Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering

Lehrbücher der Informatik

Herausgegeben von
Prof. Dr.-Ing. habil. Helmut Balzert

Heide Balzert
Lehrbuch der Objektmodellierung
Analyse und Entwurf mit der UML 2, 2. Auflage

Helmut Balzert
Lehrbuch Grundlagen der Informatik
Konzepte und Notationen in UML 2, Java 5, C# und C++,
Algorithmik und Softwaretechnik, Anwendungen, 2. Auflage

Klaus Zeppenfeld
Lehrbuch der Grafikprogrammierung
Grundlagen, Programmierung, Anwendung

Helmut Balzert
Objektorientierte Programmierung
mit Java 5

Helmut Balzert
Lehrbuch der Softwaretechnik:
Softwaremanagement
2. Auflage

Helmut Balzert

Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering

3. Auflage

Unter Mitwirkung von
Heide Balzert
Rainer Koschke
Uwe Lämmel
Peter Liggesmeyer
Jochen Quante

Spektrum
AKADEMISCHER VERLAG

Autor
Prof. Dr. Helmut Balzert
E-Mail: hb@W3L.de

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

Springer ist ein Unternehmen von Springer Science+Business Media
springer.de

3. Auflage 2009

Spektrum Akademischer Verlag ist ein Imprint von Springer

09 10 11 12 13 5 4 3 2 1

Planung und Lektorat: Dr. Andreas Rüdinger
Redaktion und Gestaltung: M.Sc. Kerstin Kohl, Witten; Dagmar Fraude, Witten
Umschlaggestaltung: SpieszDesign, Neu-Ulm
Titelbild: Anna Solecka-Zach: „Ohne Titel“ (1995)
Satz: W3L GmbH, Witten – automatischer Satz aus der W3L-Plattform.

ISBN 978-3-8274-1705-3

Vorwort zur 3. Auflage

Die Softwaretechnik bildet einen Grundpfeiler der Informatik. Sie hat sich zu einem umfassenden Wissenschaftsgebiet entwickelt. Um Ihnen, liebe Leserin, lieber Leser, ein optimales Erlernen dieses Gebietes zu ermöglichen, habe ich – zusammen mit Koautoren – ein dreibändiges Lehr- und Lernbuch über die Softwaretechnik geschrieben. Es behandelt die Kerngebiete »Basiskonzepte und Requirements Engineering« (Band 1), »Entwurf und Architekturen« (Band 2) und »Softwaremanagement« (Band 3) sowie die Abhängigkeiten zwischen diesen Gebieten.

Aufbau & Gliederung

Ich habe das Gebiet der Softwaretechnik in mehrere große Bereiche gegliedert – der Tempel der Softwaretechnik (Abb. 0.0-1) veranschaulicht diese Gliederung. Ziel jeder Softwareentwicklung ist es, ein lauffähiges Softwareprodukt zu erstellen, zu warten und zu pflegen. Im Mittelpunkt der Gliederung steht daher die Softwareentwicklung (mittlere Säule). Jede der Aktivitäten der Softwareentwicklung trägt dazu bei, Teilprodukte zu erstellen, die dann in ein Gesamtprodukt münden.

Eine Softwareentwicklung läuft aber *nicht* von alleine ab. Sie basiert auf und nutzt Basistechniken, das sind Prinzipien, Methoden, Werkzeuge, *Best Practices*.

Die eigentliche Softwareentwicklung wird durch die zwei Säulen »Softwaremanagement« und »Prozess- und Qualitätsmodelle« und das Dach »Allgemeines Management« eingerahmt. Bei den Aktivitäten des Managements und der Prozess- & Qualitätssicherung handelt es sich um *begleitende* Aktivitäten, deren Ergebnisse aber selbst *nicht* Bestandteil des Endprodukts sind. Dennoch sind sie eminent wichtig – was die große Anzahl fehlgeschlagener Softwareprojekte deutlich zeigt.

In diesem Buch werden die Basistechniken, die Basiskonzepte und das *Requirements Engineering* behandelt. Dieses Buch besteht aus folgenden Teilen:

I Die Wissenschaftsdisziplin Softwaretechnik
II Basistechniken
III Basiskonzepte
IV Requirements Engineering

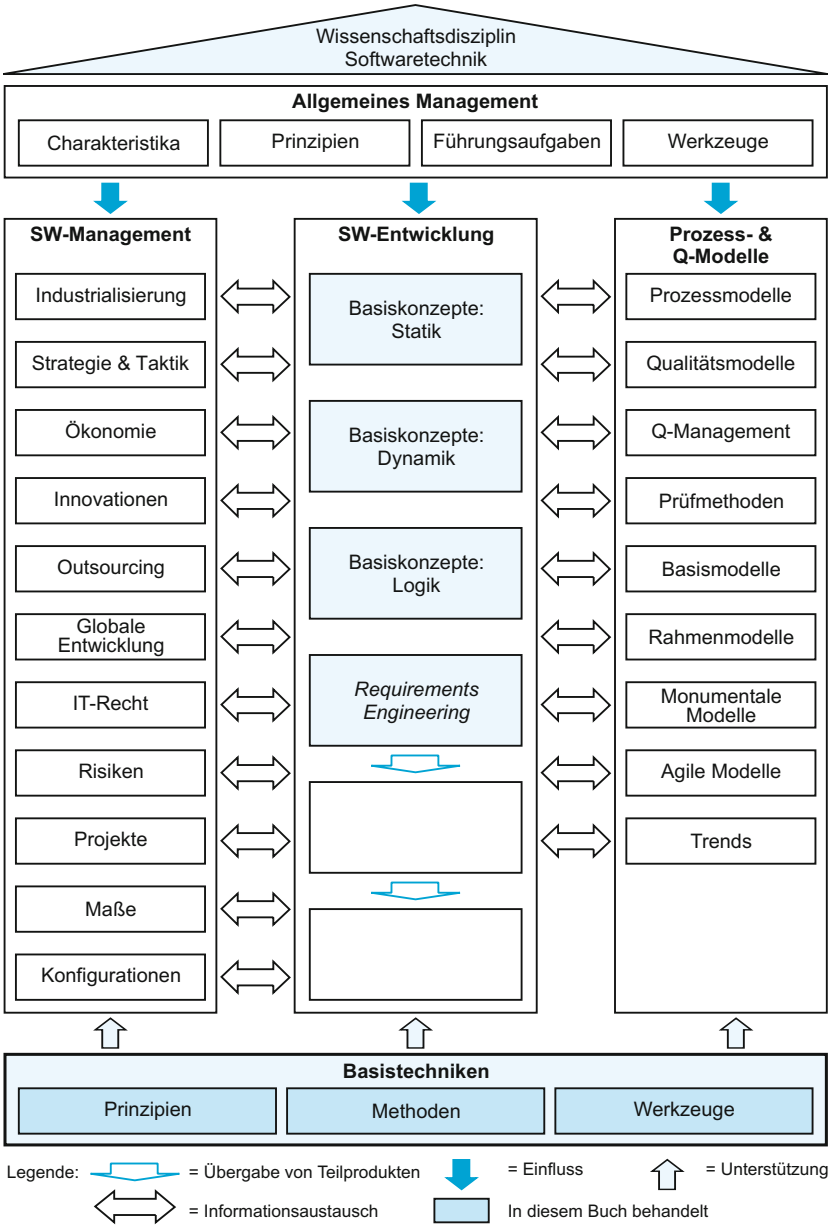

Abb. 0.0-1: Gliederung der Softwaretechnik.

Zu einem dreibändigen Lehrbuch der Softwaretechnik gehört natürlich auch eine Reflexion über die Wissenschaftsdisziplin:

■ »Die Wissenschaftsdisziplin Softwaretechnik«, S. 1

Wie in anderen dynamischen Disziplinen auch, so gibt es auch in der Softwaretechnik Trends und »Hypes«. Nach einer Euphoriephase erfolgt dann oft die Ernüchterung. Für die Lehre und das Lernen ist es

daher wichtig, das Konstante in der Softwaretechnik zu identifizieren – das unabhängig von aktuellen Trends gilt. Ich habe allgemeingültige Prinzipien, Methoden und Werkzeuge unter dem Oberbegriff Basistechniken zusammengefasst:

■ »Basistechniken«, S. 23

Analog habe ich versucht, bei den heute in der Softwaretechnik angewandten Konzepten die grundlegenden Konzepte zu isolieren und in die drei Kategorien Statik, Dynamik und Logik einzuordnen:

■ »Basiskonzepte«, S. 99

Softwaresysteme lassen sich grob in Informationssysteme und softwareintensive Systeme gliedern. Bei softwareintensiven Systemen ist die Software nur ein – wenn auch wichtiger – Bestandteil des Systems. Je nachdem, um was für eine Art von Softwaresystem es sich handelt, werden die Basiskonzepte unterschiedlich intensiv und in verschiedenen Kombinationen eingesetzt.

2 Fallstudien

Um diese Unterschiede zu verdeutlichen, werden zwei Fallstudien verwendet, auf die sich in den Beispielen zu den Basiskonzepten immer wieder bezogen wird:

■ »Fallstudie: SemOrg – Die Spezifikation«, S. 107
■ »Fallstudie: Fensterheber – Die Spezifikation«, S. 117

Zu wissen, was der Kunde will, und dies zu ermitteln, zu beschreiben, zu spezifizieren, zu analysieren und in Form einer fachlichen Lösung zu modellieren ist die Aufgabe des *Requirements Engineering* – im Deutschen auch als Systemanalyse bezeichnet. Da im *Requirements Engineering* viele Basistechniken und Basiskonzepte benötigt und eingesetzt werden, habe ich die Basistechniken und Basiskonzepte an den Anfang des Buches gestellt. Das *Requirements Engineering* wird oft noch unterschätzt – hier werden jedoch die Grundlagen für den Erfolg oder Misserfolg einer Softwareentwicklung gelegt:

■ »Requirements Engineering«, S. 433

Zur 3. Auflage

Die Inhalte der 2. Auflage wurden in zwei Bände aufgeteilt, um die Bücher handlicher und zielgruppenorientierter zu gestalten. Im Gegensatz zur 2. Auflage ist die 3. Auflage unabhängig von einem Vorgehensmodell. Die Basistechniken und Basiskonzepte sind prozessneutral beschrieben und nach sachlogischen Gesichtspunkten strukturiert – das gilt auch für das *Requirements Engineering*.

Die Methoden SA *(Structured Analysis)* und SA/RT *(Real Time Analysis)* werden nicht mehr behandelt, da ihr Einsatz in der Praxis abnimmt. Das gesamte Gebiet der Software-Ergonomie wird aus Umfangsgründen nicht mehr behandelt – obwohl es zunehmend wichtiger wird. Das Gebiet ist jedoch inzwischen so umfangreich, dass es dazu eigene Lehrbücher gibt und an Hochschulen oft auch in eigenen Lehrveranstaltungen unterrichtet wird.

Kapitel aus der 2. Auflage, die nicht mehr in dieser neuen Auflage berücksichtigt werden konnten, finden Sie als *Open Content* auf der Website `www.W3L.de/OpenContent`.

Buchaufbau

Das Buch besteht aus 28 Kapiteln. Jedes Kapitel besteht aus Unterkapiteln. Am Ende der meisten Kapitel befindet sich eine Zusammenfassung des behandelten Lehrstoffs. Am Ende des Buches sind das umfangreiche Glossar sowie das Literaturverzeichnis angeordnet. Dadurch kann das Buch auch ideal als Nachschlagewerk benutzt werden.

Didaktik & Methodik

Ziel der Didaktik ist es, einen Lehrstoff so zu strukturieren und aufzubereiten, dass der Lernende sich leicht ein mentales Modell von dem Lehrstoff aufbauen kann und genügend Übungsmöglichkeiten erhält, um zu überprüfen, ob er den Lehrstoff – nach der Beschäftigung mit ihm – entsprechend den vorgegebenen Lernzielen beherrscht. Dieses didaktische Ziel habe ich versucht in diesem Lehrbuch umzusetzen. Die Übungsmöglichkeiten befinden sich jedoch nicht im Lehrbuch, sondern in dem ebenfalls verfügbaren E-Learning-Kurs (siehe unten).

In den meisten Lehrbüchern wird die Welt so erklärt, wie sie ist – ohne dem Lernenden vorher die Möglichkeit gegeben zu haben, über die Welt nachzudenken. Ich stelle daher in vielen Kapiteln am Anfang an Sie eine Frage. Diese Frage soll Sie dazu anregen, über ein Thema nachzudenken. Erst nach dem Nachdenken sollten Sie weiter lesen. (Vielleicht sollten Sie die Antwort nach der Frage zunächst durch ein Papier abdecken).

Querverweise

Die Softwaretechnik ist komplex. Es gibt viele gegenseitigen Abhängigkeiten. Um diese Abhängigkeiten zu verdeutlichen, enthält dieses Buch viele (absolute) Querverweise auf andere Kapitel, damit Sie sich mit verschiedenen Perspektiven auf ein Themengebiet befassen können.

Einsatz des Buches

Dieses Buch kann zur Vorlesungsbegleitung, zum Selbststudium und zum Nachschlagen verwendet werden. Um den Umfang und die Kosten des Buches zu begrenzen, enthält dieses Buch *keine* Tests und Aufgaben. Für diejenigen Leser unter Ihnen, die Ihr Wissen durch Tests und Aufgaben aktiv überprüfen möchten, gibt es einen (kostenpflichtigen) E-Learning-Online-Kurs. Mentoren und Tutoren betreuen Sie bei der Bearbeitung von Tests und Aufgaben. Das Bestehen eines Abschlusstests und einer Abschlussklausur wird durch Zertifikate dokumentiert. Dieser Online-Kurs ist Bestandteil des Online-Bachelor-Studiengangs »Web- und Medieninformatik« der FH Dortmund. Sie finden den Kurs auf der W3L-Plattform (`http://www.W3L.de`).

Kostenloser E-Learning-Kurs

Ergänzend zu diesem Buch gibt es den kostenlosen E-Learning-Kurs »Prinzipien der Softwaretechnik«, der zusätzlich zahlreiche Tests enthält, mit denen Sie Ihr Wissen überprüfen können. Sie fin-

den den Kurs auf der E-Learning-Plattform `www.W3L.de`. Bitte wählen Sie auf dem Reiter »Online-Kurse« den Link »Zur TAN-Einlösung«. Registrieren Sie sich als neuer Benutzer und geben Sie anschließend folgende Transaktionsnummer (TAN) ein: `3645112138`.

Zielgruppen

Dieses Buch ist für folgende Zielgruppen geschrieben:

- Studierende der Informatik und Softwaretechnik an Universitäten, Fachhochschulen und Berufsakademien.
- Software-Ingenieure, Softwaremanager und Software-Qualitätssicherer in der Praxis.

Vorkenntnisse

Vorausgesetzt werden Kenntnisse, wie sie normalerweise in einer Einführungsvorlesung zur Informatik vermittelt werden.

Beispiele, Fallstudien

Zur Vermittlung der Lerninhalte werden Beispiele und Fallstudien verwendet. Um Ihnen diese unmittelbar kenntlich zu machen, sind sie in blauer Schrift gesetzt.

Visualisierung

Da ein Bild oft mehr aussagt als 1000 Worte, habe ich versucht, möglichst viele Sachverhalte zu veranschaulichen.

Begriffe, Glossar halbfett, blau

In diesem Lehrbuch wurde sorgfältig überlegt, welche Begriffe eingeführt und definiert werden. Ziel ist es, die Anzahl der Begriffe möglichst gering zu halten. Alle wichtigen Begriffe sind im Text halbfett und blau gesetzt. Die so markierten Begriffe sind am Ende des Buches in einem Glossar alphabetisch angeordnet und definiert. Dabei wurde oft versucht, die Definition etwas anders abzufassen, als es im Text der Fall war, um Ihnen noch eine andere Sichtweise zu vermitteln.

Weibliche vs. männliche Anrede

Um in einem Buch deutlich zu machen, dass Leser und Leserinnen gemeint sind, gibt es verschiedene Möglichkeiten für den Autor:

1. Man formuliert Bezeichnungen in der 3. Person Singular in ihrer männlichen Form. In jüngeren Veröffentlichungen verweist man in den Vorbemerkungen dann häufig darauf, dass das weibliche Geschlecht mit gemeint ist, auch wenn es nicht im Schriftbild erscheint.
2. Man redet beide Geschlechter direkt an, z. B. Leserinnen und Leser, man/frau.
3. Man kombiniert die beiden Geschlechter in einem Wort, z. B. StudentInnen.
4. Man wechselt das Geschlecht von Kapitel zu Kapitel.

Aus Gründen der Lesbarkeit und Lesegewohnheit habe ich mich für die 1. Variante entschieden. Die Variante 4 ist mir an und für sich sehr sympathisch, jedoch steigt der Aufwand für den Autor beträchtlich, da man beim Schreiben noch nicht die genaue Reihenfolge der Kapitel kennt.

Vorwort zur 3. Auflage

Als Begleitunterlage & zum Selbststudium

Bücher können als Begleitunterlage oder zum Selbststudium ausgelegt sein. In diesem Buch versuche ich einen Mittelweg einzuschlagen. Ich selbst verwende das Buch als begleitende und ergänzende Unterlage zu meinen Vorlesungen. Viele Lernziele dieses Buches können aber auch im Selbststudium erreicht werden.

Englische vs. deutsche Begriffe

Ein Problem für ein Informatikbuch stellt die Verwendung englischer Begriffe dar. Da die Wissenschaftssprache der Softwaretechnik Englisch ist, gibt es für viele Begriffe – insbesondere in Spezialgebieten – keine oder noch keine geeigneten oder üblichen deutschen Fachbegriffe. Auf der anderen Seite gibt es jedoch für viele Bereiche der Softwaretechnik sowohl übliche als auch sinnvolle deutsche Bezeichnungen, z. B. Entwurf für *Design*.

Da mit einem Lehrbuch auch die Begriffswelt beeinflusst wird, bemühe ich mich in diesem Buch, sinnvolle und übliche deutsche Begriffe zu verwenden. Ist anhand des deutschen Begriffs *nicht* unmittelbar einsehbar oder allgemein bekannt, wie der englische Begriff lautet, dann wird in Klammern und kursiv der englische Begriff hinter dem deutschen Begriff aufgeführt. Dadurch wird auch das Lesen der englischsprachigen Literatur erleichtert.

Englische Begriffe *kursiv* gesetzt

Gibt es noch keinen eingebürgerten deutschen Begriff, dann wird der englische Originalbegriff verwendet. Englische Bezeichnungen sind immer *kursiv* gesetzt, sodass sie sofort ins Auge fallen.

Lesen des Buches: sequenziell

Ziel der Buchgestaltung war es, Ihnen als Leser viele Möglichkeiten zu eröffnen, dieses Buch nutzbringend für Ihre eigene Arbeit einzusetzen. Sie können dieses Buch sequenziell von vorne nach hinten lesen. Die Reihenfolge der Kapitel ist so gewählt, dass die Voraussetzungen für ein Kapitel jeweils erfüllt sind, wenn man das Buch sequenziell liest.

Nach Teildisziplinen

Eine andere Möglichkeit besteht darin, jeweils eine der Teildisziplinen »Basistechniken«, »Basiskonzepte«, »*Requirements Engineering*« durchzuarbeiten. Auf Querbezüge und notwendige Voraussetzungen wird jeweils hingewiesen.

Themenbezogen

Außerdem kann das Buch themenbezogen gelesen werden. Möchte man sich in die Grundlagen der statischen Basiskonzepte einarbeiten, dann kann man die dafür relevanten Kapitel durcharbeiten. Will man sich auf das *Requirements Engineering* konzentrieren, dann kann man auch nur diese Kapitel lesen.

Punktuell

Durch das Buchkonzept ist es natürlich auch möglich, punktuell einzelne Kapitel durchzulesen, um eigenes Wissen zu erwerben, aufzufrischen und abzurunden, z. B. Durchlesen des Kapitels über Petrinetze.

Zum Nachschlagen

Durch ein ausführliches Sachregister und Glossar sowie durch Zusammenfassungen kann dieses Buch auch gut zum Nachschlagen verwendet werden.

Projekt-bezogen

Soll das Buch begleitend zu einem Softwareprojekt eingesetzt werden, dann kann zunächst der »Requirements Engineering«, S. 433, behandelt werden. Parallel zum Projektverlauf können dann die für das Projekt relevanten Basiskonzepte behandelt werden.

Ich habe versucht, ein innovatives wissenschaftliches Lehrbuch der Softwaretechnik zu schreiben. Ob mir dies gelungen ist, müssen Sie als Leser selbst entscheiden.

Ein Buch soll aber nicht nur vom Inhalt her gut sein, sondern Form und Inhalt sollten übereinstimmen. Daher wurde auch versucht, die Form anspruchsvoll zu gestalten. Da ich ein Buch als »Gesamtkunstwerk« betrachte, ist auf der Buchtitelseite ein Bild der Malerin Anna Solecka-Zach abgedruckt. Sie setzt den Computer als Hilfsmittel ein, um ihre künstlerischen Vorstellungen umzusetzen.

Koautoren

Die dynamische Entwicklung der Softwaretechnik und die Themenbreite machen es für einen Autor fast unmöglich, alleine ein Lehrbuch der Softwaretechnik zu schreiben. Ich habe daher einige Kollegen gebeten, Teilgebiete durch eigene Beiträge abzudecken.

Prof. Dr. rer. nat. Rainer Koschke, Universität Bremen, hat das Kapitel zum Thema »Software-Werkzeuge« verfasst:

- »Werkzeuge«, S. 59

Prof. Dr.-Ing. Uwe Lämmel, Hochschule Wismar, hat ein Kapitel zum Thema »Regelbasierte Systeme« beigetragen:

- »Regeln«, S. 404

Prof. Dr.-Ing. Peter Liggesmeyer, Technische Universität Kaiserslautern und Institutsleiter des Fraunhofer Instituts für Experimentelles Software Engineering, und seine Mitarbeiter Dr. Robert Eschbach, Dr. Mario Trapp, Johannes Kloos und Bastian Zimmer haben mit der »Fallstudie Fensterheber« und entsprechenden Beispielen in den Basiskonzepten sowie mit dem Kapitel »Formale Logik« zu diesem Buch beigetragen:

- »Fallstudie: Fensterheber – Die Spezifikation«, S. 117
- »Fallstudie: Fensterheber – Die fachliche Lösung«, S. 575
- »Formale Logik«, S. 357
- »Markov-Ketten«, S. 292
- »Generalisierte stochastische Petrinetze«, S. 316

Dr.-Ing. Jochen Quante, Universität Bremen, hat das Thema »Aufwandsschätzmethoden« bearbeitet:

- »Schätzen des Aufwands«, S. 515

Von meiner Frau, Prof. Dr. Heide Balzert, Fachhochschule Dortmund, habe ich aus ihrem »Lehrbuch der Objektmodellierung«, 2. Auflage, eine Reihe von Abschnitten übernommen [Balz05]. Die Boxen mit Checklisten stammen – in etwas abgewandelter Form – ebenfalls aus dem »Lehrbuch der Objektmodellierung«.

Vorwort zur 3. Auflage

Dank

Ich danke allen Koautoren für ihre Beiträge zu diesem Lehrbuch.

Zusätzlich danke ich Dr.-Ing. Olaf Zwintzscher, Geschäftsführer der W3L GmbH in Witten und Lehrbeauftragter für Softwaretechnik an der Ruhr-Universität Bochum, für die kritische Durchsicht des gesamten Buches und Prof. Dr. Roland Gabriel, Lehrstuhl für Wirtschaftsinformatik an der Ruhr-Universität Bochum, für Anregungen zum Kapitel »Multidimensionale Datenstrukturen«.

Die Grafiken erstellte meine Mitarbeiterin Frau Anja Schartl. Danke!

Trotz der Unterstützung vieler Personen bei der Erstellung dieses Buches enthält ein so umfangreiches Werk sicher immer noch Fehler und Verbesserungsmöglichkeiten: »*Nobody is perfect*«. Kritik und Anregungen sind daher jederzeit willkommen. Eine aktuelle Liste mit Korrekturen und Informationen zu diesem Buch finden Sie im kostenlosen E-Learning-Kurs zu diesem Buch (siehe oben).

Nach soviel Vorrede wünsche ich Ihnen nun viel Spaß beim Lesen. Möge Ihnen dieses Buch – trotz der manchmal »trockenen« Materie – ein wenig von der Faszination und Vielfalt der Softwaretechnik vermitteln. Werden Sie ein guter Softwareingenieur – Sie werden gebraucht!

Ihr

Inhalt

I Die Wissenschaftsdisziplin Softwaretechnik 1

1 Was ist Software? 3

2 Warum ist Software so schwer zu entwickeln? 9

3 Was ist Softwaretechnik? 17

II Basistechniken 23

4 Prinzipien 25
4.1 Prinzip der Abstraktion 26
4.2 Prinzip der Strukturierung 34
4.3 Prinzip der Bindung und Kopplung 37
4.4 Prinzip der Hierarchisierung 38
4.5 Prinzip der Modularisierung 40
4.6 Geheimnisprinzip 42
4.7 Prinzip der Lokalität 45
4.8 Prinzip der Verbalisierung 46
4.9 Abhängigkeiten zwischen den Prinzipien 48
4.10 Zusammenfassung 50

5 Methoden 53

6 Werkzeuge 59
6.1 Menschen, Methoden, Werkzeuge 59
6.2 Klassifikation von Werkzeugen 60
6.2.1 Von Werkzeugen behandelte Artefakte 62
6.2.2 Von Werkzeugen unterstützte Operationen 62
6.2.3 Werkzeuge zur Kollaboration und Kommunikation 73
6.2.4 Unterstützung von Prozessmodellen und Methoden 75
6.3 Integrierte Entwicklungsumgebungen 76
6.4 Modellgetriebene Entwicklung 79
6.5 Auswahlkriterien bei der Anschaffung von Werkzeugen 87
6.6 Evaluationsverfahren für die Anschaffung 90
6.7 Zusammenfassung 97

III Basiskonzepte 99

7 Fallstudie: SemOrg – Die Spezifikation 107

8 Fallstudie: Fensterheber – Die Spezifikation 117

9 Statik 127
9.1 Funktionalität 127
9.1.1 Einzelne Funktionen 128
9.1.2 Zusammenfassung von Funktionen 131
9.1.3 Box: Klassen – Methode und Checkliste 137
9.2 Funktions-Strukturen 142
9.2.1 Funktionsbaum 143
9.2.2 Pakete 145
9.2.3 Box: Pakete – Methode und Checkliste 148
9.2.4 Vererbung 150
9.2.5 Box: Vererbung – Methode und Checkliste 155
9.2.6 Assoziation 158
9.2.7 Box: Assoziationen – Methode und Checkliste 166
9.2.8 Box: Multiplizitäten – Methode und Checkliste 169
9.2.9 Aggregation und Komposition 171
9.2.10 Box: Komposition und Aggregation – Methode und Checkliste 175
9.2.11 Weitere Strukturen 177
9.3 Daten 181
9.4 Box: Attribute – Methode und Checkliste 187
9.5 Daten-Strukturen 190
9.5.1 XML, DTD und XML-Schemata 190
9.5.2 Entity-Relationship-Modell 199
9.5.2.1 ER-Konzepte und OO-Konzepte im Vergleich 200
9.5.2.2 Schlüssel, Tabellen und Dateien 204
9.5.2.3 Beispiele für semantische Datenmodelle 207
9.5.2.4 Unternehmensdatenmodelle und Weltmodelle 209
9.5.2.5 Zusammenfassung 213
9.5.3 Multidimensionale Datenstrukturen 214
9.5.3.1 *Data Warehouse* und *Data Marts* 215
9.5.3.2 OLAP und Hyperwürfel 217
9.5.3.3 Modellierungsansätze 222
9.5.3.4 Zusammenfassung 226

10 Dynamik 227
10.1 Kontrollstrukturen 227
10.1.1 Die Sequenz 229
10.1.2 Die Auswahl 229
10.1.3 Die Wiederholung 231
10.1.4 Der Aufruf 235

10.1.5 Die Nebenläufigkeit 235
10.1.6 Aktivitätsdiagramm 236
10.1.7 Box: Aktivität – Methode und Checkliste 245
10.1.8 Zusammenfassung 249
10.2 Geschäftsprozesse und *Use Cases* 250
10.2.1 Konzepte und Notationen 251
10.2.2 EKPs und Aktivitätsdiagramme 253
10.2.3 *Use Case*-Diagramme und -Schablonen 255
10.2.4 Box: *Use Case* – Methode und Checkliste 262
10.2.5 Zusammenfassung 268
10.3 Zustandsautomaten 269
10.3.1 Erstellung eines Zustandsautomaten 270
10.3.2 Notationen 272
10.3.3 Zustandsautomat mit Endzuständen 274
10.3.4 Mealy-Automat vs. Moore-Automat 275
10.3.5 Zustandsautomat nach Harel 277
10.3.6 Verhaltens- vs. Protokollzustandsautomaten 287
10.3.7 Markov-Ketten 292
10.3.8 Box: Zustandsautomat – Methode und Checkliste 295
10.3.9 Zusammenfassung 301
10.4 Petrinetze 303
10.4.1 Bedingungs/Ereignis-Netze 305
10.4.2 Stellen/Transitions-Netze 309
10.4.3 Prädikat/Transitions-Netze 311
10.4.4 Hierarchische Petrinetze 312
10.4.5 Zeitbehaftete Petrinetze 314
10.4.6 Generalisierte stochastische Petrinetze 316
10.4.7 Aktivitätsdiagramme und Petrinetze 317
10.4.8 Strukturelemente und Strukturen von Petri-Netzen 320
10.4.9 Box: Petrinetze – Methode 322
10.4.10 Analyse und Simulation von Petrinetzen 327
10.4.11 Wertung 328
10.4.12 Zusammenfassung 330
10.5 Szenarien 332
10.5.1 Sequenzdiagramm 333
10.5.2 Kommunikationsdiagramm 343
10.5.3 Box: Sequenz- und Kommunikationsdiagramm – Methode und Checkliste 346
10.5.4 Timing-Diagramm 352
10.5.5 Zusammenfassung 355

11 Logik 357
11.1 Formale Logik 357
11.1.1 Aussagenlogik 358
11.1.2 Prädikatenlogik 367

11.1.3 Temporale Logik 370
11.1.4 Zusammenfassung 377
11.2 *Constraints* und die OCL in der UML 377
11.2.1 *Constraints* in der UML 378
11.2.2 OCL 380
11.2.3 Zusammenfassung 386
11.3 Entscheidungstabellen und Entscheidungsbäume 386
11.3.1 Erstellung einer Entscheidungstabelle 387
11.3.2 Anwendung einer Entscheidungstabelle 389
11.3.3 Überprüfung und Optimierung von Entscheidungstabellen 391
11.3.4 Darstellungsformen für Entscheidungstabellen 393
11.3.5 Entscheidungstabellen-Verbunde 394
11.3.6 Erweiterte Entscheidungstabellen 399
11.3.7 Eintreffer- und Mehrtreffer-Entscheidungstabellen 400
11.3.8 Zusammenfassung & Bewertung 402
11.4 Regeln 404
11.4.1 Aufbau von Regeln 405
11.4.2 Auswahl von Regeln 407
11.4.3 Regelbasierte Software 409
11.4.4 Der Rete-Algorithmus 412
11.4.5 Verkettung von Regeln 414
11.4.6 Lösungssuche 417
11.4.6.1 Der Suchbaum 419
11.4.6.2 Tiefe-zuerst-Suche 420
11.4.6.3 Breite-zuerst-Suche 422
11.4.6.4 Heuristische Suche 425
11.4.7 Bewertete Regeln 426
11.4.8 Geschäftsregeln 428
11.4.9 Anwendungen 429
11.4.10 Zusammenfassung 430

IV *Requirements Engineering* 433

12 Problem vs. Lösung 437

13 Bedeutung, Probleme und *Best Practices* 439

14 Aktivitäten und Artefakte 443

15 Der *Requirements Engineering*-Prozess 449

16 Anforderungen und Anforderungsarten 455
16.1 Visionen und Ziele 456
16.2 Rahmenbedingungen 459

16.3 Kontext und Überblick 461
16.4 Nichtfunktionale Anforderungen 463
16.5 Box: Qualitätsmerkmale nach ISO/IEC 9126–1 468
16.6 Abnahmekriterien 471

17 Anforderungen an Anforderungen 475

18 Anforderungsattribute 479

19 Natürlichsprachliche Anforderungen 481

20 Anforderungsschablonen 485
20.1 Anforderungsschablone der IEEE 830–1998 485
20.2 Anforderungsschablonen im V-Modell XT 487
20.3 Schablonen für Lastenheft, Pflichtenheft und Glossar 492
20.4 Schablonen für agile Entwicklungen 497

21 Anforderungen ermitteln und spezifizieren 503

22 Anforderungen analysieren, validieren und abnehmen 513

23 Schätzen des Aufwands 515
23.1 Voraussetzungen und Einflussfaktoren 515
23.2 Warum ist das Schätzen des Aufwands wichtig? 517
23.3 Warum eine Aufwandsschätzung schwierig ist 518
23.4 Schätzverfahren 522
23.4.1 Analogiemethode 523
23.4.2 Expertenschätzung 523
23.4.3 Bottom-up-Methode 524
23.4.4 Prozentsatzmethode 525
23.4.5 Algorithmische Schätzung 526
23.4.6 Faustregeln 526
23.5 Die Function-Points-Methode 527
23.6 *Object Points/Application Points* 535
23.7 COCOMO II 536
23.8 Bewertung und weitere Aspekte 539
23.9 Zusammenfassung 542

24 Anforderungen priorisieren 543

25 Anforderungen modellieren 547
25.1 Beispiel: Objektorientierte Analyse 548
25.1.1 Strukturierung der OOA-Konzepte 549
25.1.2 OOA-Muster 550

25.1.3 OOA-Methode 559
25.2 Domänenspezifische Sprachen 563

26 Fallstudie: SemOrg V1.0 – Die fachliche Lösung 565

27 Fallstudie: Fensterheber – Die fachliche Lösung 575

28 Modellierte Anforderungen analysieren, verifizieren und abnehmen 587

Glossar 589

Literatur 605

Sachindex 619