt, um den Anwendungsfall durchzuführen. Im vorliegenden Fall benötigt der Controller folgende Daten:

- den Gesamtpreis aller Positionen aus dem Warenkorb,
- die Stückzahlen und Gesamtpreise der einzelnen Positionen aus den Positions-Objekten und
- den Buchtitel und die Verfügbarkeitsinformationen aus den Buch-Objekten.

Den Gesamtpreis der Artikel im Warenkorb muss man nicht unbedingt im Warenkorb-Objekt speichern. Man kann ihn auch jedes Mal aus den Preisen der Positionen neu berechnen, daher ist an dieser Stelle eine Kommunikation nicht zwingend. Das Buch-Objekt sollte aber im Diagramm nicht fehlen.

Dieses Beispiel zeigt, dass Rosenberg die Stabilitätsanalyse in seinem Beispiel nicht genau genug durchführt. Dabei verletzt er teilweise die von ihm selbst aufgestellten Regeln, indem er nicht sicherstellt, dass alle notwendigen Alternativabläufe berücksichtigt sind (Überprüfung der Vollständigkeit und Korrektheit) und nicht darauf achtet, sämtliche Objekte einzuzeichnen, die für die Durchführung des Anwendungsfalls notwendig sind. Außerdem würde man an dieser Stelle erste Überlegungen zur Softwarearchitektur erwarten. Abschnitt 6.3 zeigt exemplarisch, wodurch solche Überlegungen ausgelöst werden können. Eine Erörterung grundlegender architektonischer Entscheidungen und deren möglicher Auswirkungen auf den Entwurf bleibt Rosenberg in seinem Beispiel aber schuldig.

Tatsächlich muss die Stabilitätsanalyse als Teil der Architekturfindung gesehen werden. Die Autoren deuten diesen Sachverhalt aber bestenfalls an und geben keinerlei Werkzeuge an die Hand, die bei der Architekturfindung helfen können. In der Tat erschöpft sich die Diskussion architektonischer Fragen in dem oben erwähnten Zitat.

7. Überlegungen zur Softwarearchitektur

7.1. Allgemeine Überlegungen

Im Verlauf von Kapitel 6 wurden bereits die Entwurfsstrategien „control in the screen“ und „use case controller“ erwähnt, die Rosenberg und Scott in [RoSc99] als mögliche Strategien präsentieren, um die Controller aus der Stabilitätsanalyse in Methoden umzuwandeln. Bei genauerem Hinsehen reicht es aber nicht aus, sich für die eine oder andere Strategie zu entscheiden und sich bei der Frage nach dem Architekturmodell ansonsten darauf zurückzuziehen, dass die Entity-Boundary-Architektur vom MVC-Muster abgeleitet ist. Bei der Entwicklung nicht trivialer Softwaresysteme stellen sich noch eine ganze Reihe anderer Fragen, die zum Teil nichtfunktionaler Art sind und die nur durch ein schlüssiges Architekturmodell beantwortet werden können.

Eine dieser Fragen bezieht sich auf die *technische Infrastruktur*. Realisiert man zum Beispiel ein verteiltes System, so kann die Bandbreite und Auslastung der Netzwerkverbindung die Performanz des Systems beschränken, wenn etwa auf Antworten der Gegenstelle gewartet werden muss. Bei einem Standalone-System besteht diese Limitierung nicht.

Wie oben bereits erwähnt wurde, kann auch die Wahl der *Programmiersprache* von großer Bedeutung sein. Sprachen wie Java oder C# besitzen zum Beispiel umfangreiche Klassenbibliotheken, die dem Programmierer viele Aufgaben abnehmen, während Sprachen wie C eher spartanisch ausgestattet sind. Außerdem sind etwa Persistenzframeworks, die den jeweiligen Ansprüchen an Qualität, Leistungsfähigkeit und Preis genügen, für manche Programmiersprachen verfügbar, für andere nicht.

Auch können gewisse architektonische Rahmenbedingungen bereits vorgegeben sein. Erwägt man den Einsatz eines *Frameworks*, ist man stark von dessen Funktionsprinzipien abhängig. Daraus folgt, dass man bestimmte Entwurfsmuster einsetzen muss, die dem jeweiligen Framework zu Eigen sind.

Schließlich wird auch und vor Allem die *Qualität* der Software wesentlich von der Architektur bestimmt. Eine schwache Architektur ist durch nichts wieder gutzumachen und hat schon manches Softwareprojekt zum Scheitern gebracht. Ein gutes Softwaresystem zeichnet sich durch Performanz, Sicherheit, Verfügbarkeit, Wartbarkeit, Portierbarkeit, Wiederverwendbarkeit, Integrierbarkeit und Testbarkeit aus. Je nach dem Verwendungszweck und der Zielsetzung des Projektes wird man das eine oder andere Qualitätskriterium höher gewichten oder vernachlässigen. Erreicht man aber bei wichtigen Qualitätskriterien die Zielvorgaben nicht, so kann das im weiteren Verlauf zu Problemen führen, die unter Umständen nicht mehr lösbar sind.

7.2. Spezielle Überlegungen

In der in Kapitel 3 geschilderten Problemstellung finden sich zwei nicht funktionale Anforderungen:

- das System soll internetbasiert sein und

7. Überlegungen zur Softwarearchitektur

- es soll Accounts für bis zu 1.000.000 Kunden verwalten können.

Auswirkungen der Infrastruktur auf die Architektur

Die erste Anforderung bezieht sich direkt auf die technische Infrastruktur, mit der das System arbeitet. Es handelt sich dabei um ein Internet-System. Das bedeutet, dass es für jede Session zwei Gegenstellen geben muss - nämlich Client und Server - bei dem der Rechner des Kunden den Client stellt, und der Rechner des Händlers den Server. Es gibt verschiedene technische Möglichkeiten, Client und Server miteinander kommunizieren zu lassen. Eine Möglichkeit besteht in der Realisierung als verteiltes System, bei dem der Client mit dem Server über irgendeine Art von Middleware kommuniziert. Dies würde aber bedeuten, dass bei jedem Kunden, der das System benutzen will, ein solcher Client installiert sein müsste. Dies ist sicher für einen Internet-Shop nicht angemessen, denn dieser soll von einem Maximum an Kunden mit dem geringstmöglichen Einsatz an technischen Mitteln nutzbar sein. Clientseitig wird man also auf eine weit verbreitete Technologie setzen, die für praktisch jede Zielplattform verfügbar ist, also einen Internet-Browser.

Die Entscheidung für den Einsatz dieser Technologie erzwingt aber auch, dass man sich auf die Verwendung des HTTP-Protokolls einlässt. Das hat wiederum Auswirkungen auf die Funktionsweise des Servers:

Bei einem System, dass über das HTTP-Protokoll kommuniziert, läuft die Kommunikation zwischen Client und Server in Request-Response-Zyklen ab. Das System kann deshalb nicht zu jedem beliebigen Zeitpunkt Bildschirmansichten präsentieren, sondern immer nur als Antwort auf Client-Anfragen. Der Client stellt eine Anfrage an den Server, woraufhin der Server die Informationen, die als Antwort darauf anzuzeigen sind, an den Client zurück sendet. In aller Regel führt die Antwort des Servers zu einem unmittelbaren Neuaufbau der Bildschirmansicht im Browser.

Beispielsweise zeigt im oben dargestellten Anwendungsfall „Warenkorb editieren“ das System die Seite Warenkorb an. Der Kunde ändert im Warenkorb die Stückzahl einer Position und klickt auf „Aktualisieren“. Nun wird der Client einen HTTP-POST-Request abschicken, der die Daten des entsprechenden Formularbereichs enthält, sowie eine Action, die ebenfalls zu diesem Request gehört. Das System muss die Daten aus dem Protokoll extrahieren, die Benutzeraktion identifizieren (wahrscheinlich durch die Action), dann die entsprechende Geschäftslogik ausführen (Aktualisieren der Stückzahl im Positions-Objekt, Neu berechnen des Preises der Position und des Warenkorbs) und dem Client die Seite mit neuem Inhalt wieder anzeigen.

Das beschriebene Auswerten der Protokolle ist eine allgemeine Aufgabe, die von einem dafür spezialisierten Objekt übernommen werden sollte. Auch der Kontrollfluss kann in einem Controller zentralisiert werden, der - abhängig vom jeweiligen Zustand des Systems und der Aktion des Kunden - die passenden Seiten nacheinander aufruft. (Dies würde eher für die Verwendung der Strategie „use case controller“ sprechen.) Dieser Controller kann zum Beispiel den in Abbildung 5.4 dargestellten Automaten implementieren¹.

Ein solches spezialisiertes Objekt wäre Bestandteil der Architektur des Servers, denn jeder Request wird auf genau die gleiche Weise verarbeitet. (In der unten beschriebenen Architektur handelt es sich dabei um dem ViewHelper.)

¹Im unten dargestellten Beispiel von Singh u.a. ist diese Funktionalität im Screen Flow Manager gekapselt.

Anforderungen durch Mehrbenutzerbetrieb

Normalerweise arbeiten Internet-Systeme nebenläufig. Das bedeutet, dass mehrere Benutzer gleichzeitig so mit dem System arbeiten können müssen, als ob sie das System alleine benutzen würden. Für den Server bedeutet das in der Regel, dass das angeschlossene Datenbanksystem Transaktionen beherrschen muss. Auch die zu entwickelnde Anwendung muss darauf eingerichtet sein.

Das Stichwort Datenbank führt zu einer weiteren Überlegung. Das System muss nämlich die erzeugten Daten wie Kundenkartei, Bestellungen und Abrechnungen irgendwie persistieren, also für weitere Verwendung in einer Datenbank speichern. Man hat nun die Wahl, diese Mechanismen inklusive der Transaktionslogik selbst zu implementieren, oder dafür ein Persistenzframework zu benutzen.

Außerdem muss die Anwendung in der Lage sein, die Daten der Einkaufssitzungen verschiedener Kunden voneinander zu unterscheiden. Technisch bedeutet das, dass das System eine Sessionverwaltung haben muss. Jeder Verbindung zwischen Client und Server wird dann eine Session zugeordnet und die Daten, die zur Session gehören, werden daran gebunden. Zu diesen Daten gehören unter anderem die Identifikation und der Warenkorb des Kunden und die Bildschirmansicht, die er sich gerade ansieht.

Anforderungen an die Skalierbarkeit

Die zweite nicht funktionale Anforderung liegt in der Forderung, dass der Shop mit bis zu 1.000.000 Kunden umgehen können soll. Man wird nicht davon ausgehen, dass sich alle diese Kunden gleichzeitig im Shop befinden werden. Allerdings gilt „Je mehr, desto besser“, denn man will ja so viele Bücher wie möglich verkaufen. Am Anfang wird es vielleicht noch nicht so viele Kunden geben, später wird das Aufkommen (hoffentlich) erheblich steigen. Daraus resultieren Anforderungen an die Skalierbarkeit des Systems. Es soll praktisch mit jeder Anzahl von Kunden (bis zu der besagten Million) gleich gut zurechtkommen und das System soll mit der Anzahl an Kunden wachsen können.

Konsequenzen

Die genannten Überlegungen führen zur Auswahl einer bestimmten Technologie, die wiederum Auswirkungen auf die Softwarearchitektur hat. Eine Technologie, die speziell auf die genannten Bedürfnisse der Internetwelt zugeschnitten ist, ist das „Java 2 Enterprise Edition“-Framework (J2EE). Hier wird man insbesondere den Einsatz von Enterprise Java Beans (EJBs) anstreben. EJBs sind von Natur aus verteilt und „leben“ in Application Servern. Es gibt eine ganze Reihe verschiedener Application Server in verschiedenen Preiskategorien. Die leistungsfähigsten Application Server machen es dem Benutzer einfach, das System zu skalieren. Die Beans können nämlich auf verschiedene Rechner verteilt werden. Solche Server haben eine eingebaute Lastverteilung und bieten Ausfallsicherheit, indem sie die Beans auf einem anderen Rechner replizieren können, wenn der Rechner, auf dem sie ursprünglich vorhanden waren, ausfällt. Außerdem kann man die Anwendung mit „Container Managed Persistence“ implementieren, so dass man den ganzen Bereich der Datenpersistenz umsonst bekommt.

Außerdem kann man sich bei J2EE auch Technologien bedienen, die die Kommunikation zwischen Browser und Server über das HTTP-Protokoll ermöglichen. Dazu gehören Java Server Pages (JSPs) und Java Servlets. Diese haben auch ein integriertes Session-Management. Somit wären alle beschriebenen Probleme potenziell gelöst.

7.3. J2EE und MVC

In [SiStJa] wird eine Architektur vorgestellt, mit der man einen Internet-Shop unter Einsatz von J2EE-Technologien realisieren kann. Das dort beschriebene Beispiel zeigt, wie das MVC-Architekturmuster auch bei Internetanwendungen sinnvoll verwendet werden kann, allerdings nicht in der Weise, die das Beispiel von Rosenberg nahe legt.

Die Autoren Singh, Stearns und Johnson folgen bei der Entwicklung dieser Architektur einem Top-Down-Ansatz, bei dem die Anwendung in drei Stufen² aufgeteilt wird, und zwar in einen Client-Tier, den der Browser zur Verfügung stellt, sowie einen Web- und einen EJB-Tier, die vom Server implementiert werden.

Weiterhin wird die Anwendung in Objekte oder Komponenten unterteilt, die dann den einzelnen Schichten zugeteilt werden. Dabei kann die MVC-Architektur eingesetzt werden, die die Anwendung in drei logische Bereiche unterteilt. Ein Bereich befasst sich mit den Präsentationsaspekten (View), während sich ein zweiter um die Geschäftsregeln und -daten der Anwendung kümmert (Model). Der dritte Bereich befasst sich schließlich mit der Verarbeitung und Interpretation von Benutzeranfragen und kontrolliert die Geschäftsobjekte, die diese Anfragen bearbeiten (Controller). Abbildung 7.1 zeigt, wie die verschiedenen Funktionen der Anwendung sich auf die drei Bereiche verteilen.

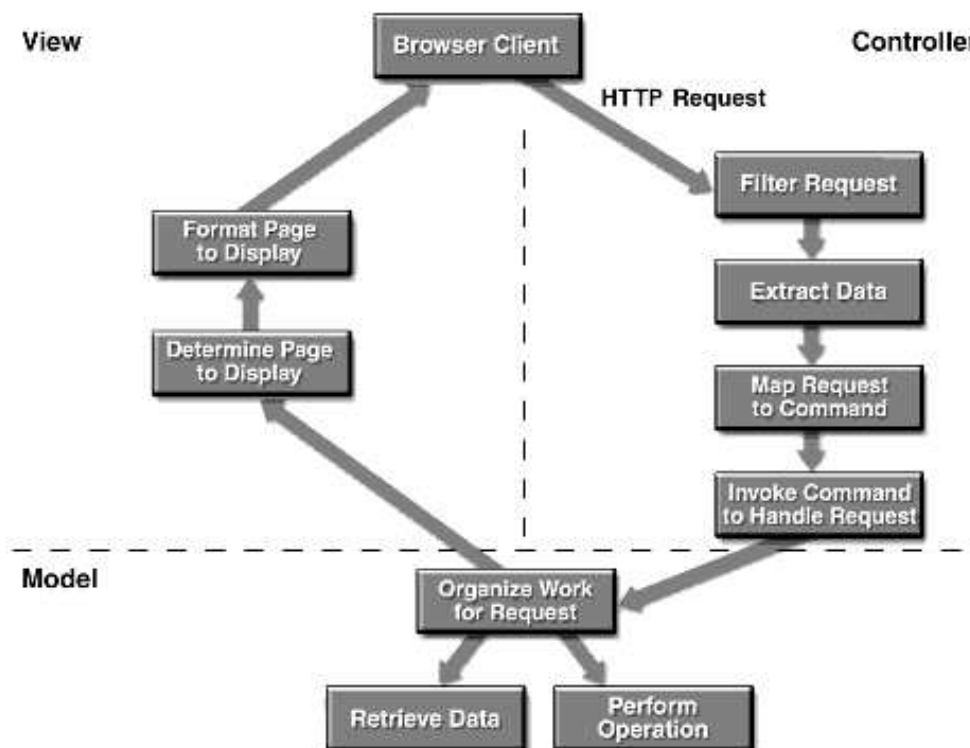


Abbildung 7.1.: Architektur eines J2EE-basierten Internetshops (aus [SiStJa, S. 366])

²engl. Tier

Eingesetzte J2EE Entwurfsmuster

Oben wurde bereits erwähnt, dass die Verwendung bestimmter Frameworks architektonische Rahmenbedingungen setzt, die sich zum Beispiel in der (unter Umständen obligatorischen) Verwendung von Entwurfsmustern niederschlägt. Singh, Stearns und Johnson erläutern in [SiStJa, S. 350f.] einige wichtige Entwurfsmuster, die typisch für J2EE-Anwendungen sind. Einige dieser Muster, die insbesondere im Controller-Teil der Anwendung eine Rolle spielen, werden nun vorgestellt:

Front Controller

„Dieses Muster ermöglicht einen zentralisierten Controller, der Requests verarbeitet. Ein Front Controller empfängt alle eingehenden Client-Requests, leitet jeden Request an den richtigen Request-Handler weiter und präsentiert dem Client eine entsprechende Antwort.“

Sessionfassade

„Dieses Muster koordiniert Operationen zwischen kooperierenden Geschäftsobjekten, wobei die Funktionen der Anwendung in einer einzigen, vereinfachten Schnittstelle vereinigt und dem aufrufenden Code präsentiert werden. Es kapselt und verbirgt die Komplexität der Klassen, die in spezieller, vermutlich komplexer Weise interagieren müssen, und schirmt die Klienten von Änderungen in der Implementierung der Geschäftsobjekte ab. Eine Sessionfassade wird üblicherweise als Session Bean implementiert und verbirgt die Interaktionen zwischen darunter liegenden Entity Beans.“

View Helper

„Ein View Helper kapselt die Präsentation und den Datenzugriff logischer Bereiche einer Ansicht, wodurch er die Ansicht verfeinert und vereinfacht. Die Präsentationslogik betrifft das Formatieren der Daten für die Anzeige auf einer Seite, während sich der Datenzugriff um das Beschaffen der Daten dreht. View Helper sind oft JSP Tags zum Rendern oder Darstellen von Daten und Java Beans zum Beschaffen der Daten.“

7.4. Architekturfragment des Internet-Shops

Im Abschnitt 6.3 wurde bereits auf das Problem des Kontrollflusses eingegangen. Während sich die Stabilitätsanalyse noch auf sehr abstraktem Niveau abspielt (in der Art von „Anwendungsfall A ruft Anwendungsfall B auf“), ist es nun an der Zeit, sich die Dinge genauer anzusehen. Am Anfang dieses Kapitels wurde bereits erläutert, dass die Anwendung nicht ohne weiteres zu jeder Zeit eine Bildschirmansicht präsentieren kann. Dies stellt das Entwurfsprinzip „control in the screen“ in Frage, denn hier wird so getan, als könne eine Bildschirmseite eine andere zur Ansicht bringen. Dies ist jedoch aufgrund der Beschränkungen des HTTP-Protokolls nicht möglich.

Die in [SiStJa] vorgestellte Architektur stellt eine Referenzarchitektur für Internet-Shops auf der Basis von J2EE-Technologien dar. Im Folgenden soll ein vereinfachtes Architekturfragment erläutert werden (siehe auch Abbildung 7.2), das von der dort vorgestellten Architektur abgeleitet ist und

7. Überlegungen zur Softwarearchitektur

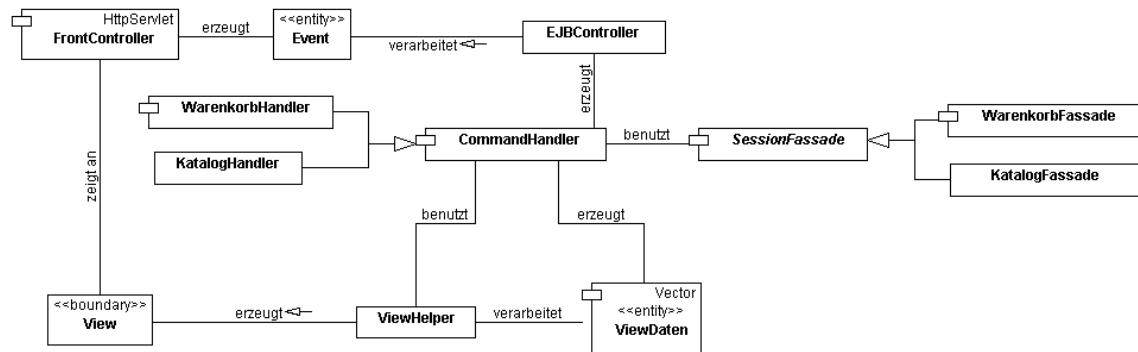


Abbildung 7.2.: Architekturfragment für den Buchshop

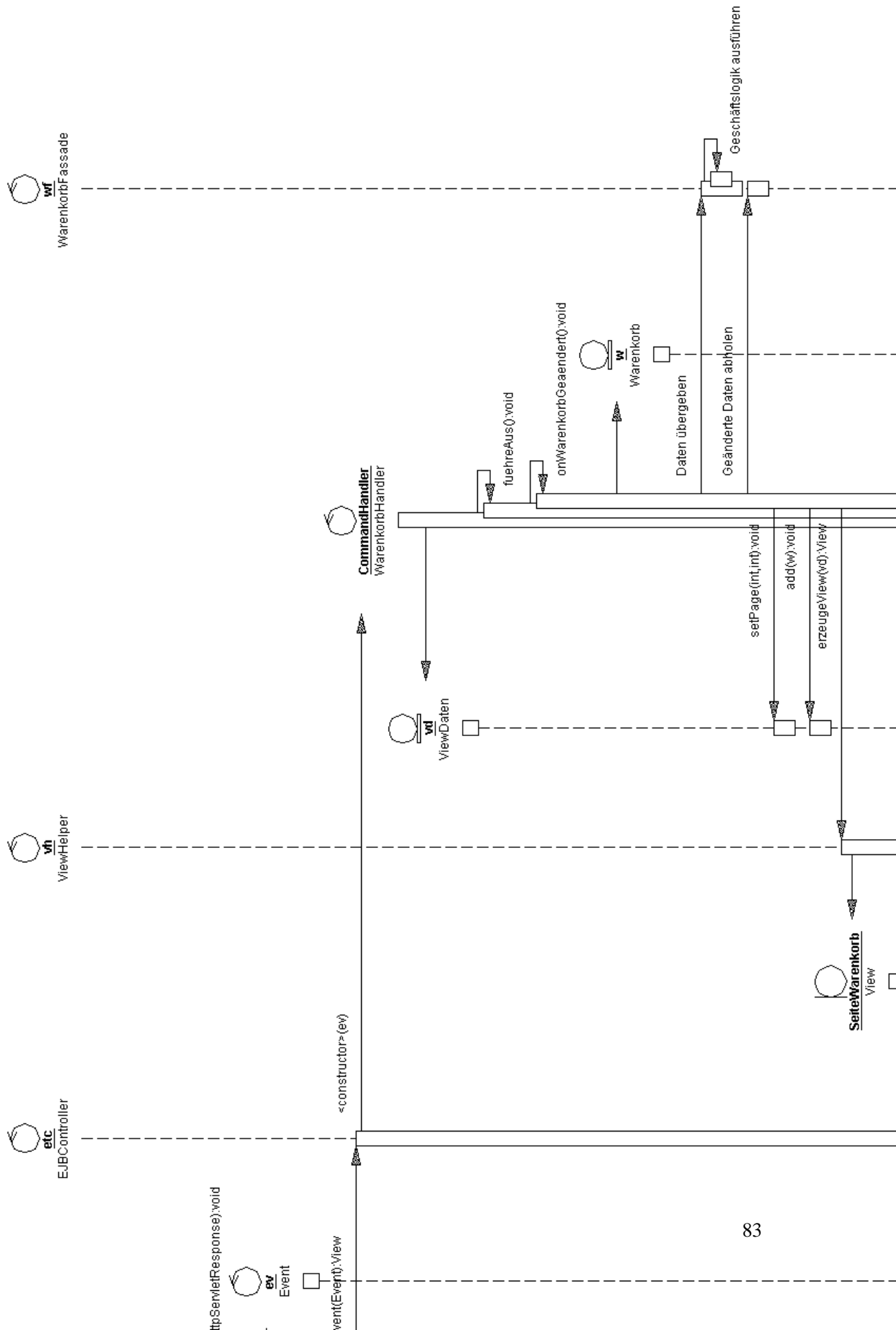
das für den Buchshop zur Anwendung kommen könnte. Es handelt sich dabei hauptsächlich um den Controller-Teil der Architektur. Deren Funktionsprinzipien sollen nun kurz erläutert werden:

Der Front Controller bearbeitet sämtliche HTTP-Anfragen, die vom Client kommen. Er extrahiert die Daten aus dem HTTP-Protokoll und identifiziert die Eingabe des Kunden. Dann erzeugt er ein Event-Objekt, das die Daten der Anfrage enthält und leitet es an den EJBController weiter. Dieser erzeugt einen dazu passenden CommandHandler. Der CommandHandler benutzt Methoden der verschiedenen Sessionfassaden, um die Geschäftsprozesse zu aktivieren, die von der Client-Anfrage angestoßen wurden, und anschließend die Geschäftsdaten auszulesen. Somit ist seine Hauptbedeutung, die Benutzeranfragen auf die Geschäftsprozesse abzubilden. Die Sessionfassaden sind nach den Bereichen aufgeteilt, die sie bearbeiten. Zum Beispiel kann es Sessionfassaden für den Warenkorb, für die Bestellungsabwicklung und den Katalog geben. Eine sinnvolle Aufteilung kann hier nach den Anwendungsfallpaketen erfolgen.

Unter Umständen kann es daher auch nötig sein, dass ein CommandHandler auf mehrere Sessionfassaden zugreift, etwa um Bestellungen auszuführen. Hier muss er eine WarenkorbFassade und eine BestellungsFassade bedienen, um einen Warenkorb in eine Bestellung zu überführen. Normalerweise deklariert die Command-Schnittstelle nur eine einzige Methode: `fuehreAus()`. Um jedoch das Verhalten eines Anwendungsfalls komplett in einem CommandHandler unterbringen zu können, gibt es nicht für jede mögliche Benutzeraktion einen CommandHandler, sondern diese sind nach Anwendungsfällen unterteilt. Jeder Handler hat einen Satz von Methoden, die die einzelnen Teilabläufe der Anwendungsfälle in sich tragen. Die Methode `fuehreAus()` entscheidet anhand des Events, aus dem der Handler gebildet wurde, welche Methode jeweils aufgerufen werden muss.

Das HTTP-Protokoll sieht vor, dass ein HTTP-Request immer durch eine HTTP-Response beantwortet wird. Dies führt in der Regel zum Aufbau einer Bildschirmseite im Client. Zum Erzeugen der Struktur dieser Bildschirmdarstellung wird der ViewHelper benötigt. Er baut aus den Geschäftsdaten, die der CommandHandler von der SessionFassade abfragen kann, und einer Beschreibungsdatei für die Struktur der Seite ein View-Objekt zusammen, das der FrontController dann in eine HTTP-Response übersetzt und dem Client zuschickt. Bei der Bestückung des ViewDaten-Objekts mit den Geschäftsdaten aus dem Model-Tier kann der CommandHandler unter anderem die Struktur der Daten ändern, so dass deren Implementierungsdetails vor dem ViewHelper verborgen wird. Auch kann die Datenstruktur auf diese Weise in eine Form gebracht werden, die für den ViewHelper geeigneter ist, um daraus eine Bildschirmseite aufzubauen.

7.4. Architekturfragment des Internet-Shops



7. Überlegungen zur Softwarearchitektur

Der CommandHandler bildet die Schnittstelle zwischen dem Model-Tier und Control-Tier. Er ist das einzige Objekt im Control-Tier, das Zugriff auf die SessionFassade hat. Außerdem kapselt er durch die Verwendung des Command-Musters die Geschäftsprozesse in sich.

Abbildung 7.3 zeigt, wie das System einen HTTP-Request verarbeitet. Der dargestellte Vorgang bezieht sich auf den Anwendungsfall „Warenkorb editieren“, ist aber auch auf andere Anwendungsfälle übertragbar.

Zu Beginn stellt das System die Seite „Warenkorb“ dar. Der Benutzer ändert die Stückzahl einer Position und klickt auf „Aktualisieren“. Daraufhin schickt der Client einen HTTP-Request ab. Der FrontController analysiert den Request und erzeugt ein entsprechendes Event-Objekt, das der EJBController in einen spezialisierten CommandHandler vom Typ WarenkorbHandler übersetzt. Der Handler weiß nun, was er zu tun hat: Er übergibt die Daten des Events an die WarenkorbFassade, die daraufhin die Geschäftslogik für die Datenänderung ausführt, nämlich die Neuberechnung der Preise (siehe dazu auch Kapitel 8). Anschließend liest er die Daten des Warenkorbs wieder aus und überträgt sie in ein ViewDaten-Objekt. Dieses übergibt er dem ViewHelper, der daraus und aus der (nicht eingezeichneten) Layout-Beschreibungsdatei den Inhalt der Seite erzeugt und in einem View-Objekt speichert.

Sobald der EJBController wieder die Kontrolle hat, fragt er im CommandHandler nach dem erzeugten View-Objekt, das er dann dem FrontController zurück gibt. Der FrontController verpackt den im View-Objekt enthaltenen Seiteninhalt in eine HTTP-Response und übermittelt sie dem Client.

8. Interaktionsmodellierung

In dieser Phase des Entwurfsprozesses werden die Softwarefunktionen, die in der Stabilitätsanalyse identifiziert werden, auf die Klassen des Klassenmodells verteilt. Das Mittel zur Darstellung sind dabei Sequenzdiagramme. Als Grundlage für die Interaktionsmodellierung dienen die Diagramme, die bei der Stabilitätsanalyse entwickelt wurden. Diese Diagramme werden erneut durchdacht und dabei versucht man, ein genaues Bild der Abläufe zu erlangen, die dort dargestellt sind.

Dabei werden folgende Tätigkeiten ausgeführt:

- Man eruiert, welche Boundary-, Entity- und Control-Objekte welches Verhalten haben müssen.
- Man stellt dar, wie die genauen Interaktionen aussehen, die zwischen den Objekten auftreten, die zu den verschiedenen Anwendungsfällen gehören.
- Man schließt die Verteilung der Operationen auf die Klassen ab.

Grundsätzlich orientieren sich die Sequenzdiagramme sehr stark an den Stabilitätsdiagrammen. In den Stabilitätsdiagrammen wird allerdings nur ein vorläufiger Entwurf gemacht, wobei hier der Entwurf detailliert stattfinden sollte. Es können daher Entscheidungen, die im vorigen Kapitel getroffen wurden, geändert werden. Dies führt aber nicht zwangsläufig dazu, dass die Stabilitätsdiagramme überarbeitet werden müssen. Sie haben ihren Zweck erfüllt - nämlich dem Designer eine Vorstellung davon zu geben, wie die Dinge funktionieren sollen - und werden nicht weiter erwartet.

Der Ablauf, nach dem ein Sequenzdiagramm entsteht, ist folgender:

- Kopieren des Anwendungsfalltextes ins Diagramm,
- Übernehmen der im Stabilitätsdiagramm gefundenen Entity-Objekte ins Diagramm,
- Übernehmen der im Stabilitätsdiagramm gefundenen Boundary-Objekte ins Diagramm,
- Umsetzen jedes Controllers im Stabilitätsdiagramm in (ggf. mehrere) Operationen und Verteilen der Operationen auf die richtigen Objekte.

Außerdem sollte in dieser Phase Wert auf einen guten Softwareentwurf gelegt werden, denn hier fallen Entscheidungen, die sich drastisch auf die Qualität der Software auswirken. Hilfreich für die Realisierung eines guten Entwurfs ist dabei der Einsatz von Entwurfsmustern.

Schließlich soll in dieser Phase noch die „Infrastruktur“ hinzugefügt werden, die sich vom reinen Informationsmodell abhebt, wie zum Beispiel Hilfsklassen. Der wichtige Gesichtspunkt dabei ist, dass sich nach Rosenberg an dieser Stelle die Perspektive vom Problembereich zum Lösungsbereich verschiebt¹ (siehe auch [RoSc99, S. 103]).

¹Tatsächlich führt diese Vorgehensweise, ohne dass man vorher ein Bild von der Gesamtarchitektur des Systems entwickelt, zu erheblichen Schwierigkeiten. Mehr dazu findet sich in den folgenden Abschnitten und in Kapitel 7.

8.1. Warenkorb editieren

Oben wurde bereits erwähnt, dass die Wahl der Architektur bestimmt, nach welcher Strategie die Control-Objekte in Operationen von Klassen umgewandelt werden. Unterstellt man hier die Architektur, die in Abschnitt 7.4 vorgestellt wurde, so spricht dies für die Verwendung der Strategie „use case controller“. Die gesamte auszuführende Logik eines Anwendungsfalls befindet sich nämlich bei dieser Architekturvariante in einem einzigen CommandHandler-Objekt.

8.1.1. Entwicklung des Detailentwurfs aus dem Stabilitätsdiagramm

Abbildung 6.2 im vorigen Kapitel zeigt das vorläufige Design des Anwendungsfalls „Warenkorb editieren“. In diesem Diagramm befinden sich 7 Controller:

Im Standardablauf befinden sich die Controller

1. Warenkorb suchen,
2. Positionen auslesen und Inhalt anzeigen,
3. Preis neu berechnen,
4. Anzeigen und
5. Stückzahl aktualisieren,

und in den Alternativabläufen

1. Position löschen und
2. Warenkorb erzeugen und zuordnen.

Ableiten der Methoden aus den Controllern

In diesem Abschnitt soll beschrieben werden, wie sich die Controller auf die Objekte verteilen und in welche Methoden oder Sätze von Methoden sie sich verwandeln. Der *vollständige* Ablauf des Anwendungsfalls kann aber nur im Kontext der Architektur sinnvoll betrachtet werden. Daher kann die Realisierung unter Umständen etwas vom vorläufigen Design in Kapitel 6 abweichen.

Warenkorb suchen Für die Kommunikation mit den Entity-Objekten des Systems, die als Java Entity Beans realisiert sind, sind die verschiedenen Sessionfassaden zuständig. Speziell für den Warenkorb gibt es die WarenkorbFassade. Um zu prüfen, ob ein Warenkorb vorhanden ist, bekommt die WarenkorbFassade eine Methode `sucheWarenkorb():boolean`. Diese muss ihrerseits versuchen, den Warenkorb aus der Session des Kunden auszulesen. Deshalb benötigt die Klasse „Session“ eine `get`-Methode für den Warenkorb.

Positionen auslesen und Inhalt anzeigen Genau genommen handelt es sich hier um zwei verschiedene Funktionen, die in einem Controller dargestellt sind. Gemeint ist damit, dass die Daten des Warenkorbs über irgend einen Mechanismus an die anzeigende Bildschirmseite übermittelt werden müssen. Der Warenkorb setzt sich aber aus Positionen zusammen, die wiederum Bücher enthalten. Aus allen drei Klassen müssen Daten in der Bildschirmansicht angezeigt werden. Über eine geschickte Verwendung des Fassaden- und Iteratormusters kann man diese Daten übermitteln, ohne die Struktur und die Implementierung des Warenkorbs offen legen zu müssen. Näheres dazu findet sich in Abschnitt 8.2. Vorgreifend kann aber schon verraten werden, dass man mit einem Iterator arbeitet, der relativ abstrakte Warenkorb-Positionen zurück liefert, über die man auf die wirklichen Positions- und Buch-Objekte zugreift. Diese neue Klasse „WkPosition“ bekommt get-Methoden für alle gewünschten Attribute von „Position“ und „Buch“.

Das Anzeigen des Warenkorb Inhalts funktioniert über einen eigenen Controller, nämlich den ViewHelper. Er baut aus einer Beschreibung des Seitenlayouts und den Daten, die er vom WarenkorbHandler im ViewDaten-Objekt übermittelt bekommt, eine entsprechende Ansicht zusammen. Der WarenkorbHandler muss dazu allerdings das Attribut „page“ mit der Methode „setPage(int,int)“ setzen, damit der ViewHelper weiß, welche Seite anzuzeigen ist. Das erste Argument beschreibt dabei die anzuzeigende Seite, während das zweite Argument einen optionalen Anzeigemodus bezeichnet. Damit kann der ViewHelper erfahren, ob zum Beispiel Fehlermeldungen in die Seite einzubauen sind oder Ähnliches.

Stückzahl aktualisieren und Preis neu berechnen Wenn der Kunde eine neue Stückzahl eingegeben hat, muss das System diese Stückzahl in die passende Position des Warenkorbs übernehmen. Das System weiß aber nicht, welche Daten der Benutzer im Warenkorb geändert hat. Deshalb gibt es eine Methode aktualisiereWarenkorb() in der Klasse „WarenkorbFassade“, die aus den Daten des Events einen neuen Warenkorb erzeugen und mit dem gespeicherten Warenkorb vergleichen kann. Dazu dient die Methode vergleicheWarenkorb() der WarenkorbFassade. Da die WarenkorbFassade die Struktur und Implementierung des Warenkorbs kennt, kann sie über die Methode setStueckzahl() der Klasse „Position“ die Stückzahl der richtigen Position aktualisieren.

Nach dem Aktualisieren der Stückzahl muss das System allerdings den Preis der Position neu berechnen. Außerdem muss es auch den Gesamtpreis des Warenkorbs neu berechnen, weil sich dieser ja aus der Summe der Positionspreise zusammensetzt. Dazu benötigt man neben get- und set-Methoden für die Preise in „Position“ und „Buch“ auch noch zwei Methoden berechnePreis() in Position und Warenkorb.

Anzeigen Das Anzeigen wurde oben bereits beschrieben. Der Mechanismus ist immer gleich, da das HTTP-Protokoll verlangt, dass auf jede Client-Anfrage eine Antwort des Servers erfolgt. Die in Kapitel 7 beschriebene Architektur des Systems trägt dieser Forderung Rechnung.

Position löschen Dieser Controller führt zu keiner neuen Methode. Er würde in Sprachen ohne Garbage Collection zu einem Destruktor für die Klasse „Position“ führen, doch Java kommt ohne Destruktoren aus.

Warenkorb erzeugen und zuordnen Dieser Controller ist beinahe ebenso unspektakulär. Er führt nur zu einem Konstruktor von Warenkorb, der einen leeren Warenkorb erzeugt und einer Methode setWarenkorb() in der Session. Das Erzeugen eines Warenkorbs sollte an dieser Stelle normaler-

8. Interaktionsmodellierung

weise gar nicht nötig sein. Es handelt sich lediglich um eine Sicherheitsprüfung, denn normalerweise sollte der Warenkorb im Anwendungsfall „Artikel in den Warenkorb legen“ erzeugt und mit Inhalt gefüllt werden.

8.1.2. Ablauf des Anwendungsfalls

Mit den soeben entwickelten Methoden ergibt sich der in Abbildung 8.1 dargestellte Ablauf des Anwendungsfalls:

Standardablauf Der Anwendungsfall wird aufgerufen, wenn der Kunde auf der Seite „Produktdetails“ auf „Artikel in den Warenkorb legen“ klickt. Das System ruft dann zunächst den Anwendungsfall „Artikel in den Warenkorb legen“ auf und direkt anschließend den Anwendungsfall „Warenkorb editieren“ (siehe Anwendungsfall B.1.1). Entsprechend der im vorigen Kapitel erläuterten Architektur erzeugt das System dann ein CommandHandler-Objekt, das als „Use Case Controller“ agiert. Es verwendet Methoden der WarenkorbFassade, um auf den Datenbestand zuzugreifen.

Im Diagramm ist gut zu erkennen, dass das Objekt O1 vom Typ „WarenkorbHandler“ die ganze Zeit über die Abläufe steuert, was dem „use case controller“-Paradigma entspricht.

Da der CommandHandler aus einem Event erzeugt wird, weiß er, welche Aktion der Benutzer ausgeführt hat, und steuert die Abläufe entsprechend. Wir nehmen an, dass der CommandHandler verschiedene Methoden besitzt, aus denen er - abhängig von dem Event, aus dem er gebildet wurde - eine Methode auswählt und deren Funktionalität ausführt. Hat der Kunde den Anwendungsfall neu gestartet, wird der Handler zunächst die Methode onWarenkorbAnzeigen() ausführen:

Zuerst prüft der WarenkorbHandler, ob ein Warenkorb vorhanden ist. Dazu benutzt er die Methode sucheWarenkorb() der WarenkorbFassade O2, die ihrerseits die Session nach dem Warenkorb-Objekt O4 des Kunden befragt. Da hier zunächst nur der Standardablauf betrachtet wird, kann man erwarten, dass tatsächlich ein Warenkorb gefunden wird und der Rückgabewert nicht „null“ ist. Daher kann der CommandHandler mit den im vorigen Kapitel erörterten Mechanismen die Seite Warenkorb erneut anzeigen. (Dazu erzeugt er ein ViewDaten-Objekt, gibt ihm einen Parameter mit, der angibt, welche Seite anzuzeigen ist, und füllt es über die Schnittstelle der WarenkorbFassade mit dem Inhalt des Warenkorbs. Abbildung 8.2 zeigt die Details dieses Zugriffs.)

Wenn der Kunde nun auf der Warenkorb-Seite auf „Aktualisieren“ klickt, startet der CommandHandler die Methode onDataGeändert(). (Ein Klick auf „Löschen“ würde zum Aufruf der Methode onPositionLöschen() führen.) Der WarenkorbHandler ruft die Methode aktualisiereWarenkorb(Event) der WarenkorbFassade auf, woraufhin diese mit der Methode erzeugeWarenkorb(Event) einen zweiten Warenkorb aus dem Event erzeugt. Dann vergleicht er diesen neuen Warenkorb mit dem aktuell gespeicherten und findet heraus, dass sich (in diesem Beispiel) eine Stückzahl geändert hat. Daraufhin aktualisiert die WarenkorbFassade die Stückzahl in der geänderten Position über die Methode setStueckzahl(). Die Position selbst berechnet nun ihren Preis mit berechnePreis() neu, weil eine geänderte Stückzahl zu einem geänderten Positionspreis führen muss. Dazu liest sie aus dem Buch-Objekt O6 den Preis für das betreffende Buch mit getPreis() aus und multipliziert ihn mit der neuen Stückzahl.

Die Neuberechnung des Preises könnte natürlich auch vom Objekt O2 angestoßen werden. Es ist aber sinnvoller, das Setzen der Stückzahl und die Neuberechnung in der gleichen Methode der Klasse „Position“ zusammen zu halten, damit keine Konsistenzprobleme zwischen Stückzahl und Preis entstehen. Diese Strategie verhindert jedoch, dass O2 von der Neuberechnung des Preises erfährt. O2

8.1. Warenkorb editieren

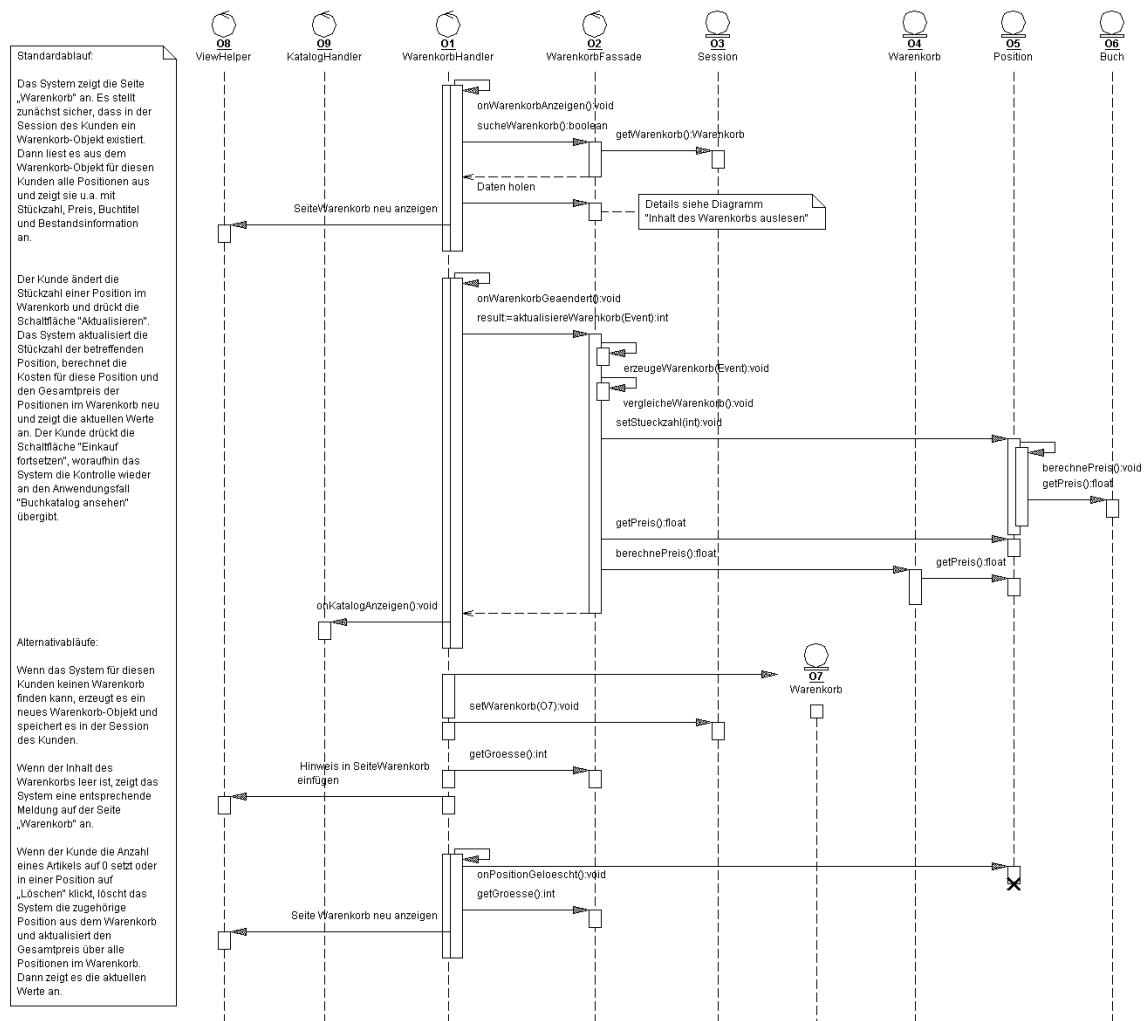


Abbildung 8.1.: Sequenzdiagramm „Warenkorb editieren“

8. Interaktionsmodellierung

weiß jedoch, dass es die Stückzahl geändert hat und fordert daher den Preis mit `getPreis()` neu an². Anschließend muss auch im Warenkorb der Gesamtpreis neu berechnet werden, weil sich der Preis mindestens einer Position geändert hat. Da hier aber die Neuberechnung vom Objekt O2 aus initiiert wird, kann der neue Preis sofort über den Rückgabewert mitgeteilt werden. Jedoch muss auch O4 vor der Neuberechnung des Gesamtpreises bei allen beteiligten Positionen (hier nur O5) die Preise abfragen.

An dieser Stelle angekommen, hat O2 dafür gesorgt, dass alle anzuzeigenden Informationen wieder aktuell sind, und könnte diese nun durch den ViewHelper darstellen lassen. Der Anwendungsfall sieht aber vor, dass das System jetzt auf die Katalogseite wechselt. Daher ruft der WarenkorbHandler die Methode `onKatalogAnzeigen()` des KatalogHandlers auf, wodurch die Kontrolle auf den Anwendungsfall „Katalog ansehen“ übergeht.

Wenn der Kunde schließlich auf „Einkauf fortsetzen“ klickt, führt dies von vornherein zum Erzeugen eines anderen CommandHandlers vom Typ KatalogHandler. Damit wird die Kontrolle von vornherein an einen anderen Use Case Controller übergeben und der Standardablauf ist beendet.

Alternativabläufe³ Der erste Alternativablauf tritt ein, wenn der WarenkorbHandler bei der anfänglichen Suche des Warenkorbs „false“ zurückbekommt. In diesem Fall erzeugt er mit dem Default-konstruktor einen neuen Warenkorb und ordnet ihn mit einem Aufruf von `setWarenkorb()` im Objekt O3 dem Kunden zu.

Der zweite Alternativablauf tritt ein, wenn der Warenkorb leer ist. Der WarenkorbHandler bemerkt diese Tatsache, wenn er einen WkIterator anfordert, der aber keine Ergebnisse liefert (siehe Abbildung 8.2). Der Handler bricht dann die Verarbeitung ab und ruft die Methode `setPage()` des ViewHelpers auf, wobei auch der zweite Parameter der Methode gesetzt wird. Der Wert dieses Parameters signalisiert dem ViewHelper, die entsprechende Fehlermeldung auf der Seite anzuzeigen.

Der dritte Alternativablauf kann zwei verschiedene Auslöser haben. Entweder klickt der Kunde in einer Position auf „Löschen“, woraufhin ein Event erzeugt wird, der die gelöschte Position mit enthält, oder die WarenkorbFassade merkt beim Vergleichen des aus den Event-Daten erzeugten Warenkorbs mit dem aktuell gespeicherten, dass eine Stückzahl auf 0 gesetzt wurde. In beiden Fällen ist der weitere Ablauf nahezu mit dem im Standardablauf identisch. Die WarenkorbFassade löscht die betreffende Position aus dem Warenkorb und setzt die bereits oben beschriebene Berechnungskaskade für die Preise in Gang.

8.2. Verwendung von Entwurfsmustern

Ein guter Softwareentwurf zeichnet sich unter anderem dadurch aus, dass auf bestimmte Qualitätskriterien Wert gelegt wird. Zu diesen Qualitätskriterien gehört zum Beispiel auch die Wiederverwendbarkeit von Teilen der Software. Im Allgemeinen wird auf dieses Kriterium besonderer Wert gelegt, weil Software in aller Regel gewartet und weiterentwickelt werden muss, meist über mehrere Jahre hinweg.

²Vorsicht ! Hier wird die implizite Annahme getroffen, dass eine Änderung der Stückzahl zur Neuberechnung der Preise führt. Dies kann bei späterer Änderung des Verhaltens zu Problemen führen, wenn sich andere Teile des Systems darauf verlassen. Hier könnte der Einsatz des Observer-Musters helfen, diese Abhängigkeit zu eliminieren.

³Der ICONIX-Prozess legt großen Wert auf die Zuordenbarkeit der Abläufe zu den entsprechenden Stellen in den Sequenzdiagrammen. Das erschwert allerdings die Darstellung, da man die Alternativabläufe nicht an der Stelle der Fallunterscheidungen darstellen kann. Daher werden für die Alternativabläufe nur die wichtigsten Elemente eingezeichnet.

Rosenberg nennt in [RoSc99] ebenfalls einige klassische Qualitätskriterien, auf die beim Softwareentwurf geachtet werden sollte: Kopplung, Geschlossenheit, Hinlänglichkeit, Vollständigkeit und Einfachheit.

Im Folgenden wird beschrieben, welche Auswirkungen die Anwendung der Entwurfsmuster „Fassade“ und „Iterator“ auf den Softwareentwurf haben. Dabei führt der Einsatz des Fassaden-Musters zu einer Reduzierung der Kopplung zwischen den Objekten, und der Einsatz des Iterator-Musters ermöglicht das Traversieren der verschiedenen im Modell vorkommenden Listen, ohne dass deren innere Struktur nach außen hin offen gelegt werden muss⁴.

Fassade

In [Gam96] wird die Fassade als ein Muster beschrieben, das verwendet werden kann, um einem komplexen Subsystem eine einfachere Schnittstelle zu geben. Anfragen von Klienten an das Subsystem finden üblicherweise über die besagte Fassadenklasse statt, die von der internen Verteilung der Verantwortlichkeiten über die Klassen innerhalb des Subsystems Kenntnis hat. Diese Fassadenklasse reicht dann die Anfragen des Klienten an die richtigen Klassen weiter und gibt dem Klienten die Ergebnisse zurück. Auf diese Weise muss ein Klient normalerweise die Klassen, die sich hinter der Fassade befinden, nicht kennen, was die Kopplung zwischen den Objekten stark reduziert.

Dieses Muster findet auch bei der WarenkorbFassade Anwendung. Dabei dient die WarenkorbFassade dem WarenkorbHandler als zentraler Zugriffspunkt auf den Warenkorb, seine aggregierten Positionen und deren aggregierte Bücher. Der CommandHandler weiß nichts von der Implementierung der Warenkorb-Struktur und davon, wie die Objekte zusammenarbeiten. Er kennt nur folgende Vereinbarung: *Ein Warenkorb besteht aus einer Reihe logischer Einheiten, die eine feste Reihenfolge haben, und die Attribute wie Autor, Titel, Einzel- und Stückpreis etc. haben. Der Warenkorb liefert einen Iterator zurück, mit dem man durch diese Einheiten iterieren kann.*

Abbildung 8.2 stellt das Auslesen des Warenkorbbinhalts durch den WarenkorbHandler dar. Dieses Beispiel macht deutlich, wie die Abhängigkeiten zwischen Objekten durch das Fassadenmuster im Entwurf des Buchshops effektiv reduziert werden können. Während hier nämlich der WarenkorbHandler die Methode `getGesamtpreis()` der WarenkorbFassade benutzt, um den Gesamtpreis des Warenkorbbinhalts zu erfahren, müsste er ohne die Fassade direkt mit dem Objekt O4 kommunizieren. Damit wäre die Trennung zwischen Model und Controller aufgeweicht, denn der WarenkorbHandler müsste eine Referenz auf die Warenkorb-Klasse besitzen und könnte nicht ohne diese Referenz funktionieren.

Entscheidet man sich nun bei der Entwurfsvariante mit Fassade für eine andere Art der Implementierung, wie zum Beispiel das Ableiten der Klasse Warenkorb von einer anderen Container-Klasse, hat das keinerlei Auswirkungen auf den Control-Teil der Architektur. Auch die innere Organisation des Warenkorbs kann geändert werden, solange die oben erwähnte Bedingung erhalten bleibt.

Iterator

Das Iterator-Muster dient, wie bereits erwähnt, zum Traversieren von Listenstrukturen, ohne die internen Strukturen der zu traversierenden Objekte offen zu legen. Da im vorliegenden Modell relativ viele Listen vorkommen, ist es wünschenswert, das Iterator-Muster so oft wie möglich einzusetzen. Dadurch wird sichergestellt, dass das Traversieren von Listen (oder anderen Containern) immer auf die gleiche Weise erfolgt.

⁴Eine genaue Beschreibung der verwendeten Entwurfsmuster findet sich in [Gam96].

8. Interaktionsmodellierung

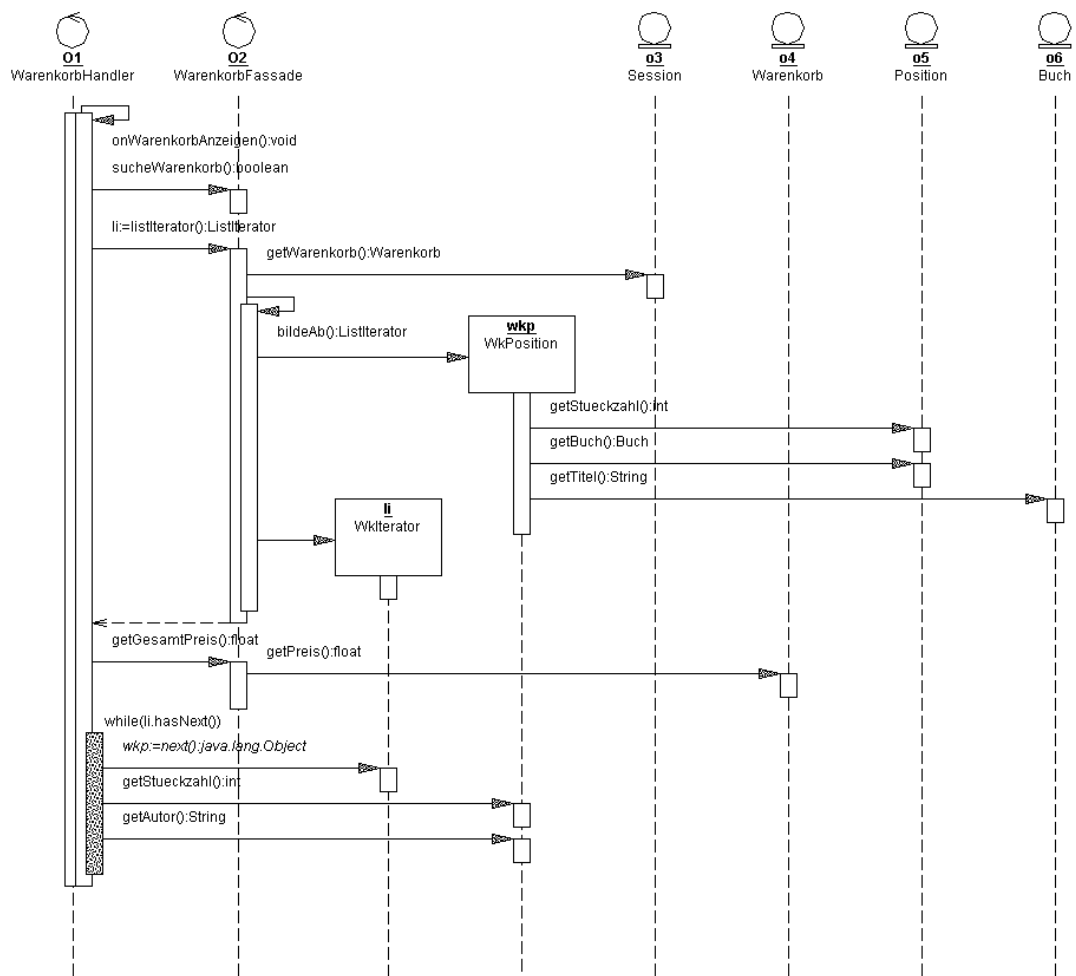


Abbildung 8.2.: Sequenzdiagramm „Inhalt des Warenkorbs auslesen“

Rosenberg schreibt dazu in [RoSc99, S. 104]:

„[...] Ich habe zwei Strategien vorgestellt, nach denen Controller in Stabilitätsdiagrammen umgewandelt werden können: 'control in the screen' und 'use case controller'. Wenn Sie sich in die eine oder andere Richtung bewegen, während Sie Ihre Sequenzdiagramme zeichnen, handelt es sich dabei um die Anwendung von Mustern.

Die Idee dabei ist, dass die Teammitglieder, die für die verschiedenen Diagramme zuständig sind, bereits früh in dieser Phase Entwurfsstandards vereinbaren sollten, die in allen Diagrammen angewandt werden können.“

Der konsequente Einsatz des Iterator-Musters etabliert einen solchen Standard. Abbildung 8.2 zeigt, wie das Iterator-Muster in der WarenkorbFassade eingesetzt werden kann, um noch weiter von der internen Organisation des Warenkorbs zu abstrahieren:

Ein Warenkorb ist aufgebaut wie ein Baum. Er besteht aus 0 bis n Positionen mit je einem Buch. Jedes Buch hat dabei mindestens einen Autor. Dies ist allerdings für die Darstellung eines Buches im Warenkorb völlig unerheblich. Hier hat man eine Informationszeile, die den Bestand, die Autoren und den Titel des Buches, die Anzahl der zu bestellenden Bücher, den Preis pro Exemplar und pro Position enthält. Außerdem gibt der Warenkorb Aufschluss über den Gesamtwert der Bestellung.

Hier kann das Iterator-Muster weiterhelfen, indem man die genannten Attribute, die zu einer solchen Zeile gehören, auf ein neues Objekt abbildet, das eine Zeile darstellt. Dieses Objekt ist in dem Beispiel vom Typ „WkPosition“. Klienten können nun bei der WarenkorbFassade einen Iterator auf WkPositionen anfordern, mit dem sie durch den Warenkorb iterieren können. Die „WkPositionen“ kennen die Objekte, die tatsächlich die Daten enthalten, und leiten die Anfragen nach den Werten der Attribute dorthin weiter. Der Klient greift also weder auf die Klassen des Models, die die tatsächlichen Daten enthalten, direkt zu noch kennt er die Struktur über die oben genannte Vereinbarung hinaus.

Implementiert würde die Liste der Warenkorb-Positionen als Vector von Objekten des Typs WkPosition in der WarenkorbFassade. Dabei kommt ein Vorteil zum Tragen, den die Wahl der Programmiersprache Java bietet. Die umfangreiche Klassenbibliothek von Java bietet den Vector-Typ schon fertig an. Glücklicherweise bietet Vector auch die Möglichkeit des Zugriffs über verschiedene bereits implementierte Iteratoren, da Vector verschiedene Interfaces implementiert. Dazu gehört auch das ListIterator-Interface. Wie in Abbildung 8.2 gezeigt, kann man also über die Methode listIterator() einen Iterator auf die Elemente des WkPositionen-Vectors erhalten. Diesen Iterator kann der WarenkorbHandler verwenden, um mit der Methode next() so lange nacheinander die Elemente des Vectors auszulesen, bis hasNext() „false“ zurück liefert.

8.3. Buchkatalog ansehen

8.3.1. Entwicklung des Detailentwurfs aus dem Stabilitätsdiagramm

In der in Abschnitt 6.1 durchgeführten Stabilitätsanalyse wurden die folgenden 9 Controller gefunden:

1. Auswahlzustand suchen,
2. Auswahlzustand auslesen,
3. Auswahlzustand erzeugen,

8. Interaktionsmodellierung

4. Auswahlzustand speichern,
5. Top-Level-Kategorien suchen,
6. Temp. Kategorielliste erzeugen / updaten,
7. Subkategorien suchen,
8. Bücher suchen,
9. Anzeigen.

Diese müssen wie folgt auf die verschiedenen Objekte verteilt werden:

Auswahlzustand Der Auswahlzustand ist genau genommen ein Objekt, das nur für die Wiederherstellung der Bildschirmansicht verwendet wird. Dennoch muss es an die Session gebunden werden, denn dieser Zustand muss über die gesamte Dauer der Einkaufssitzung gespeichert bleiben, damit der Kunde beim nächsten Ansehen der Seite wieder die gleiche Ansicht präsentiert bekommt. Zum Erzeugen des Auswahlzustands wird einfach ein Konstruktor benötigt. Dieser sollte die gewählte Kategorie und einen Vector von Kategorien als Parameter haben, die den Pfad von der Wurzel zu dieser Kategorie bezeichnen. Der Auswahlzustand muss der Session zuzuordnen sein, diese benötigt also get- und set-Methoden für den Auswahlzustand. Zum Suchen benötigt die KatalogFassade eine Methode `pruefeAuswahlzustand():boolean`, anhand deren Ergebnis sie das weitere Vorgehen bestimmen kann. Außerdem benötigt sie eine Methode `getAuswahlzustand()`, um dem KatalogHandler das Auszulesen zu ermöglichen, ohne dass dieser direkt auf die Session zugreifen muss.

Top-Level-Kategorien suchen Dieser Controller verwandelt sich in die Methode `getTopLevelKategorien():TmpKategorieListe` der Klasse „Kategorielliste“. Sie durchstöbert die Liste nach denjenigen Kategorien, deren Attribut „`istTopLevel`“ auf „`true`“ steht. Natürlich benötigt auch hier die KatalogFassade eine entsprechende Methode. Diese liefert einen Iterator auf die Kategorien zurück, die das Ergebnis der Anfrage darstellen.

Temp. Kategorielliste erzeugen / updaten Zum Speichern der temporären Kategoriellisten, die das System zum Aufbau der Katalogansicht benötigt, dient die Klasse „`TmpKategorieListe`“, die als Vector implementiert ist und somit alle benötigten Funktionen bereits besitzt. Dazu gehören die Methoden `add()` zum Hinzufügen von Elementen, `removeAllElements()` zum Zurücksetzen der Liste für den nächsten Anzeigezyklus und `listIterator()`, die einen Iterator auf die Elemente liefert.

Subkategorien suchen Zum Suchen der Subkategorien benötigt der KatalogHandler ebenfalls eine Methode in der KatalogFassade, nämlich `getSubKategorien():ListIterator`. Sie liefert einen Iterator auf Kategorien zurück, die der KatalogHandler dann dem ViewHelper übergeben kann. Zusätzlich benötigt natürlich die Kategorielliste solch eine Funktion, da die Fassade an dieser Stelle die Anfrage nur weiterleitet.

Bücher suchen Für die Büchersuche ist die Situation ähnlich gelagert. Der Buchkatalog bekommt eine Methode `getBuecher(Kategorie)`, mit deren Hilfe er einen Vector von Büchern erzeugen kann, die alle zur entsprechenden Kategorie gehören. Diese wird er allerdings in ein Suchergebnis überführen, so dass die KatalogFassade eine Methode `getBuecher(Kategorie):Suchergebnis` erhält. Dies ist im Anwendungsfall so vorgesehen, damit die Liste der gefundenen Bücher gesammelt an die Methode `getKurzinfs()` übergeben werden kann, die den Anwendungsfall „Kurzinfs anzeigen“ abhandelt.

Anzeigen Der Anzeigemechanismus wurde bereits in Kapitel 8.1.1 dargelegt und unterscheidet sich hier nicht.

8.3.2. Ablauf des Anwendungsfalls

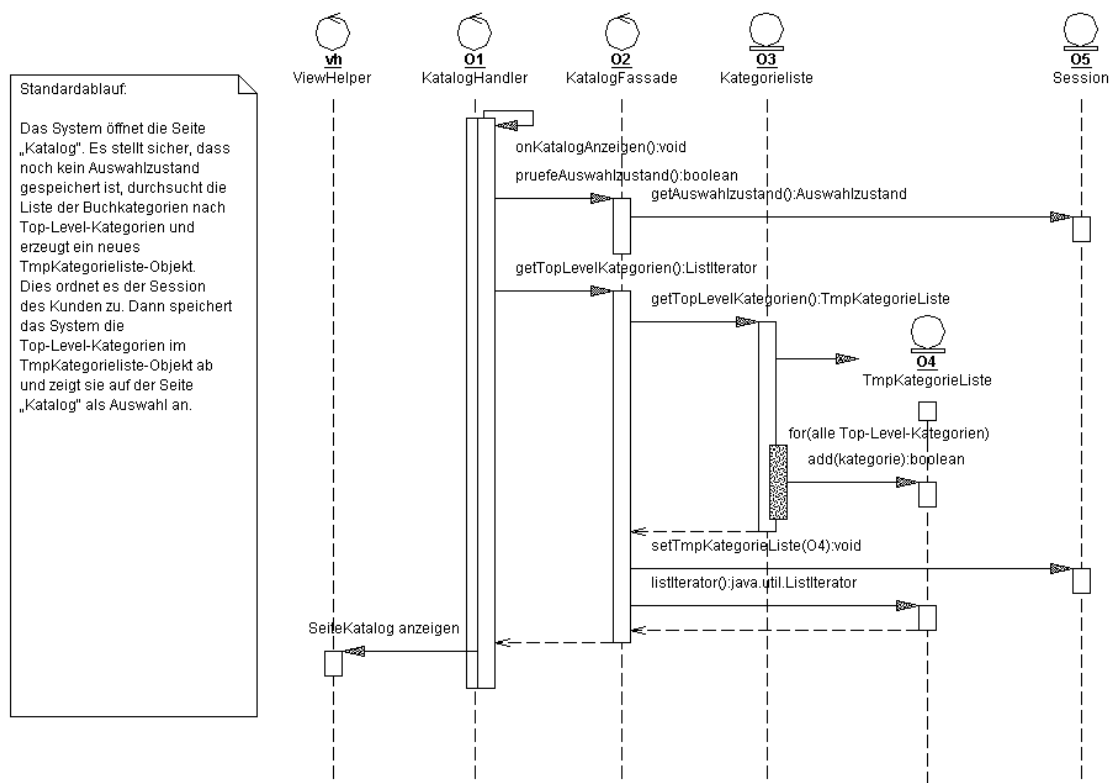


Abbildung 8.3.: Sequenzdiagramm „Buchkatalog ansehen“, Teil 1

Standardablauf Der Anwendungsfall startet durch den Aufruf der Methode `onKatalogAnzeigen()` des `KatalogHandler` O1. Dieser fragt über die Methode `pruefeAuswahlzustand()` bei der `KatalogFassade` O2 an, ob bereits ein Auswahlzustand gespeichert ist. Die Fassade versucht dann, aus der `Session` O5 mit `getAuswahlzustand()` den Auswahlzustand auszulesen. Im Standardablauf ist kein Auswahlzustand vorhanden, somit kommt hier „null“ zurück und die `KatalogFassade` meldet „false“. Mit `getTopLevelKategorien()` fordert O1 dann bei O2 die Kategorien höchster Ebene an, um sie dem

8. Interaktionsmodellierung

ViewHelper zur Darstellung zu übergeben. Dann startet O2 die Methode `getTopLevelKategorien()` in der KategorieListe. Diese ermittelt die Top-Level-Kategorien, erzeugt ein Objekt vom Typ „TmpKategorieListe“ und fügt die gefundenen Kategorien mit `add()` ein. Dann speichert es dieses Objekt mit `setTmpKategorieListe()` zur späteren Verwendung für diesen Kunden in der Session, fordert mit `listIterator()` einen Iterator auf den Inhalt bei O4 an und gibt diesen an das Objekt O2 zurück, das es an O1 weiterreicht. Dann startet O1 den ViewHelper zum Anzeigen der Seite.

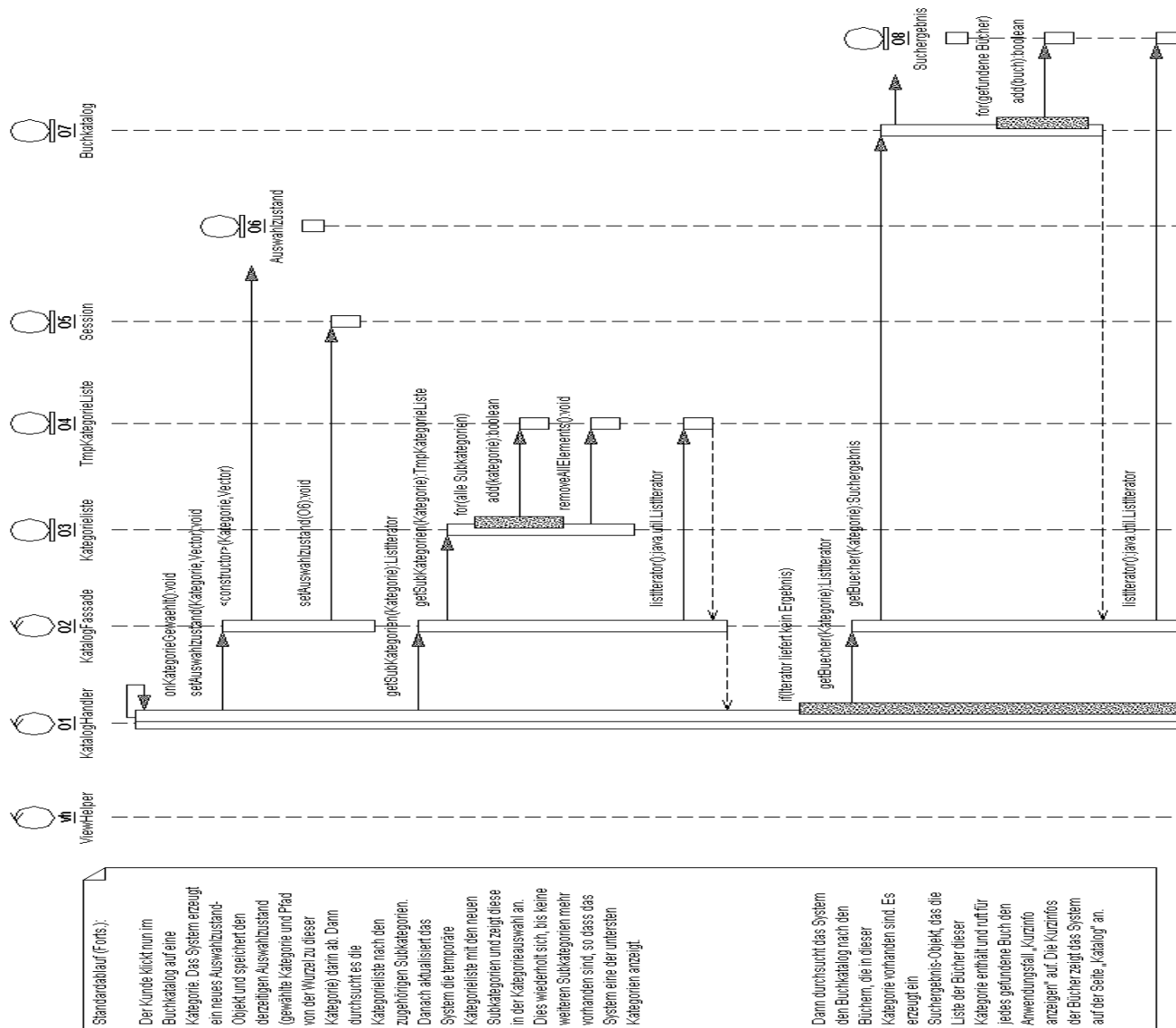


Abbildung 8.4.: Sequenzdiagramm „Buchkatalog ansehen“, Teil 2

Wenn der Kunde eine Kategorie auswählt, führt dies letztlich zum Aufruf der Methode `onKategorieGewählt()` des `KatalogHandler` O1. Dieser kennt die Daten des Events, wozu auch die ausgewählte Kategorie und der Pfad von der Wurzel zu dieser Kategorie gehören. (Der Pfad wird ja oben im Hauptanzeigebereich dargestellt.) Diese Daten werden als Parameter des Aufrufs von `setAuswahlZustand()`

8.4. Vergleich mit der Lösung von Rosenberg und Scott

übergeben. O2 erzeugt aus diesen Daten das Objekt O6 vom Typ „Auswahlzustand“ und speichert es zur späteren Verwendung in der Session O5 ab.

Nun ermittelt der KatalogHandler die Subkategorien, die zu der gewählten Kategorie gehören. Dazu ruft er die Methode `getSubKategorien()` von O2 auf. O2 leitet die Anfrage an die Kategorieliste O3 weiter. O3 ermittelt dann die Subkategorien, leert die `TmpKategorieListe` O4 mit `removeAllElements()` und fügt die neuen Kategorien mit `add()` ein. Dann gibt es die Liste an O2 zurück. O2 fordert dann einen Iterator bei O4 an, den es an O1 weitergibt.

Es ist allerdings auch möglich, dass die Kategorie eine Kategorie unterster Stufe ist. Dann enthält sie keine weiteren Subkategorien, dafür aber Bücher. Wenn O1 also feststellt, dass der Iterator kein Ergebnis liefert, versucht er, für die betreffende Kategorie die Bücherliste abzufragen. Dazu ruft der die Methode `getBuecher()` von O2 auf. O2 leitet die Anfrage an den Buchkatalog weiter, der die Suche durchführt und ein Suchergebnis O8 erzeugt. Dort fügt er die gefundenen Bücher ein und gibt das Suchergebnis an O2 zurück. O2 fordert einen Iterator bei O8 an und gibt ihn als Ergebnis der Suche an O1 weiter.

Abschließend fordert O1 für alle Bücher, die der Iterator findet, die Kurzinfos an und signalisiert dem ViewHelper, dass die Katalogseite neu aufgebaut werden muss.

Alternativabläufe Zu Beginn des Anwendungsfalls prüft das System mit `pruefeAuswahlzustand()`, ob ein Auswahlzustand gespeichert ist. Wenn das der Fall ist, werden die gleichen Abläufe wie in Teil 2 des Sequenzdiagramms durchgeführt, wobei allerdings sichergestellt werden muss, dass der Kunde nicht gerade eine Kategorie gewählt hat. Die Kategorie wird dann aus dem Auswahlzustand ausgelesen. Der obere Teil von Abbildung 8.5 stellt diesen Sachverhalt dar. Die Abläufe sind, wie gesagt, mit denen in Teil 2 identisch. Dieser Teil wird hier nicht dupliziert, sondern lediglich mit `onKategorieGewaeht()` angedeutet. Tatsächlich sind die Abläufe nicht komplett identisch, sondern erst ab `getSubKategorien()`. Man betrachte also den Methodenaufruf als Platzhalter für die erwähnte Funktionalität. Man könnte dies sinnvoll implementieren, indem man den Inhalt von `onKategorieGewaeht()` auf zwei weitere Methoden verteilt, die von `onKategorieGewaeht()` aufgerufen werden. Die eine Methode kann dann auch für den Alternativablauf genutzt werden.

Der zweite Alternativablauf tritt ein, wenn das System für eine Kategorie keine Bücher finden kann. Die Bildschirmansicht muss dann auf eine Weise präsentiert werden, die den Hinweistext enthält. Dies geschieht durch Setzen des zweiten Parameters von `setPage()`, wie bereits im vorigen Abschnitt geschildert wurde.

8.4. Vergleich mit der Lösung von Rosenberg und Scott

Abbildung 8.6 zeigt das Sequenzdiagramm, das Rosenberg und Scott für den Anwendungsfall „Warenkorb editieren“ erstellt haben. Vergleicht man die Lösung der Autoren mit der hier vorgestellten, verwundert es nicht weiter, dass sich die Mängel, die bereits in der Stabilitätsanalyse festgestellt wurden, hier fortsetzen. Nach wie vor bleibt der Fall unberücksichtigt, dass zu Beginn des Anwendungsfalls möglicherweise kein Warenkorb existiert, das Auslesen des Warenkorbs aus der Session fehlt und nach dem Ändern der Stückzahl wird der aktuelle Zustand nicht erneut angezeigt.

Auch das Fehlen von Überlegungen architektonischer Art setzt sich im Sequenzdiagramm fort. Das Sequenzdiagramm hält sich strikt an den Verlauf des Anwendungsfalls und geht zum Beispiel nicht darauf ein, wie das System die Kontrolle an den „Check Out“-Anwendungsfall übergeben kann. Dennoch ist es den Autoren ohne Weiteres möglich, ein Sequenzdiagramm zu zeichnen, das den Ablauf

8. Interaktionsmodellierung

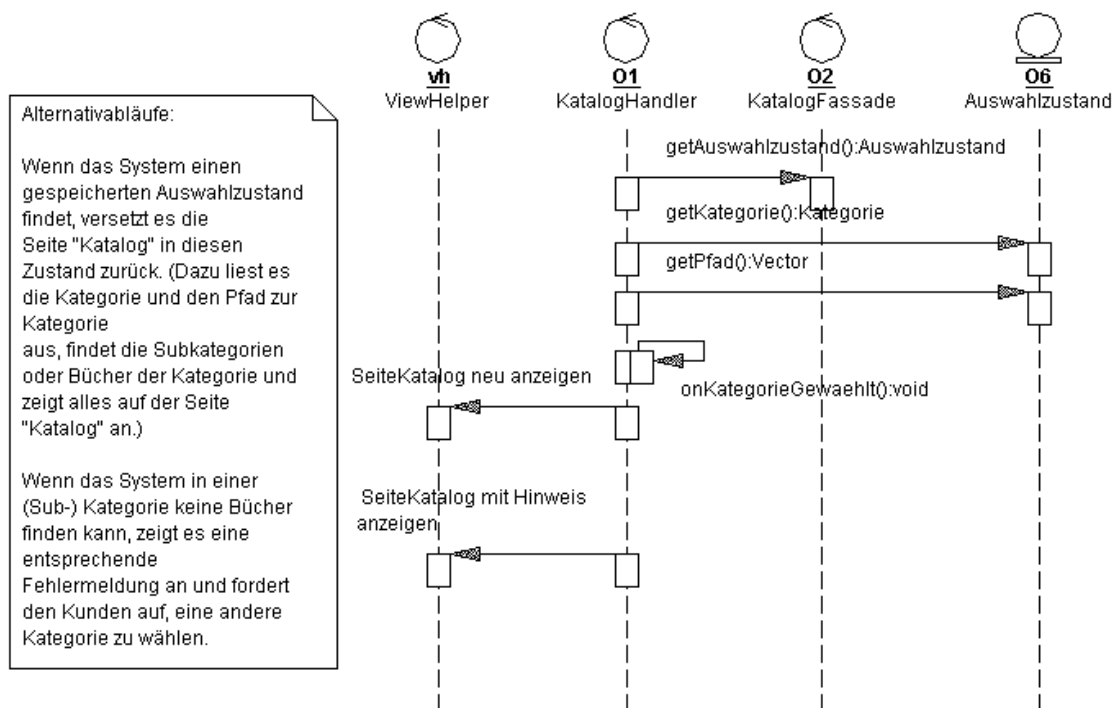


Abbildung 8.5.: Sequenzdiagramm „Buchkatalog ansehen“, Teil 3

8.4. Vergleich mit der Lösung von Rosenberg und Scott

des Anwendungsfalls darstellt. Dies enthält aber wenige Details und ist eher konzeptioneller Art. Die Autoren bezeichnen es in [RoSc01] als einen der häufigsten Fehler der Interaktionsmodellierung, nicht die Details zu zeigen, sondern die Diagramme auf einem hohen Abstraktionsniveau zu halten. Im Diagramm 8.6 soll der Unterschied zwischen der guten und der schlechten Lösung in [RoSc01] unter anderem durch das Fehlen der destroy()-Methoden am schlechten Beispiel gezeigt werden. Dieses Beispiel wirkt jedoch künstlich. Die wesentlichen Details sind überhaupt nicht erwähnt. Dazu gehört, *wie* die Preise neu berechnet werden, wenn die Stückzahl in einer Position geändert wird, und wie ein Anwendungsfall die Kontrolle an einen anderen Anwendungsfall übergibt (siehe dazu auch Kapitel 8).

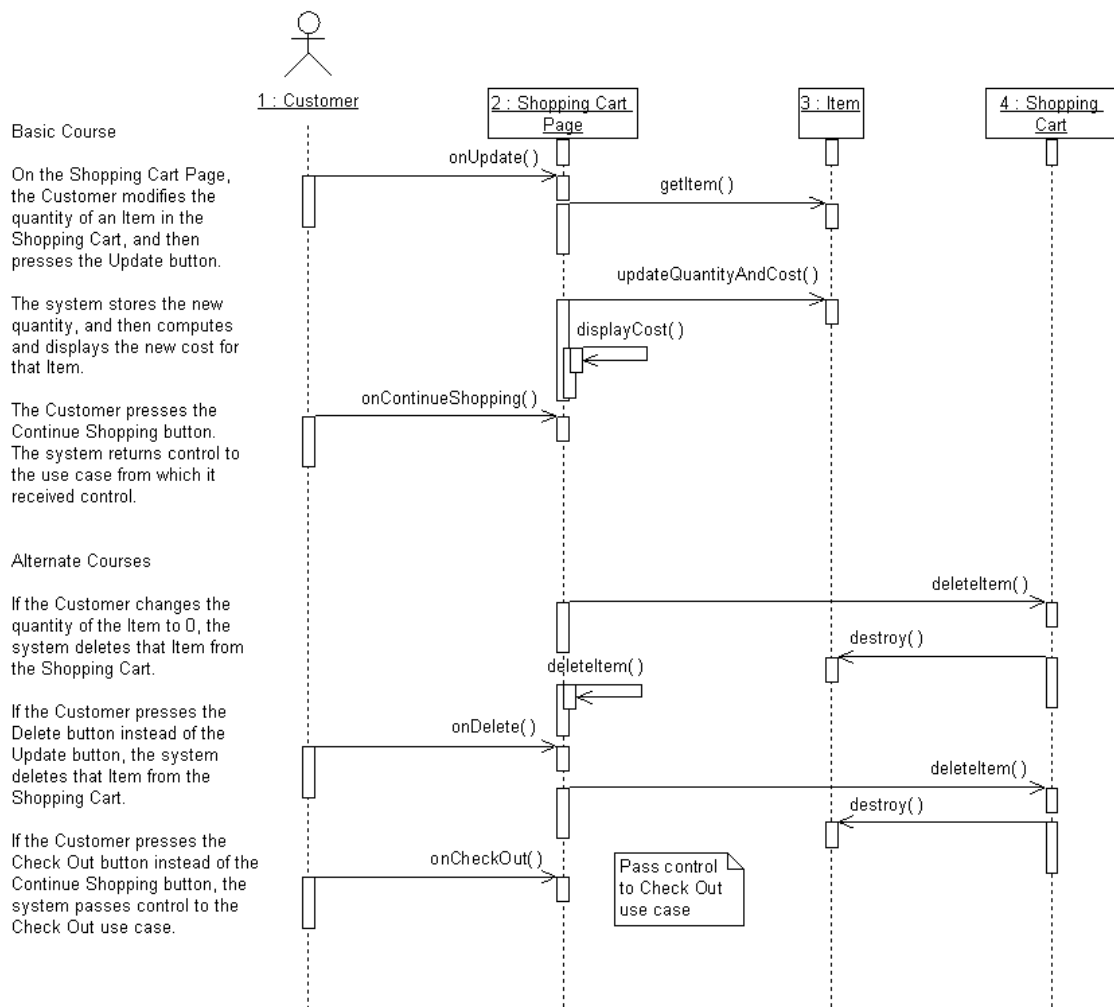


Abbildung 8.6.: Sequenzdiagramm „Warenkorb editieren“ von Rosenberg und Scott

Das weitgehende Fehlen architektonischer Überlegungen wird beim direkten Vergleich der Diagramme ebenfalls deutlich. Das Rosenberg-Diagramm enthält nämlich - abgesehen von der „Shopping Cart Page“ - keinerlei Objekte aus dem Lösungsbereich, sondern nur solche aus dem Problembereich. Das, was Rosenberg und Scott mit „Adding Infrastructure“ bezeichnen, fehlt. Wenn man sich über die in

8. Interaktionsmodellierung

Kapitel 7 beschriebenen Gegebenheiten wie Infrastruktur, Programmiersprache und Framework aber wirklich Gedanken macht, kann man es kaum vermeiden, in den Sequenzdiagrammen Objekte einzuzeichnen, die rein technischer Natur sind - außer man verzichtet bewusst darauf, weil man ohnehin vorhat, nur ein konzeptionelles Modell zu entwerfen.

Rosenberg und Scott schreiben in [RoSc99], dass es wichtig sei, in den Sequenzdiagrammen Entwurfsmuster einzusetzen, um so die Qualität des Entwurfes zu erhöhen. Auch sollen spätestens im Klassenmodell die technischen Klassen wie EJBs enthalten sein (was nahe legt, sie auch in den Sequenzdiagrammen dort einzuzeichnen, wo es sinnvoll erscheint). Beides wird der geneigte Leser in den Beispielen jedoch vergeblich suchen.

Zusammenfassend kann gesagt werden, dass ein Designer, der das von Rosenberg und Scott proklamierte Ziel der Interaktionsmodellierung ernst nimmt, nämlich ein Modell zu entwerfen, dass man mehr oder weniger unmittelbar nachprogrammieren kann, mit den Beispielen aus [RoSc01] nicht besonders glücklich sein wird, denn die wichtigen Aspekte der Interaktionsmodellierung werden dort nicht wirklich behandelt.

9. Überarbeitetes Klassendiagramm

Während der Stabilitätsanalyse und der Interaktionsmodellierung finden sich neue Objekte, Attribute und Methoden. Diese übernimmt man nach und nach in das Informationsmodell, so dass dieses zu einem vollständigen Klassenmodell heranwächst, das man als Schablone zur Programmierung des Systems nutzen kann. Im Verlaufe dieses Kapitels wird dargelegt, welche Auswirkungen die in den vorigen Kapiteln gemachten Entwurfsentscheidungen auf das statische Modell des Systems haben.

Aufgrund des Detailreichtums des Klassenmodells wird das Gesamtmodell sehr umfangreich. Die dargestellten Klassensymbole werden sehr groß und es sind viele Assoziationen vorhanden. Um der besseren Übersicht und Darstellbarkeit willen wird das Diagramm daher auf zwei kleinere Diagramme verteilt, die in etwa den Anwendungsfallpaketen zugeordnet werden können.

9.1. Kategorielliste und Buchkatalog

Abbildung 9.1 zeigt hauptsächlich den Teil des statischen Modells, der durch den Anwendungsfall „Buchkatalog ansehen“ beeinflusst wird. Dieser Teil enthält nach wie vor die betreffenden Klassen des Informationsmodells. Jetzt sind aber die Attribute und Operationen mit ihren Sichtbarkeiten in den Klassen enthalten. Der Anwendungsfall rankt sich um die Navigation durch den Produktkatalog, die maßgeblich durch die Kategorielliste und deren Organisation bestimmt wird. Die Kategorielliste war allerdings im Informationsmodell noch nicht enthalten. Sie ist eine Klasse, von der im Informationsmodell noch nicht klar war, dass sie für die Navigation im Buchkatalog wichtig sein würde. Andere neue Klassen, die in der Stabilitätsanalyse entdeckt wurden, sind die Entities „Auswahlzustand“, „TmpKategorieListe“ und „Session“. Außerdem gehören sowohl die Boundary-Klasse „SeiteKatalog“ als auch die während der Architekturfindung entdeckten Controller „KatalogHandler“, „KatalogFassade“ und „ViewHelper“ zu den neuen Klassen.

Struktur der Kategorielliste Für den Anwendungsfall „Buchkatalog ansehen“ ist die Klasse „Kategorielliste“ die wichtigste Klasse. Sie hat zwei Beziehungen zur Klasse „Kategorie“. Bei der ersten Beziehung nimmt die Kategorie die Rolle der „Wurzel“ ein. Die Kardinalität ist 1:1. Das liegt daran, dass es im ganzen System nur eine einzige Kategorielliste gibt (man würde sie gegebenenfalls als Singleton implementieren) und nur eine einzige Kategorie die Wurzel sein kann. Unterhalb der Wurzel liegen die Top-Level-Kategorien, die als erste im Buchkatalog angezeigt werden. Bei diesen Kategorien ist zusätzlich zu der Tatsache, dass sie Subkategorien der Wurzel sind, noch das Attribut „istTopLevel“ auf „true“ gesetzt. Dadurch bieten sich zwei alternative Wege zur Suche nach Top-Level-Kategorien.

Die zweite Beziehung zur Klasse „Kategorie“ ist eine Komposition mit der Kardinalität 1:n. Die Kategorien existieren nämlich nur innerhalb der Kategorielliste. Wird diese gelöscht, werden alle Kategorien ebenfalls gelöscht.

9. Überarbeitetes Klassendiagramm

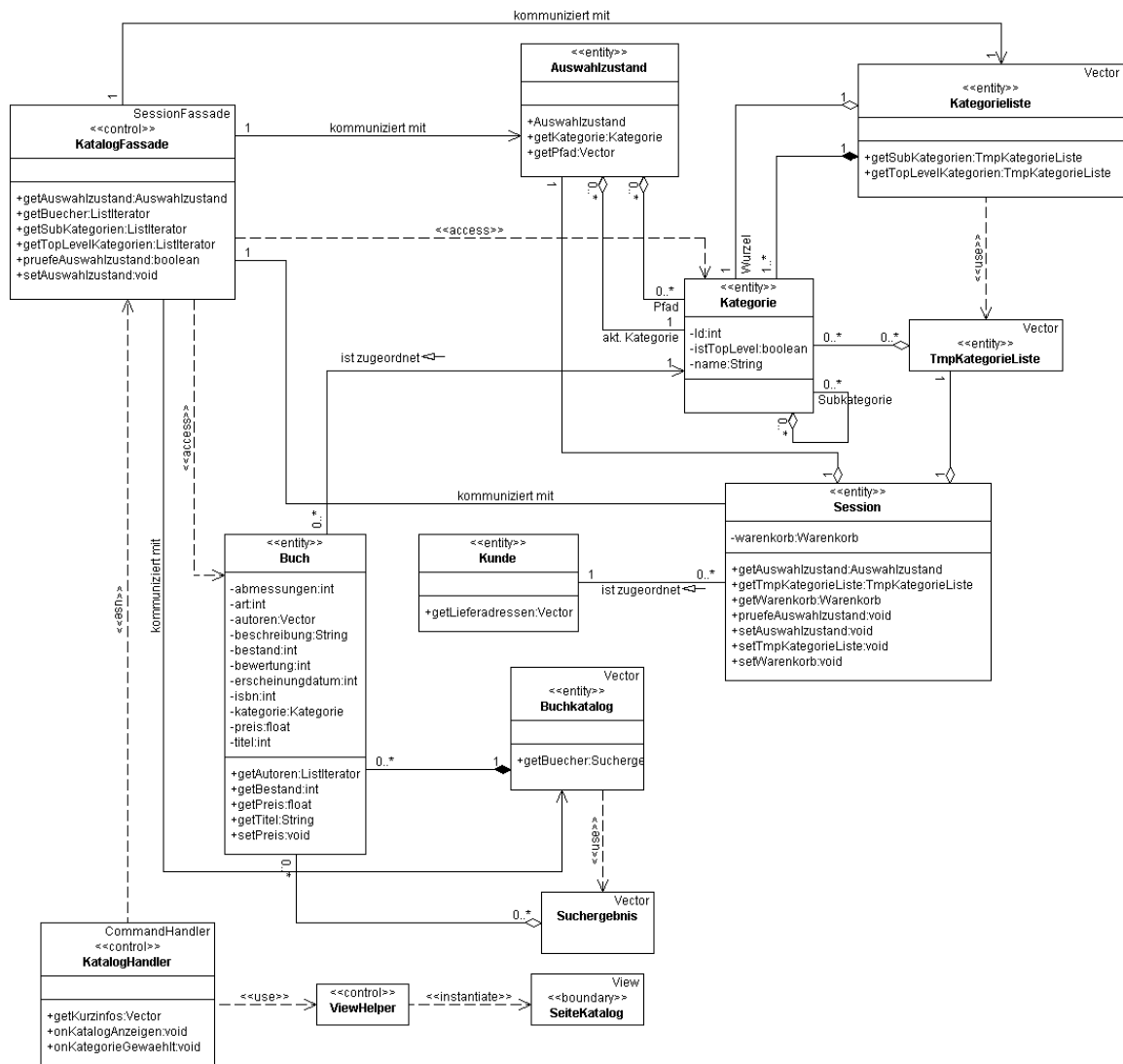


Abbildung 9.1.: Klassendiagramm „Buchkatalog“

Beziehungen der Kategorielliste Wie in früheren Kapiteln bereits mehrfach erwähnt wurde, kommuniziert der KatalogHandler nur mit der KatalogFassade. Über die Methoden `getTopLevelKategorien()` und `getSubKategorien()` erhält er von der Fassade Iteratoren, mit denen er die Kategorien auslesen kann. Dies versteckt die Implementierung und Struktur des Modells vor den Klienten der Fassade. Damit dies funktionieren kann, ist die KatalogFassade mit der Kategorielliste über die Beziehung „kommuniziert mit“ verbunden. Die Kategorielliste liefert der Fassade Referenzen auf die `TmpKategorieListe` als Ergebnis der Anfrage. Daher rührt die „use“-Abhängigkeit zwischen der Kategorielliste und der `TmpKategorieListe`. Die KatalogFassade fordert dann von der `TmpKategorieListe` den Iterator auf die Kategorien an, durch den sie auf die Klasse „Kategorie“ Zugriff bekommt. Daher rührt die „access“-Abhängigkeit zwischen KatalogFassade und Kategorie.

Damit die Zwischenergebnisse, die in den Objekten der `TmpKategorieListe` gespeichert sind, dem richtigen Kunden zugeordnet werden können, werden sie in der Session gespeichert (Aggregation von Session zu `TmpKategorieListe`). Jede Session ist genau einem Kunden zugeordnet. Der Kunde kann aber mehrere Sessions parallel benutzen.

Auswahlzustand Wenn der Kunde eine Kategorie ausgewählt hat, speichert die KatalogFassade den Auswahlzustand in der Session ab. Dazu benötigt die Session-Klasse eine Aggregationsbeziehung zum Auswahlzustand. Die Kardinalität der Beziehung ist 1:1, denn eine Session speichert genau einen Auswahlzustand (beide sind dem gleichen Kunden zugeordnet und wenn der Kunde die Auswahl ändert, ändert auch das Objekt seinen Zustand) und ein Objekt vom Typ „Auswahlzustand“ kann immer nur in einer Session gleichzeitig vorkommen. Die KatalogFassade kommuniziert mit dem Auswahlzustand, indem sie ihn instantiiert oder die Kategorie oder den Pfad ausliest. Der Auswahlzustand besteht aus einer Liste von Referenzen auf Kategorien, die den Pfad von der Wurzelkategorie zur ausgewählten Kategorie bezeichnen, und aus einer Referenz auf die aktuell gewählte Kategorie.

Buchkatalog Das Suchen von Büchern nach Kategorien funktioniert ähnlich, wie bei der Suche von Kategorien. Die Hilfsklasse ist hier das Suchergebnis, die vom Buchkatalog instantiiert und benutzt wird. Die Fassade erhält hier über den Iterator auf die Bücher, den das Suchergebnis zurück gibt, Zugriff auf die Objekte der Buch-Klasse.

9.2. Warenkorb

Struktur des Warenkorbs Dieser Teil des statischen Modells bezieht sich im Wesentlichen auf die Aspekte, die durch den Anwendungsfall „Warenkorb editieren“ beeinflusst werden. Die Kernabstraktion für diesen Anwendungsfall ist natürlich der Warenkorb. Er besteht aus Positionen, wobei die Klasse „Position“ durch Aggregation mit dem Warenkorb in Beziehung steht. Man würde vielleicht an dieser Stelle eine Komposition erwarten, da die Positionen ein fester Bestandteil des Warenkorbs sind. Wenn der Warenkorb dann zerstört würde, würden auch die Positionen verworfen. Dies ist aber nicht unbedingt gewünscht, denn der Warenkorb kann in eine Bestellung übergehen (Anwendungsfall „Buch bestellen“). Dann wäre es sinnvoll, die Positionen dort weiter zu verwenden, auch wenn der Warenkorb nicht mehr existiert.

Eine Position des Warenkorbs bezieht sich auf genau ein Buch. Ein Buch kann aber in mehreren Positionen (in verschiedenen Warenkörben) vorkommen. Daher lautet die Kardinalität zwischen Buch und Position 1:n. Schließlich kann ein Buch noch einen oder mehrere Autoren haben (es wird angenom-

9. Überarbeitetes Klassendiagramm

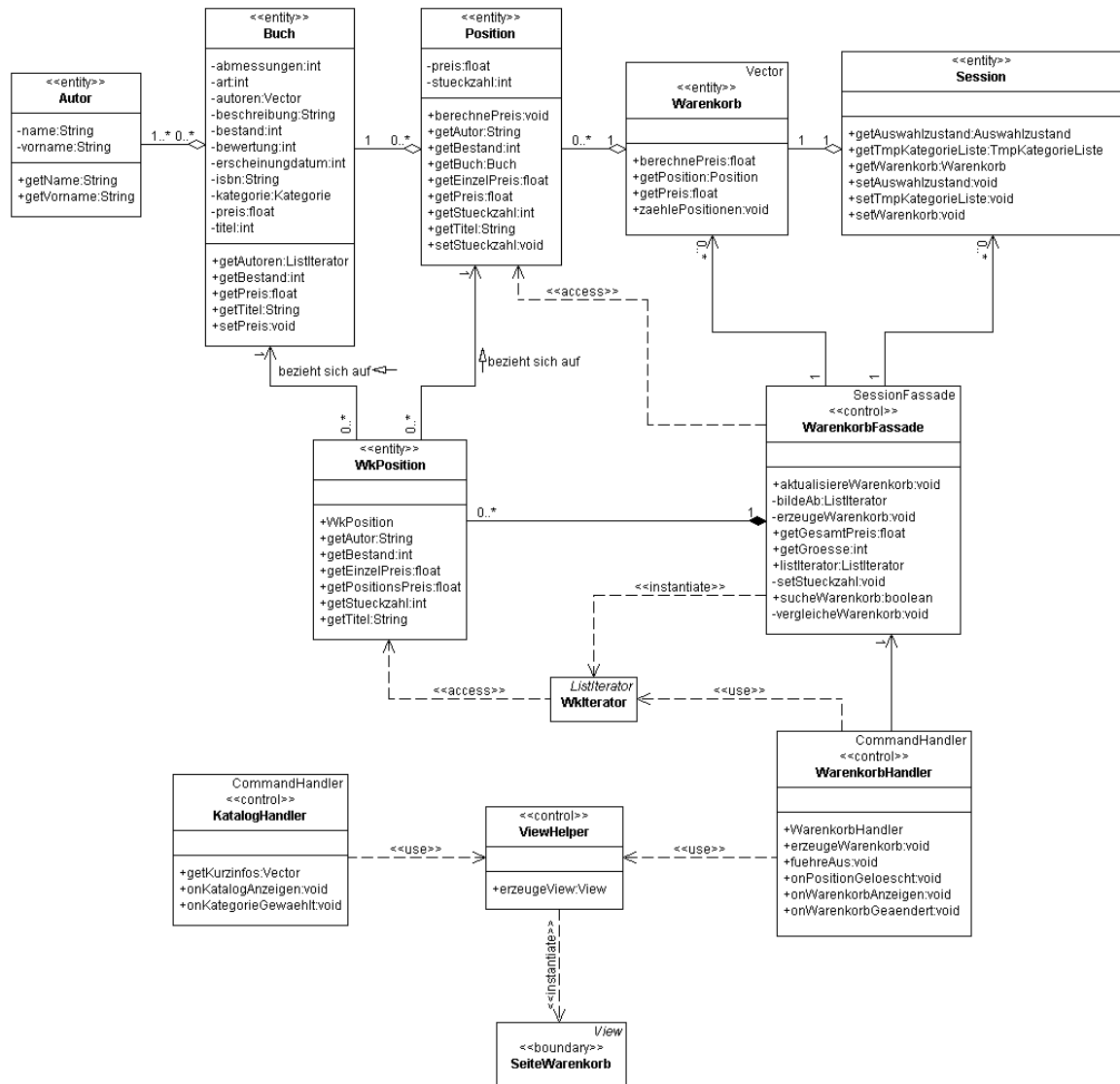


Abbildung 9.2.: Klassendiagramm „Warenkorb“

men, dass das Attribut Autor auch einen Herausgeber beschreiben kann), wobei ein Autor beliebig viele Bücher geschrieben haben kann.

Zugriff auf den Warenkorb Durch die Verwendung des Iterator-Musters wird der Zugriff auf den Warenkorb sehr elegant gelöst. Die WarenkorbFassade besitzt einen Vector von „WkPositionen“, die durch Komposition mit der WarenkorbFassade assoziiert sind. Wenn ein Klient, wie etwa der WarenkorbHandler, auf den Inhalt des Warenkorbs zugreifen will, fordert er bei der WarenkorbFassade einen Iterator auf „WkPositionen“ an. Die Fassade instantiiert den Iterator und der Klient kann ihn benutzen, um damit auf Objekte vom Typ „WkPosition“ zuzugreifen. Eine WkPosition bezieht sich auf genau eine Position des Warenkorbs und das dazugehörige Buch. Sie kommuniziert mit diesen Klassen, um über die entsprechenden Zugriffsmethoden deren Attribute auszulesen und ggf. zu speichern. Dadurch sind keine weiteren Beziehungen zwischen dem Klienten und den tieferen Strukturen des Warenkorbs nötig, was das Diagramm sehr übersichtlich macht. Dies ist auch ein Zeichen für ein gutes Design, denn der Grad der Kopplung zwischen den Klassen ist insgesamt niedrig. Damit allerdings die WkPositionen wissen, von welcher Position sie die Daten beziehen müssen, muss die WarenkorbFassade vom Warenkorb mit getPosition(int) Referenzen auf die betreffenden Positionen anfordern und den WkPositionen im Konstruktor übergeben. Dadurch entsteht eine Abhängigkeit zwischen der WarenkorbFassade und der Klasse „Position“.

Weitere Beziehungen und Abhängigkeiten Die WarenkorbFassade benötigt eine Assoziation zur Session und zum Warenkorb. Erstere wird benötigt, um mit der Methode getWarenkorb() eine Referenz auf den Warenkorb des Kunden zu erhalten, letztere dient dazu, den Gesamtpreis des Warenkorbs zu ermitteln. Die Beziehungen bzw. Abhängigkeiten zwischen den verschiedenen Controllern wurden im Kapitel 7 bereits erläutert.

Neue Objekte Dieser Teil des Modells kommt mit relativ wenigen neuen Klassen aus. Abgesehen von den architekturbedingten Control-Klassen sind dies nur die Klassen „WkIterator“ und „WkPosition“, die durch den Einsatz des Iteratormusters, der in Kapitel 8 beschrieben wurde, zustande kommt.

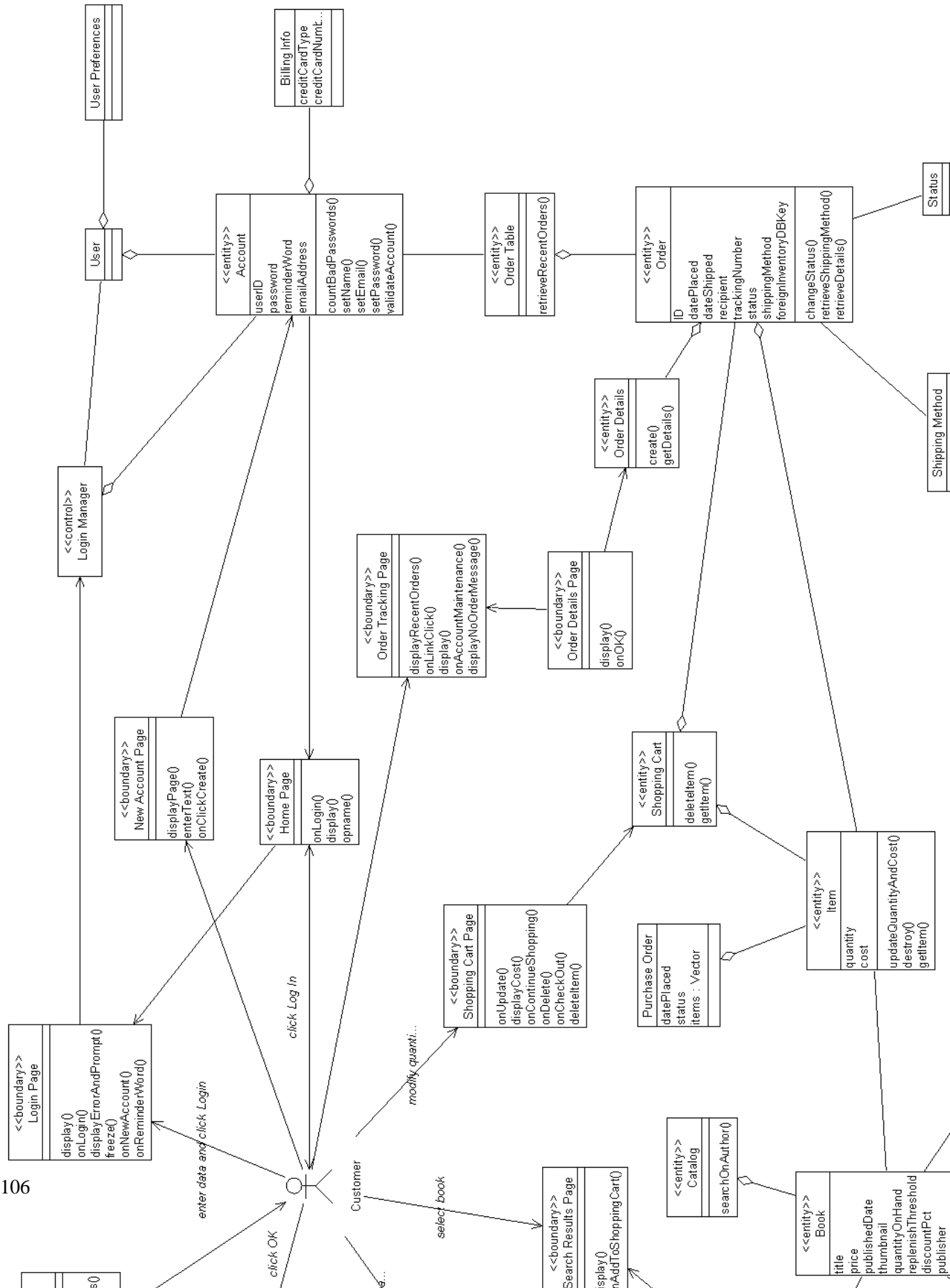
9.3. Vergleich mit der Lösung von Rosenberg und Scott

Abbildung 9.3 zeigt das vollständige Klassenmodell von Rosenberg und Scott aus dem „Full Workbook Model“. Um der besseren Vergleichbarkeit willen wurden allerdings die Teile, die die internen Abläufe des Buchshops betreffen, entfernt.

Das Modell zeigt im oberen linken Teil die Boundary-Klassen, die aus den modellierten Anwendungsfällen hervorgegangen sind. Diese besitzen die Methoden, die die Benutzeraktionen verarbeiten. In der vorliegenden Arbeit sind die entsprechenden Methoden in den CommandHandlern untergebracht. Dieser Unterschied rührt daher, dass Rosenberg und Scott bei der Umwandlung der Controller in Methoden die „control in the screen“-Strategie verfolgen, während der Autor dieser Arbeit die „use case controller“-Strategie verwendet, wobei die CommandHandler als Use Case Controller operieren.

Die wichtigen Abstraktionen, wie „Book“ und „Order“ enthalten bei Rosenberg und Scott jetzt einen vollständigen Satz von Attributen. Dies ist auch in dieser Arbeit für die Buch-Klasse der Fall. Die Klasse „Bestellung“ wird in diesem Kapitel nicht beschrieben, da für die Buchbestellung keine Sequenzdiagramme erstellt wurden. Auch sind die notwendigsten Methoden in den Klassen von Rosenberg enthalten, allerdings mit Ausnahme der get- und set-Methoden. Rosenberg und Scott empfehlen,

9. Überarbeitetes Klassendiagramm



9.3. Vergleich mit der Lösung von Rosenberg und Scott

diese Methoden nach Möglichkeit nicht mit darzustellen, da sie von den zu modellierenden Geschäftsabläufen des Systems ablenken. Der Autor dieser Arbeit ist jedoch der Ansicht, dass sie unter gewissen Umständen unbedingt auch einzuzeichnen sind, nämlich dann, wenn diese Methoden einen wichtigen Bestandteil der Geschäftsabläufe darstellen. Andernfalls gelangt man zu Sequenzdiagrammen, die vergleichsweise nichts sagend sind.

Weiterhin fehlen in Rosenberg's Modell noch die Hilfsklassen und die Klassen, die durch den Einsatz von Entwurfsmustern, durch die Wahl des Frameworks oder durch die Funktionsweise des Systems im Sinne seiner Gesamtarchitektur entstehen. Zwar betont er in [RoSc01, RoSc99] die Wichtigkeit solcher Klassen und ermahnt den Leser, sie unbedingt mit einzuzeichnen, in seinem Beispielmmodell wird man sie aber vergeblich suchen.

Auch enthält das Klassenmodell aus Abbildung 9.3 nicht sehr viele Details über die reine Nennung der Methodennamen, und teilweise auch der Attribute, hinaus. Es fehlen vor allem die Kardinalitäten und die Sichtbarkeiten. Außerdem sind keine Kompositionen eingezeichnet, die Rückgabetyphen der Methoden fehlen und an den Assoziationen stehen keine Texte, was das Modell schwerer verständlich macht.

Von einem Detailentwurf ist dies alles aber zu erwarten. Ein genauerer Blick in [TFWM] zeigt, dass die Ausgabe der besagten Details nicht einfach bei der Erstellung der Diagramme unterdrückt wurde, sondern dass sie tatsächlich nicht modelliert wurden. Das Klassenmodell erweckt also nur den Anschein eines vollständigen Modells. In Wirklichkeit kann man es in dieser Form - ähnlich wie bereits bei den Sequenzdiagrammen gesehen - noch nicht zur Programmierung des Systems verwenden.

9. Überarbeitetes Klassendiagramm

Teil III.

Schluss

10. Diskussion der Ergebnisse

Rosenberg und Scott beschreiben ihren Prozess als schlank, iterativ und inkrementell, mit einem hohen Maß an Rückverfolgbarkeit zu den anfangs formulierten Anforderungen. Auch sagen Rosenberg und Scott, dass nur der Teil der UML benutzt wird, der für das Entwickeln eines Softwareprojekts am Nützlichsten ist.

Diese Aussagen haben im Großen und Ganzen auch Bestand. Verglichen etwa mit dem Rational Unified Process, der mit Artefakten sehr überladen ist, ist der ICONIX-Prozess in der Tat schlank und zielgerichtet. Die Anzahl der verschiedenen Artefakte ist überschaubar und die eingesetzten Diagrammtypen sind gut dazu geeignet, die gefundenen Ergebnisse zu dokumentieren. Im Informationsmodell und im abschließenden Klassenmodell werden Klassendiagramme benutzt, in der Anwendungsfallanalyse Anwendungsfalldiagramme und in der Interaktionsmodellierung Sequenzdiagramme. Optional können auch noch Kollaborationsdiagramme und Zustandsdiagramme dazu kommen, sowie Verteilungs- und Komponentendiagramme für die Implementierung (die aber in dieser Arbeit nicht behandelt wird).

Die Autoren vergleichen ihren Prozess gerne mit dem Vorgehensmodell Extreme Programming (XP), mit dem Unterschied, dass im ICONIX-Prozess das Design nicht vernachlässigt wird. Auch dies ist richtig. In der Tat beschäftigen sich drei der vier Phasen des Prozesses mit der Analyse der Problemstellung, und der Prozess enthält eine eigene Designphase.

Analyse der Problemstellung Die von Rosenberg in der ersten Phase „Informationsmodell“ des ICONIX-Prozesses verwendete Methode der grammatikalischen Analyse hat sich in dieser Arbeit (Kapitel 4) als ein sehr nützliches Werkzeug erwiesen um aus der Problembeschreibung des Kunden das Informationsmodell zu gewinnen. Die im Informationsmodell gefundenen Objekte helfen sehr dabei, Anwendungsfalltexte zu formulieren, die konkret genug sind, um aus ihnen später den Entwurf der Software zu gewinnen. Das Informationsmodell ist auch deshalb sehr hilfreich, weil es dafür sorgt, dass kein System entwickelt wird, in dem die Objekte des Problembereichs keine nennenswerte Rolle spielen. Ein solches System ginge potentiell am Problem vorbei.

Parallel dazu wird in der in Kapitel 5 durchgeführten Anwendungsfallanalyse ein dynamisches Modell angefertigt. Hier schlagen Rosenberg und Scott vor, einen GUI-Prototypen zu benutzen, um sich mit dem Kunden über das gewünschte Systemverhalten zu einigen. Zwar konnte dies in Ermangelung eines realen Kunden in dieser Arbeit nicht ausprobiert werden, doch ist es leicht vorstellbar (und der Autor hat diese Erfahrung auch schon in eigener Tätigkeit gemacht), dass es viel produktiver ist, wenn man das gewünschte Systemverhalten an einem Prototypen diskutieren kann. Ein interessanter Weg, den der ICONIX-Prozess dabei aufzeigt, ist der, für den Prototypen ein Benutzerhandbuch zu schreiben und dieses in Anwendungsfälle zu übertragen. Auch diese Vorgehensweise hat sich in Kapitel 5 als sehr praktikabel erwiesen. Ist das Benutzerhandbuch erst einmal geschrieben, so sind die Anwendungsfälle daraus leicht entwickelt. Problematisch war an dieser Stelle aber, dass Rosenberg und Scott keine konkreten Wege und Hilfestellungen aufzeigen, um solch einen Prototypen zu entwickeln.

10. Diskussion der Ergebnisse

Hier zeigt diese Arbeit einen Weg auf, mit dem man - wieder durch grammatikalische Analyse der Problemstellung - zum Ziel gelangen kann, indem man erste Analyse-Anwendungsfälle schreibt, die man aus der Problemstellung gewinnt, und daraus den Prototypen entwickelt.

Einer der wichtigsten, aber leider auch am schlechtesten erklärten Schritte ist die Stabilitätsanalyse (Kapitel 6 dieser Arbeit). Sie bildet die Brücke zwischen Analyse und Entwurf und hat zwei Funktionen: Durchführen eines abschließenden Review der Anwendungsfälle und Entwickeln eines vorläufigen Designs.

Dieser Schritt ist prinzipiell sinnvoll, denn er hilft, den Blick von den zu lösenden Problemen selbst auf die *Art* zu lenken, wie diese Probleme zu lösen sind. Der Anwendungsfalltext bleibt dabei (wie auch beim späteren Design) der zentrale Aspekt. Dies führt dazu, dass man stets im Bilde ist, welche funktionalen Anforderungen umgesetzt sind und welche nicht. Dies ist bei anderen Prozessen oft ein Problem, da die Anwendungsfälle nicht 1:1 auf das Design abgebildet werden können. Daher halten Rosenberg und Scott es auch für so wichtig, ein Stabilitätsdiagramm und später ein Sequenzdiagramm aus jedem Anwendungsfall zu machen. Der Autor dieser Arbeit hält diesen Aspekt ebenfalls für sinnvoll, denn er erhöht die Rückverfolgbarkeit zu den Anforderungen und schützt davor, beim Entwurf des Systems Benutzungsszenarien zu vergessen.

In ihrer Bedeutung für die Analyse - im Sinne eines zusätzlichen Reviews der Anwendungsfälle - ist die Stabilitätsanalyse durchaus hilfreich. Während dieser Tätigkeit wurden in der Tat bei den hier durchgeführten Beispielen noch einige Schwachstellen in den Anwendungsfällen aufgedeckt.

So hilfreich die Stabilitätsanalyse allerdings in ihrer Rolle bei der Anforderungsanalyse ist, so problematisch ist sie in ihrer Rolle beim Entwurf des Systems.

Stabilitätsanalyse und Design Die größte Schwäche im ICONIX-Prozess ist das Fehlen einer Architekturfindungsphase. Rosenberg deutet an, dass erste architektonische Entscheidungen in der Stabilitätsanalyse fallen sollten. Er gibt jedoch keine Handhaben zur Architekturfindung und in seinem Beispiel sind architektonische Entscheidungen schlicht nicht zu sehen.

Wie die Ergebnisse von Kapitel 6 zeigen, führt die nach der von Rosenberg und Scott dargelegten Methodik durchgeführte Stabilitätsanalyse zu Diagrammen, nach denen man nicht unbedingt ein System programmieren kann. Kapitel 6 beschreibt am Beispiel des Kontrollflusses bei der Buchbestellung, welche Überlegungen während einer gründlich durchgeführten Stabilitätsanalyse dazu führen können, eine Architektur zu entwickeln. Dennoch kann man die Stabilitätsanalyse durchführen, ohne sich Gedanken über die Softwarearchitektur zu machen, und kann - wie Rosenberg zeigt - diese Verfahrensweise auch bei der Interaktionsmodellierung beibehalten, indem man sich streng an den Abläufen der Anwendungsfälle orientiert. Was dabei herauskommt, ist ein Modell, das (nur) gut für ein System geeignet ist, das nur einmal entwickelt wird und danach nicht weiter gewartet oder weiterentwickelt wird. Es zeichnet sich dadurch aus, dass es

- sehr statisch ist und eine starke Kopplung zwischen den Bildschirmseiten aufweist,
- sich streng an den funktionalen Anforderungen orientiert und
- kaum Wert auf Qualitätskriterien, wie etwa Wiederverwendbarkeit legt.

Letzteres kann man durch den Einsatz von Entwurfsmustern während der Interaktionsmodellierung zum Teil wieder gutmachen, aber an dieser Stelle verweist Rosenberg auf die Literatur anderer Autoren und sagt nur, dass der Designer darauf achten sollte, beim Entwurf bestimmte Qualitätskriterien einzuhalten, und dass es eine Sache von Erfahrung und Können sei, einen Entwurf gut zu machen.

An dieser Stelle wird deutlich, wo die Schwäche eines so zielgerichteten Ansatzes liegt, der sich streng an den funktionalen Anforderungen orientiert: Man verliert leicht den Blick für das Ganze. Obwohl der ICONIX-Prozess vorgibt, ein Top-Down-Ansatz zu sein, hat diese Fokussierung auf die Anwendungsfälle doch eher den Charakter eines Bottom-Up-Ansatzes.

Nach Meinung des Autors sollten die Entscheidungen über die Architektur, das Framework, die Infrastruktur und die Gewichtung der Qualitätskriterien gefallen sein, bevor man sich an das Design eines beliebigen Softwaresystems begibt. Kapitel 7 zeigt, wie man durch die Einbeziehung dieser Überlegungen zu einem geeigneten und flexiblen Architekturmodell kommt. Die dort beschriebene Architektur erzeugt, abhängig von den Eingaben des Benutzers, Events und bildet diese unter Einsatz des Command-Musters auf „CommandHandler“-Objekte ab. Da dabei die Logik eines jeden Anwendungsfalls in einem Objekt gekapselt ist, ist sowohl das Verhalten des Systems als auch die Darstellung der Daten leicht änderbar, womit vor allem der Wartbarkeit Rechnung getragen wird.

Der Grund dafür, dass die Architekturfundungsphase zu kurz kommt, ist sicherlich darin zu suchen, dass der ICONIX-Prozess anwendungsfallgesteuert ist. Anwendungsfälle beschreiben die funktionalen Anforderungen an das zu entwickelnde System. Ein anwendungsfallorientiertes Vorgehen konzentriert somit die Sicht des Entwicklers auf die funktionalen Anforderungen. Das engt aber den Blick des Entwicklers stark ein. Es fehlt die Einbeziehung der nicht funktionalen Anforderungen. Dazu gehören auch die oben beschriebenen Einflussfaktoren auf das System. Rosenberg und Scott erwähnen in ihrer Problembeschreibung zwei nicht funktionale Anforderungen: Das System soll im Internet funktionieren und es soll 1.000.000 Kunden verwalten. Beide Anforderungen bleiben bis zuletzt unberücksichtigt.

Vielleicht hilft ein Blick auf das von Karl J. Lieberherr formulierte Inventor's Paradox weiter, um sich von der bloßen Erfüllung der funktionalen Anforderungen ein wenig zu lösen. Nach diesem Muster vereinfacht man, indem man ein generelleres Problem löst, bzw. ein allgemeineres Programm schreibt. Dabei soll auf unnötiges Verteilen oder Duplizieren von Informationen im Code verzichtet werden, und besonders beachtet werden, welche Programmteile für andere bedeutsam sind.

Die leidvolle Erfahrung vieler Systemarchitekten zeigt, dass es oft unter dem Strich einfacher (und auch billiger) sein kann, wenn man eine allgemeinere Lösung für ein spezielles Problem entwickelt. Im vorliegenden Fall ist damit zum Beispiel die Eventerzeugungsmaschine in der hier entwickelten Architektur gemeint. Durch die Übersetzung der Benutzeraktionen in Events und deren Abbildung auf die in den CommandHandlern gekapselten Abläufe kann man jeden der beschriebenen Anwendungsfälle lösen - und noch viele andere mehr.

Der Autor weiß aus eigener Erfahrung in einem kommerziellen Projekt, wie teuer es werden kann, wenn man sich beim Entwurf einer Software zu sehr auf bestimmte funktionale Anforderungen beschränkt. Wenn sich später die Anforderungen ändern (und das tun sie immer), dann sollte man ein System gebaut haben, das sich nicht zu sehr gegen diese Änderungen „sträubt“.

Konkret handelte es sich um ein System zur internetbasierten Dokumentation medizinischer Studien, bei dem die Kontrolllogik mit den Datenbeschreibungen vermengt war. Dies führte dazu, dass es später kaum mehr möglich war, das Verhalten des Systems zu ändern, da es keine richtige Ablaufsteuerung gab. Letztlich wurde dieses Projekt eingestellt¹.

¹Mehr zu diesem System und seinen Problemen findet sich in der Diplomarbeit von Christoph Schönfeld: „Analyse und Architekturbaseline für ein System zur Online-Dokumentation bei klinischen Studien“ vom April 2002

10. *Diskussion der Ergebnisse*

11. Bewertung des ICONIX Prozesses

Der ICONIX-Prozess ist in der Tat ein schlanker und zielgerichteter Prozess. Er erzeugt keine Artefakte, die ungeeignet sind, den Entwicklungsprozess unmittelbar voranzutreiben. Sowohl das anfänglich entwickelte Informationsmodell als auch die Anwendungsfälle ziehen sich als treibende Kräfte durch den gesamten Prozess bis zum Detailentwurf. Seine Stärken hat der Prozess jedoch eher in der Analyse. Die dafür eingesetzten Methoden - grammatikalische Analyse und GUI-Prototyping - sind gut geeignet, um die funktionalen Anforderungen zu erheben. Was dabei aber völlig unbeachtet bleibt, sind die nicht funktionalen Anforderungen, die man auf andere, dem Prozess nicht innewohnende Weise einbringen muss.

Dennoch kann man mit der vorgestellten Methode Systeme entwickeln. Es sollten jedoch interaktive Systeme sein, die eine grafische Schnittstelle haben. Die MVC-Architektur, die dem Prozess durch die Einteilung der Objekte in Entity, Boundary und Control immanent ist, ist nämlich für solche Systeme besonders gut geeignet. Das Fehlen von Handhaben zur Entwicklung eines richtigen Architekturmodells für die Zielanwendung erhöht jedoch die Gefahr, dass man statische Systeme baut, die nicht besonders wartungsfreundlich sind und - je nach der Entwurfsstrategie, die man verwendet - eine hohe Kopplung zwischen den Seiten aufweisen. Hier muss der Entwickler viel durch Erfahrung und Können wettmachen.

Hat man jedoch vor, nur ein konzeptionelles Modell zu entwickeln, das nicht wirklich programmiert werden soll, kann man das mit diesem Prozess ohne Weiteres tun. Man kann - wie Rosenberg in seinem Beispiel zeigt - mit dem ICONIX Prozess ein Modell entwickeln, das auf den ersten Blick programmierbar wirkt und das die Geschäftsabläufe des Systems komplett darstellt. Praxistauglich ist ein solches System aber nicht.

11. Bewertung des ICONIX Prozesses

Abbildungsverzeichnis

2.1. Die Phasen des ICONIX-Prozesses (aus [RoSc99, S. 1])	15
4.1. Erste Klassen	31
4.2. Informationsmodell	33
4.3. Kunde mit Aggregationen	33
4.4. Informationsmodell von Rosenberg und Scott	34
5.1. Erste Anwendungsfälle	38
5.2. Navigation durch den Bestellprozess	40
5.3. Anwendungsfälle vor der Zerlegung	42
5.4. Navigationskonzept	43
5.5. Übersicht über die Anwendungsfälle	45
5.6. Buchbestellung	50
5.7. Buchrezension	53
5.8. Paket Buchbestellung	56
5.9. Paket Buchkatalog	56
5.10. Paket Buchrezension	57
5.11. Paket Systemanmeldung und Kontaktinformationen	57
5.12. Anwendungsfälle von Rosenberg und Scott	58
6.1. Stabilitätsdiagramm Buchkatalog ansehen	66
6.2. Stabilitätsdiagramm Warenkorb editieren	69
6.3. Stabilitätsdiagramm Buch bestellen	72
6.4. Stabilitätsdiagramm Lieferdaten bestimmen	74
6.5. Stabilitätsdiagramm „Warenkorb editieren“ von Rosenberg und Scott	75
7.1. Architektur eines J2EE-basierten Internetshops (aus [SiStJa, S. 366])	80
7.2. Architekturfragment für den Buchshop	82
7.3. Abarbeiten eines HTTP Requests	83
8.1. Sequenzdiagramm „Warenkorb editieren“	89
8.2. Sequenzdiagramm „Inhalt des Warenkorbs auslesen“	92
8.3. Sequenzdiagramm „Buchkatalog ansehen“, Teil 1	95

Abbildungsverzeichnis

8.4. Sequenzdiagramm „Buchkatalog ansehen“, Teil 2	96
8.5. Sequenzdiagramm „Buchkatalog ansehen“, Teil 3	98
8.6. Sequenzdiagramm „Warenkorb editieren“ von Rosenberg und Scott	99
9.1. Klassendiagramm „Buchkatalog“	102
9.2. Klassendiagramm „Warenkorb“	104
9.3. Vollständiges Klassenmodell von Rosenberg und Scott	106
A.1. Startseite	121
A.2. Systemanmeldung	122
A.3. Account anlegen	123
A.4. Produktübersicht	123
A.5. Produktdetails	124
A.6. Rezensionen verfassen	125
A.7. Buchbewertung abgeben	126
A.8. Büchersuche	127
A.9. Suchergebnis	128
A.10. Warenkorb	128
A.11. Lieferdaten	129
A.12. Zahlungsmodalitäten	130
A.13. Bestellung bestätigen	131
A.14. Historie der Bestellungen	132

Literaturverzeichnis

- [RoSc01] Doug Rosenberg, Kendall Scott
Applying Use Case Driven Object Modeling with UML
Addison-Wesley, Boston - San Francisco - New York u.a.
Juni 2001
- [RoSc99] Doug Rosenberg, Kendall Scott
Use Case Driven Object Modeling With UML
Addison-Wesley, Boston - San Francisco - New York u.a.
6. Auflage, August 2001
- [TFWM] The Full Workbook Model
<http://www.iconixsw.com/WorkbookExample.zip>
- [BoRuJa99] Grady Booch, Jim Rumbaugh, Ivar Jacobson
Das UML-Benutzerhandbuch
Addison-Wesley, Bonn - Reading, Massachusetts - Menlo Park, California u.a.
1. Auflage, 1999
- [HiKa99] Martin Hitz, Gerti Kappel
UML@Work
dpunkt.verlag, Heidelberg
1. Auflage, 1999
- [Jac00] Ivar Jacobson
The Road to the Unified Software Development Process
Cambridge University Press, Cambridge - New York - Melbourne - Madrid
1. Auflage, 2000
- [Rum94] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen
Objektorientiertes Modellieren und Entwerfen
Hanser Verlag, München
November 1994
- [Gam96] E. Gamma, R. Helm, R. Johnson, J Vlissides
Entwurfsmuster - Elemente wiederverwendbarer objektorientierter Software
Addison-Wesley, München - Boston - San Francisco u.a.
- [SiStJa] I. Singh, B. Stearns, M. Johnson, and the Enterprise Team
Designing Enterprise Applications with the J2EE Platform, Second Edition
Addison-Wesley, Boston - San Francisco - New York u.a.
1. Auflage, März 2002

Literaturverzeichnis

A. Prototyp und Benutzerhandbuch

A.1. Startseite (Azamon)



Abbildung A.1.: Startseite

Wenn Sie den Shop betreten, sehen Sie als erstes diese Seite. Hier können Sie auf einen Blick unsere Angebote und Neuheiten sehen. Falls Sie noch nicht wissen, was Sie kaufen sollen, können Sie sich hier einige Anregungen holen. Klicken Sie auf „Details“ unter jeder Buchbeschreibung, um in die Detailansicht des Katalogs zu wechseln und mehr über diesen Artikel zu erfahren.

Über den Link „Anmelden“ in der Titelseite der Seite können Sie sich am System anmelden. Dadurch vermeiden Sie, dass Sie sich später „zwischendurch“ anmelden müssen, wenn Sie eine Bestellung durchführen oder ein Buch rezensieren wollen.

Systemanmeldung Wenn Sie auf den Link „Anmelden“ geklickt haben, öffnet das System den Anmeldedialog. Wenn Sie schon ein Kundenkonto (Account) besitzen, so geben Sie hier einfach Ih-

The screenshot shows a web interface for logging in. At the top, there's a dark bar with the word 'Anmeldung' in white. Below this, a navigation bar contains 'Azamon' and 'Anmeldung'. A sidebar on the left lists various site functions. The central area is for login, featuring a link for password hints, a section header for login information, and fields for username and password. Two buttons at the bottom allow for logging in or creating a new account.

Abbildung A.2.: Systemanmeldung

re Benutzerkennung und Ihr Passwort ein und klicken Sie auf „Anmelden“. Falls Sie Ihr Passwort vergessen haben und bei der Registrierung Ihres Accounts einen Passworthinweis angegeben haben, können Sie über die Schaltfläche „Passworthinweis“ den Hinweis abrufen. Dann öffnet sich ein neues Fenster mit Ihrem Passworthinweis.

Neuen Account anlegen Wenn Sie noch keinen Account bei Azamon haben, klicken Sie auf die Schaltfläche „Neuen Account anlegen“. Es öffnet sich dann die Seite „Account anlegen“ (Abbildung A.3). Wählen Sie hier eine Benutzerkennung und ein Passwort aus, das Sie später zur Systemanmeldung verwenden wollen. Damit Ihnen keine Schreibfehler unterlaufen, müssen Sie das Passwort zweimal eingeben. Anschließend können Sie - wenn Sie wollen - noch den bereits erwähnten Passworthinweis angeben, den das System Ihnen präsentiert, wenn Sie Ihr Passwort vergessen haben.

A.2. Katalog

Übersicht Abbildung A.4 zeigt die Produktübersicht im Buchkatalog. Der Katalog ist der Dreh- und Angelpunkt des Shops. Hier können Sie nach Herzenslust unser Büchersortiment durchstöbern, wobei die Bücher nach Kategorien geordnet sind. Im linken oder oberen Teil des Hauptanzeigebereichs finden Sie die Kategorien, für die der Buchshop Artikel führt. Wenn Sie auf der linken Seite eine Kategorie (hier nicht zu sehen) angeklickt haben, für die es Subkategorien gibt, werden nach dem Klick auf die Kategorie die Subkategorien hier angezeigt. Wenn keine weiteren Subkategorien verfügbar sind, bleibt dieser Bereich (wie in der Abbildung) leer. Am oberen Rand des Anzeigebereichs können Sie zu den Oberkategorien zurückkehren, die Sie vorher ausgewählt hatten.

Wenn für eine Kategorie keine Bücher vorhanden sind (etwa, weil es sich um eine Oberkategorie handelt, die noch weiter aufgegliedert werden kann), erscheint im rechten Teil des Anzeigebereichs eine entsprechende Meldung. Andernfalls sehen Sie dort die Kurzzinfos zu den Büchern in dieser

Abbildung A.3.: Account anlegen

Abbildung A.4.: Produktübersicht

A. Prototyp und Benutzerhandbuch

Kategorie. Wenn Sie auf den Link „Details“ unter einer solchen Kurzinfo klicken, gelangen Sie zur Detailansicht für das entsprechende Buch.



Abbildung A.5.: Produktdetails

Detailansicht In der Detailansicht in Abbildung A.5 sehen Sie alle relevanten Informationen zum ausgewählten Buch. Dazu gehören unter anderem Angaben zu Titel, Autor, Preis und Verfügbarkeit des Artikels. Außerdem sehen Sie hier eventuell existierende Rezensionen anderer Kunden zu diesem Buch. Sie können nun das Buch über den Link oben links in den Warenkorb legen, wenn Sie es kaufen wollen. Wenn Sie selbst das Buch bewerten und/oder rezensieren wollen, klicken Sie auf den Link in der rechten oberen Ecke.

Um zurück zur zuletzt angezeigten Katalogseite zu gelangen, klicken Sie auf den Link „zurück zum Katalog“ links unter der Rezension (hier nicht sichtbar). Mit dem Link „weitere Rezensionen ansehen“ rechts unter der Rezension können Sie - falls verfügbar - weitere Rezensionen anderer Kunden zu diesem Artikel einsehen.

Rezensionen schreiben Wenn Sie in der Detailansicht auf den Link „Rezension schreiben“ klicken, gelangen Sie auf die Seite „Rezension“. Im oberen Teil des Hauptanzeigebereichs wird - zur Erinnerung - noch einmal der Titel des Buches angezeigt, das Sie rezensieren wollen. Im unteren

Rezension

Katalog Suche Warenkorb Bestellen Historie Kontakt

Azamon Katalog Produktdetails **Rezension**

Use Case Driven Object Modeling with UML
von Doug Rosenberg, Kendall Scott

Geben Sie bitte hier den Text der Rezension ein:

Rezension abschicken

Abbildung A.6.: Rezensionen verfassen

A. Prototyp und Benutzerhandbuch

Teil können Sie den Text Ihrer Rezension eingeben. Wenn Sie fertig sind, klicken Sie einfach auf „Rezension abschicken“. Bitte beachten Sie, dass es nicht möglich ist, ein Buch zu rezensieren, ohne es auch zu bewerten. Daher öffnet das System nun die Seite „Bewertung“, auf der Sie das Buch bewerten können (siehe unten). Nachdem Sie die Bewertung durchgeführt haben, zeigt das System wieder die Detailansicht des Buches an, das Sie rezensiert haben. Dort wird unterhalb der Detailinformationen über das Buch die Rezension angezeigt, die Sie soeben verfasst haben. Wenn Ihnen beim Lesen auffällt, dass Sie noch etwas korrigieren möchten, klicken Sie einfach auf den Link „Zurück zur Rezension“ unten links unter dem Rezensionstext und Sie gelangen auf die Seite „Rezension“ zurück, wo Sie ihren Text noch einmal überarbeiten und neu bewerten können.

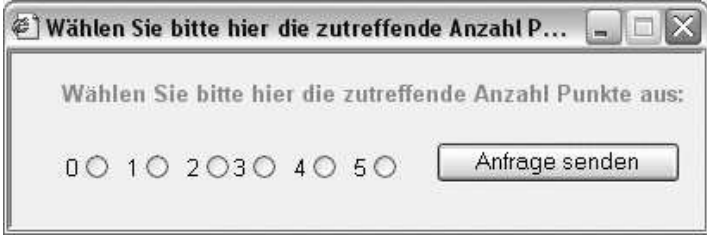


Abbildung A.7.: Buchbewertung abgeben

Bücher bewerten Zum Bewerten von Büchern steht Ihnen die Seite „Bewertung“ zur Verfügung. Diese öffnen Sie entweder von der Detailansicht des Buches mit dem Link „Buch bewerten“ oder das System präsentiert Ihnen die Seite automatisch, wenn Sie eine Rezension eintragen (s.o.). Geben Sie einfach eine Bewertung ein, indem Sie den Radiobutton anklicken, der Ihrer Meinung nach die Qualität des Buches am ehesten beschreibt. Null ist am Schlechtesten, fünf am Besten. Dann bestätigen Sie Ihre Auswahl durch Drücken der Schaltfläche „Absenden“. Das System berechnet nun einen neuen Durchschnitt aus allen Kundenbewertungen und zeigt Ihnen die aktualisierten Informationen in der Detailansicht an. Dort wird der Durchschnitt aller Kundenbewertungen in „Sternen“ ausgedrückt - ebenfalls von 0 bis 5.

A.3. Büchersuche

Diese Seite dient dazu, gezielt nach Büchern zu suchen. Wählen Sie ein Suchkriterium aus, geben Sie den Suchbegriff im unteren Feld ein und drücken Sie auf „Suche starten“. Das System durchsucht jetzt den Produktkatalog nach Büchern, die dem gewählten Suchkriterium entsprechen und zeigt sie auf der Seite „Suchergebnis“ an.

Suchergebnis Auf der Seite „Suchergebnis“ werden die gefundenen Bücher mit einer kurzen Produktbeschreibung untereinander aufgelistet. Neben jedem gefundenen Buch finden Sie einen Link, der Sie zur Detailansicht für dieses Buch bringt, wenn Sie mehr darüber wissen wollen. Wie im Abschnitt „Katalog“ beschrieben, können Sie von dort aus den Artikel auch in den Warenkorb legen. In der Detailansicht finden Sie unten links unter der Artikelbeschreibung (oder falls vorhanden der Rezension) für dieses Buch einen Link, der Sie wieder zurück zum Suchergebnis bringt, so dass sie die Detailansicht des nächsten gefundenen Buches ansehen können.

Abbildung A.8.: Büchersuche

Um eine neue Suche (ggf. mit geänderten Suchkriterien) durchzuführen, klicken Sie im Suchergebnis einfach auf „Neue Suche durchführen“.

A.4. Warenkorb

Im Warenkorb befinden sich stets die von Ihnen zum Kauf vorgemerkten Artikel. Über die entsprechende Eingabezeile können Sie hier die Anzahl der Artikel in den einzelnen Positionen ändern. Nach einer solchen Änderung aktualisiert sich der Gesamtpreis der Artikel im Warenkorb nicht automatisch. Um eine Neuberechnung anzustoßen, klicken Sie bitte auf die Schaltfläche „Aktualisieren“. Sie haben auch die Möglichkeit, über den Link „löschen“ eine Position komplett aus dem Warenkorb zu entfernen. Das gleiche erreichen Sie, wenn Sie die Anzahl der Artikel in der Position auf 0 setzen und anschließend auf „aktualisieren“ klicken. Außerdem ist es möglich, über den Link „Einkauf fortsetzen“ zurück zum Katalog zu kommen (und zwar zu der Seite, die Sie zuletzt besucht haben) oder über den Link „Bestellen“ aus dem Inhalt des Warenkorbs eine echte Bestellung zu machen, wie im nächsten Abschnitt beschrieben wird.

A.5. Bestellen

Das Bestellen von Büchern kann entweder vom Hauptmenü aus oder von der Seite „Warenkorb“ aus erfolgen. Klicken Sie auf den Link oder die Schaltfläche „Bestellen“ und die Bestellung kann beginnen. Voraussetzung, um eine Bestellung auszuführen, ist allerdings, dass der Warenkorb überhaupt Bücher enthält.

Der Bestellvorgang erfolgt in drei Schritten:

A. Prototyp und Benutzerhandbuch

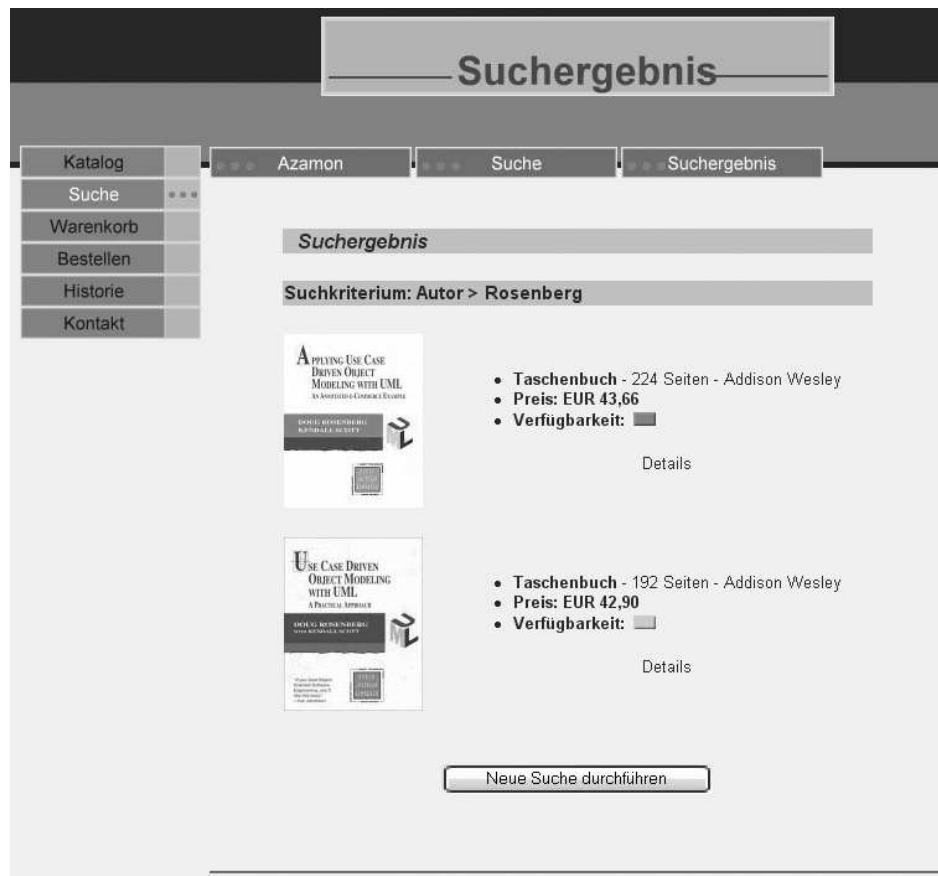


Abbildung A.9.: Suchergebnis



Abbildung A.10.: Warenkorb

Bestellen

- Katalog
- Suche
- Warenkorb
- Bestellen**
- Historie
- Kontakt

...
Azamon
...
Bestellen

Bestellung: abbrechen
⚙ NEXT

⚙ Lieferadresse auswählen

<input checked="" type="radio"/> Hans Mustermann Musterstraße 23 12345 Musterstadt Deutschland Tel. 0123-45678	<input type="radio"/> Firma Mustermann z.Hd. Hans Mustermann Musterallee 23 12345 Mustestadt Deutschland Tel. 0123-88888
--	---

Neue Adresse anlegen
Ausgewählte Adresse ändern
Ausgewählte Adresse löschen

⚙ Versandart auswählen

<input checked="" type="radio"/> Deutsche Post Express	<input type="radio"/> UPS
<input type="radio"/> Trans-O-Flex	<input type="radio"/> German Parcel

Adresse und Versandart verwenden

Abbildung A.11.: Lieferdaten

Lieferdaten angeben Im oberen Teil des Hauptanzeigebereichs finden Sie eine Liste der Adressen, die Sie als mögliche Lieferadressen angegeben haben. Wählen Sie hier eine Adresse aus, indem Sie den entsprechenden Radiobutton markieren. Falls Sie noch keine Lieferadresse hinterlegt haben, klicken Sie auf „Neue Adresse anlegen“, um die Daten für eine neue Adresse einzugeben. Sie können auch falsch eingegebene Adressdaten korrigieren, wenn Sie die Adresse markieren und auf „Ausgewählte Adresse ändern“ klicken. In beiden Fällen öffnet sich dann die Seite „Adressdaten ändern“. Machen Sie dort ihre Eingaben und klicken Sie auf „Daten abschicken“. Die Daten werden dann in die Datenbank übernommen und Sie gelangen hier auf diese Seite zurück. Sie haben hier auch die Möglichkeit, ungültig gewordene Adressdaten zu löschen, indem Sie die Adresse markieren und auf „Ausgewählte Adresse löschen“ klicken.

Anschließend müssen Sie noch die Versandart festlegen. Wählen Sie eine der Optionen aus und klicken Sie danach auf „Adresse und Versandart verwenden“, um zur nächsten Seite zu gelangen.

Abbildung A.12.: Zahlungsmodalitäten

Zahlungsmodalitäten festlegen Der zweite Schritt besteht im Angeben der Zahlungsmodalitäten. Wenn Sie noch Neukunde sind, haben Sie zunächst nur die Möglichkeit, per Kreditkarte zu bezahlen. Nach drei erfolgreichen Transaktionen können Sie dann auch auf Rechnung bestellen. Wählen Sie also aus, ob Sie per Rechnung oder per Kreditkarte bestellen wollen und prüfen Sie die Rechnungsanschrift, bzw. die Kreditkartendaten. Falls bestimmte Angaben nicht korrekt sind, klicken Sie auf

„Zahlungsdaten ändern“. Sie gelangen dann zur Seite „Zahlungsdaten anlegen“, wo Sie die betreffenden Korrekturen durchführen können. Wenn Sie auf dieser Seite dann auf „Daten ändern“ geklickt haben, gelangen Sie zurück zur Seite „Zahlungsweise“. Nachdem Sie eine Zahlungsweise markiert haben, klicken Sie auf „Ausgewählte Zahlungsweise verwenden“, um zur nächsten Seite zu gelangen.

Bestätigung

Katalog Suche Warenkorb **Bestellen** Historie Kontakt

... Azamon ... Bestellen ... **Bestätigung**

[Bestellung abbrehen](#) [BACK](#)

Bestellung bestätigen: 0000001-05-2002

Versenden an:
Hans Mustermann
Musterstraße 23
12345 Musterstadt
Deutschland
Tel. 0123-45678

Versandart:
Deutsche Post Express

Zahlungsweise: Kreditkarte
Kartennummer: 1234 1223 1234 5678
Karteninhaber: Hans Mustermann
gültig bis: 03/04

Bestellung:
1 St. **Applying Use Case Driven Object Modeling with UML**
Doug Rosenberg, Kendall Scott
EUR 46,33
Verfügbarkeit: **im Zulauf**

[Bestellung ausführen](#)

Abbildung A.13.: Bestellung bestätigen

Daten bestätigen Als letzten Schritt müssen Sie noch einmal alle Angaben auf ihre Richtigkeit überprüfen. Falls Sie noch einen Fehler entdecken, können Sie mit den „Back“ und „Next“-Symbolen am oberen Rand des Hauptanzeigebereichs durch den gesamten Bestellprozess vor und zurück navigieren und die betreffenden Änderungen durchführen. Sie können den Bestellprozess auch an jeder Stelle abbrechen, wenn Sie sich entscheiden sollten, doch nicht zu bestellen. Sind jedoch alle Daten korrekt, dann klicken Sie auf „Bestellung ausführen“, um die Bestellung in Auftrag zu geben.

A. Prototyp und Benutzerhandbuch

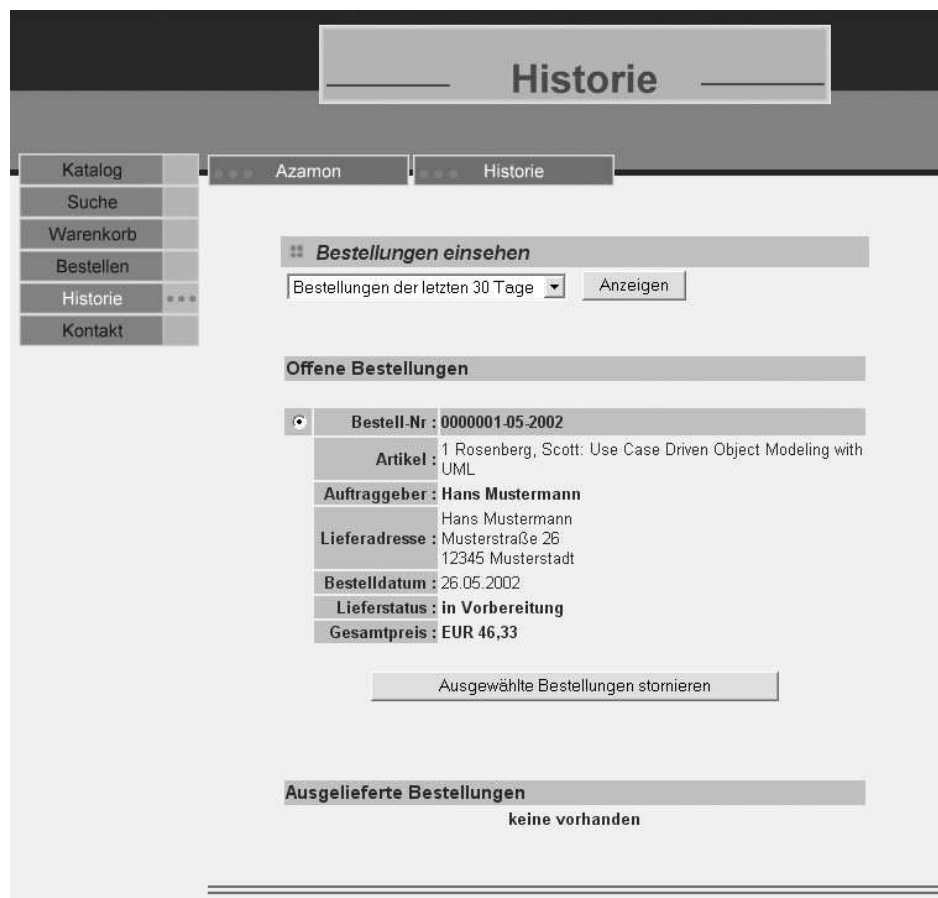


Abbildung A.14.: Historie der Bestellungen

A.6. Historie

Auf dieser Seite finden Sie eine komplette Zusammenstellung aller Bestellungen, die Sie bei uns getätigt haben. Direkt unter der Überschrift „Bestellungen einsehen“ befindet sich eine Auswahlliste mit verschiedenen Filtermöglichkeiten, von offenen Bestellungen über die Bestellungen der letzten 3 Monate bis hin zu allen Bestellungen, die Sie jemals bei uns aufgegeben haben. Suchen Sie einfach den passenden Filter heraus und klicken Sie auf „Anzeigen“. Die Liste der Bestellungen erscheint dann unmittelbar unter diesen Eingabeelementen. Falls eine Bestellung noch offen ist, also unser Lager noch nicht verlassen hat, können Sie sie auch noch stornieren.

Stornieren von Bestellungen Um eine oder mehrere Bestellungen zu stornieren, die noch nicht ausgeliefert wurden, markieren Sie einfach die entsprechende(n) Bestellung(en), indem Sie die Check-box neben der Bestellnummer aktivieren, und klicken anschließend am unteren Ende der Liste offener Bestellungen auf „Ausgewählte Bestellungen stornieren“. Die Bestellungen verschwinden dann aus der Liste und werden nicht ausgeliefert. Sie müssen sie natürlich auch nicht bezahlen, da die Rechnungsstellung immer erst erfolgt, wenn die Ware unseren Shop auch wirklich verlassen hat.

A.7. Kontakt

Falls Sie einmal technische Schwierigkeiten mit unserem Shop haben oder Fragen zu einer Lieferung auftreten, finden Sie auf dieser Seite die e-mail Adressen und die Telefonnummern unserer Vertriebsabteilung und der technischen Hotline. Wir sind stets bemüht, Sie bei auftretenden Problemen nach besten Kräften zu unterstützen.

A. *Prototyp und Benutzerhandbuch*

B. Anwendungsfälle vor der Stabilitätsanalyse

B.1. Paket Buchbestellung

B.1.1. Artikel in den Warenkorb legen

Standardablauf: Der Benutzer klickt auf der Seite „Produktdetails“ auf „In den Warenkorb legen“. Das System stellt daraufhin sicher, dass der Artikel noch verfügbar ist (also das Attribut Bestand des Buch-Objekts mindestens den Wert 1 hat), und fügt daraufhin dem Warenkorb-Objekt eine neue Position hinzu, die dieses Buch enthält und die Stückzahl 1 aufweist. Anschließend reduziert es den Bestand im betreffenden Buch-Objekt um 1 und ruft den Anwendungsfall „Warenkorb editieren“ auf.

Alternativabläufe: Wenn das Buch nicht mehr verfügbar ist, also das Attribut „Bestand“ des Buch-Objekts den Wert 0 hat, zeigt das System dem Benutzer eine entsprechende Hinweismeldung in einem eigenen Dialog an. Der Kunde klickt auf „OK“ und das System kehrt zur Seite „Produktdetails“ zurück.

Wenn für dieses Buch bereits eine Position im Warenkorb existiert, erhöht das System die Stückzahl der Position im Warenkorb um 1. Anschließend reduziert es das Bestand-Attribut des Buch-Objekts um den Wert 1.

B.1.2. Buch bestellen

Standardablauf: Das System stellt zunächst sicher, dass der Kunde angemeldet ist und dass der Warenkorb für diese Einkaufssitzung mindestens eine Position enthält. Dann erzeugt es ein neues Bestells-Objekt für diesen Kunden und kopiert den Inhalt des Warenkorbs hinein. Anschließend ruft das System nacheinander die Anwendungsfälle „Lieferdaten bestimmen“ und „Zahlungsdaten bestimmen“ auf. Dann stellt das System sicher, dass der Kunde in keinem der aufgerufenen Anwendungsfälle die Bestellung abgebrochen hat.

Anschließend zeigt das System die Seite „Bestätigung“ an. Auf dieser Seite stellt es noch einmal alle Daten der Bestellung dar. Der Kunde überprüft die Richtigkeit der Daten und klickt auf „Bestellung abschicken“. Das System fügt dann der Bestellliste die Bestellung hinzu und übergibt die Kontrolle an den Anwendungsfall, der vorher die Kontrolle hatte.

Alternativabläufe: Wenn der Kunde nicht am System angemeldet ist, ruft das System zunächst den Anwendungsfall „Anmelden“ auf.

B. Anwendungsfälle vor der Stabilitätsanalyse

Wenn der Warenkorb für diese Einkaufssitzung keine Positionen enthält, zeigt das System eine entsprechende Meldung an, bricht diesen Anwendungsfall ab und übergibt die Kontrolle an den Anwendungsfall „Buchkatalog ansehen“.

Wenn der Kunde die Bestellung abbricht, verwirft das System das Bestellungs-Objekt und kehrt zum aufrufenden Anwendungsfall zurück.

B.1.3. Bestellung ansehen

Standardablauf: Der Kunde klickt im Hauptmenü auf „Historie“. Das System stellt daraufhin sicher, dass der Kunde angemeldet ist. Dann öffnet es die Seite „Historie“. Der Kunde wählt nun ein Filterkriterium aus, nach dem das System die Bestellungen dieses Kunden filtern soll. Anschließend klickt er auf „Anzeigen“. Das System durchsucht nun die Bestellliste und zeigt alle dort gespeicherten Bestellungen des Kunden an, die dem Filterkriterium entsprechen, wobei es diese nach offenen und gelieferten Bestellungen gruppiert und absteigend chronologisch sortiert. Für alle offenen Bestellungen erzeugt das System eine Möglichkeit zur Markierung.

Alternativabläufe: Wenn der Kunde nicht am System angemeldet ist, ruft das System zunächst den Anwendungsfall „Anmelden“ auf.

Wenn das System keine offenen oder gelieferten Bestellungen finden kann, zeigt es in der entsprechenden Rubrik eine Meldung an.

Wenn der Kunde mindestens eine offene Bestellung markiert und auf „Ausgewählte Bestellungen stornieren“ klickt, speichert das System die vom Kunden gewählten Filterkriterien ab und übergibt die Kontrolle an den Anwendungsfall „Bestellung stornieren“.

Wenn dieser Anwendungsfall die Kontrolle vom Anwendungsfall „Bestellung stornieren“ zurück erhalten hat und nicht vom Kunden direkt aufgerufen wurde, lässt das System die Anmeldungsprüfung aus.

Es durchsucht die Bestellliste erneut nach den vom Kunden eingegebenen Kriterien und aktualisiert die Rubrik „Offene Bestellungen“ mit den jetzt gefundenen Bestelldaten.

Wenn der Kunde auf „Zurück zum Katalog“ klickt, übergibt das System die Kontrolle an den Anwendungsfall „Buchkatalog ansehen“.

B.1.4. Bestellung stornieren

Standardablauf: Das System löscht alle Bestellungen, die der Kunde auf der Seite „Historie“ markiert hat, aus der Bestellliste und aktualisiert die Bestandszahlen der betreffenden Buch-Objekte. Dann übergibt es die Kontrolle wieder an den aufrufenden Anwendungsfall.

B.1.5. Fehlerhafte Dateneingabe behandeln

Standardablauf: Das System stellt die betreffende Seite erneut dar, wobei es darauf hinweist, dass es Fehler bei der Dateneingabe gab. Es markiert die Felder auf der Seite, die die fehlerhaften Daten enthalten und fordert den Kunden zur Neueingabe der Daten auf.

B.1.6. Lieferdaten bestimmen

Standardablauf: Das System sucht im Kunden-Objekt nach den Lieferadressen, die für diesen Kunden gespeichert sind, und zeigt diese Adressen auf der Seite „Bestellen“ an. Außerdem ermittelt das System die zur Verfügung stehenden Versandarten und stellt sie ebenfalls auf dieser Seite dar. Der Kunde wählt eine Adresse und eine Versandart aus und klickt anschließend auf „Adresse und Versandart verwenden“. Das System stellt nun sicher, dass wirklich eine Adresse und Versandart ausgewählt ist und versieht dann die Bestellung mit der gewünschten Adresse und Versandart. Anschließend übergibt es die Kontrolle wieder an den aufrufenden Anwendungsfall.

Alternativabläufe: Wenn das System keine Lieferadressen finden kann oder der Kunde auf „Neue Adresse anlegen“ klickt, wechselt das System auf die Seite „Adressdaten“. Der Kunde gibt dort eine gültige Lieferadresse an und klickt auf „Daten abschicken“. Das System stellt daraufhin die Vollständigkeit und (soweit möglich) die Korrektheit der Adressdaten sicher und fügt die Angaben der Liste der Lieferadressen für diesen Kunden hinzu. Dann stellt es wieder die Seite „Bestellen“ mit den aktualisierten Informationen dar. Falls der Kunde keine Daten eingegeben hat, kehrt das System zur Seite „Bestellen“ zurück, ohne Daten zu speichern.

Wenn der Kunde auf „Ausgewählte Adresse ändern“ klickt, wechselt das System auf die Seite „Adressdaten“. Es trägt in die dort vorgesehen Felder für die Adressdaten die gespeicherten Daten des betreffenden Adress-Objekts ein. Der Kunde ändert die Daten und klickt auf „Daten abschicken“. Das System validiert daraufhin die Adressdaten (s.o) und aktualisiert bei erfolgreicher Validierung das Adress-Objekt. Dann zeigt es wieder die Seite „Bestellen“ mit den aktuellen Daten an.

Wenn der Kunde eine Adresse markiert und auf „Ausgewählte Adresse löschen“ klickt, löscht das System das betreffende Adress-Objekt aus der Liste der Lieferadressen für diesen Kunden und stellt die Seite „Bestellen“ erneut mit den aktuellen Daten dar.

Wenn der Kunde auf „Bestellung abbrechen“ klickt, übergibt das System die Kontrolle wieder an den aufrufenden Anwendungsfall und signalisiert diesem, dass die Bestellung abgebrochen wurde.

Wenn das System feststellt, dass der Kunde keine Versandart oder keine Lieferadresse ausgewählt hat, fordert es den Kunden auf, eine Adresse und Versandart auszuwählen.

Wenn die Validierung der Adressdaten fehlschlägt, ruft das System den Anwendungsfall „Fehlerhafte Dateneingabe behandeln“ auf.

B.1.7. Warenkorb editieren

Standardablauf: Das System zeigt die Seite „Warenkorb“ an. Dann öffnet es das Warenkorb-Objekt für diesen Kunden und zeigt alle dort gespeicherten Positionen mit Stückzahl, Preis, Buchtitel und Bestandsinformation an. Der Kunde ändert die Stückzahl einer Position im Warenkorb und klickt auf „Aktualisieren“. Das System speichert die neue Stückzahl der betreffenden Position, berechnet die Kosten für diese Position und den Gesamtpreis über alle Positionen im Warenkorb neu und zeigt die aktuellen Werte an. Der Kunde klickt dann auf „Einkauf fortsetzen“, woraufhin das System die Kontrolle wieder an den Anwendungsfall „Buchkatalog ansehen“ übergibt.

Alternativabläufe: Wenn der Kunde die Anzahl eines Artikels auf 0 setzt, löscht das System die zugehörige Position aus dem Warenkorb und aktualisiert den Gesamtpreis über alle Positionen im Warenkorb.

B. Anwendungsfälle vor der Stabilitätsanalyse

Wenn der Kunde anstatt auf „Aktualisieren“ auf „Löschen“ klickt, löscht das System die zugehörige Position aus dem Warenkorb und aktualisiert den Gesamtpreis über alle Positionen im Warenkorb.

Wenn der Kunde anstatt auf „Einkauf fortsetzen“ auf „Bestellen“ klickt, ruft das System den Anwendungsfall „Buch bestellen“ auf.

B.1.8. Zahlungsdaten bestimmen

Standardablauf: Das System zeigt die Seite „Zahlungsweise“ an. Es durchsucht das Kunden-Objekt nach den Kreditkartendaten und der Rechnungsanschrift des Kunden und stellt sicher, dass diese vorhanden sind. Dann zeigt es die Daten in den jeweils dafür vorgesehenen Abschnitten an. Der Kunde wählt als Zahlungsweise Kreditkartenzahlung oder Rechnungsbestellung aus und klickt auf „Ausgewählte Zahlungsweise verwenden“. Das System stellt nun sicher, dass der Kunde wirklich eine Auswahl getroffen hat und fügt der Bestellung die gewählten Zahlungsdaten hinzu. Dann gibt es die Kontrolle an den aufrufenden Anwendungsfall zurück.

Alternativabläufe: Wenn das System keine Rechnungsanschrift finden kann und der Kunde bereits mindestens dreimal erfolgreich per Kreditkarte bezahlt hat, weist das System den Kunden auf der Seite „Zahlungsweise“ darauf hin, dass er auch per Rechnung bestellen darf und dass er dafür jetzt per „Zahlungsdaten ändern“ Rechnungsdaten anlegen kann.

Wenn der Kunde auf „Zahlungsdaten ändern“ klickt, öffnet das System die Seite „Zahlungsdaten anlegen“. Das System zeigt nun die Kreditkartendaten des Kunden im entsprechenden Abschnitt an. Wenn bereits eine Rechnungsanschrift vorliegt, zeigt das System diese ebenfalls im vorgesehenen Abschnitt an. Der Kunde ändert nun die gewünschten Daten und klickt anschließend auf „Daten abschicken“. Das System prüft dann die Kreditkartendaten und die Rechnungsanschrift soweit wie möglich auf Konsistenz, aktualisiert die entsprechenden Attribute des Kunden-Objekts und speichert sie ab. Anschließend wechselt es wieder auf die Seite „Zahlungsweise“.

Wenn das System die Seite „Zahlungsdaten ändern“ anzeigt und der Kunde noch nicht dreimal erfolgreich per Kreditkarte bezahlt hat, bietet das System keine Möglichkeit zur Eingabe der Rechnungsdaten.

Wenn das System die Seite „Zahlungsdaten ändern“ anzeigt und keine Rechnungsdaten gespeichert sind, der Kunde aber schon mindestens dreimal erfolgreich per Kreditkarte bezahlt hat, stellt das System die Felder für die Rechnungsbestellung leer dar, so dass der Kunde Daten eintragen kann.

Wenn das System im Kunden-Objekt keine Kreditkartendaten finden kann, öffnet es die Seite „Zahlungsdaten anlegen“. Es zeigt dort im Abschnitt für die Kreditkartendaten leere Felder an, in denen der Kunde die Daten seiner Kreditkarte einträgt. Der Kunde klickt dann auf „Daten abschicken“. Das System prüft dann die Kreditkartendaten auf Konsistenz, aktualisiert die entsprechenden Attribute des Kunden-Objekts und speichert sie ab. Anschließend wechselt es wieder auf die Seite „Zahlungsweise“.

Stellt das System bei der Datenvalidierung Fehler fest, so ruft es den Anwendungsfall „Fehlerhafte Dateneingabe behandeln“ auf.

Wenn das System feststellt, dass der Kunde keine Zahlungsweise ausgewählt hat, fordert es den Kunden auf, eine Auswahl zu treffen.

B.2. Paket Buchrezension

B.2.1. Buch bewerten

Standardablauf: Das System öffnet die Dialogbox „Buch bewerten“. Dort wählt der Kunde aus der Liste möglicher Buchbewertungen eine Option aus und klickt auf „Bewertung abschicken“. Das System stellt daraufhin sicher, dass wirklich eine Bewertung ausgewählt wurde. Anschließend errechnet es aus den bisher abgegebenen Bewertungen für das Buch und der soeben eingegebenen einen neuen Bewertungsdurchschnitt und aktualisiert das Attribut „Bewertung“ im betreffenden Buch-Objekt. Danach kehrt das System zum aufrufenden Anwendungsfall zurück.

Alternativabläufe: Wenn der Kunde keine Bewertung aus der Liste auswählt, bevor er auf „Bewertung / Rezension abschicken“ klickt, fordert das System ihn auf, eine Bewertung abzugeben.

B.2.2. Buch rezensieren

Standardablauf: Auf der Seite „Rezension“ gibt der Kunde im vorgesehenen Feld für den Rezensionstext einen Text ein. Um sicherzustellen, dass der Kunde das Buch außerdem auch bewertet, ruft das System den Anwendungsfall „Buch bewerten“ auf.

Das System fügt nun der Liste der Rezensionen für dieses Buch ein neues Rezensions-Objekt mit dem soeben eingegebenen Text und der Kennung des Kunden als Autor hinzu. Anschließend kehrt es auf die Seite „Produktdetails“ zurück, wobei es die Rezension des Kunden als erste Rezension anzeigt. Außerdem erzeugt das System auf dieser Seite einen Link, mit dem der Kunde den Anwendungsfall „Rezension überarbeiten“ aufrufen kann.

B.2.3. Rezension vorbereiten

Standardablauf: Der Kunde klickt auf der Seite „Produktdetails“ auf „Rezension schreiben“.

Das System stellt sicher, dass der Kunde angemeldet ist. Dann durchsucht es die Liste der Rezensionen zum betreffenden Buch und stellt sicher, dass es noch keine Rezension gibt, die den Kunden als Autor hat. Anschließend ruft es den Anwendungsfall „Buch rezensieren“ auf.

Alternativabläufe: Wenn der Kunde nicht angemeldet ist, ruft das System zunächst den Anwendungsfall „Anmelden“ auf.

Wenn das System zu diesem Buch ein Rezensions-Objekt findet, das den Kunden als Autor hat, ruft es den Anwendungsfall „Rezension überarbeiten“ anstelle von „Buch rezensieren“ auf.

B.2.4. Rezension überarbeiten

Standardablauf: Das System wählt aus der Liste der Rezensionen für dieses Buch die vom Kunden geschriebene aus und zeigt auf der Seite „Rezension“ den Text im Feld für den Rezensionstext an. Der Kunde ändert den Text der Rezension und klickt auf „Rezension abschicken“.

Das System aktualisiert nun den Rezensionstext im entsprechenden Rezensions-Objekt. Danach kehrt das System zur Seite „Produktdetails“ zurück, wo es diese Rezension als erste Rezension anzeigt. Außerdem zeigt es ein Navigationselement an, mit dem der Kunde den Anwendungsfall „Rezension überarbeiten“ erneut aufrufen kann.

B.3. Paket Systemanmeldung und Kontaktinformationen

B.3.1. Anmelden

Standardablauf: Der Kunde klickt auf der Startseite auf den Link „Anmelden“. Das System öffnet die Seite „Anmeldung“. Dort gibt der Kunde seine Benutzerkennung und sein Passwort ein und klickt dann auf die Schaltfläche „Anmelden“. Das System validiert die Anmeldeinformationen gegen die in der Datenbank gespeicherten Benutzerdaten und stellt dabei sicher, dass der betreffende Account nicht gesperrt ist. Nach der erfolgreichen Validierung meldet es den Kunden an, indem es das Account-Objekt in den Zustand „angemeldet“ versetzt. Danach leitet das System den Kunden zurück zur Startseite.

Alternativabläufe: Wenn der Kunde diesen Anwendungsfall nicht von der Startseite aus aufgerufen hat, sondern das System ihn aufrief, weil der Benutzer eine autorisierte Aktion durchführen wollte ohne angemeldet zu sein, leitet das System den Benutzer nach erfolgreicher Abwicklung des Anmeldeprozesses zu dem Anwendungsfall weiter, den der Benutzer eigentlich aufgerufen hatte.

Wenn der Kunde auf der Anmeldeseite auf die Schaltfläche „Neuen Account anlegen“ klickt, startet das System den Anwendungsfall „Neuen Account eröffnen“.

Wenn der Kunde auf der Anmeldeseite auf die Schaltfläche „Passworthinweis“ klickt, zeigt das System den für den Kunden gespeicherten Passworthinweis in einem separaten Dialogfenster an. Wenn der Kunde in diesem Fenster auf „OK“ klickt, wird es wieder geschlossen und er befindet sich wieder auf der Anmeldeseite.

Wenn das System die Benutzerdaten nicht erfolgreich validieren kann (der Kunde gibt eine falsche Benutzerkennung oder ein falsches Passwort an), gibt das System eine Meldung aus, die besagt, dass die Kennung oder das Passwort falsch sind. Anschließend fordert es den Kunden zur Neueingabe der Daten auf und weist ihn darauf hin, dass er vielleicht noch keinen Account hat.

Gibt der Kunde das Passwort für eine existierende Benutzerkennung dreimal hintereinander falsch ein, sperrt das System diesen Account und fordert den Kunden auf, sich beim Kundendienst zu melden.

Wenn der Account des Kunden gesperrt ist, meldet das System die Daten der fehlgeschlagenen Logins, die zur Sperrung führten und fordert den Kunden auf, sich beim Kundendienst zu melden.

B.3.2. Kontaktinformationen ansehen

Standardablauf: Der Kunde klickt im Hauptmenü auf „Kontakt“. Das System öffnet daraufhin die Seite „Kontakt“. Der Kunde liest nun die benötigten Kontaktinformationen.

B.3.3. Neuen Account eröffnen

Standardablauf: Der Kunde klickt auf der Seite „Anmelden“ auf „Neuen Account eröffnen“. Das System zeigt dann die Seite „Account anlegen“ an. Der Kunde gibt hier eine frei gewählte Benutzerkennung und eine gültige e-mail Adresse an und wählt ein Passwort aus, das er zweimal eingibt. Optional kann er auch einen Passworthinweis eingeben, der helfen kann, wenn der Kunde sein Passwort vergessen hat. Der Kunde drückt anschließend die Schaltfläche „Neuen Account anlegen“. Das System stellt darauf hin sicher, dass die angegebene e-mail Adresse gültig ist und dass nicht bereits ein

Account mit der gleichen Benutzerkennung vorhanden ist, und fügt anschließend einen neuen Account mit diesen Daten in die Accounttabelle ein. Dann meldet es den Kunden am System an, indem es das Account-Objekt in den Zustand „angemeldet“ versetzt. Abschließend übergibt es die Kontrolle wieder an den aufrufenden Anwendungsfall.

Alternativabläufe: Wenn der Kunde keine Benutzerkennung angibt, zeigt das System eine entsprechende Fehlermeldung an und fordert den Kunden auf, eine Kennung einzugeben.

Wenn der Kunde eine e-mail Adresse angibt, die nicht der korrekten Syntax entspricht, oder die e-mail Adresse fehlt, zeigt das System eine entsprechende Fehlermeldung an und fordert den Kunden zur Eingabe einer gültigen e-mail Adresse auf.

Wenn der Kunde ein zu kurzes Passwort angegeben hat, zeigt das System eine entsprechende Fehlermeldung an und fordert den Kunden zur Eingabe eines längeren Passworts auf.

Wenn der Kunde das Passwort nicht wiederholt hat, oder die zweite Eingabe nicht der ersten entspricht, zeigt das System eine entsprechende Fehlermeldung an und fordert den Kunden auf, die Eingabe zu wiederholen.

Wenn bereits ein Account in der Accounttabelle existiert, der die gleiche Kennung aufweist, zeigt das System eine entsprechende Fehlermeldung an und fordert den Kunden auf, eine andere Kennung auszuwählen.

B.4. Paket Buchkatalog

B.4.1. Buchkatalog ansehen

Standardablauf: Das System öffnet die Seite „Katalog“. Es durchsucht die Liste der Buchkategorien nach Top-Level-Kategorien und zeigt diese an.

Der Kunde klickt nun im Buchkatalog auf eine Kategorie. Das System speichert daraufhin die gewählte Kategorie ab und zeigt alle möglichen Subkategorien für diese Kategorie an. Dies wiederholt sich, bis keine weiteren Subkategorien mehr vorhanden sind, so dass das System eine der untersten Subkategorien anzeigt. Dann durchsucht das System den Buchkatalog nach den Büchern, die in dieser Kategorie vorhanden sind, und ruft für jedes gefundene Buch den Anwendungsfall „Kurzinformat anzeigen“ auf.

Alternativabläufe: Wenn das System von einem anderen Anwendungsfall zu diesem zurückkehrt, wählt das System die abgespeicherte Kategorie aus und zeigt die darin enthaltenen Bücher an. (Es stellt den Zustand wieder her, den der Kunde vor dem Verlassen der Seite gesehen hat).

Wenn das System in einer (Sub-)Kategorie keine Bücher finden kann, zeigt es eine entsprechende Fehlermeldung an und fordert den Kunden auf, eine andere Kategorie zu wählen.

B.4.2. Buchkatalog durchsuchen

Standardablauf: Der Kunde klickt im Hauptmenü auf „Suche“. Daraufhin öffnet das System die Seite „Suche“. Der Kunde wählt ein Suchkriterium aus der Liste aus und gibt den oder die Suchbegriffe ein. Dann klickt er auf „Suche starten“. Das System durchsucht nun den Buchkatalog nach Büchern, die den gewählten Kriterien entsprechen. Daraufhin öffnet das System die Seite „Suchergebnis“ und ruft für alle gefundenen Bücher den Anwendungsfall „Kurzinformat anzeigen“ auf.

B. Anwendungsfälle vor der Stabilitätsanalyse

Alternativabläufe: Wenn der Kunde keinen Suchbegriff eingibt, öffnet das System einen Dialog mit einem Warnhinweis, der den Benutzer zur Eingabe eines Suchbegriffs auffordert. Der Kunde klickt in diesem Dialog auf „OK“ und gelangt auf die Seite „Suche“ zurück.

Wenn das System kein Buch finden kann, das den ausgewählten Kriterien genügt, zeigt es auf der Seite „Suchergebnis“ eine entsprechende Meldung an.

Wenn der Kunde auf der Seite „Suchergebnis“ auf „Neue Suche durchführen“ klickt, führt es diesen Anwendungsfall erneut aus.

B.4.3. Kurzinfo anzeigen

Standardablauf: Das System liest aus allen übergebenen Buch-Objekten die wichtigsten Informationen über das Buch sowie dessen Verfügbarkeit aus und zeigt sie zusammen mit einem Bild des Buches an. Außerdem erzeugt das System für jedes Buch ein Navigationselement „Details“, mit dem der Kunde den Anwendungsfall „Produktinformationen ansehen“ aufrufen kann.

Anschließend übergibt das System die Kontrolle wieder an den aufrufenden Anwendungsfall.

Alternativabläufe: Wenn der Kunde auf „Details“ klickt, übergibt das System die Kontrolle an den Anwendungsfall „Produktinformationen ansehen“.

B.4.4. Neuheiten und Empfehlungen ansehen

Standardablauf: Wenn der Kunde den Shop betritt, zeigt das System die Startseite an. Es sucht nun aus der Liste der Neuheiten und aus der Liste der Empfehlungen alle eingetragenen Buch-Objekte heraus und ruft für alle gefundenen Bücher den Anwendungsfall „Kurzinfo anzeigen“ auf.

Alternativabläufe: Wenn das System keine Neuheiten oder Empfehlungen finden kann, unterdrückt es die entsprechende Rubrik. Wenn es in beiden Kategorien keine Bücher finden kann, zeigt es statt dessen einen Standardtext an, der für den Buchshop wirbt.

B.4.5. Produktinformationen ansehen

Standardablauf: Der Kunde klickt in einer der Produktübersichten auf „Details“. Das System öffnet nun die Seite „Produktdetails“. Es liest aus dem Buch-Objekt alle Informationen über das betreffende Buch und dessen Verfügbarkeit sowie die zugehörigen Rezensionen aus und zeigt sie zusammen mit einem Bild des Buches an. Außerdem erzeugt das System Navigationselemente, mit denen der Kunde die Anwendungsfälle „Artikel in den Warenkorb legen“ und „Rezension vorbereiten“ aufrufen kann, sowie ein Navigationselement, das den Kunden zu dem Anwendungsfall zurückbringt, von dem aus er die Detailansicht geöffnet hat. Dann speichert das System eine Referenz auf das angezeigte Buch-Objekt ab. Der Kunde klickt nun auf das Navigationselement für den aufrufenden Anwendungsfall. Daraufhin gibt das System die Kontrolle an den aufrufenden Anwendungsfall zurück.

Alternativabläufe: Wenn der Kunde diesen Anwendungsfall von „Rezension vorbereiten“ oder von „Artikel in den Warenkorb legen“ aus aufgerufen hat, stellt das System die Ansicht für das gespeicherte Buch wieder her.

B.4. Paket Buchkatalog

Wenn der Kunde auf das Navigationselement klickt, das den Anwendungsfall „Rezension vorbereiten“ aufruft, übergibt das System nach dem Speichern der Buchreferenz die Kontrolle an den Anwendungsfall „Rezension vorbereiten“.

Wenn der Kunde auf das Navigationselement klickt, das den Anwendungsfall „Artikel in den Warenkorb legen“ aufruft, übergibt das System nach dem Speichern der Buchreferenz die Kontrolle an den Anwendungsfall „Artikel in den Warenkorb legen“.

B. Anwendungsfälle vor der Stabilitätsanalyse

C. Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig unter Zuhilfenahme der angegebenen Quellen erstellt habe. Wörtlich oder sinngemäß übernommene Abschnitte sind als solche gekennzeichnet. Die Vorlage dieser Arbeit zur Erlangung des akademischen Grades Diplom-Informatiker (FH) bei der Prüfungsbehörde der Fachhochschule Gießen-Friedberg ist die erste Veröffentlichung.

Gießen, August 2002