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

Springer Vieweg

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.



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 versetzen. 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 Kuhrmann 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.

Juli 2020 Hasso Plattner

# 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**

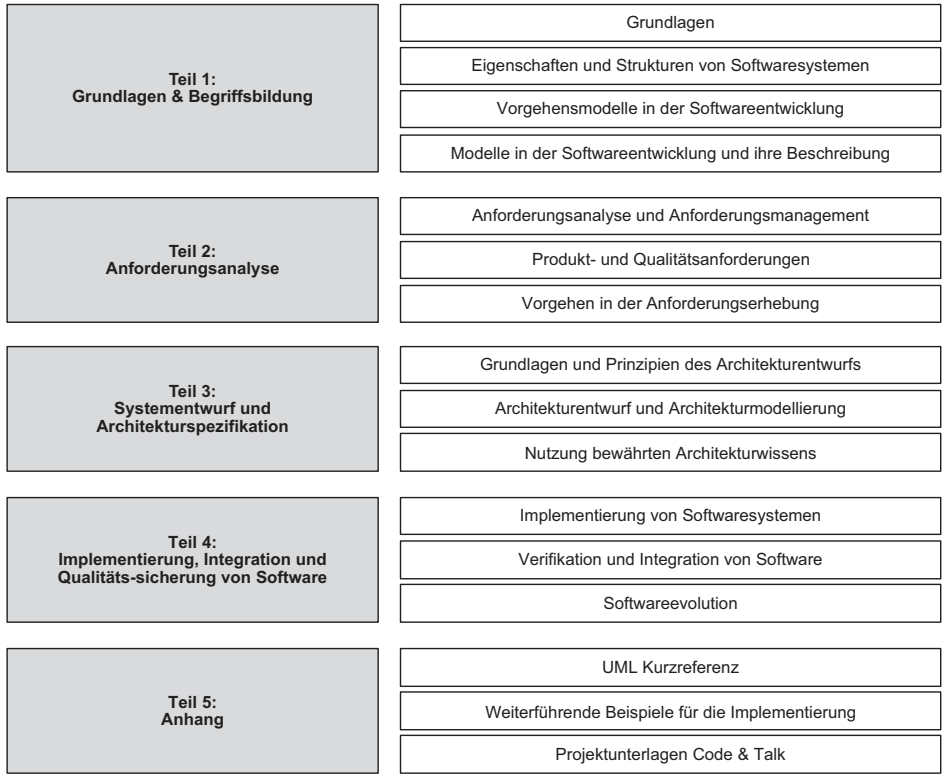
Dieses Lehrbuch richtet sich an Studierende in Bachelor-, Master- und Diplom-Studiengängen der Informatik, Wirtschaftsinformatik und verwandter Disziplinen aber auch an Praktiker, insbesondere Berufs- und Quereinsteiger. Es liefert einen Einstieg in wissenschaftlich fundierte und praktisch relevante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




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

[1]ACM/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 — Manfred Broy
2020 — Marco Kuhrmann

# Inhaltsverzeichnis