Xpert.press

Bernhard Rumpe

Modellierung mit UML

2. Auflage

Springer

Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Bernhard Rumpe

Modellierung mit UML

Sprache, Konzepte und Methodik

2. Auflage

Springer

Prof. Dr. Bernhard Rumpe
RWTH Aachen
Informatik/Software Engineering
Ahornstr. 55
52062 Aachen
Deutschland
http://www.se-rwth.de

ISSN 1439-5428
ISBN 978-3-642-22412-6 e-ISBN 978-3-642-22413-3
DOI 10.1007/978-3-642-22413-3
Springer Heidelberg Dordrecht London New York

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Einbandentwurf: KünkelLopka GmbH, Heidelberg

Gedruckt auf säurefreiem Papier

Springer ist Teil der Fachverlagsgruppe Springer Science+Business Media (www.springer.com)

Vorwort

Vorwort zur 2ten Auflage

Vor zehn Jahren war absehbar, dass agile Methoden sich zumindest für einen Teilbereich der Softwareentwicklung durchsetzen würden, auch wenn sie damals von vielen Entwicklern noch eher belächelt wurden. Mittlerweile sind agile Methoden ein etablierter Bestandteil des Portfolios der Softwaretechnik und wurden an vielen Stellen ergänzt und für mehrere Domänen angepasst.

Parallel hat die Unified Modelling Language ihren Siegeszug angetreten und hat heute praktisch alle anderen echten Modellierungssprachen in sich aufgesogen oder verdrängt, wobei wir Matlab/Simulink nicht als echte Modellierungssprache sondern als grafische Programmiersprache zählen wollen. Die UML ist einerseits groß und leidet weiterhin an den vielen Optionen und Interpretationsmöglichkeiten, die aufgrund ihrer vielen Einsatzgebiete auch nicht so ohne weiteres reduziert werden können. Stattdessen ist es wohl besser ein expliziteres Variabilitätsmodell für syntaktische, methodische und semantische Unterschiede zu erstellen und durch geeignete Auswahl auf einzelne Projekte zu konfigurieren [Grö10].

Noch erfolgreicher hat sich die Programmiersprache Java sowohl als primäre Web- und Business-System-Sprache, als auch als Lehrsprache für Informatikstudenten durchgesetzt.

In diesem sowie auch dem darauf aufbauendem Buch „Agile Modellierung mit UML“ werden die UML als auch Java konsolidiert, in Maßen ergänzt und gleichzeitig weiterentwickelt. UML liegt in Version 2.3 und Java in Version 6 vor. Die in diesem Buch vorgestellte UML/P stellt zwar eine eigenständige Fassung, ein sogenanntes Profil, dar, wurde aber durch die Änderungen von UML 1.4 nach UML 2.3 dennoch an einigen Stellen angepasst. Da Java als Ziel von Generierungs- und Testvorgängen zum Einsatz kommt, ist es natürlich von Interesse, auf die neuen Möglichkeiten von Java wie zum Beispiel den Generics oder dem assert-Statement einzugehen.

Die Kluft zwischen den Welten der modellbasierten Softwareentwicklung mit der UML und den agilen Methoden hat sich trotz oder vielleicht gerade wegen des Erfolgs beider Ansätze nicht wirklich geschlossen. Während agile Methoden durchaus gerne Code generieren statt von Hand schreiben wollen, sehen viele Entwickler im Moment noch die Hürde zur Generierung als relativ groß an. Dies liegt häufig an der Unhandlichkeit bzw. Schwergewichtigkeit der Generierungsprozesses und des relativ großen initialen Aufwands zur Einführung von Generierungswerkzeugen in den Entwicklungsprozess. Diese Lücke gilt es noch zu schließen.

An der Erstellung der ersten und der Überarbeitung zur zweiten Fassung dieses Buchs haben eine Reihe von Personen direkt oder indirekt mitgewirkt. Mein besonderer Dank gilt Manfred Broy für die Unterstützung, die dieses Buch erst ermöglicht hat und den Mitarbeitern und Studierenden, insbesondere Christian Berger, Marita Breuer, Angelika Fleck, Hans Grönniger, Sylvia Gunder, Tim Gülke, Arne Haber, Christoph Herrmann, Roland Hildebrandt, Holger Krahn, Thomas Kurpik, Markus Look, Shahar Maoz, Philip Martzok, Anonio Navarro Pérez, Class Pinkernell, Dirk Reiss, Holger Rendel, Jan Oliver Ringert, Martin Schindler, Mark Stein, Christopher Vogt, Galina Volkova, Steven Völkel und Ingo Weisenmöller, die dieses Buch als Grundlage ihrer Arbeit einsetzen oder geholfen haben es für die zweite Auflage zu ergänzen und zu verbessern. Gerne danke ich dem ehemaligen bayerischen Minister für Wissenschaft, Forschung und Kunst Hans Zehetmair für die Verleihung des Habilitationsstipendiums und meinem geschätzten Kollegen und Vorgänger Prof. Dr.-Ing. Manfred Nagl für eine wohlwollende Unterstützung beim Aufbau des Lehrstuhl in Aachen.

Herzlicher Dank gilt meinen Freunden, Kolleginnen und Kollegen, wissenschaftlichen Mitarbeitern, sowie den Studierenden für konstruktive Diskussionen, Mitarbeit an dem Anwendungsbeispiel und Reviews von Zwischenständen dieses Buchs in erster Auflage aus München: Samer Alhunaty, Hubert Baumeister, Markus Boger, Peter Braun, Maria Victoria Cengarle, David Cruz da Bettencourt, Ljiljana Döhring, Jutta Eckstein, Andreas Günzler, Franz Huber, Jan Jürjens, Ingolf Krüger, Konstantin Kukushkin, Britta Liebscher, Barbara Paech, Markus Pister, Gerhard Popp, Jan Philipps, Alexander Pretschner, Mattias Rahlf, Andreas Rausch, Stefan Rumpe, Robert Sandner, Bernhard Schätz, Markus Wenzel, Guido Wimmel und Alexander Wisspeintner.

Bernhard Rumpe

Aachen im Juni 2011

Vorwort zur 1ten Auflage

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 schwergewichtigten, 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 Implementierungsbeschreibungs-

mittel 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 der beiden Bücher 2
- 1.2 Überblick 3
- 1.3 Notationelle Konventionen 4
- 1.4 Einordnung der UML/P 5
  - 1.4.1 Bedeutung und Anwendungsbereiche der UML 5
  - 1.4.2 UML-Sprachprofile 6
  - 1.4.3 Die Notationen in der UML/P 7
  - 1.4.4 Modellbegriff und Modellbasierung 8
- 1.5 Ausblick: Agile Modellierung mit UML 12

2 Klassendiagramme 15
- 2.1 Bedeutung der Klassendiagramme 16
- 2.2 Klassen und Vererbung 19
  - 2.2.1 Attribute 19
  - 2.2.2 Methoden 22
  - 2.2.3 Vererbung 23
  - 2.2.4 Interfaces 24
- 2.3 Assoziationen 25
  - 2.3.1 Rollen 26
  - 2.3.2 Navigation 26
  - 2.3.3 Kardinalität 27
  - 2.3.4 Komposition 27
  - 2.3.5 Abgeleitete Assoziationen 28
  - 2.3.6 Assoziationsmerkmale 29
  - 2.3.7 Qualifizierte Assoziation 30
- 2.4 Sicht und Repräsentation 31
- 2.5 Stereotypen und Merkmale 34
  - 2.5.1 Stereotypen 36
  - 2.5.2 Merkmale 37
  - 2.5.3 Einführung neuer Elemente 38

3 Object Constraint Language ... 41
3.1 Übersicht über OCL/P ... 43
3.1.1 Der Kontext einer Bedingung ... 43
3.1.2 Das **let**-Konstrukt ... 46
3.1.3 Fallunterscheidungen ... 48
3.1.4 Grunddatentypen ... 49
3.2 Die OCL-Logik ... 50
3.2.1 Die boolesche Konjunktion ... 50
3.2.2 Zweiwertige Semantik und Lifting ... 52
3.2.3 Kontrollstrukturen und Vergleiche ... 54
3.3 Container-Datenstrukturen ... 55
3.3.1 Darstellung von Mengen und Listen ... 56
3.3.2 Mengen- und Listenkomprehension ... 58
3.3.3 Mengenoperationen ... 61
3.3.4 Listenoperationen ... 64
3.3.5 Container-Operationen ... 66
3.3.6 Flachdrücken von Containern ... 68
3.3.7 Typisierung von Containern ... 69
3.3.8 Mengen- und listenwertige Navigation ... 71
3.3.9 Qualifizierte Assoziation ... 76
3.3.10 Quantoren ... 77
3.3.11 Spezialoperatoren ... 82
3.4 Funktionen in OCL ... 85
3.4.1 Queries ... 85
3.4.2 ≪OCL≫-Methoden ... 89
3.4.3 Methodenspezifikation ... 91
3.4.4 Bibliothek von Queries ... 104
3.5 Ausdrucksmächtigkeit der OCL ... 105
3.5.1 Transitive Hülle ... 105
3.5.2 Die Natur einer Invariante ... 109
3.6 Zusammenfassung ... 111

4 Objektdiagramme ... 113
4.1 Einführung in Objektdiagramme ... 116
4.1.1 Objekte ... 116
4.1.2 Attribute ... 118
4.1.3 Links ... 118
4.1.4 Qualifizierte Links ... 120
4.1.5 Komposition ... 121
4.1.6 Merkmale und Stereotypen ... 123
4.2 Bedeutung eines Objektdiagramms ... 125
4.2.1 Unvollständigkeit und Exemplarizität ... 125
4.2.2 Prototypische Objekte ... 126
4.2.3 Instanz versus Modellinstanz ... 127
4.3 Logik der Objektdiagramme ... 129

4.3.1 Namen für ein Diagramm 130
4.3.2 Bindung von Objektnamen 130
4.3.3 Integration von Objektdiagramm und OCL 132
4.3.4 Anonyme Objekte 133
4.3.5 OCL-Bedingungen im Objektdiagramm 134
4.3.6 Abstrakte Objektdiagramme 135
4.4 Methodische Verwendung von Objektdiagrammen 137
4.4.1 Komposition von Objektdiagrammen 137
4.4.2 Negation 138
4.4.3 Alternative Objektstrukturen 139
4.4.4 Objektdiagramme in einer Methodenspezifikation 139
4.4.5 Objekterzeugung 141
4.4.6 Gültigkeit von Objektdiagrammen 142
4.4.7 Initialisierung von Objektstrukturen 143
4.5 Zusammenfassung 145

5 Statecharts 147
5.1 Eigenschaften von Statecharts 149
5.2 Automatentheorie und Interpretation 150
5.2.1 Erkennende und Mealy-Automaten 151
5.2.2 Interpretation 153
5.2.3 Nichtdeterminismus als Unterspezifikation 155
5.2.4 ε-Transitionen 156
5.2.5 Unvollständigkeit 156
5.2.6 Lebenszyklus 157
5.2.7 Beschreibungsmächtigkeit 158
5.2.8 Transformationen auf Automaten 159
5.3 Zustände 160
5.3.1 Zustandsinvarianten 162
5.3.2 Hierarchische Zustände 167
5.3.3 Start- und Endzustand 170
5.4 Transitionen 171
5.4.1 Bedingungen innerhalb der Zustandshierarchie 171
5.4.2 Start- und Endzustand in der Zustandshierarchie 172
5.4.3 Stimuli für Transitionen 174
5.4.4 Schaltbereitschaft 176
5.4.5 Unvollständiges Statechart 179
5.5 Aktionen 183
5.5.1 Prozedurale und beschreibende Aktionen 183
5.5.2 Aktionen in Zuständen 185
5.5.3 Zustandsinterne Transitionen 190
5.5.4 do-Aktivität 190
5.6 Statecharts im Kontext der UML 191
5.6.1 Vererbung von Statecharts 191
5.6.2 Transformation von Statecharts 192

5.6.3 Abbildung in die OCL 205
5.7 Zusammenfassung 207

6 Sequenzdiagramme 209
6.1 Konzepte der Sequenzdiagramme 211
6.2 OCL in Sequenzdiagrammen 215
6.3 Semantik eines Sequenzdiagramms 217
6.4 Sonderfälle und Ergänzungen für Sequenzdiagramme 222
6.5 Sequenzdiagramme in der UML 225
6.6 Zusammenfassung 228

A Sprachdarstellung durch Syntaxklassendiagramme 229

B Java 237

C Die Syntax der UML/P 245
C.1 UML/P-Syntax Übersicht 245
C.2 Klassendiagramme 246
C.2.1 Kernteile eines Klassendiagramms 247
C.2.2 Textteile eines Klassendiagramms 248
C.2.3 Merkmale und Stereotypen 250
C.2.4 Vergleich mit dem UML-Standard 251
C.3 OCL 254
C.3.1 Syntax der OCL 254
C.3.2 Unterschiede zu dem OCL-Standard 257
C.4 Objektdiagramme 260
C.4.1 Kontextfreie Syntax 261
C.5 Statecharts 263
C.5.1 Abstrakte Syntax 264
C.5.2 Vergleich mit dem UML-Standard 267
C.6 Sequenzdiagramme 269
C.6.1 Abstrakte Syntax 269
C.6.2 Vergleich mit dem UML-Standard 270

D Anwendungsbeispiel: Internet-basiertes Auktionssystem 273
D.1 Auktionen als E-Commerce Applikation 274
D.2 Die Auktionsplattform 275

Literatur 279

Index 289

1

Einführung

Das Streben nach Wissen ist eine
natürliche Veranlagung aller Menschen.
Aristoteles

Die Softwaretechnik hat sich in den letzten Jahren zu einer wirkungsvollen Ingenieursdisziplin entwickelt. Aufgrund der kontinuierlich anwachsenden Komplexität ihrer Aufgabenstellungen und Diversität der Anwendungsdomänen konnte ein *Portfolio von Softwareentwicklungstechniken* gebildet werden, das für jede Anwendungsdomäne, Kritikalität und Komplexität des zu entwickelnden Systems eine weitgehend maßgeschneiderte Auswahl an geeigneten Vorgehensweisen und Konzepten bietet. Techniken für Projekt-, Konfigurations-, Varianten- und Qualitätsmanagement, Software Produktlinien, Vorgehensmodelle, Spezifikationstechniken, Analyse- und Entwurfsmuster und „Best Practices" für spezielle Aufgabenstellungen sind nur einige der Elemente dieses Portfolios.

In diesem Portfolio stehen zum einen konkurrierende Ansätze mit jeweils problemspezifischen Vorteilen zur Verfügung. Zum anderen ermöglicht und bedingt die Weiterentwicklung der eingesetzten Sprachen, Frameworks und Werkzeuge eine kontinuierliche Ergänzung und Erweiterung des Portfolios auch in methodischer Hinsicht. Programmiersprachen wie Java, ausgereifte Klassenbibliotheken und die permanent verbesserten Softwareentwicklungswerkzeuge erlauben heute Vorgehensweisen, die noch vor wenigen Jahren undenkbar waren. So ist beispielsweise die werkzeuggestützte Weiterentwicklung oder Modifikation einer bereits im Einsatz befindlichen Softwarearchitektur, mittlerweile wesentlich einfacher geworden.

Weiterführendes Material:

http://www.se-rwth.de/mbse

B. Rumpe, *Modellierung mit UML*, 2. Aufl., Xpert.press, DOI 10.1007/978-3-642-22413-3_1,

Die sich schnell wandelnde Technologie, die von den Anwendern beispielsweise im E-Service-Bereich erwartete Flexibilität und Erweiterbarkeit der Systeme sowie die hohe Kritikalität von Geschäftsanwendungen machen es notwendig, Vorgehensweisen und die dabei benutzten Entwicklungstechniken kontinuierlich zu optimieren und anzupassen. Nur so lässt sich unter Nutzung der vorhandenen Softwareentwicklungstechniken und der gegebenen zeitlichen und personellen Ressourcen in flexibler Weise ein qualitativ hochwertiges und für die Wünsche des Kunden geeignetes System entwickeln und kontinuierlich ergänzen.

Die Verbreitung des Internets erlaubt auch die zunehmende Integration von Geschäftsanwendungen über Unternehmensgrenzen hinweg, sowie die Einbindung der Anwender über Rückkopplungsmechanismen sozialer Netzwerke, so dass besonders im Bereich internetbasierter Software komplexe Netzwerke von E-Service- und E-Business-Anwendungen entstehen. Dafür sind adäquate Softwareentwicklungstechniken notwendig. In diesem Bereich wird vor allem Objekttechnologie genutzt und zur Modellierung die sich derzeit zum Standard entwickelnde Unified Modeling Language (UML) eingesetzt.

1.1 Ziele der beiden Bücher

Mission Statement: Es ist ein Kernziel, für das genannte Portfolio einige grundlegende Techniken zur modellbasierten Entwicklung zur Verfügung zu stellen. Dabei wird in diesem Buch eine Variante der UML vorgestellt, die speziell zur effizienten Entwicklung qualitativ hochwertiger Software und Software-basierter Systeme geeignet ist.

UML-Standard: Der UML-Standard muss sehr viele Anforderungen aus unterschiedlichen Gegebenheiten heraus erfüllen und ist daher notwendigerweise überladen. Viele Elemente des Standards sind für unsere Zwecke nicht oder nicht in der gegebenen Form sinnvoll, während andere Sprachkonzepte fehlen. Deshalb wird in diesem Buch ein angepasstes und mit UML/P bezeichnetes Sprachprofil der UML vorgestellt. UML/P wird dadurch für die vorgeschlagenen Entwicklungstechniken im Entwurf, in der Implementierung und in der Wartung optimiert und so in agilen Entwicklungsmethoden besser einsetzbar.

Dieses Buch konzentriert sich vor allem auf die Einführung des Sprachprofils. In einem zweiten Buch „Agile Modellierung mit UML“ werden darüber hinausgehend vor allem modellbasierte Vorgehenstechniken Generierung, Testfalldefinition und Evolution beschrieben.

Die UML/P ist als Ergebnis mehrerer Grundlagen- und Anwendungsprojekte entstanden. So wurde zum Beispiel die in Anhang D beschriebene Anwendung soweit möglich unter Verwendung der hier beschriebenen Prinzipien entwickelt. Das beschriebene Auktionssystem ist auch deshalb ideal zur Demonstration der in den beiden Büchern entwickelten Techniken, weil