

Xpert.press

Springer-
Verlag
Berlin
Heidelberg
GmbH

Die Reihe Xpert.press des Springer-Verlags vermittelt Professionals in den Bereichen Betriebs- und Informationssysteme, Software Engineering und Programmiersprachen aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Bernhard Rumpe

Modellierung mit UML

Sprache, Konzepte und Methodik

Mit 210 Abbildungen und Tabellen



Springer

Bernhard Rumpe
Software Systems Engineering
TU Braunschweig
Mühlenpfordtstr. 23
38106 Braunschweig
Deutschland
b.rumpe@tu-bs.de

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation
in der Deutschen Nationalbibliografie; detaillierte bibliografische
Daten sind im Internet über <http://nb.ddb.de> abrufbar.

ISBN 978-3-642-62268-7 ISBN 978-3-642-18733-9 (eBook)
DOI 10.1007/978-3-642-18733-9

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2004

Ursprünglich erschienen bei Springer-Verlag Berlin Heidelberg New York in 2004

Softcover reprint of the hardcover 1st edition 2004

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutzgesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften. Text und Abbildungen wurden mit größter Sorgfalt erarbeitet. Verlag und Autor können jedoch für eventuell verbliebene fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Satz: Computer fo film von Autoredaten

Umschlaggestaltung: KünkelLopka Werbeagentur, Heidelberg

Gedruckt auf säurefreiem Papier 33 - 3142PS - 5 4 3 2 1 0

Vorwort

Der Entwurf großer Software Systeme ist eine der großen technischen Herausforderungen unserer Zeit. Umfang und Komplexität von Software haben mittlerweile Größenordnungen erreicht, die alle bekannten Ansätze und Methoden ihrer Entwicklung an ihre Grenzen bringen.

Vor diesem Hintergrund haben auch die Softwareentwickler das in den Ingenieurwissenschaften altbewährte Rezept der Modellbildung stärker für sich entdeckt. In den letzten Jahren ist unter dem Stichwort modellbasierter Softwareentwicklung eine große Zahl unterschiedlicher Ansätze entstanden, die eine umfangreiche Modellbildung zur Unterstützung der Entwicklung von Softwaresystemen zum Ziel haben. Modellbildung erlaubt es, wichtige Eigenschaften und Aspekte eines zu analysierenden oder zu erstellenden Softwaresystems gezielt modellhaft darzustellen. Ein Anliegen dabei ist eine angemessene Abstraktion, die zu einer Komplexitätsreduktion und einer besseren Beherrschbarkeit von Softwaresystemen führt. Trotz aller Fortschritte auf diesem Gebiet und seiner durchaus gegebenen Einsatzreife für die Praxis stehen noch viele ungelöste Fragen für die Forschung offen.

Ein kritischer Faktor der Modellbildung ist natürlich der zusätzliche Aufwand in der Entwicklung. Hier stellt sich die Frage, wie weit man den Aufwand bei der Modellbildung überhaupt treiben sollte und wie man die oft schwergewichtigsten, modellbasierten Vorgehensweisen mit genügend Flexibilität versehen kann, so dass sie die Profile der durchgeführten Projekte besser berücksichtigen.

Neben der Modellorientierung ist ein weiterer Trend in den letzten Jahren im Software Engineering der Einsatz sogenannter agiler Methoden, insbesondere unter dem Stichwort „extreme Programming“. Hierunter werden

leichtgewichtige Vorgehensmodelle für die Software Entwicklung verstanden, die eine Reduzierung der Softwarebürokratie sicherstellen und eine viel höhere Flexibilität in der Softwareentwicklung erlauben. Für Projekte bestimmten Profils können agile Methoden ein bedeutend effektiveres Vorgehen ermöglichen. Voraussetzung dafür sind jedoch entsprechend hinreichend kompetente Entwickler sowie ein deutlich begrenzter Projektumfang. So können agile Methoden erfolgreich nur in kleinen Projekten eingesetzt werden mit nur einer Handvoll Entwickler über einen überschaubaren Zeitraum, so dass für eine schnelle Kommunikation Rückkopplung im Projekt tatsächlich funktionieren kann.

Zunächst scheint es, als dass modellbasierte Ansätze mit ihrer starken Systematik und ihrer bewußten - von der eigentlichen Codierung losgelösten Modellierungstechnik - den agilen Methoden, die in der Regel codezentriert sind, nicht vereinbar sind. Die vorliegende Arbeit zeigt eindrucksvoll, dass es doch möglich ist, modellbasierte Ansätze mit agilen Methoden zu kombinieren und dies unter Einsatz wohlbekannter Modellierungssprachen wie der UML. Dazu muss allerdings sorgfältig überlegt werden, welche UML-Konstrukte als Modellierungs-, Test- und Implementierungsbeschreibungsmittel eingesetzt werden können und wie methodisch vorgegangen werden soll.

Eine Antwort auf diese Frage leistet die Arbeit von Herrn Rumpe. Herr Rumpe setzt sich gleichermaßen zum Ziel, relevante, praktische Ansätze, wie agiles Vorgehen und die weit verbreitete Sprache UML einzusetzen, ohne dabei aber auf saubere wissenschaftliche Fundierung und gut dokumentiertes Vorgehen zu verzichten. Deutlich wird insbesondere dargestellt, welche Programmkonstrukte der UML geeignet sind, um beispielsweise Testfälle rigoros zu entwickeln oder über perfekte Regeln eine evolutionäre Weiterentwicklung anzustoßen.

Die vorliegende Arbeit demonstriert, wie die beiden recht unterschiedlichen Paradigmen der agilen Methoden und der Modellorientierung doch miteinander korrespondieren und sich ergänzen. Es entsteht ein Ansatz, der gleichermaßen dem Anspruch einer praxisnahen, gut einsetzbaren Vorgehensweise, wie dem Anspruch einer sauberen wissenschaftlichen Fundierung gerecht wird.

Der beiliegende Text ist sehr gut lesbar, ohne dabei den Anspruch aufzugeben, eine sorgfältige inhaltliche und fachliche Darstellung zu leisten. Die Vorgehensweise, die Herr Rumpe hier vorschlägt, hat er selbst in einer Reihe von kleineren Projekten erfolgreich erprobt.

Die Arbeit stellt damit einen wertvollen Beitrag dar, der für Praktiker eine gute Handlungsanleitung gibt und Ihnen zusätzliche Informationen zeigt,

wie sie aktuelle Trends der Softwaretechnik - wie agiles Vorgehen und modellbasierte Entwicklung - erfolgreich und mit guten Ergänzungen miteinander kombinieren können. Für Studierende wird eine umfassende Einführung in das Themengebiet und eine solide Fundierung geleistet.

Das vorliegende und das darauf aufbauende Buch „Agile Modellierung mit UML“ sind somit gleichermaßen für Praktiker geeignet, die an einem entsprechenden Ansatz für ihre Entwicklungsprojekte interessiert sind, wie auch für auf praktische Fragestellungen ausgerichtete Vorlesungen, die aber nicht auf einen grundlegenden wissenschaftlichen Anspruch verzichten möchten.

Manfred Broy

Garching im Februar 2004

Inhaltsverzeichnis

1	Einführung	1
1.1	Ziele dieses Buchs	2
1.2	Überblick	3
1.3	Notationelle Konventionen	4
1.4	Ausblick: Agile Modellierung mit UML	5
2	Agile und UML-basierte Methodik	9
2.1	Das Portfolio der Softwaretechnik	11
2.1.1	Strukturierung der Softwaretechnik	11
2.1.2	Trends in der Softwaretechnik	11
2.1.3	Zusammenspiel von Technologien und Fähigkeiten ...	14
2.2	Extreme Programming (XP)	15
2.3	Ausgesuchte Entwicklungspraktiken	22
2.3.1	Test-First-Ansatz	22
2.3.2	Refactoring	25
2.4	Agile UML-basierte Vorgehensweise	26
2.5	Übersicht zur UML/P	33
2.5.1	Bedeutung und Anwendungsbereiche der UML	33
2.5.2	UML-Sprachprofile	33
2.5.3	Die Notationen in der UML/P	34
2.5.4	Modellbegriff und Modellbasierung	35
2.6	Zusammenfassung	40
3	Klassendiagramme	41
3.1	Bedeutung der Klassendiagramme	42
3.2	Klassen und Vererbung	45
3.2.1	Attribute	45
3.2.2	Methoden	48
3.2.3	Vererbung	49
3.2.4	Interfaces	50
3.3	Assoziationen	51

3.3.1	Rollennamen	52
3.3.2	Navigation	52
3.3.3	Kardinalität	53
3.3.4	Komposition	53
3.3.5	Abgeleitete Assoziationen	54
3.3.6	Assoziationsmerkmale	55
3.3.7	Qualifizierte Assoziation	55
3.4	Sicht und Repräsentation	57
3.5	Stereotypen und Merkmale	60
3.5.1	Stereotypen	61
3.5.2	Merkmale	63
3.5.3	Einführung neuer Elemente	64
3.6	Zusammenfassung	66
4	Object Constraint Language	67
4.1	Übersicht über OCL/P	69
4.1.1	Der Kontext einer Bedingung	69
4.1.2	Das let -Konstrukt	72
4.1.3	Fallunterscheidungen	74
4.1.4	Grunddatentypen	75
4.2	Die OCL-Logik	76
4.2.1	Die boolesche Konjunktion	76
4.2.2	Zweiwertige Semantik und Lifting	78
4.2.3	Kontrollstrukturen und Vergleiche	80
4.3	Container-Datenstrukturen	81
4.3.1	Aufschreibung von Mengen und Listen	83
4.3.2	Mengen- und Listenkomprehension	84
4.3.3	Mengenoperationen	88
4.3.4	Listenoperationen	91
4.3.5	Container-Operationen	93
4.3.6	Flachdrücken von Containern	94
4.3.7	Typisierung von Containern	96
4.3.8	Mengen- und listenwertige Navigation	98
4.3.9	Qualifizierte Assoziation	103
4.3.10	Quantoren	104
4.3.11	Spezialoperatoren	109
4.4	Funktionen in OCL	112
4.4.1	Queries	113
4.4.2	«OCL»-Methoden	116
4.4.3	Methodenspezifikation	118
4.4.4	Bibliothek von Queries	129
4.5	Ausdrucksmächtigkeit der OCL	131
4.5.1	Transitive Hülle	131
4.5.2	Die Natur einer Invariante	135
4.6	Zusammenfassung	137

- 5 **Objektdiagramme** 139
 - 5.1 Einführung in Objektdiagramme 141
 - 5.1.1 Objekte 141
 - 5.1.2 Attribute 144
 - 5.1.3 Links 144
 - 5.1.4 Qualifizierte Links 146
 - 5.1.5 Komposition 147
 - 5.1.6 Merkmale und Stereotypen 150
 - 5.2 Bedeutung eines Objektdiagramms 151
 - 5.2.1 Unvollständigkeit und Exemplarizität 152
 - 5.2.2 Prototypische Objekte 152
 - 5.2.3 Instanz versus Modellinstanz 154
 - 5.3 Logik der Objektdiagramme 156
 - 5.3.1 Namen für ein Diagramm 156
 - 5.3.2 Bindung von Objektnamen 157
 - 5.3.3 Integration von Objektdiagramm und OCL 159
 - 5.3.4 Die Bedeutung anonymer Objekte 160
 - 5.3.5 OCL-Bedingungen im Objektdiagramm 161
 - 5.3.6 Abstrakte Objektdiagramme 162
 - 5.4 Methodische Verwendung von Objektdiagrammen 164
 - 5.4.1 Komposition von Objektdiagrammen 164
 - 5.4.2 Negation 165
 - 5.4.3 Alternative Objektstrukturen 166
 - 5.4.4 Objektdiagramme in einer Methodenspezifikation 167
 - 5.4.5 Objekterzeugung 169
 - 5.4.6 Gültigkeit von Objektdiagrammen 169
 - 5.4.7 Initialisierung von Objektstrukturen 170
 - 5.5 Zusammenfassung 172

- 6 **Statecharts** 175
 - 6.1 Eigenschaften von Statecharts 177
 - 6.2 Automatentheorie und Interpretation 179
 - 6.2.1 Erkennende und Mealy-Automaten 179
 - 6.2.2 Interpretation 181
 - 6.2.3 Nichtdeterminismus als Unterspezifikation 183
 - 6.2.4 ϵ -Transitionen 184
 - 6.2.5 Unvollständigkeit 184
 - 6.2.6 Lebenszyklus 185
 - 6.2.7 Beschreibungsmächtigkeit 186
 - 6.2.8 Transformationen auf Automaten 188
 - 6.3 Zustände 189
 - 6.3.1 Zustandsinvarianten 190
 - 6.3.2 Hierarchische Zustände 195
 - 6.3.3 Start- und Endzustand 198
 - 6.4 Transitionen 199

6.4.1	Bedingungen innerhalb der Zustandshierarchie	199
6.4.2	Start- und Endzustand in der Zustandshierarchie	201
6.4.3	Stimuli für Transitionen	202
6.4.4	Schaltbereitschaft	208
6.4.5	Unvollständiges Statechart	211
6.5	Aktionen	216
6.5.1	Prozedurale und beschreibende Aktionen	216
6.5.2	Aktionen in Zuständen	218
6.5.3	Zustandsinterne Transitionen	222
6.5.4	do-Aktivität	223
6.6	Statecharts im Kontext der UML	224
6.6.1	Vererbung von Statecharts	224
6.6.2	Transformation von Statecharts	225
6.6.3	Abbildung in die OCL	238
6.7	Zusammenfassung	240
7	Sequenzdiagramme	241
7.1	Konzepte der Sequenzdiagramme	243
7.2	OCL in Sequenzdiagrammen	247
7.3	Semantik eines Sequenzdiagramms	249
7.4	Sonderfälle und Ergänzungen für Sequenzdiagramme	254
7.5	Sequenzdiagramme in der UML	257
7.6	Zusammenfassung	259
A	Sprachdarstellung durch Syntaxklassendiagramme	261
B	Java	267
C	Die Syntax der UML/P	275
C.1	UML/P-Syntax Übersicht	275
C.2	Klassendiagramme	276
C.2.1	Kernteile eines Klassendiagramms	277
C.2.2	Textteile eines Klassendiagramms	278
C.2.3	Merkmale und Stereotypen	280
C.2.4	Vergleich mit dem UML-Standard	281
C.3	OCL	284
C.3.1	Syntax der OCL	284
C.3.2	Unterschiede zu dem OCL-Standard	287
C.4	Objektdiagramme	290
C.4.1	Kontextfreie Syntax	291
C.4.2	Konformität zu Klassendiagrammen	293
C.5	Statecharts	294
C.5.1	Abstrakte Syntax	295
C.5.2	Vergleich mit dem UML-Standard	298
C.6	Sequenzdiagramme	300

C.6.1	Abstrakte Syntax	300
C.6.2	Vergleich mit dem UML-Standard	301
D	Anwendungsbeispiel: Internet-basiertes Auktionssystem	303
D.1	Auktionen als E-Commerce Applikation	304
D.2	Die Auktionsplattform	306
Literatur	311