
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.

Weitere Bände in der Reihe <http://www.springer.com/series/4393>

Manfred Broy · Marco Kuhrmann

Einführung in die Softwaretechnik

Manfred Broy
Software & Systems Engineering,
Fakultät für Informatik
Garching, Bayern, Deutschland

Marco Kuhrmann 
Fakultät für Informatik und
Mathematik, Universität Passau
Passau, Bayern, Deutschland

ISSN 1439-5428

ISSN 2522-0667 (electronic)

Xpert.press

ISBN 978-3-662-50262-4

ISBN 978-3-662-50263-1 (eBook)

<https://doi.org/10.1007/978-3-662-50263-1>

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

© Springer-Verlag GmbH Deutschland, ein Teil von Springer Nature 2021

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von allgemein beschreibenden Bezeichnungen, Marken, Unternehmensnamen etc. in diesem Werk bedeutet nicht, dass diese frei durch jedermann benutzt werden dürfen. Die Berechtigung zur Benutzung unterliegt, auch ohne gesonderten Hinweis hierzu, den Regeln des Markenrechts. Die Rechte des jeweiligen Zeicheninhabers sind zu beachten.

Der Verlag, die Autoren und die Herausgeber gehen davon aus, dass die Angaben und Informationen in diesem Werk zum Zeitpunkt der Veröffentlichung vollständig und korrekt sind. Weder der Verlag, noch die Autoren oder die Herausgeber übernehmen, ausdrücklich oder implizit, Gewähr für den Inhalt des Werkes, etwaige Fehler oder Äußerungen. Der Verlag bleibt im Hinblick auf geografische Zuordnungen und Gebietsbezeichnungen in veröffentlichten Karten und Institutionsadressen neutral.

Planung: Sybille Thelen

Springer Vieweg ist ein Imprint der eingetragenen Gesellschaft Springer-Verlag GmbH, DE und ist ein Teil von Springer Nature.

Die Anschrift der Gesellschaft ist: Heidelberger Platz 3, 14197 Berlin, Germany

Geleitwort

Die durch Software kreierte und getragene digitale Welt unterscheidet sich deutlich von unserer physischen Welt mit ihren naturgesetzlichen Beschränkungen. Wir können in Bruchteilen von Sekunden rund um die Welt kommunizieren, zusammenarbeiten und auch mit der weit entfernten physischen Welt dort interagieren. Räume und räumliche Entfernungen spielen keine Rolle und auch die Gravitation, die unsere physische Welt prägt, hat keinerlei Wirkung in der digitalen Welt. Mit Hilfe von Software kann fast alles abgebildet werden, was die menschliche Vorstellungskraft hergibt, in vielerlei Hinsicht Fluch und Segen zugleich.

Als die ersten Computer in die Welt kamen, war Programmierung eine Magie, eine geheime Kunst, die nur wenige beherrschten und mit der die ersten Programmierer die Welt verzauberten und in Erstaunen versetzten. Die damals entwickelten Softwareprogramme kamen nur auf wenigen Rechnern zum Einsatz. Wenn etwas nicht funktionierte, war nie ganz klar, ob es am Programm, an der Betriebssystemumgebung oder doch an der ausführenden Hardware lag, alles wurde aufwendig in Handarbeit erstellt und geprüft.

Als dann die Hardware leistungsfähiger und die Ansprüche an die Programmierung immer komplexer wurden, die entwickelten Systeme aufeinander aufbauen und gegenseitig integrierbar sein sollten, reichte die Kunst des einzelnen Programmierers nicht mehr aus, solche Systeme zu entwickeln, und es gelang nicht mehr, die Wechselwirkungen zu übersehen oder gar zu beherrschen. Wird Software erweitert und integriert, steigt die Komplexität eben exponentiell und nicht nur linear an. Um diese Komplexität zu meistern, braucht es eine fundierte wissenschaftliche Methodik, grundlegende Entwicklungsprinzipien und standardisierte, wo immer möglich automatisierte, Techniken. Die wissenschaftliche Durchdringung der systematischen industriellen Softwareentwicklung ist das Feld des Software Engineering.

Heute nun steigen die Anforderungen an die Softwareentwicklung rasant weiter an. Selbstfahrende Autos, vernetzte Fabriken, interaktive digitale Markt- und Informationsplätze, KI-gesteuerte Systeme funktionieren nur, wenn ganze Heerscharen von Programmierern, unterstützt durch leistungsfähige Tools und Methoden, in die Lage versetzt werden, zu kooperieren und performante, wo immer möglich, energieeffiziente

Software herzustellen, auszurollen und zum Einsatz zu bringen. Sie müssen denselben Entwicklungsprinzipien folgen, Prozesse modellieren, abgestimmte Methoden anwenden und Projekte managen können. Bereits vor der Erstellung der ersten Programmzeile müssen sich Softwareentwickler im Requirement Engineering der Frage stellen, was das zu entwickelnde Programm genau leisten soll, wie es sich in die vorhandene Systemlandschaft integriert und wie es nutzerfreundlich zu bedienen ist. Gleichzeitig muss die Architektur des Systems im Blick behalten werden, was durch Modularisierung und objektorientierte Programmierung gelingen kann. Für die Integration verschiedener Komponenten braucht es eine eindeutig definierte Schnittstellensystematik, und über den ganzen Entwicklungsprozess hinweg muss auf Qualität, Effizienz, Wartbarkeit, Dokumentation und Anpassungsfähigkeit der Software geachtet werden.

Software Engineering vereint Kenntnisse und Techniken aus ganz verschiedenen Disziplinen. Programmierer sind heute keine Magier mehr, sondern methodisch ausgebildete und arbeitende Ingenieure, die auf Entwicklungsstandards aufsetzen, Frameworks nutzen, auf Software-Bausteine und Interaktionsmethoden zurückgreifen können, die im Software Engineering erforscht wurden. Die Entwicklung von komplexen Softwaresystemen ist heute ein industrialisiert durchorganisierter Prozess, der in großen Teilen sogar automatisiert abläuft. Nur so kann überhaupt sichergestellt werden, dass die millionenfach in allen Bereichen unseres Lebens zum Einsatz kommenden Anwendungen nutzerfreundlich und fehlerfrei funktionieren und miteinander interagieren können.

Die Informatiker Manfred Broy und Marco Kuhmann widmen dem spannenden Feld des Software Engineering mit dem vorliegenden Band ein Lehrbuch, in das ihre Erfahrungen aus über 30 Jahren Forschung, Lehre und Projektarbeit in diesem Bereich eingeflossen sind. Das Buch vereint die Darstellung der klassischen Tugenden der hierarchisch-deduktiven ingenieurmäßigen Softwareentwicklung – dem „Butter und Brot“-Geschäft der Softwareentwicklung – mit neuen Arbeitsparadigmen und agilen Methoden, wie zum Beispiel Scrum. Das Spannungsverhältnis zwischen diesen zwei Arbeitsweisen wird im Lehrbuch explizit beschrieben und für den Leser in verständlicher Weise aufbereitet. Damit ist es nicht nur interessant für Softwareingenieure und Informatiker, sondern auch für Fachleute anderer Professionen, die mit der Konzeption, der Entwicklung und dem Einsatz von digitalen Technologien in Wirtschaft, Wissenschaft und Verwaltung befasst sind.

Vorwort

Software Engineering ist eine Disziplin in der Informatik, die ein ingenieurmäßiges Vorgehen bei der Entwicklung umfangreicher, leistungsstarker Softwaresysteme beschreibt. Dies umfasst Aufgaben der *Softwaretechnik* – der fachlichen, methodischen und technischen Entwicklung von Softwaresystemen – und Themen der *Projektorganisation und des Managements von Softwareprojekten* und projektübergreifende Aufgaben, welche die Fähigkeit von Unternehmen zur Entwicklung softwareintensiver Systeme sicherstellen. In nur wenigen Jahrzehnten hat das Software Engineering eine hohe wirtschaftliche und technische, aber auch eine große gesellschaftliche Bedeutung erlangt. Kaum ein Produkt kommt heute noch ohne Software aus und kaum eine Innovation ist heute ohne Software möglich. Verwaltung, Verkehr und Logistik, Infrastruktur und Produktion sind maßgeblich auf komplexen IT-Ökosystemen aufgebaut, deren Funktion und Leistungsfähigkeit fast ausschließlich durch Software bestimmt wird. Somit kommt der Fähigkeit der Wirtschaft, schnell, kostengünstig und nutzergerecht Software hoher Qualität zu entwickeln, entscheidende Bedeutung zu.

Klassisch, modern oder klassisch-modern? Wie andere Disziplinen auch, insbesondere solche mit großer praktischer Bedeutung, ist das Software Engineering nicht ganz frei von Ideologien, Philosophien und Modeströmungen. Ein *umfassender Ansatz* für Software Engineering muss eine klare, konzeptuelle, ja auch eine philosophische Basis haben und in sich stimmig sein. Wenn jedoch diese Basis zu ideologisch wird, bestimmt von nicht eindeutig belegten Behauptungen, so ist das mit Vorsicht zu genießen.

Im Software Engineering finden sich zwei große Philosophien für die Softwareentwicklung: Einerseits findet sich eine stringente, planorientierte, stärker modellbasierte, stark auf Dokumente abgestützte und arbeitsteilige Vorgehensweise, die oft als *konventionelle* Vorgehensweise bezeichnet wird, als Vorgehen nach Phasenmodellen wie dem V-Modell oder auch als Vorgehen nach dem Wasserfallmodell. Andererseits haben sich die *agilen* Vorgehensweisen etabliert, ohne dass das Wort „agil“ hinreichend eindeutig definiert wird. Agil zielt darauf ab, dass einem Projektteam viel Verantwortung gegeben wird und dass das Team selbstbestimmt arbeitet, unsinnige Dokumentationen

und Beharren auf Festlegungen, die nicht haltbar sind, vermeidet. Hinzu kommt die Vorstellung, dass es vorteilhaft ist, schnell in die Codierung einzusteigen, da erfahrene Programmierer dadurch schnell Rückkopplung bekommen, ob gewählte Lösungsansätze funktionieren. Zwischen diesen beiden Extrempositionen der dokumentenzentrierten Entwicklung und der agilen Entwicklung gibt es natürlich jede Menge Zwischenformen.

Das Wesentliche ... Für dieses Buch musste im Text ein Weg gefunden werden, die unterschiedlichen philosophischen Ansätze gleichermaßen zu behandeln und sie gleichberechtigt nebeneinander zu beschreiben. Wir gehen dabei wie folgt vor:

Bei der Entwicklung von Softwaresystemen gibt es *Kernaufgaben*, wie die Festlegung der Anforderungen, der Entwurf der Architektur, die eigentliche Umsetzung in den Code, die Qualitätssicherung, insbesondere die Verifikation durch Tests, die Integration, die Auslieferung und die Evolution. Diese Aufgaben werden – welcher Ansatz auch immer gewählt wird – im Rahmen einer Softwareentwicklung notwendigerweise bearbeitet, ob das bewusst explizit und gesondert passiert oder eher implizit, mit anderen Kernaufgaben verzahnt und ohne viel Dokumentation.

Das Buch ist nach diesen Kernaufgaben organisiert, wobei wir klar herausstellen, dass die einzelnen Aufgaben oft nicht isoliert oder zwingend sequenziell behandelt, sondern häufig miteinander verzahnt iterativ und inkrementell bearbeitet werden.

Ziele des Buchs

Mit diesem Buch adressieren wir die grundlegenden Themen der *Softwaretechnik* als wesentlichen Teil des Software Engineerings. Dabei legen wir einen starken Fokus auf die systematische und modellbasierte Software- und Systementwicklung unter Einbezug moderner Techniken des agilen Vorgehens auch für die Entwicklung sicherheitskritischer Systeme. Erfahrungen und praktische Beispiele geben Orientierung. Dieses Buch gibt einen Einstieg in die für Software- und Systementwicklungsprojekte besonders relevanten Grundlagen und Kernthemen:

- Erfassung von Anforderungen
- Entwurf von situationsangemessenen Architekturen
- Qualitätsorientierte Implementierung von Software
- Durchgängiger Entwicklungsprozess

Dieses Buch ist kein Kompendium und erhebt nicht den Anspruch ein möglichst umfangreiches und vollständiges Repertoire aller Methoden und Werkzeuge des Software Engineerings darzustellen. Mit der Fokussierung auf die vier oben genannten Themen wird das grundlegende Instrumentarium behandelt, welches durch Softwareingenieure beherrscht werden **muss**, da es die Grundlage für alle weiteren Aufgaben in Software- und Systementwicklungsprojekten darstellt. Alle Themen werden durch Methoden, Erfahrungen, Anwendungsbeispiele und Übungsaufgaben umfassend beschrieben. Ziel dieses Buchs ist eine Einführung in das Thema *Softwaretechnik* für Informatiker und

Softwareingenieure und die Bereitstellung einer Hilfe bei der Organisation und Durchführung der technischen Aufgaben in Software- und Systementwicklungsprojekten.

Tiefes Verständnis für die Aufgaben des Software Engineerings erfordert gleichermaßen Einsichten zum Stand der Wissenschaft, zu den Methoden, Grundlagen und Erfahrungen bei der Erstellung von Software in den unterschiedlichsten Unternehmen und Anwendungsgebieten. Beides ist in dieses Buch eingeflossen.

Achtung *Wie kurz dargestellt, leben Menschen mehr und mehr in einer Welt, die durch Software geprägt ist. Daraus ergibt sich eine hohe Verantwortung für die Gestaltung von Softwaresystemen. Softwareentwickler sollten sich dieser Verantwortung stets bewusst sein und sie in ihren Aufgaben aktiv wahrnehmen.*

Zielgruppe und erforderliches Vorwissen

Dieses Lehrbuch richtet sich an Studierende in Bachelor-, Master- und Diplom-Studiengängen der Informatik, Wirtschaftsinformatik und verwandter Disziplinen aber auch an Praktiker, insbesondere Berufs- und Quereinsteiger. Es liefert einen Einstieg in wissenschaftlich fundierte und praktisch relevante Grundlagen der Softwaretechnik.

Dieses Buch behandelt die methodischen Grundlagen der Softwaretechnik wie Anforderungserhebung, Modellierung, Architektur, Implementierung und Qualitätssicherung. Einen Schwerpunkt dieses Buchs bildet die Propädeutik mit dem Ziel, auch den relevanten Begriffsapparat wissenschaftlich fundiert einzuführen. Hilfreich für das Verständnis sind Grundkenntnisse der Mathematik und insbesondere in der Programmierung, insbesondere in der objektorientierten Programmierung. Auch dazu behandelt das Buch die wesentlichen Grundlagen, jedoch ist es kein Programmierkurs. Vorteilhaft sind erste Erfahrungen in Projekten und in Teamarbeit.

Aufbau des Buchs

Das Buch besteht aus fünf Teilen. Im Teil 1 werden die Grundlagen behandelt und die Begriffsbildung vorgenommen. In diesem Rahmen werden die *Eigenschaften und Strukturen von Softwaresystemen*, die *Vorgehensmodelle in der Softwareentwicklung* und die *Modelle in der Softwareentwicklung und ihre Beschreibung* eingeführt. Die Teile 2 bis 4 greifen dann die Kernaufgaben der Softwareentwicklung auf. Der Teil 2 stellt die Grundlagen der Anforderungsanalyse – dem Requirements Engineering – dar. Insbesondere werden die Themen *Anforderungsanalyse und Anforderungsmanagement*, *Produkt- und Qualitätsanforderungen* sowie das *Vorgehen in der Anforderungsanalyse* besprochen. Teil 3 behandelt den Systementwurf und die Architekturspezifikation. Es werden zunächst die *Grundlagen und Prinzipien des Architekturentwurfs* behandelt, gefolgt von Techniken im Bereich *Architekturentwurf und Architekturmodellierung* und der *Nutzung bewährten Architekturwissens*. Die Teile 2 und 3 sind dabei so gestaltet, dass ein Bogen von den Grundlagen hin zur Anwendung gespannt wird. Teil 4 widmet sich schließlich der Implementierung, Integration und Verifikation von Software. In diesem Rahmen werden die *Implementierung von Softwaresystemen*, die *Verifikation und*

Integration von Software und die *Softwareevolution* behandelt. Den fünften Teil dieses Buchs bildet ein Anhang. Im Anhang finden sich eine *UML Kurzreferenz*, welche die für dieses Buch erforderlichen Notationselemente der Unified Modeling Language einführt, sowie *weiterführende Beispiele für die Implementierung* von Software, insbesondere unter Verwendung von Mustern. Abschließend finden sich im Anhang noch die *Projektunterlagen Code & Talk*, ein Übungsbeispiel welches die Grundlage für viele Übungsaufgaben in diesem Buch legt.

Teil 1: Grundlagen & Begriffsbildung	Grundlagen
	Eigenschaften und Strukturen von Softwaresystemen
	Vorgehensmodelle in der Softwareentwicklung
	Modelle in der Softwareentwicklung und ihre Beschreibung
Teil 2: Anforderungsanalyse	Anforderungsanalyse und Anforderungsmanagement
	Produkt- und Qualitätsanforderungen
	Vorgehen in der Anforderungserhebung
Teil 3: Systementwurf und Architekturspezifikation	Grundlagen und Prinzipien des Architekturentwurfs
	Architekturentwurf und Architekturmodellierung
	Nutzung bewährten Architekturwissens
Teil 4: Implementierung, Integration und Qualitätssicherung von Software	Implementierung von Softwaresystemen
	Verifikation und Integration von Software
	Softwareevolution
Teil 5: Anhang	UML Kurzreferenz
	Weiterführende Beispiele für die Implementierung
	Projektunterlagen Code & Talk

Dieses Buch befasst sich mit Methoden und Techniken der Softwareentwicklung. Es steht somit in Beziehung zur etablierten Standardliteratur der *Softwaretechnik* und entsprechenden Empfehlungen für den Aufbau von *Software Engineering Programmen*, wie etwa dem *ACM/IEEE Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*¹. Die nachfolgende Tabelle stellt eine Abbildung der Inhalte dieses Buchs zu den empfohlenen Inhalten des ACM/IEEE- Curriculums dar. Die Tabelle

¹ACM/IEEE Joint Task Force on Computing Curricula, *Software Engineering 2014 – Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*, 23. Februar 2015, Online: <https://dl.acm.org/doi/book/10.1145/2594168>.

benennt im linken Teil die sogenannten *Knowledge Areas* (inklusive der Abkürzungen) und stellt diesen die Kapitel und, wo erforderlich, die Unterkapitel dieses Buchs mit den entsprechenden Inhalten gegenüber.

Titel Knowledge Area		Kapitel
Computing Essentials	CMP	Kap. 4 (CMP.cf.4), Kap. 10 (CMP.ct.1–3)
Mathematical and Engineering Fundamentals	FND	–
Professional Practice	PRF	–
Software Modeling and Analysis	MAA	Kap. 4 (MAA.md und MAA.tm)
Requirements Analysis and Specification	REQ	Kap. 5, 6 und 7 (vollständig)
Software Design	DES	Kap. 8, 9, 10 und Kap. A (vollständig bis auf DES.str.4; DES.ar.3, 6, 7; DES.dd.2, 3; DES.ev.2, 3; DES.hci)
Software Verification and Validation	VAV	Kap. 12 (vollständig VAV.fnd, VAV.rev, VAV.tst; bis auf VAV.tst.12, 13), Kap. 2 (VAV.fnd.4)
Software Process	PRO	Kap. 3 (PRO.con.1, 7; PRO.imp.2), Kap. 11.5 (PRO.cm.1–3), Kap. 12.4 (PRO.cm.4, 5), Kap. 13 (PRO.evo)
Software Quality	QUA	Kap. 2 (QUA.cc.5, QUA.pda.5), Kap. 12.1 (QUA.cc.1), Kap. 13 (QUA.cc.3)
Security	SEC	–

Hier fehlt aber noch etwas ...

Trotz der Breite in den Themen, die in diesem Buch behandelt werden, darf nicht vergessen werden, dass es sich um eine *Einführung* handelt. Insbesondere die Auswahl von Techniken, Methoden und Werkzeugen in diesem Buch ist geprägt von unserem Erfahrungsschatz und den Verfügbarkeiten einzelner Standards und Werkzeuge zum Zeitpunkt der Erstellung dieses Buchs. Es besteht daher kein Anspruch auf Vollständigkeit und – aufgrund der schnellen Innovationszyklen – ebenfalls kein Anspruch auf umfassende Aktualität. Wir erheben aber auch nicht den Anspruch, dass Software Engineering nur und ausschließlich mit den Inhalten

dieses Buchs gestaltet werden kann. Daher laden wir den geneigten Leser dazu ein, seine eigenen Vorstellungen und Erfahrungen in das Rahmenwerk dieses Buchs einzubringen.

Danksagung

Das vorliegende Buch ist über einen Zeitraum von mehr als 25 Jahren im Rahmen einer mehrfach gehaltenen Vorlesungsreihe zum Thema Software Engineering und Softwaretechnik an der Technischen Universität München und entsprechenden Veranstaltungen an den Universitäten Odense und Passau entstanden. Eingeflossen sind dabei Erfahrungen bei der Durchführung der Vorlesungen und den begleitenden Übungen, aus den mehrfach durchgeführten Softwaretechnik-Praktika und Semesterprojekten, in denen 5 bis 20 Studenten über ein Semester größere Softwareprojekte durchgeführt haben und nicht zuletzt aus einer Reihe von Erkenntnissen aus der Zusammenarbeit und Projekten mit führenden Wirtschaftsunternehmen zu Themen der Softwareentwicklung.

Vor diesem Hintergrund ist es uns ein Bedürfnis, den Studenten, Doktoranden, Mitarbeitern, Kollegen und Gesprächspartnern aus der Industrie für ihre Beiträge und kritischen Anregungen zu danken. Insbesondere gilt unser Dank Dr. Herbert Ehler, PD Dr. Bernhard Schätz, Dr. Marc Sihling, Dr. Klaus Bergner, Prof. Dr. Andreas Rausch und Prof. Dr. Daniel Méndez Fernández für ihre Unterstützung bei der Erarbeitung der Foliensätze und Vorlesungsmanuskripte, welche in dieses Buch mit eingeflossen sind. Weiterhin danken wir insbesondere unseren stets kritischen Reviewern Dr. Jens Calamé und Prof. Dr. Oliver Linssen. Ohne sie wäre es nicht möglich gewesen, dem vorliegenden Text eine hinreichend tiefe wissenschaftliche Dimension und stark praktische Note zu geben. Dr. Sebastian Eder danken wir für die kritischen Kommentare zum fast fertigen Manuskript. Nicht vergessen wollen wir auch Hermann Engesser, der uns dieses Buchprojekt ermöglicht hat und, vor allem, Dorothea Glaunsinger, Dr. Sabine Kathke und Sybille Thelen für ihre unendliche Geduld und Hilfe. Wir hoffen, dass dieses Buch dem Leser nützlich ist, die eigenen Vorstellungen vom Thema Software Engineering zu festigen und weiter zu entwickeln.

München, Garching, Passau
2020

Manfred Broy
Marco Kuhrmann

Inhaltsverzeichnis

Teil I Grundlagen und Begriffsbildung

1	Grundlagen	3
1.1	Software is Eating the World	4
1.1.1	Herausforderungen in der Softwareentwicklung	7
1.1.2	Zielsetzung des Software Engineering	12
1.1.3	Prinzipien und Erfolgsfaktoren	14
1.2	Grundlegende Begriffe	16
1.3	Kernthemen der Softwaretechnik	22
1.3.1	Erfassung und Verfeinerung der Anforderungen	27
1.3.2	Architektur	30
1.3.3	Implementierung, Integration und Verifikation	33
1.3.4	Betrieb und Evolution	35
1.3.5	Vorgehensweisen in der Softwareentwicklung	36
	Literatur	38
2	Eigenschaften und Strukturen von Softwaresystemen	41
2.1	Charakterisierung von Softwaresystemen	41
2.1.1	Der Kontext	45
2.1.2	Systemverhalten, Schnittstellen und Funktionen	46
2.1.3	Sichten auf Softwaresysteme	48
2.2	Qualitätseigenschaften von Softwaresystemen	51
2.2.1	Produkt- und Nutzungsqualität	53
2.2.2	Qualität in der Nutzung	54
2.2.3	Qualität in der Entwicklung und Evolution	58
2.2.4	Qualität im Betrieb	60
2.2.5	Qualität in der Vermarktung (Vermarktbarkeit)	61

2.3	Messung von Systemeigenschaften	61
2.3.1	Messung und Vermessung von Software	62
2.3.2	Festlegung von Metriken	64
2.3.3	Softwaremetriken	65
2.4	Weiterführende Literatur und Übungen	75
	Literatur	78
3	Vorgehensmodelle in der Softwareentwicklung	83
3.1	Was ist ein Vorgehensmodell?	83
3.2	Grundlegende Vorgehensmodelle und Prozessbeschreibungen	85
3.2.1	Phasenorientierte Modelle und sequenzielles Vorgehen	86
3.2.2	Iteratives und inkrementelles Vorgehen	89
3.2.3	Prototyping	92
3.2.4	Agile Vorgehensweisen	94
3.3	Das V-Modell XT	98
3.3.1	Rollen im V-Modell XT	98
3.3.2	V-Modell XT Produkte für die Systementwicklung	99
3.3.3	Vorgehensweisen im V-Modell XT	100
3.4	Scrum	104
3.4.1	Rollen in Scrum	104
3.4.2	Scrum Artefakte und Prozess	105
3.4.3	Anforderungen an die Organisation	107
3.4.4	Scrum in der Praxis	107
3.4.5	Skalierung von Scrum	108
3.4.6	Scrumban	112
3.5	Rollen und Verantwortlichkeiten	115
3.5.1	Der Product Owner	115
3.5.2	Weitere zentrale Projektrollen	116
3.6	Weiterführende Literatur und Übungen	117
	Literatur	121
4	Modelle in der Softwareentwicklung und ihre Beschreibung	125
4.1	Warum Modellierung?	125
4.2	Modelle und ihre Beschreibung	127
4.2.1	Grundsätzliches zur Modellbildung in der Informatik	129
4.2.2	Mathematische Modellierung von Informationsverarbeitung	131
4.2.3	Grafische Modellierung am Beispiel UML	135
4.3	Modellierung von Daten	138
4.3.1	Algebraische Spezifikation	140
4.3.2	Entity-Relationship-Modelle	144

4.3.3	UML-Klassendiagramme	147
4.3.4	Typ- und Sortendeklarationen	148
4.4	Spezifikation von Funktionen und Prozeduren	149
4.4.1	Spezifikation von Funktionen	149
4.4.2	Spezifikation von Prozeduren	151
4.4.3	Operationen in UML-Klassendiagrammen	155
4.5	Modellierung von Zuständen	156
4.5.1	Zustandsmaschinen mit Ein- und Ausgabe	156
4.5.2	Zustandsmaschinen in der UML	161
4.6	Modellierung von verteilten Systemen	163
4.6.1	Komponenten und Schnittstellen	163
4.6.2	Architekturen	175
4.6.3	Daten- und Kontrollfluss	178
4.6.4	Prozesse	184
4.7	Weiterführende Literatur und Übungen	189
	Literatur.	192

Teil II Anforderungsanalyse

5	Anforderungsanalyse und Anforderungsmanagement	199
5.1	Ziele und Aufgaben einer Software	199
5.2	Die zentrale Rolle der Anforderungen	200
5.2.1	Die Nutzer und ihr Erlebnis stehen im Zentrum	201
5.2.2	Grundstruktur der Anforderungen	202
5.2.3	Anforderungsanalyse	203
5.2.4	Klassifikation von Anforderungen	206
5.2.5	Anforderungen an die Systemverlässlichkeit.	209
5.3	Kernaufgaben in der Erarbeitung der Systemanforderungen.	211
5.3.1	Kernartefakte in der Anforderungsanalyse.	211
5.3.2	Grundlegendes Vorgehen in der Anforderungserhebung	212
5.4	Rollen in der Anforderungsanalyse	214
5.5	Management von Anforderungen im Produktlebenszyklus	215
5.5.1	Änderungsmanagement	216
5.5.2	Versions- und Konfigurationsmanagement	217
5.5.3	Anforderungsverifizierbarkeit	217
5.5.4	Anforderungsverfolgung	218
5.5.5	Anforderungsbegründung	218
5.6	Weiterführende Literatur und Übungen	219
	Literatur.	221

6	Produkt- und Qualitätsanforderungen	223
6.1	Produktanforderungen	223
6.1.1	Funktionsarchitektur und Anwendungsfälle	226
6.1.2	Mensch-Maschine Interaktion	237
6.2	Qualitätsanforderungen	241
6.2.1	Funktionsbezogene Qualitätsanforderungen	241
6.2.2	Nutzungsbezogene Qualitätsanforderungen	242
6.2.3	Entwicklungsbezogene Qualitätsanforderungen	242
6.2.4	Qualitätsattribute	243
6.3	Weiterführende Literatur und Übungen	245
	Literatur	248
7	Vorgehen in der Anforderungserhebung	251
7.1	Grundsätzliches Vorgehen in der Anforderungserhebung	251
7.1.1	Domänenanalyse	252
7.1.2	Allgemeiner Prozess der Anforderungserhebung	252
7.1.3	Das Lastenheft und Anwenderforderungen	257
7.1.4	Anforderungsfestlegung und Dokumentation	258
7.2	Techniken für die Anforderungserhebung	259
7.2.1	Anforderungsquellen	260
7.2.2	Kreativitätstechniken	260
7.3	Methoden für die anforderungsgetriebene Entwicklung	263
7.3.1	Design Thinking	263
7.3.2	Feature-driven Development	268
7.4	Anforderungserhebung im Agilen Vorgehen	271
7.4.1	User Stories	273
7.4.2	Agile Praktiken in der Anforderungserhebung	275
7.4.3	Definition of Ready und Definition of Done	280
7.5	Modellierung von Anforderungen	284
7.5.1	Modellierung von Anwendungsfällen	285
7.5.2	Detaillierte Modellierung von Anforderungen	287
7.6	Qualitätssicherung für Anforderungen	294
7.6.1	Allgemeine Qualitätsaspekte von Anforderungen	295
7.6.2	Validierung von Anforderungen	295
7.6.3	Verifizierbarkeit von Anforderungen	296
7.7	Weiterführende Literatur und Übungen	297
	Literatur	300

Teil III Systementwurf und Architekturspezifikation

8	Grundlagen und Prinzipien des Architekturentwurfs	307
8.1	Strukturierung von Systemen	307
8.1.1	Softwarekomponenten	309
8.1.2	Erarbeitung der System- und Softwarearchitektur	310
8.1.3	Funktionen und Funktionsarchitektur	311
8.1.4	Architekturmodelle für Softwaresysteme	312
8.1.5	Perspektiven des Architekturentwurfs	320
8.1.6	Die zentrale Rolle des Architekten	327
8.2	Prinzipien des Architekturentwurfs	329
8.2.1	Das Gebot der Einfachheit: KISS	329
8.2.2	Kopplung und Kohäsion	330
8.2.3	Kapselung und Information Hiding	332
8.2.4	Separation of Concerns	335
8.2.5	Teile und Herrsche	336
8.2.6	Design by Contract	340
8.3	Wiederverwendung im Architekturentwurf	341
8.3.1	Was ist Wiederverwendung?	342
8.3.2	Herausforderungen in der Wiederverwendung	344
8.3.3	Strategien der Wiederverwendung	345
8.4	Weiterführende Literatur und Übungen	346
	Literatur	349
9	Architekturentwurf und Architekturmodellierung	353
9.1	Grundsätzliches Vorgehen im Architekturentwurf	353
9.1.1	Allgemeiner Prozess des Architekturentwurfs	354
9.1.2	Entwurfsstrategien im Architekturentwurf	356
9.1.3	Modellierung von Daten	357
9.1.4	Modellierung von Nutzungsschnittstellen	359
9.1.5	Berücksichtigung von Datensicherheit und Datenschutz	362
9.1.6	Fehlerbehandlung	363
9.2	Der Grobentwurf: Entwicklung der Systemarchitektur	366
9.2.1	Prinzipien im Architektur- und Komponentenentwurf	367
9.2.2	Black-Box Spezifikation und Schnittstellen	367
9.2.3	Wiederverwendung von Komponenten und Modulen	369
9.3	Der Feinentwurf: Entwicklung der Softwarearchitektur	370
9.3.1	Softwarearchitektur zur Entwurfs- und Laufzeit	371
9.3.2	Abstraktion und Verfeinerung	371
9.4	Dokumentation der Architektur: Architekturspezifikation	372
9.4.1	Das Pflichtenheft und die Systemanforderungen	373
9.4.2	Nachhaltige Dokumentation der Softwarearchitektur	375
9.4.3	Dokumentation von Entwurfsentscheidungen	377

9.5	Qualitätssicherung des Architekturentwurfs	378
9.5.1	Evaluation von Systementwürfen	379
9.5.2	Architekturreviews	381
9.6	Weiterführende Literatur und Übungen	383
	Literatur	386
10	Nutzung bewährten Architekturwissens	389
10.1	Wiederverwendung im Architekturentwurf	389
10.2	Klassenbibliotheken und Idiome und Entwurfsmuster	390
10.2.1	Frameworks	391
10.2.2	Generische Klassenbibliotheken für die Softwareentwicklung	392
10.2.3	Domänenspezifische Frameworks und Klassenbibliotheken	394
10.2.4	Referenzarchitekturen	395
10.3	Entwurfsmuster	396
10.3.1	Allgemeine Beschreibung von Entwurfsmustern	396
10.3.2	Klassifizierung von Entwurfsmustern	397
10.3.3	Anwendung von Entwurfsmustern	398
10.4	Architekturmuster	400
10.4.1	Klassifizierung von Architekturmustern	400
10.4.2	Anwendungsbereiche von Architekturmustern	401
10.5	Entwurfsregel S.O.L.I.D.	405
10.5.1	Single Responsibility Principle	406
10.5.2	Open-Closed Principle	407
10.5.3	Liskov's Substitution Principle	410
10.5.4	Interface Segregation Principle	412
10.5.5	Dependency Inversion Principle	415
10.6	Weiterführende Literatur und Übungen	418
	Literatur	422
 Teil IV Implementierung, Integration und Qualitätssicherung von Software		
11	Implementierung von Softwaresystemen	427
11.1	Implementierung von Software	427
11.1.1	Codearchitektur und Codequalität	428
11.1.2	Entscheidungen vor Beginn der Implementierung	429
11.2	Grundsätzliche Aufgaben in der Implementierung	431
11.2.1	Implementierung des Datenmodells	431
11.2.2	Implementierung der Programmlogik und der Module	432
11.2.3	Realisierung der Nutzerschnittstelle	435

11.3	Codierung	436
11.3.1	Codierungsrichtlinien	436
11.3.2	Dokumentation von Quellcode	438
11.3.3	Clean Code	439
11.3.4	Pair Programming	442
11.4	Modellbasierte Entwicklung und Codegenerierung	443
11.4.1	Grundidee der modellbasierten Entwicklung	443
11.4.2	Beispiel der Modellbasierten Entwicklung	446
11.5	Source Code Management	449
11.5.1	Versionskontrollsysteme	450
11.5.2	Organisation und Umgang mit Versionskontrollsystemen	451
11.5.3	Techniken und Werkzeuge	454
11.6	Weiterführende Literatur und Übungen	455
	Literatur	460
12	Verifikation und Integration von Software	463
12.1	Qualitätssicherung der Implementierung	463
12.1.1	Begriffe und Konzepte	465
12.1.2	Codeinspektionen und Codereviews	468
12.1.3	Funktionale Korrektheit	471
12.1.4	Verifikation durch Korrektheitsbeweis und Model Checking	472
12.2	Grundsätzliches Vorgehen im Testen von Software	472
12.2.1	Prinzipien	473
12.2.2	Allgemeiner Testprozess	474
12.2.3	Methodik des Testens	476
12.2.4	Entwurfsregel F.I.R.S.T.	483
12.2.5	Unit Testing	484
12.2.6	Test-Driven Development	488
12.3	Strategien für Software- und Systemintegration	492
12.3.1	Grundsätzliche Test- und Integrationsstufen	494
12.3.2	Vorgehen bei der Integration	505
12.4	Kontinuierliche Integration und Auslieferung	511
12.4.1	Vorgehen in der Kontinuierlichen Softwareentwicklung	512
12.4.2	DevOps	516
12.5	Weiterführende Literatur und Übungen	524
	Literatur	531
13	Softwareevolution	535
13.1	Transition in die Einsatzumgebung	535
13.1.1	Auslieferung und Abnahme	536

13.1.2	Installation und Inbetriebnahme.	538
13.1.3	Übergang in die Wartung	540
13.2	Wartung, Pflege und Weiterentwicklung von Software	540
13.2.1	Was ist Softwarewartung?	541
13.2.2	Aufgaben in der Wartung von Software.	546
13.2.3	Methoden, Techniken und Werkzeuge in der Wartung von Software	551
13.3	Weiterführende Literatur und Übungen	569
	Literatur.	571
Epilog	575
Anhang A UML Kurzreferenz	587
Anhang B Weiterführende Beispiele für die Implementierung	605
Anhang C Projektunterlagen Code & Talk	635
Glossar	653
Stichwortverzeichnis	661