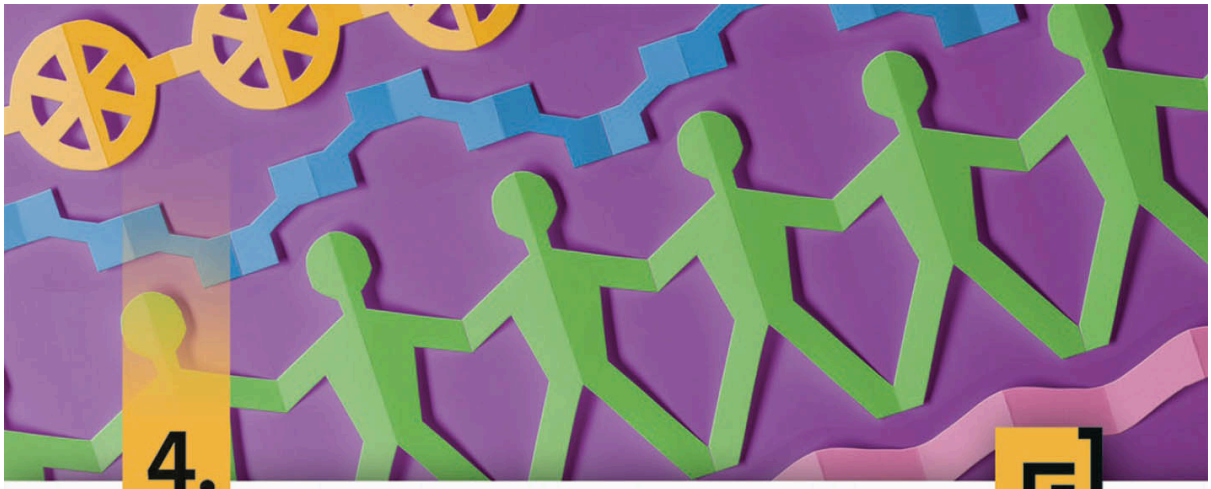


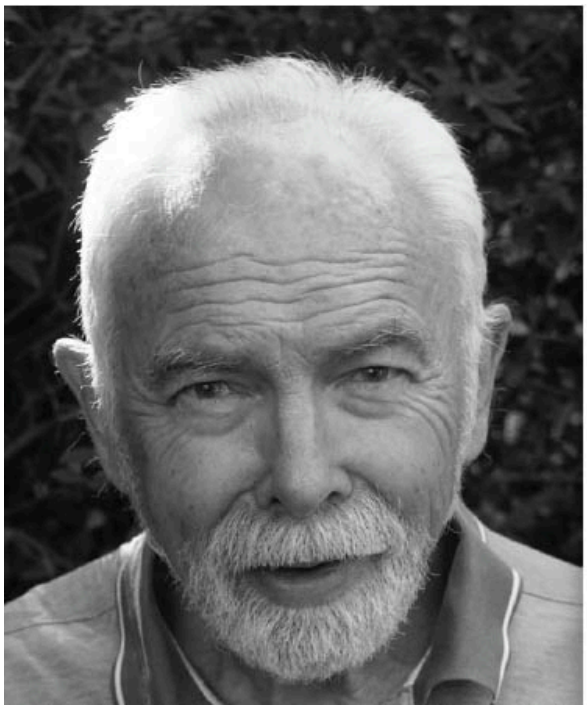
4. Auflage

Jochen Ludewig · Horst Lichter

Software Engineering

Grundlagen, Menschen, Prozesse, Techniken

dpunkt.verlag



Jochen Ludewig und Horst Lichter, Oktober 2022

Prof. Dr. rer. nat. Jochen Ludewig geboren 1947 in Hannover. Studium der Elektrotechnik (TU Hannover) und Informatik (TU München); Promotion 1981. 1975 bis 1980 Gesellschaft für Kernforschung, Karlsruhe, dann Brown Boveri Forschungszentrum in Baden/Schweiz. 1986 Assistenzprofessor an der ETH Zürich, 1988 Ruf auf den neuen Lehrstuhl Software Engineering an der Universität Stuttgart. Arbeitsgebiete: Softwareprojekt-Management, Software-Prüfung und Software-Qualität, Software-Wartung. Ab 1996 Konzeption und Aufbau des Diplomstudiengangs Softwaretechnik, der inzwischen in einen Bachelor- und einen Masterstudiengang umgewandelt wurde. Seit 2009 Fellow der Gesellschaft für Informatik.

Verheiratet, zwei erwachsene Töchter, zwei Enkel.

Prof. Dr. rer. nat. Horst Lichter geboren 1960 in Trier. Studium der Informatik und Betriebswirtschaftslehre (TU Kaiserslautern). Wissenschaftlicher Mitarbeiter (ETH Zürich und Universität Stuttgart), Promotion 1993. Anschließend Schweizerische Bankgesellschaft Zürich und ABB Forschungszentrum Heidelberg. 1998 Ruf an die RWTH Aachen University, Leiter des Lehr- und Forschungsgebiets Software-Konstruktion. Arbeitsgebiete: Software-Architektur, Qualitätssicherung, Software-Evolution. Seit 2005 Lecturer an der Thai German Graduate School of Engineering (TGGS) in Bangkok. Von 2018-2021 Adjunct Lecturer an der Universiti Teknologi Petronas (UTP) Malaysia.

Verheiratet, drei Söhne.

Jochen Ludewig · Horst Lichter

Software Engineering

Grundlagen, Menschen, Prozesse, Techniken

4., überarbeitete und erweiterte Auflage

dpunkt.verlag

Jochen Ludewig

mail@jludewig.de

Horst Lichter

horst.lichter@rwth-aachen.de

Lektorat: Christa Preisendanz

Lektoratsassistenz: Julia Griebel

Copy-Editing: Ursula Zimpfer, Herrenberg

Satz: Horst Lichter, Aachen

Herstellung: Stefanie Weidner, Frank Heidt

Umschlagmotiv: Jochen Ludewig, Stuttgart

Umschlaggestaltung: Helmut Kraus, *www.exclam.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.

ISBN:

Print 978-3-86490-598-8

PDF 978-3-96088-546-7

ePub 978-3-96088-547-4

mobi 978-3-96088-548-1

4., überarbeitete und erweiterte Auflage 2023

Wieblinger Weg 17

69123 Heidelberg

Schreiben Sie uns:

Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: *hallo@dpunkt.de*.

5 4 3 2 1 0

Unseren Frauen und Kindern gewidmet:
Jutta · Gabi · Linda · Nora · Bastian · Jannis · Moritz

Vorwort

Entstehungsgeschichte

Dieses Buch hat seine allerersten Anfänge in einer Vorlesungsreihe »Software Engineering« am Neu-Technikum in Buchs, St. Gallen, Schweiz (Ludewig, 1985). Das Material wurde durch viele weitere Vorlesungen beider Autoren an der ETH Zürich, der Universität Stuttgart und der RWTH Aachen University immer wieder bearbeitet, ergänzt und erweitert.

2006 endlich ging unser Buch in Druck. 2010 erschien die zweite, 2013 die dritte Auflage. Nun, 2022, war es an der Zeit, die vierte Auflage fertigzustellen. Nachdem in den Auflagen 2 und 3 Korrekturen und kleine Ergänzungen im Vordergrund standen, gibt es nun einige substanzielle Erweiterungen, vor allem bei den Software-Prozessen und im Software-Entwurf.

Die lange Geschichte des Buches steht scheinbar im Widerspruch zum raschen Verderb des Informatik-Wissens. Wir sehen das weniger ängstlich und halten das Gerede von der »kurzen Halbwertzeit des Wissens« für modisches Geschwätz. Da wir uns nicht mit dem Leistungsstand von Mikroprozessoren befassen, sondern die Leserinnen und Leser, soweit es in unseren Kräften steht, mit dem Grundwissen für ein hoffentlich langes und befriedigendes Berufsleben ausrüsten, sollte das hier präsentierte Material viele Jahre brauchbar bleiben. Dazu trägt ohne Zweifel bei, dass sich Software Engineering vor allem mit den Menschen befasst, die Software in Auftrag geben, entwickeln und ändern oder benutzen. Was die Philosophen vor mehr als zweitausend Jahren über Menschen gesagt haben, gilt zum größten Teil noch heute, und so wird es wohl noch ein paar weitere Jahre gültig bleiben.

Rollenbezeichnungen

Auch in diesem Buch bleibt das Problem ungelöst, eine befriedigende Form der Rollenbezeichnungen zu finden, die nicht suggeriert, dass die Person in dieser Rolle ein männliches, ein weibliches oder ein diverses Wesen ist. Wir haben uns

an anderer Stelle mit diesem Problem näher befasst (Deininger et al., 2017), freilich ohne eine gute, allgemein akzeptierte Lösung zu finden.

Darum gehen wir den üblichen Weg, alle Rollenbezeichnungen in ihrer männlichen Grundform, dem generischen Maskulinum, zu verwenden. Es dürfte überflüssig sein, darauf hinzuweisen, dass es keine einzige Rolle auf dem Gebiet des Software Engineerings gibt, die vorzugsweise oder ausschließlich mit Personen eines bestimmten Geschlechts besetzt sein sollte. Nach unserer Erfahrung sind gemischte Teams immer den ungemischten vorzuziehen.

Sprachstil

Vor drei, vier Jahrzehnten wäre eine Vorbemerkung zum Sprachstil nur töricht gewesen. Bis auf wenige bekannte Riffe im »Wörtersee« war allgemein klar und anerkannt, was als Deutsch gilt.

Das ist heute nicht mehr so. Ursache ist anscheinend nicht die Rechtschreibreform, sondern ein dramatischer Verfall der sprachlichen Disziplin. Viele Texte, die von Studierenden abgeliefert werden, können kaum mehr als Deutsch bezeichnet werden, und viele derer, die eigentlich Vorbild sein sollten, also Journalisten, Politiker und natürlich auch Hochschullehrer, verstärken das Problem, statt es zu bekämpfen.

Wir bekennen uns ausdrücklich zum Ziel, unsere Gedanken in einer akzeptablen Sprache zu formulieren. Wir sind dabei vor Fehlern nicht sicher, und wir werden keine literarische Qualität erreichen, aber wir bemühen uns. Den Leserinnen und Lesern wird darum auch an einigen Stellen ein Sprachgebrauch auffallen, der unmodern erscheint, aber – wenigstens in unseren Ohren oder Augen – richtig ist.

Anglizismen lassen sich gerade in Informatik-Büchern nicht vermeiden; das beginnt schon mit unserem Fachgebiet und mit dem Titel dieses Buches. Wo es eine gute deutsche Übersetzung gibt, ziehen wir sie vor. Gerade beim »Software Engineering« ist das zweifelhaft, denn »Softwaretechnik« ist ein sprachlicher Zwitter. Leider wurde versäumt, das Wort »Software« in unsere Sprache zu übertragen, wie es den Franzosen mit »Logiciel« glänzend gelungen ist. Dass solche Wortschöpfungen auch im Deutschen möglich sind, zeigt sich an künstlichen, aber heute geläufigen Wörtern wie »Rechner« oder »Datei«.

Dank

»Wir sind Zwerge, die auf den Schultern von Riesen sitzen.« So hat es laut Wikipedia Bernhard von Chartres um 1120 formuliert. Das gilt natürlich vor allem für die Verfasser wissenschaftlicher Arbeiten. Wo wir bewusst auf Material

Bezug nehmen, das von anderen Autoren stammt, haben wir (hoffentlich vollständig und korrekt) zitiert. Das ist in besonders hohem Maße bei zwei Büchern der Fall, an denen einer von uns beteiligt ist (Frühauf, Ludewig, Sandmayr, 2002 und 2007). Vor allem unsere Kapitel 13 (*Software-Qualitätssicherung und -Prüfung*), 18 (*Programmtest*), 20 (*Konfigurationsverwaltung*) und 21 (*Software-Wartung*) enthalten Anleihen aus diesen Büchern. Herzlichen Dank an die beiden Kollegen in Baden/Schweiz, die mit ihrer INFOGEM AG gutes Software Engineering erfolgreich vermitteln und praktizieren!

Einige Koryphäen des Software Engineerings haben uns fast ein halbes Jahrhundert lang geprägt: Das waren vor allem David L. Parnas (Jahrgang 1941), Barry W. Boehm (1935–2022), Fred Brooks (1931–2022) und Michael Jackson (1936), auch heute kaum noch bekannte Pioniere wie Daniel Teichroew (1925–2003), der das erste Spezifikationssystem geschaffen hat (PSL/PSA). In Deutschland ist vor allem Friedrich L. Bauer (1924–2015) zu nennen, der den Begriff »Software Engineering« 1968 wenn nicht erfunden so doch populär gemacht hat. Ohne diese (und viele andere) Leute wäre das Fachgebiet deutlich ärmer und ein Lehrbuch darüber ein »Leerbuch«.

Wenn es Unklarheiten über Begriffe oder Quellen gibt, schaut man ins Internet – und findet dort sehr viel Müll. Umso erfreulicher sind die Ausnahmen; Martin Glinz (bis zur Emeritierung 2017 an der Universität Zürich) sei stellvertretend für alle genannt, die Qualitätssicherung nicht nur lehren, sondern auch leben und (z.B. durch ihre Webseiten) demonstrieren.

Kornelia Kuhle hat (bis zur dritten Auflage) mit scharfem Blick auch kleinste Unregelmäßigkeiten des Textes erkannt und markiert. Einzelne Kapitel wurden von unseren Mitarbeiterinnen und Mitarbeitern kommentiert. Ihnen allen herzlichen Dank! Und allen, die uns zugehört und auch widersprochen haben, sind wir zu großem Dank verpflichtet.

Die Zusammenarbeit mit dem dpunkt.verlag, stets bestens vertreten durch Christa Preisendanz, war über die vielen Jahre reibungslos und immer erfreulich. Ursula Zimpfer hat wieder einmal mit bewunderungswürdiger Präzision die Endkontrolle des Buches vorgenommen; was jetzt noch falsch ist, geht auf unsere Kappe.

Unseren Familien danken wir dafür, dass sie uns immer den nötigen Freiraum und die Zeit gewährt haben, um an diesem Buch zu arbeiten.

Natürlich freuen wir uns auch in Zukunft über Ihre Kritik, positiv oder negativ, und Ihre Hinweise auf Lücken und Mängel. Dafür schon heute vielen Dank!

Jochen Ludewig, Horst Lichter
Stuttgart und Aachen, Dezember 2022

Inhalt und Aufbau, Zielgruppen

Inhalt

Vier Kriterien entscheiden darüber, welche Themen in einem Lehrbuch behandelt werden:

- Welches Wissen bringen die Autoren mit?
- Welche Themen sind für die Leser wichtig und attraktiv?
- Welche Aussagen und Erkenntnisse sind mutmaßlich für einige Jahre stabil, nicht den kurzfristigen Moden unterworfen?
- Was kann und sollte in einer Vorlesung über Software Engineering behandelt werden?

Jedes der Kriterien definiert, mehr oder minder scharf, eine Menge; die Schnittmenge liefert den Themenkatalog, der dem Buch zugrunde liegt.

Dabei müssen auch – wie immer im Software Engineering – Kompromisse gefunden, also Einschränkungen beim einen oder anderen Kriterium in Kauf genommen werden. Für die Leserinnen und Leser sollte aber in allen Teilen deutlich werden, welches Verständnis wir von unserem Fach haben: Software Engineering ist eine auf der Informatik beruhende Ingenieurdisziplin, die wie alle Ingenieurfächer darauf abzielt, die Praxis zu verbessern.

Software Engineering stellt sich für uns wie ein weitgehend unerforschter Kontinent dar. Wir sind weit davon entfernt, eine vollständige und präzise Beschreibung anbieten zu können. Wenigstens aber die Konturen, die bisherige Entwicklung (in groben Zügen) und die als gesichert geltenden Einsichten soll dieses Buch abdecken. Wer es in der Lehre einsetzt, wird eigene Erfahrungen und spezielle Literatur hinzufügen.

Aufbau

Für den Aufbau eines Lehrbuches gibt es einige sinnvolle Prinzipien:

- vom Allgemeinen zum Speziellen
- vom Leichten zum Schwierigen
- von den Grundlagen zu den Anwendungen

Im Software Engineering kommt noch hinzu:

- dem Gang des Software-Projekts folgend

Leider lässt sich daraus keine konkrete Reihenfolge ableiten, denn diese Prinzipien haben unterschiedliche Konsequenzen. Zudem sind viele Themen zyklisch verbunden. Ein typisches Beispiel ist die Software-Wartung: Im Projektablauf steht sie ganz am Ende. Da aber die Schwierigkeiten der Wartung sehr stark von den Entscheidungen bei der Konstruktion der Software beeinflusst sind, sollten Überlegungen zur Wartung nicht erst angestellt werden, wenn die Entwicklung abgeschlossen ist, sondern bereits in der Projektplanung. Die Wartung wirkt sich also auf den Entwurf aus, weil sich der Entwurf auf die Wartung auswirkt.

Wir können dieses Dilemma nicht auflösen. Wir haben darum einen anderen Aufbau gewählt: In sechs Teilen wird das Thema aus verschiedenen Perspektiven betrachtet.

- Teil I: *Grundlagen*
  Die Inhalte dieses Teils sind fundamental und damit nicht von einer speziellen Technologie geprägt. Hier geht es um das Basiswissen des Software-Ingenieurs. Am Anfang steht ein Kapitel über Modelle, weil dieses Thema für das Software Engineering wirklich grundlegend ist und damit das Fundament aller weiteren Kapitel darstellt.
- Teil II: *Menschen und Prozesse*
  Software ist enger als andere technische Artefakte mit dem Denken der Menschen verknüpft, die die Software herstellen oder benutzen. Die organisatorischen Rahmenbedingungen haben großen Einfluss auf den Erfolg der Projekte. Diese Punkte werden im Teil II behandelt.
- Teil III: *Daueraufgaben im Software-Projekt*
  Viele Tätigkeiten wie Dokumentation oder Prüfung können nicht einzelnen Schritten der Entwicklung oder speziellen Dokumenten zugeordnet werden, sie finden laufend statt. Teil III behandelt diese Daueraufgaben.
- Teil IV: *Techniken der Software-Bearbeitung*
  Die einzelnen Schritte der Entwicklung, von der Analyse bis zur Integration, werden im Teil IV erläutert.

- Teil V: *Verwaltung und Erhaltung von Software*
  Die Wartung ist eng verknüpft mit der Konfigurationsverwaltung, dem Reengineering und der Wiederverwendung. Darum werden alle diese Themen im Teil V des Buches behandelt. Untereinander ist die Abgrenzung unklar; beispielsweise wird die Änderungsverwaltung im Kapitel über die Wartung besprochen, sie könnte ebenso gut im Kontext der Konfigurationsverwaltung behandelt werden.

  Die Themen dieses Teils könnten auch dem Teil III, den Daueraufgaben, zugeordnet werden. Hier handelt es sich aber um Aufgaben und Arbeiten, die ganz oder vorwiegend anfallen, nachdem die Software an den Kunden ausgeliefert ist.

- Teil VI: *Software Engineering lehren*
  Am Ende des Buches befassen wir uns mit der Frage, wie sein Inhalt in Studiengängen und in anderen Formen vermittelt werden kann. Dazu wird ein konkreter Studiengang kurz vorgestellt, und es werden einige Zahlen zur Verbreitung solcher Studiengänge angegeben. Auf Kurse und Seminare zu Teilgebieten des Software Engineerings wird kurz hingewiesen.
- Teil VII: *Literatur und Index*
  Die im Buch zitierte Literatur haben wir nach Verfassern, die zitierten Normen nach Normenreihen geordnet; wir haben uns sehr nachdrücklich bemüht, alle Quellen richtig und vollständig anzugeben. Das Stichwortverzeichnis steht ganz am Schluss des Buches.

Wer dieses Buch im Unterricht einsetzt, sollte das Inhaltsverzeichnis nicht als Vorgabe für die Gliederung seiner Lehrveranstaltung betrachten. Wir hoffen, unsere Kolleginnen und Kollegen durch eine klare und nachvollziehbare Struktur bei der individuellen Auswahl der Themen und bei der Gestaltung ihrer Lehre zu unterstützen. Und natürlich gibt es Bedarf und Raum für Ergänzungen durch andere Themen, die in diesem Buch fehlen oder nur kurz besprochen werden. Die formale Spezifikation und eine ganze Reihe moderner Entwicklungstechnologien sind naheliegende Beispiele solcher Gebiete.

Zielgruppen

Da wir regelmäßig Lehrveranstaltungen über Software Engineering an zwei deutschen Universitäten durchführen bzw. durchgeführt haben, sind Studierende die Adressaten dieses Buches. Unsere Erfahrungen in anderen Ländern und in anderen Lehranstalten legen die Vermutung nahe, dass das Buch für alle deutschsprachigen Länder und auch für andere Hochschulen geeignet

ist. Dazu zählen wir auch Schulungseinrichtungen in der Industrie, wo wir selbst immer gern unterrichtet haben und unterrichten.

Aus dieser Erfahrung und den Erfahrungen aus vielen Kooperationen mit Industrieunternehmen wissen wir, dass es in der Industrie sehr viele Menschen gibt, die an Software arbeiten, ohne eine einschlägige Ausbildung zu haben. Die meisten von ihnen sind gelernte Ingenieure, Naturwissenschaftler, Mathematiker oder Kaufleute; sie bringen durch ihren ursprünglichen Beruf wichtige Voraussetzungen für das Software Engineering mit und haben Studierenden sehr viel praktische Erfahrung voraus. Dieses Buch gibt ihnen die Möglichkeit, einige Grundlagen und spezielle Themen im Selbststudium zu erarbeiten. Wir haben uns besonders für diese Gruppe darum bemüht, die Ideen und Gedanken nicht nur aufzulisten, sondern durch einen hoffentlich gut strukturierten und formulierten Text auch verständlich und angenehm lesbar zu machen, also quasi eine Vorlesung in Buchform anzubieten.

Wir haben zu diesem Buch eine Webseite (*se-buch.de*) erstellt. Dort finden Sie unter anderem Materialien, beispielsweise alle im Buch enthaltenen Abbildungen, und auch Korrekturen zu gemeldeten Fehlern.

Inhaltsverzeichnis

Teil I

Grundlagen

1 Modelle und Modellierung

Modelle sind ein fundamentales Konzept unseres Umgangs mit der Welt. Alle Naturwissenschaftler und Ingenieure verwenden und schaffen Modelle, um allgemeingültige Aussagen zu treffen und um ihre Vermutungen zu konkretisieren. Oft markieren die Modelle Zwischenschritte auf dem Weg zu neuen Artefakten, also zu Brücken, Autos oder Funktelefonen. Im Software Engineering ist die Bedeutung der Modelle noch größer, weil sie nicht Zwischenschritte, sondern Endpunkte unserer Arbeit darstellen: Eine Spezifikation, aber auch ein Programm ist ein Modell. Natürlich gehören auch die Prozessmodelle dazu, nach denen die Projekte organisiert werden. Wir legen also mit diesem Kapitel, das auf Ludewig (2003) basiert, den Grundstein für unser Buch. Das gilt auch für die beiden letzten Abschnitte, die sich mit Skalen und Skalentypen befassen.

1.1 Modelle, die uns umgeben

1.1.1 Die Bedeutung der Modelle

Lebenswichtig für uns sind die Modelle, die wir als *Begriffe* kennen und verwenden, um uns ein Bild (d. h. ein Modell) der Realität zu machen. Ohne die Begriffe wäre für uns jeder Gegenstand ganz neu; weil wir aber zur Abstraktion fähig sind, können wir die Identität eines Gegenstands, seinen Ort, seinen Zustand und u. U. viele andere individuelle Merkmale ausblenden, um ihn einer Klasse von Gegenständen, eben dem *Begriff*, zuzuordnen. Auf diese Weise erkennen wir auch einen Gegenstand, den wir noch nie gesehen haben, als Bleistift, Stuhl, Auto oder was immer in unserer Vorstellung am besten passt.

Diese Fähigkeit ist uns bereits von Natur aus gegeben; sie ist weder bewusst steuerbar, noch lässt sie sich unterdrücken. Darum sind wir auch nicht davor

geschützt, falsche (d. h. ungeeignete) Modelle zu wählen. Wir erleben das erheitert bei optischen Täuschungen, wir erleiden es, wenn wir direkt oder indirekt Opfer von Vorurteilen werden: Auch das sind Modelle.

Dagegen steht es uns frei, Modelle bewusst einzusetzen, um auf diese Weise Phänomene zu erklären oder Entscheidungen zu überprüfen, bevor sie wirksam werden. Beispielsweise können wir ein geplantes Bauwerk durch eine Zeichnung oder ein Papiermodell darstellen und dann den Entwurf anhand des Modells überprüfen.

Wo Modelle offensichtliche Schwächen zeigen, neigen wir dazu, sie zu belächeln. Das gilt etwa für die Puppe, die ein spielendes Kind in einem länglichen Stück Holz sieht. Wo Modelle dagegen sehr überzeugend wirken, besteht die Gefahr, dass sie mit der Realität verwechselt werden. Darum ist zunächst festzuhalten: Ein Modell ist ein Modell, es ist nicht die Realität. Die Schwierigkeit, die wir haben, wenn wir von Modellen auf die Realität zu schließen versuchen, hat bereits der griechische Philosoph Platon (428/427–348/347 v. Chr.) in seinem berühmten *Höhlengleichnis* angesprochen. Darin beschreibt er die Situation eines Menschen, der, seit seiner Kindheit in einer Höhle mit dem Gesicht zur Wand gefesselt, nur die Schatten der Menschen sieht, die an der Höhle vorbeilaufen. Der Gefangene muss die Schatten als Realität nehmen, da ihm die *wirkliche* Realität nicht zugänglich ist. Die Botschaft dieses Gleichnisses ist, dass wir alle in dieser Situation sind, also nicht die Realität sehen können, sondern nur die Schatten.

1.1.2 Beispiele für Modelle

Alle Begriffe sind Modelle; diese sehr allgemeine Betrachtungsweise spielt hier weiter keine Rolle mehr, wir konzentrieren uns auf »richtige« Modelle. Auch diese sind uns so geläufig, dass wir sie in der Regel nicht mehr als Modelle wahrnehmen: Jedes Bild, jedes Schema ist ein Modell. Beispielsweise sind Modelle eines bestimmten oder unbestimmten Menschen

- ein Personenfoto,
- die anatomische Darstellung der Blutbahnen,
- Strichmännchen und Piktogramme, die einen Menschen zeigen,
- ein Fingerabdruck,
- ein Gemälde, das einen Menschen zeigt,
- ein Steckbrief (mit oder ohne Bild),
- eine Matrikelnummer.

Unmittelbar leuchtet das für solche Modelle ein, die (wie das Foto oder die anatomische Darstellung) eine optische oder strukturelle Ähnlichkeit mit dem

Original aufweisen. Aber auch die übrigen Beispiele sind, wie unten gezeigt wird, korrekt.

Im Software Engineering treffen wir Modelle auf verschiedenen Ebenen an:

- Software wird auf unterschiedliche Arten repräsentiert, typischerweise durch eine Spezifikation, einen Entwurf, durch Diagramme und Quellcode, Kennzahlen (Metriken) und Prospekte. Offenkundig haben wir hier Modelle vor uns; dabei bleibt zunächst noch unklar, welcher Gegenstand denn eigentlich das Original darstellt. Wenn man bedenkt, dass man (nach dem Lehrbuch) den Code auf der Grundlage einer Spezifikation entwickelt, dass andererseits in der Praxis sehr oft versucht wird, den Sinn eines Programms (d. h. seine Spezifikation) aus dem Code abzuleiten, dann sieht man, dass die Frage nicht eindeutig zu beantworten ist.
- Die Abläufe bei der Arbeit an Software werden durch Prozessmodelle beschrieben. Gutes Software Engineering ist weitgehend gleichbedeutend mit der Wahl eines geeigneten Prozessmodells und seiner Umsetzung in die Praxis.

Allgemeiner gesagt ist jede Theorie ein Modell. Im Software Engineering steckt die Theoriebildung noch in den Kinderschuhen, wir müssen uns gerade darum mit ihr befassen (siehe Abschnitt 1.6).

1.2 Modelltheorie

Herbert Stachowiak (1921–2004) hat das Standardwerk zur Modelltheorie verfasst. Die folgenden Aussagen zur Modelltheorie geben Gedanken aus diesem Buch wieder (Stachowiak, 1973).

1.2.1 Deskriptive und präskriptive Modelle

Modelle, wie wir sie aus dem Alltag kennen, sind entweder Abbilder von etwas oder Vorbilder für etwas; sie werden auch als *deskriptive* (d. h. beschreibende) bzw. *präskriptive* (d. h. vorschreibende) Modelle bezeichnet. Eine Modelleisenbahn ist ein Abbild; eine technische Zeichnung, nach der ein Mechaniker arbeitet, ist ein Vorbild. Wenn ein Gegenstand fotografiert wird und dann Änderungen im Foto skizziert werden, geht das eine in das andere über. Den Modellen kann man im Allgemeinen nicht ansehen, ob sie deskriptiv oder präskriptiv sind; eine Modelleisenbahn kann auch der Entwurf für eine erst zu bauende reale Bahn sein, eine Zeichnung kann nach einem realen Gegenstand entstanden sein.

1.2.2 Die Modellmerkmale

Jedem Modell (wie in Abb. 1–1 schematisch dargestellt) sind drei Merkmale zu eigen (sonst ist es kein Modell):

- Das *Abbildungsmerkmal*
  Zum Modell gibt es das Original, ein Gegenstück, das wirklich vorhanden, geplant oder fiktiv sein kann. (Beispiel: Zu einem Foto gibt es das fotografierte Motiv, zum Rauschgoldengel gibt es den – wenn auch fiktiven – Engel der Fantasie.)
- Das *Verkürzungsmerkmal*
  Ein Modell erfasst nicht alle Attribute des Originals, sondern nur einen Ausschnitt (der vor allem durch den Zweck des Modells bestimmt ist, siehe pragmatisches Merkmal).

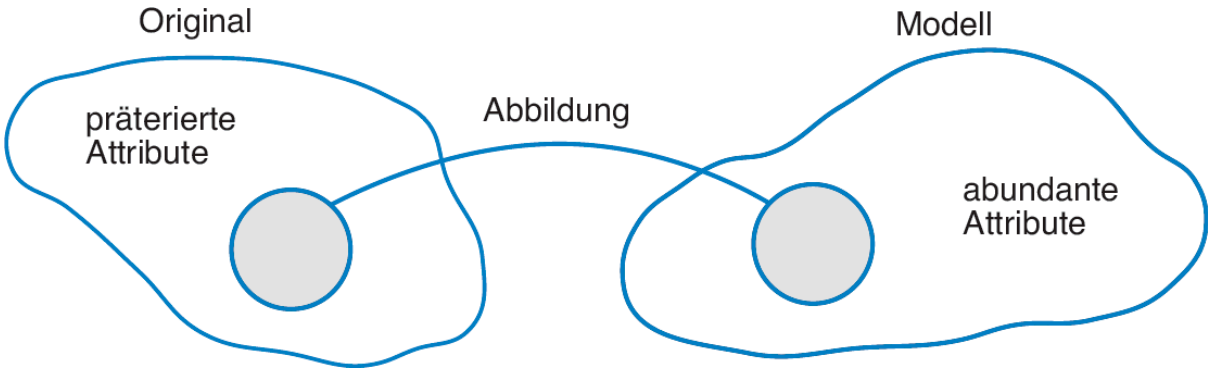


Abb. 1–1 *Original und Modell nach Stachowiak*

  Damit fallen die *präterierten Attribute* weg (von lat. praeter: außer, ausgenommen). Das sind alle Attribute des Originals, die im Modell nicht repräsentiert sind. Das Modell weist stattdessen *abundante Attribute* auf (von lat. abundans: übervoll, überreich), die nichts mit dem Original zu tun haben. So sind auf dem Foto einer Person die meisten ihrer Attribute präteriert (ihr Gewicht, ihr Name, ihre Blutgruppe usw.). Andererseits sind Attribute des Modells abundant, sie haben nichts mit der Person zu tun (die Qualität des Fotopapiers, das Format). Der Fingerabdruck gibt nur einen winzigen Ausschnitt der Merkmale eines Menschen wieder, alle anderen Merkmale sind präteriert; die Farbe des Abdrucks ist dagegen ein abundantes Attribut.

- Das *pragmatische Merkmal*
  Modelle können unter bestimmten Bedingungen und bezüglich bestimmter Fragestellungen das Original ersetzen. (Beispiel: Ein Foto erlaubt die Beurteilung eines Unfalls, der sonst nur durch Anwesenheit am Unfallort zu beurteilen gewesen wäre. Der Fingerabdruck gestattet es

U

V

W

X

Z

Ziffern