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).

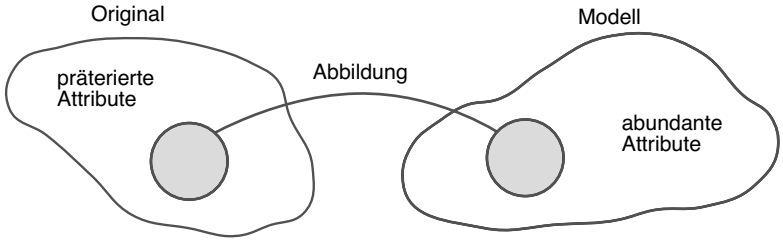


Abb. 1–1 *Original und Modell nach Stachowiak*

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 eventuell, die Identität einer Person festzustellen, auch wenn die Person selbst nicht zur Verfügung steht.)

1.3 Ziele beim Einsatz von Modellen

Modelle werden mit unterschiedlichen Zielen (Zwecken) eingesetzt. Diese Ziele sind nicht präzise unterscheidbar, sie gehen ineinander über.

1.3.1 Modelle für den Einsatz in der Lehre und zum Spielen

Modelle werden vielfach in der Ausbildung, in der Werbung, für Spiele usw. eingesetzt, wo die Originale aus ethischen oder praktischen Gründen nicht zur Verfügung stehen. Dies ist wohl die bekannteste Art von Modellen. Charakteristisch für ein solches Modell ist die Nachahmung eines existierenden oder fiktiven Originals, d. h., das Modell ist deskriptiv und dem Original ähnlich. Beispiele sind

- Modelleisenbahnen,
- Modelle der menschlichen Anatomie,
- die meisten Computerspiele.

1.3.2 Formale (mathematische) Modelle

Formale Modelle sind ebenfalls deskriptiv, den Originalen aber äußerlich gar nicht ähnlich; ihr wesentliches Kennzeichen ist, dass sie die Möglichkeit eröffnen, reale Situationen und Vorgänge formal darzustellen und damit Hypothesen über eine vergangene, zukünftige oder denkbare Realität zu begründen. In anderen, komplexeren Modellen sind solche mathematischen Modelle meist enthalten. Typische Beispiele sind

- die Formeln der Physik und der Chemie,
- die Formeln der Statistik (soweit sie empirisch begründet sind),
- das Modell der Regelschleife.

1.3.3 Modelle für die Dokumentation

Eine weitere in der Praxis ständig angewandte Form von Modellen ist die Dokumentation. Wir fertigen Modelle an, damit wir uns besser und genauer an bestimmte Situationen, Abläufe, Personen und Gegenstände erinnern und die Erinnerungen austauschen und überliefern können. Beispiele dafür sind

- Fotos und Fotoalben, Aufzeichnungen einer Überwachungskamera,
- Geschichtsbücher, Gerichtsprotokolle,
- Buchungsbelege, Tagebücher,
- Logbücher, Fehlerreports.

1.3.4 Explorative Modelle

Explorative Modelle (Abb. 1–2) werden eingesetzt, wenn die Folgen einer vorgeschlagenen, noch nicht beschlossenen Änderung der Realität beurteilt werden sollen.

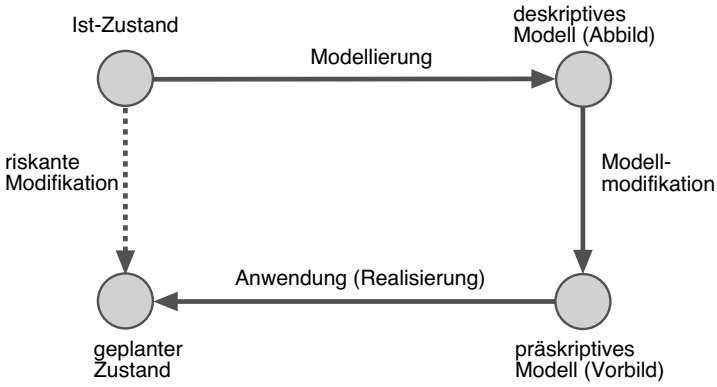


Abb. 1–2 *Anwendung des Modells als Experiment für eine geplante Änderung*

Beispielsweise ist die Erstellung eines Gebäudes heute zwar technisch in der Regel nicht besonders schwierig, doch besteht das Risiko, dass es dem Kunden dann nicht gefällt oder dass es die Landschaft verschandelt. Darum wird ein Modell angefertigt, das es gestattet, diese Risiken zu vermindern. Wir gehen also nicht direkt vom Ist-Zustand in den geplanten Zustand über, sondern zunächst in das deskriptive Modell (im Fall des Gebäudes ein Modell der Umgebung, also der Landschaft) und von dort zum präskriptiven Modell (Landschaft mit dem geplanten Gebäude). Erscheint dieses – u. U. nach mehreren Versuchen – akzeptabel, so wird der Schritt zurück in die Realität vollzogen (das Haus wird gebaut).

Stachowiak (1973, S. 139 f.) beschreibt die wesentliche Funktion solcher Modelle wie folgt (verkürztes Zitat):

> *... Dabei ist bei allen Modellierungen, die zum Informationsgewinn über das Original führen sollen, die folgende Vorgehensweise zu beobachten. Das Original wird in sein Modell abgebildet [a], wobei zumeist zahlreiche Originalattribute fortgelassen [b] und oft Modellattribute neu eingeführt werden [c]. (...) Mittels zielgerichteter Modelloperationen wird dann das ursprüngliche Modell in ein verändertes übergeführt [d]. Ist die attributenmäßige Original-Modell-Zuordnung umkehrbar eindeutig [e], so sind den modellseitigen Operationen bestimmte originalseitige zugeordnet, und die faktisch-operative Überführung des ursprünglichen in das veränderte Modell zieht eine wohlbestimmte hypothetisch-operative Überführung des ursprünglichen Originals in ein verändertes [f] nach sich. (...)*

Zwei Beispiele (aus der Medizin und aus dem Finanzwesen) sollen den Ablauf anschaulich machen. Dabei wird auf die in den Text von Stachowiak eingefügten Buchstaben Bezug genommen.

[a]	Das gebrochene Bein (Original) wird geröntgt.	Die Steuereinnahmen der kommenden Jahre (Original) werden geschätzt.
[b]	Die Beschaffenheit der Haut ist im Röntgenbild nicht sichtbar.	Die individuellen Quellen der Steuer, die exakten Zahlungstermine usw. spielen keine Rolle.
[c]	Die physikalisch-chemische Beschaffenheit des Röntgenfilms ist nicht vom Patienten beeinflusst.	Die konkrete mathematische Darstellung des Modells ist nicht durch die Realität bestimmt.
[d]	Das Bild der Bruchstelle wird im Hinblick auf eine Therapie untersucht.	Die Folgen einer Änderung der Steuergesetze werden im Modell ermittelt.
[e]	Wenn das Bild den Bruch richtig zeigt und nichts Wesentliches ausblendet (beispielsweise eine andere Erkrankung des Patienten),	Wenn die Zusammenhänge im Modell, z. B. über die Auswirkungen vermehrter Kaufkraft auf den Konsum, der Realität entsprechen,
[f]	können die realen Bruchstücke so zusammengefügt werden, wie es auf dem Bild durchgespielt wurde.	kann auf das Steueraufkommen geschlossen werden, das nach der Änderung der Steuergesetze entsteht.

Stachowiak wertet wie folgt:

> *Der Gewinn dieser Vorgehensweise liegt auf der Hand: Modellseitig gewonnene Einsichten und Fertigkeiten lassen sich – bei Erfülltsein gewisser Transferierungskriterien – auf das Original übertragen, der Modellbildner gewinnt neue Kenntnisse über das modellierte Original, er bekommt dieses besser als bisher in den Griff, kann es auf neue Weise zweckdienlich umgestalten oder als verbessertes Hilfsmittel für neue Aktionen verwenden.*

Darin stecken auch bereits die Gefahren, die von Modellen ausgehen: Die Transferierungskriterien, von denen Stachowiak spricht, sind in der Praxis oft unklar. Auf diese Weise kommt es oft vor, dass die Aussagekraft eines Modells überschätzt wird und falsche Schlüsse gezogen werden. Ein (auch bei Politikern) sehr populäres Beispiel ist das folgende: Aus Erfahrungsdaten wird abgeleitet (»bewiesen«), dass das Wachstum (einer bestimmten Industrie, des Energiebedarfs, der Erdbevölkerung, des Mülls) exponentiell verläuft. Das kann, da alle Ressourcen unserer Erde endlich sind, nicht lange richtig sein. Trotzdem wird gern damit argumentiert.

1.4 Entwicklung und Validierung von Modellen

Die Entwicklung von Modellen ist der Zweck jeder Forschung. Unabhängig vom Fachgebiet und von der Art des Modells geht man dabei so vor, dass man ein Modell entwirft und es anschließend zu validieren versucht. »Validieren« bedeutet nicht »beweisen«; Modelle sind nicht richtig oder falsch, sondern zweckmäßig oder unzweckmäßig.

1.4.1 Die Entwicklung eines Modells

Modelle entstehen durch Beobachtungen, Analogieschlüsse und Intuition. Beobachtungen geben meist den Anstoß: Wer beobachtet, dass Programme in der Programmiersprache A typischerweise weit weniger syntaktische Fehler enthalten als solche in B, hat schon eine Theorie entwickelt. Er könnte nun ein Modell schaffen, indem er behauptet, dass es (unter bestimmten Bedingungen) in der Sprache A im Mittel F_A Fehler pro 1000 Zeilen gibt, bei Sprache B im Mittel F_B.

1.4.2 Die Validierung eines Modells

Wie die Naturwissenschaftler können wir auch im Software Engineering nur durch die Resultate von Experimenten und Beobachtungen seriös validieren. Leider stehen Experimenten mindestens drei große Hindernisse im Wege:

- Die individuellen Eigenschaften der Probanden, also der Personen, die bei einem Experiment mitwirken, streuen in einem sehr weiten Bereich (siehe dazu Abschnitt 6.3.2); selbst wenn die niedrigsten in der Literatur genannten Zahlen stimmen, geht es um eine volle Größenordnung. Darum müssen für ein Experiment entweder sehr spezielle Gruppen gebildet werden (was die Allgemeingültigkeit der Resultate stark einschränkt) oder sehr viele Probanden herangezogen werden (was inakzeptable Kosten verursacht). Im Beispiel oben müsste zudem sichergestellt werden, dass die Hilfsmittel (z. B. die Editoren) und die Vorkenntnisse der Probanden bezüglich Programmiersprachen und Hilfsmitteln gleich sind.
- Experimente im Bereich des Software Engineerings sind selbst bei wenigen Probanden schon extrem kostspielig und darum im Allgemeinen unmöglich; wo sie durchgeführt werden, handelt es sich bei den Probanden in der Regel um Studierende, die aber kaum repräsentativ für diejenigen sind, über die Aussagen gemacht werden sollen (meist die Software-Entwickler im Allgemeinen).
- Die durch ein Experiment zu beantwortenden Fragen müssen sehr präzise gestellt werden, um das Experiment reproduzierbar zu machen, denn Reproduzierbarkeit ist ein fundamentales Prinzip der wissenschaftlichen Arbeit. Es reicht also nicht aus, danach zu fragen, wie viele Zeilen Code ein Entwickler pro Tag schreibt, sondern es muss exakt definiert sein, welche Art von Code, in welcher Programmiersprache, aufgrund welcher Vorgaben, unter welchen Arbeitsbedingungen, mit welcher Vorbildung und bei welchen weiteren persönlichen Voraussetzungen. Die Erfüllung dieser Forderungen bedeutet aber, dass die Ergebnisse nur für einen winzigen Ausschnitt der Welt gültig sind.

Bei Beobachtungen (Feldstudien) untersucht man aktuelle oder dokumentierte Abläufe, meist Software-Projekte oder Teilprojekte. Gegenüber den Experimenten fallen dabei einige Probleme weg, vor allem das der extrem hohen Kosten. Dafür gibt es andere Probleme: In aller Regel sind Daten weder im erforderlichen Umfang noch in der gewünschten Qualität vorhanden und zugänglich, und die untersuchten Abläufe unterliegen oder unterlagen sehr vielen Einflüssen, sodass es unmöglich ist, die Wirkung eines bestimmten Parameters zu analysieren.

1.4.3 Beispiel: Wie wirkt sich die Methode der Entwicklung aus?

Betrachten wir als Beispiel den Versuch, die Effekte einer bestimmten Methode bei der Software-Entwicklung festzustellen.

Im Experiment bildet man n Entwicklergruppen; n/2 Gruppen entwickeln nach Methode A, n/2 nach Methode B. Die Probanden werden den Gruppen durch Los zugeordnet. Es muss sichergestellt sein, dass die Vorkenntnisse der Probanden hinsichtlich A und B gleich sind. (Das ist kaum zu schaffen.) Über einen solchen (mit einigen Mängeln behafteten) Versuch berichten Boehm, Gray und Seewaldt (1984).

Eine Feldstudie zur selben Frage steht vor dem Problem, dass niemals die gleiche Aufgabe von mehreren Gruppen unter ähnlichen Bedingungen, aber nach verschiedenen Methoden bearbeitet wird. Darum müssen unterschiedliche Auswirkungen der Methoden aus völlig verschiedenen Projekten abgeleitet werden. Dabei ist aber zu befürchten, dass die Projekte weniger durch die Methoden als durch andere Einflüsse, beispielsweise die Zusammensetzung der Entwicklergruppen oder die Unterschiede der Aufgaben und Rahmenbedingungen, geprägt sind.

1.5 Modelle im Software Engineering

Die Übertragung der Modell-Begriffe in die Software ist an sich klar, das Resultat aber oft überraschend, weil sich zeigt, dass wir fast nur mit Modellen arbeiten. Beispielsweise kann eine Software-Spezifikation als Modell des Codes betrachtet werden, der Code als Modell des ausführbaren Programms, dieses als Modell der Ausführung. Wir haben es also mit mehrstufigen Modellen zu tun. Solche Modelle werden in vielen Fällen durch Graphen dargestellt, z. B. durch Datenflussdiagramme.

Auch ein Balkendiagramm mit der Rechenzeit verschiedener Algorithmen ist ein Modell, wobei das Original die Rechenprozesse sind, deren Attribute bis auf Bezeichner und Rechenzeit präteriert sind. Die Form (Balkendiagramm) ist abundant. Ähnliches gilt für alle Metriken (siehe Kap. 14).

Eine wichtige Unterscheidung der Modelle ist die in

- *Software-Modelle*
  Beispiele: Natürlichsprachliche Spezifikation, Entity-Relationship-Diagramm, Use-Case-Diagramm, Z-Spezifikation eines Moduls
- *Vorgehens- und Prozessmodelle*
  Beispiele: Wasserfallmodell, Prototyping, V-Modell

In diesem Zusammenhang sei erneut der Unterschied zwischen deskriptiven und präskriptiven Modellen betont. Wo immer Vorgaben gemacht werden, sei es für eine konkrete Software, sei es für das Vorgehen, haben wir präskriptive Modelle vor uns.

Die Vorgaben für das Projekt kranken oft daran, dass die Realität ihnen nicht folgt. Ein typisches Beispiel sind die Richtlinien, die nicht beachtet werden, sondern in den Schubladen der Entwickler einstauben. Ein solches Modell ist nicht nur nutzlos, sondern u. U. kontraproduktiv, weil es einen allgemeinen Zynismus gegenüber Regelungen und Vorgaben fördert.

Der Flut gutgemeinter, aber wirkungsloser Regelungen steht oft ein Mangel an nüchternen Beschreibungen gegenüber: In vielen Projekten fehlt eine realistische Einschätzung der Risiken, des Entwicklungsstands, der Aufwands- und Terminschätzungen. Die Ist-Analyse (siehe Abschnitt 15.2.3) ist der Schlüssel zum Erfolg – das gilt auch im Projektmanagement.

1.6 Theoriebildung

Eine Theorie ist ein Modell, von dem behauptet wird, dass es Zusammenhänge des Originals erklärt. So ist die Relativitätstheorie ein Modell bestimmter physikalischer Phänomene, sie erklärt Zusammenhänge zwischen Masse, Energie, Raum und Zeit.

Eine Theorie wird als brauchbar betrachtet, wenn das pragmatische Merkmal möglichst umfassend gegeben ist. Beispielsweise erlaubt es die Relativitätstheorie, Aussagen über das Licht zu treffen, die sich experimentell bestätigen lassen (etwa die Ablenkung des Lichts durch Gravitation). Wenn eine neue Theorie mehr leistet als die bisher bekannten, werden diese obsolet. Das war der Fall, als die Kepler'schen »Gesetze« (die natürlich auch nur eine Theorie mit begrenzter Gültigkeit sind) die komplizierteren und zugleich weniger mächtigen Modelle älterer Astronomen ersetzten.

Die sogenannten exakten Wissenschaften sind dadurch gekennzeichnet, dass ihre Theorien ganz überwiegend formalisiert sind und daher ihren Ausdruck in Formeln finden. Die Formel

$$s = \frac{g}{2} \cdot t^2$$

für den Fallweg im Vakuum (mit dem Fallweg s, der Falldauer t und der Fallbeschleunigung g) ist eine solche formalisierte (und hier auch quantifizierte) Theorie. Der zweite Hauptsatz der Thermodynamik (die Entropie S eines geschlossenen Systems kann mit der Zeit t nicht fallen, vgl. Abb. 22–1 auf S. 609) ist durch eine Ungleichung formalisiert:

$$dS/dt \geq 0$$

Die nichtexakten Wissenschaften arbeiten dagegen mit nichtformalen Theorien: Die Arbeiten von Darwin zur Evolution und die Arbeiten von Freud und Jung zur Psychologie enthalten viele solcher Theorien.

Die genannten Beispiele sind nicht repräsentativ: Eine Theorie muss weder eine zentrale wissenschaftliche Frage betreffen noch von einer Berühmtheit entdeckt worden sein. Wer stirnrunzelnd seinen streikenden PC betrachtet und ihn schließlich aus-, dann wieder einschaltet, hat eine Theorie, nämlich die, dass ein Neustart die Probleme beheben wird. Diese Theorie kann im konkreten Fall zutreffen (wenn ein fehlerhaftes Programm hauptspeicherresidente Informationen beschädigt hat) oder auch nicht (wenn die Hardware defekt ist). Sie ist vielleicht gut genug für einen einfachen Ratgeber, aber nicht der Stoff, aus dem Lehrbücher gemacht werden.

Brooks' Law »Adding manpower to a late project makes it even later« (Brooks, 1975) ist weder formal noch präzise, hat aber nach allgemeiner Einschätzung einen wahren Kern. Im Software Engineering gibt es viele Theorien dieser Art.

Abschnitt 3.1.1 (*Die Ausgaben für Hard- und Software*) enthält das Beispiel einer Theorie, die inzwischen als widerlegt gilt. Überhaupt sind alte Prognosen ein unerschöpflicher Quell widerlegter Theorien.

1.7 Modellierung durch Graphen und Grafiken

Graphen, also Gebilde aus meist beschrifteten Knoten und Kanten, spielen im Software Engineering eine besonders wichtige Rolle. Das ist verständlich, denn Ingenieure haben schon immer vorzugsweise mit einfachen Zeichnungen gearbeitet.

Das Thema wird hier nur kurz angesprochen; die Grundlagen der Graphentheorie werden als bekannt vorausgesetzt. Hier geht es nur um eine knappe Einordnung der Graphen in den Kontext des Software Engineerings. Nicht betrachtet werden Bäume als Datenstrukturen, die zur effizienten Speicherung von Informationen dienen.

Da Graphen im Software Engineering typischerweise grafisch dargestellt werden, erläutern wir in Abschnitt 1.7.5 eine konkrete und in diesem Buch benutzte Notation.

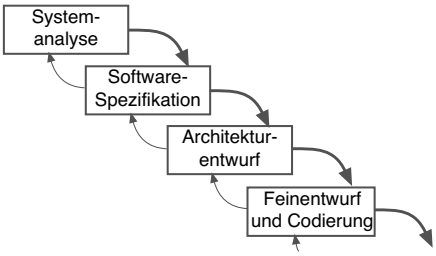
1.7.1 Beispiele für Graphen

Bevor wir Graphen näher untersuchen, betrachten wir einige Beispiele (siehe Abb. 1–3). Der Inhalt wird an dieser Stelle nicht weiter erörtert (siehe dazu die Verweise), es geht hier um das Prinzip. Bei jedem Graph sind einige spezielle Merkmale aufgezählt.

1	Steuerung (Hauptprogramm)
2	Hauptfunktionen im Sinn der Anforderungen
3	Datenstrukturen und Algorithmen
4	Bibliotheken, Basisfunktionen
5	virtueller Rechner (Betriebssystem usw.)

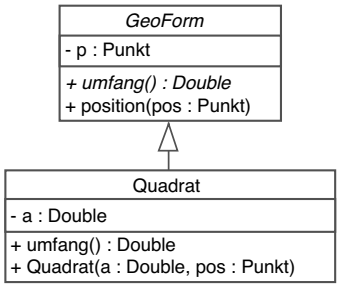
Schichtenstruktur

Die Knoten sind nur durch die Texte erkennbar (sie sind also beschriftet), die (gerichteten) Kanten sind implizit durch die Anordnung der Knoten angegeben.



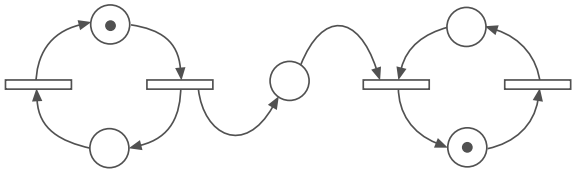
Wasserfallmodell

Die Knoten des gerichteten Graphen sind beschriftet. Es gibt zwei Arten von Kanten (mager und fett dargestellt).



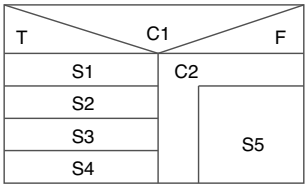
UML-Klassendiagramm

Die Knoten besitzen eine ausgeprägte Substruktur (d. h., die Knoten bilden hierarchisch untergeordnete Graphen), verschiedene gerichtete Kantentypen repräsentieren unterschiedliche Relationen zwischen Knoten; analog gibt es verschiedene Knotentypen.



Petri-Netz

Gerichteter bipartiter Graph (Stellen und Transitionen). Die Stellen sind mit beliebig vielen Marken markiert.



Struktogramm
(Nassi-Shneiderman-Diagramm; Inhalt nur angedeutet)

Die Knotentypen des hierarchischen Graphen sind durch ihre Strukturen unterschieden. Die gerichteten Kanten sind nur implizit dargestellt.

Abb. 1–3 *Beispiele für Graphen im Software Engineering*

Diese Beispiele legen folgenden Schluss nahe: Wenn wir im Software Engineering Graphen verwenden, so sind diese in der Regel gerichtet und die Knoten sind beschriftet. Viele Graphen sind hierarchisch aufgebaut, d. h., Knoten der einen Ebene repräsentieren Graphen der nächsten Ebene. Kanten werden nicht immer in der üblichen Form als Linien dargestellt, sie ergeben sich oft durch die Anordnung der Knoten. Grafische Merkmale werden verwendet, um verschiedene Typen von Knoten, eventuell auch Kanten, zu unterscheiden.

1.7.2 Graphen der Mathematik und Graphen des Software Engineerings

Im Software Engineering verwenden wir Graphen praktisch niemals so, wie sie in den Büchern über Graphentheorie eingeführt werden.

- Der auffälligste Unterschied liegt in der Beschriftung: Sie schafft die logische Verbindung zwischen dem Graphen und der übrigen Software. Knoten sind immer, Kanten oft beschriftet.
- In der Informatik finden wir kaum Beispiele für ungerichtete Graphen; in aller Regel stellen die Kanten Flüsse oder Kausalität dar und sind darum gerichtet.
- In der Mathematik sind die Kanten arm an Information, sie haben nur ein ungeordnetes oder geordnetes Paar von Endknoten. In der Informatik sind sie dagegen oft beschriftet und haben damit wie die Knoten Identität. In diesem Fall ist es auch sinnvoll, für ein bestimmtes Paar von Endknoten mehrere Kanten zu notieren. Die Identität von Kanten führt dazu, dass ein Informatik-Graph, wenn er in einer Datenstruktur abgelegt werden soll, in einen bipartiten Graphen übersetzt wird, bei dem die Knoten der einen Klasse die ursprünglichen Knoten repräsentieren, die der anderen Klasse die ursprünglichen Kanten. Damit entsteht auch die oft gewünschte Möglichkeit, mehr als zwei Knoten durch eine einzige Kante zu verbinden, also mehrstellige Relationen darzustellen.
- Ein weiterer wichtiger Unterschied liegt in der Interpretation der Graphen in ihrer visualisierten Form: Während die Mathematik Graphen als etwas völlig Abstraktes behandelt (»Ein Graph ist ein geordnetes Paar zweier Mengen«), das man nur ungern zweidimensional darstellt, ist für die Ingenieure diese zweidimensionale Darstellung der eigentliche Sinn der Graphen: Die Darstellung *ist* der Graph. Diese Darstellung enthält aber unvermeidlich Attribute, die mit dem abstrakten Graphen nichts zu tun haben (also formal gesehen abundant sind). Beispielsweise kann man einen Knoten höher oder tiefer zwischen die anderen Knoten ordnen, man kann Teilmengen der Knoten optisch zusammenfassen oder auseinanderreißen, man kann Kanten gerade oder rund zeichnen.

Wir verbinden oft auch mit solchen Attributen eine (meist formal nicht definierte) Vorstellung: Der größer dargestellte Knoten repräsentiert ein größeres Programm oder eine schwierigere Aufgabe, die dick gezeichnete Kante ist wichtiger als die gestrichelte usw. Diese Konnotationen sind gefährlich, weil sie Aussagen suggerieren können, die so nicht gemeint sind. Darum sollte man es sich zur Regel machen, alle Attribute entweder zu erklären oder auf Variationen nach Möglichkeit zu verzichten; notfalls sollte man explizit darauf hinweisen, dass darin keine Aussagen stecken.

Betrachten wir als Beispiel die Abbildung 1–1 auf Seite 6. Hier handelt es sich um eine Darstellung, in der ein Graph steckt; in Abbildung 1–4 ist die Information auf den Graphen reduziert. Der gestrichelte Pfeil repräsentiert die Beziehung »modelliert durch«, die übrigen Pfeile repräsentieren die Beziehung »enthält«. Weggefallen ist die durch Abbildung 1–1 nahegelegte Interpretation, dass nur ein *kleiner* Teil des Originals abgebildet wird. Trotzdem ist auch diese Darstellung nicht frei von Konnotationen (beispielsweise über die Ähnlichkeit von Original und Modell, die sich in der sehr ähnlichen Struktur links und rechts äußert).

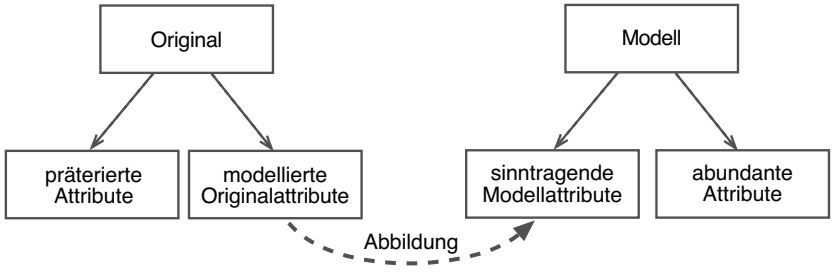


Abb. 1–4 *Original und Modell, schematisch*

1.7.3 Spezielle Eigenschaften der Graphen

Gerichtete Graphen, die einen Baum bilden, sind im Software Engineering oft besonders nützlich: Der Baum gestattet eine einfache Abstraktion, weil jeder Teilbaum (wie auch der ganze Baum) durch seinen Wurzelknoten repräsentiert wird. Beispielsweise kann man – eine entsprechende Baumstruktur vorausgesetzt – einen Programmteil abtrennen, indem man seinen Wurzelknoten löscht.

Auch wenn ein Graph keinen Baum bildet, kann er eine Hierarchie repräsentieren, vorausgesetzt, er ist zyklenfrei. Zyklen erschweren erfahrungsgemäß die Analyse und das Verständnis erheblich. Darum versuchen wir, Zyklen zu vermeiden. Auf niedrigem Abstraktionsniveau gilt das nicht: Rekursive Programme enthalten offensichtlich zyklische Aufrufe, die aber durchaus zur Eleganz und Verständlichkeit beitragen können.

1.7.4 Grafiken und Schaubilder

Weitläufig mit den Graphen verwandt sind Grafiken und Schaubilder. Dort geht es meist darum, Funktionen, beispielsweise die Zahl der gefundenen Fehler als Funktion der Zeit, suggestiv darzustellen.

Für solche Grafiken gelten die Aussagen oben sinngemäß. Insbesondere ist sicherzustellen, dass zu jeder Koordinatenachse genau eine Dimension gehört (z. B. Zeit, Programmgröße, Laufzeit, Fehlerzahl). Natürlich sollte erkennbar sein, woher die dargestellten Daten stammen. Wo nur wenige Datenpunkte verfügbar sind, ist es tollkühn, durch eine spekulative Kurve eine wohldefinierte Funktion zu suggerieren.

1.7.5 Skizzen im Software Engineering

Wie die Beispiele in Abbildung 1–3 auf Seite 14 zeigen, verwendet man im Software Engineering – wie in allen Ingenieurdisziplinen – gern Graphen, um statische oder dynamische Strukturen im ursprünglichen Sinne *anschaulich* darzustellen. Im einfachsten Fall sind das Graphen mit Knoten, dargestellt durch Rechtecke oder Kreise, die die Beschriftung enthalten, und gerichtete Kanten (Pfeile). Abbildung 1–5 zeigt ein einfaches Beispiel. Dabei ist natürlich die konkrete Form der Knoten und der Kanten irrelevant, wenn die Darstellung nicht durch einen Standard vorgegeben ist; wo die Kanten ansetzen, ob sie gerade (und eckig) oder gebogen sind, hängt nur vom Aufwand und vom Geschmack der Urheber ab.

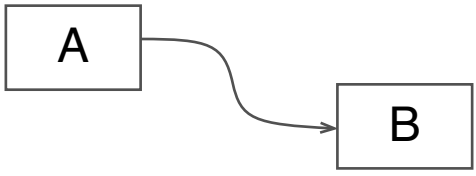


Abb. 1–5 *Typischer Ingenieur-Graph*

In vielen Fällen bietet es sich an, eine Notation zu verwenden, die an die Entity-Relationship-Diagramme nach Chen (1976) und an die Klassendiagramme aus UML (Rumbaugh, Jacobson, Booch, 2004) angelehnt ist. Dabei werden folgende Elemente verwendet:

- Rechtecke für Begriffe (mit dem jeweiligen Begriff beschriftet)
- Kanten (ohne Pfeilspitze oder mit normaler Pfeilspitze) zwischen zwei Begriffen für zweistellige Beziehungen (Assoziationen), die auf Instanzenebene ausgeprägt werden (z. B. für eine Kommunikation zwischen Objekten). Wenn die Kante benannt ist, dann gibt die Pfeilspitze die Leserichtung an.

- Kanten zwischen zwei Begriffen mit einer kleinen Raute an einem Ende für die Beziehungen »ist Teil von«. Der Begriff auf der Rautenseite der Beziehung ist dabei das Aggregat. Eine gefüllte Raute beschreibt eine Komposition; die Komponente ist ausschließlich Teil eines einzigen Aggregats. Bei einer leeren Raute ist dies nicht der Fall, die Komponente kann an mehreren Stellen aggregiert sein.
- Pfeile mit einem leeren Dreieck als Pfeilspitze für Spezialisierungs-/Generalisierungsbeziehungen; ein Pfeil von X nach Y repräsentiert die Beziehung »X ist eine Spezialisierung von Y«, gleichbedeutend mit »Y ist eine Generalisierung von X«.
- Rauten, die durch Kanten mit Begriffen verbunden sind, für Beziehungen zwischen (meist drei oder mehr) Instanzen der Begriffe.
- Rollen als Beschriftungen der Kantenenden; diese sind besonders wichtig, wenn ein Begriff mehrfach in einer Beziehung erscheint, z. B. eine Person als Verkäufer und Käufer.
- Zahlenbereiche als Beschriftungen an den Kantenenden; sie definieren Kardinalitäten, also die Mindest- und Höchstzahl der an einer Beziehung beteiligten Instanzen.

Je nach Kontext kann »Begriff« auch in »Entität« oder »Klasse« übersetzt werden.

Achtung, es geht hier nicht um einen Crashkurs in UML (Unified Modeling Language), sondern nur um die Erläuterung einer Notation, die leicht verständlich und nicht an eine spezielle Version von UML gebunden ist.

Die Abbildung 1–6 zeigt eine typische Anwendung dieser Notation. Links sieht man eine dreistellige Beziehung (eine abgelegte Prüfung) zwischen Student, Prüfung und Prüfungsergebnis. Die Beschriftungen an den Enden der Kanten geben die Rollenbezeichnung der an der Beziehung beteiligten Begriffe an. Das Rechteck »Prüfung« repräsentiert wie alle Rechtecke einen Typ, darunter sieht man Spezialisierungsbeziehungen; Erstprüfungen und Wiederholungsprüfungen sind damit allesamt Prüfungen.

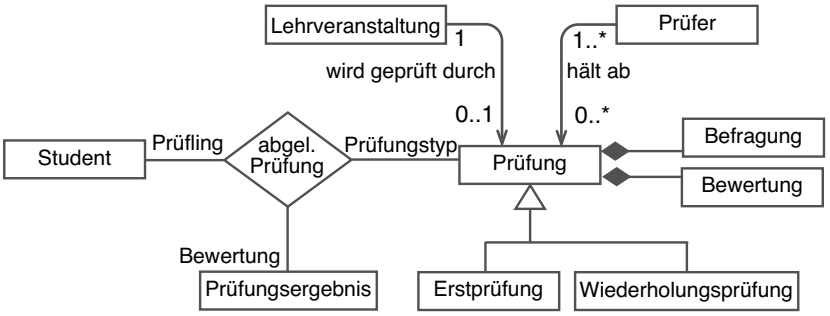


Abb. 1–6 *Beispiel eines Begriffsmodells*

Prüfer und Prüfung stehen in einer m:n-Beziehung, wobei jede Prüfung mindestens einen Prüfer braucht, während ein Prüfer auch keine Prüfung haben kann. Ähnlich gibt es zu einer Lehrveranstaltung eine Prüfung oder auch nicht; zur Prüfung gehört genau eine Lehrveranstaltung. Die Prüfung besteht aus zwei Teilen: aus einer Befragung und einer Bewertung (Kompositionsbeziehungen).

Wir nutzen diese Art von Grafiken an mehreren Stellen dieses Buches. Wo es sich anbietet, verwenden wir auch andere UML-Diagramme, denn sie sind standardisiert und auch in der Industrie akzeptiert. Eine sehr gute Einführung in UML geben Rupp et al. (2012); die vollständigen Spezifikationen zur Sprache UML stellt die OMG (Object Management Group) zur Verfügung (OMG-UML, o. J.).

1.8 Modellierung durch Zahlen: Skalen und Skalentypen

Eine Kennzahl, z. B. der Entwicklungsaufwand in einem Projekt, ist ein Modell, bei dem die Verkürzung alle Merkmale bis auf ein einziges beseitigt hat. Wir bilden alle Projekte auf eine Skala ab, die mit einer Einheit wie »Entwicklerstunde« bezeichnet ist, und machen sie damit vergleichbar. Diese Art der Modellierung ist die Grundlage der Metriken (siehe Kap. 14). Nachfolgend präsentieren wir die Skalentypen und diskutieren ausführlich Fragen, die sich beim Übergang von einem zum anderen Skalentyp stellen.

1.8.1 Die Skalentypen

Auf einer Messlatte ist eine Skala angebracht. Andere Skalen finden wir auf einer Uhr, auf einem Thermometer oder auf einem Voltmeter. Alle Skalen geben neben den Teilungen und Zahlen auch eine Einheit an, beispielsweise auf der Messlatte die Einheit m oder cm. Auch wenn wir, wie bei einem Digitalvoltmeter, keine eigentliche Skala sehen, können wir uns die abgelesenen Werte leicht auf einer geeigneten Skala vorstellen.

Abstrahiert man von den Einheiten und von den speziellen Erscheinungsformen der Skalen (groß oder klein, linear oder rund usw.), so reduziert sich die Vielfalt auf wenige *Skalentypen*[1]. Der Skalentyp definiert gewisse Merkmale und Beschränkungen, die für alle Skalen dieses Skalentyps gelten.

Insbesondere ist die Menge der zulässigen Operationen durch den Skalentyp bestimmt. Sind auf Skala B alle Operationen zugelassen, die auf Skala A zugelassen sind, und mindestens eine, die auf A nicht zugelassen ist, so nennen wir die Skala B *stärker* als A. Unter dieser Relation bilden die Skalentypen eine strenge Ordnung, können also selbst auf einer Skala angeordnet werden (Abb. 1–7).

1. Die Arbeit von Stevens (1946) scheint die Originalquelle zu den Skalentypen zu sein. Darin kommt aber die Absolutskala noch nicht (oder nur implizit) vor.

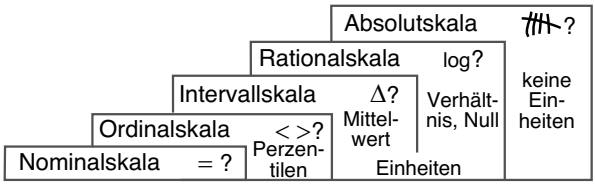


Abb. 1–7 *Skala der Skalen*

Eine Abbildung auf eine (ungeordnete) Menge bringt nur sehr wenig. Beispielsweise sind die Programmiersprachen durch ihre Namen auf eine Menge abgebildet. Wir sprechen in diesem Fall von einer *Nominalskala*. Unterschiedliche Sprachen liegen auf einer Nominalskala, zwischen ihnen ist keine Ordnung definiert.

Von einer echten Metrik wird mindestens gefordert, dass das Resultat der Bewertung in einer geordneten Menge (»Rangliste«) liegt. Dann haben wir eine *Ordinalskala* vor uns. In dieser ist die Reihenfolge von Bedeutung, es gibt aber keine definierten Abstände zwischen den Elementen. Man kann die Gesamtheit durch den *Median* oder andere *Perzentile* charakterisieren. Beispielsweise kann man auch ohne zu messen Bleistifte nach ihrer Länge ordnen. Der Median wird von dem Bleistift repräsentiert, der an n-ter Stelle von 2n-1 Bleistiften liegt, also an der Grenze zwischen der Hälfte der kürzeren und der Hälfte der längeren Bleistifte. Analog liegt die x %-Perzentile am oberen Rand der unteren x %. Der Median ist also die 50 %-Perzentile (siehe auch Abschnitt 1.9.1). (Bei einer geraden Anzahl weicht man auf den Unter- oder den Obermedian aus, also bei hundert Bleistiften auf den 50. oder den 51.)

Stärker noch als die Ordinalskala ist eine *Intervallskala*, in der auch Differenzen zwischen Bewertungen eine Bedeutung haben. Mit Werten dieser Skala kann gerechnet und gemittelt werden.

Am stärksten ist eine *Rationalskala*, die einen natürlichen (d. h. nicht willkürlichen) Nullpunkt hat (z. B. die Rechenzeit in s) und darum auch dem *Verhältnis* der Bewertungen einen Sinn gibt.

Die *Absolutskala* ist eine Rationalskala, bei der der Zahlenwert nicht nur *proportional* zur gesuchten Größe, sondern *selbst* die gesuchte Größe ist; das gilt, wenn es sich um eine natürliche Zahl handelt, die durch Zählen ermittelt wird.

Die Aussage, dass eine höhere Position in Abbildung 1–7 »stärker« bedeutet und damit eine größere Auswahl an Rechenoperationen impliziert, gilt für alle Skalen außer der höchsten, der Absolutskala. Da die Werte einer Absolutskala auf die natürlichen Zahlen beschränkt sind, fallen wir aus der Skala heraus, wenn wir dividieren oder logarithmieren. Damit können wir also auf einer Absolutskala den Median, jedoch nicht den Mittelwert bilden. Wir können aber die Skala zu einer Rationalskala erweitern und dann alle dort verfügbaren Operationen verwenden; so entstehen Aussagen wie »Die Projektgruppen haben im Mittel 7,8 Mitarbeiter«. Darum wird in der Praxis kaum zwischen Absolutskalen und

Rationalskalen unterschieden, auch die durch Zählung ermittelten Werte werden meist als rationale Zahlen behandelt.

Verwendet man Metriken, so muss man sich stets darüber im Klaren sein, auf welcher Skala man sich bewegt und welche Operationen man anwenden darf. In der Regel gilt, dass wir die stärkste verfügbare Skala verwenden, also beispielsweise lieber eine Intervallskala als eine Ordinalskala. Manchmal ist aber das Gegenteil sinnvoll (siehe Abschnitt 1.9.1).

1.8.2 Beispiele für die Skalentypen

Tabelle 1–1 zeigt für die Skalentypen allgemeine Beispiele und Beispiele aus dem Software Engineering.

Skalentyp	Beispiel allgemein	Beispiel Software Engineering
Nominalskala	Nationalität, Geschlecht	Programmiersprache, Passwort
Ordinalskala	Reihenfolge des Posteingangs	Skalentypen, CMMI-Skala
Intervallskala	Temperatur in Grad Celsius	Zeitpunkt (Datum und Uhrzeit)
Rationalskala	Masse, Druck, Stromstärke	Laufzeit, Aufwand
Absolutskala	Kapazität eines Reisebusses, Anzahl der Ferientage	Programmumfang (Code-Zeilen), Fehlerzahl

Tab. 1–1 *Größen und ihre Skalen*

Bei Nominalskalen entsteht oft ein Missverständnis, weil durch die lineare Anordnung (meist alphabetisch) eine Ordinalskala vorgespiegelt wird. Die Ordnung nach dem Alphabet ist aber nur eine Krücke, die mit dem Inhalt der Information nichts zu tun hat und durch die Willkür der Reihenfolge bestimmt ist, in der unsere Buchstaben stehen, außerdem durch die Willkür der Bezeichnungen. Ein anderes Alphabet (»ü« nach »ue« oder nach »uz«) oder ein anderer Name (»hellgrün« oder »lindgrün«) verändert die Reihenfolge. Wenn Nummern vergeben werden, um die eindeutige Identifikation zu erleichtern (beispielsweise Matrikel- oder Passnummern), dann haben diese Nummern den Charakter von Namen, sie liegen (nur) auf einer Nominalskala.

In vielen anderen Fällen repräsentieren Nummern eine Ordnung, eine Rangliste, etwa die Nummern, die an die Wartenden in einer Behörde ausgegeben werden. Diese Nummern liegen also auf einer Ordinalskala, die nicht in eine Intervallskala umgedeutet werden darf (siehe Abschnitt 1.9.2); Ordinalia (der Erste, der Zweite, ...) sind keine Kardinalia (eins, zwei, ...).

Intervallskalen sind rar, die »lineare Zeit«[2] (also im landläufigen Sinne das *Datum*) ist aber eine ganz wichtige Größe, auch in der Informatik. Die Wahl eines Nullpunkts für die Zeit ist willkürlich, darum haben viele Kulturen und Religionen ihre eigenen Bezugspunkte (z. B. Christi Geburt), selbst wenn die Einheiten (Tage, Jahre) übereinstimmen. Rechner verwenden als Nullpunkt ihres Kalenders typischerweise einen Zeitpunkt, der so weit zurückliegt, dass frühere Zeiten (in Berechnungen) nicht vorkommen, z. B. den 1.1.1970 (Unix) oder den 1.1.1904 (Excel für Mac OS), jeweils 0 Uhr GMT. Alle Zeitpunkte werden als Abstand von diesem Zeitpunkt in Sekunden oder in Tagen dargestellt.

In den Naturwissenschaften ist die Rationalskala weit wichtiger als alle anderen; im Software Engineering begegnet sie uns nur bei der Zeitmessung. Dafür gibt es im Software Engineering viel mehr Absolutskalen als in der (Newton'schen) Physik, weil die Welt der Rechner diskret ist und damit das Abzählen erlaubt. In der Teilchenphysik ist das ähnlich.

1.9 Übergänge zwischen verschiedenen Skalentypen

Generell gilt, dass man eine Metrik hinsichtlich der Skala nicht durch einen Trick verbessern kann; dagegen bewirkt die Kombination verschiedener Skalen eine Verschlechterung. Wenn eine Ordinalskala beteiligt ist, dürfen wir nicht mehr rechnen. Diese Einschränkung ist sehr hinderlich, wenn wir verschiedene Systeme, Komponenten oder Prozesse vergleichen wollen. Wir setzen uns darum mit diesem Problem ausführlich in den Abschnitten 1.9.2 bis 1.9.4 auseinander.

1.9.1 Der Wechsel auf eine schwächere Skala

Gelegentlich nimmt man eine Schwächung der Skalen in Kauf, um dadurch eine praktischere Darstellung zu erhalten, vor allem, wenn man mit Werten arbeitet, deren Verhältnisse (Quotienten) weit mehr aussagen als ihre Differenzen. Das trifft speziell für Werte zu, die Sinnesreize repräsentieren, etwa die Helligkeit oder die Lautstärke; dies wurde von E. H. Weber schon 1834 festgestellt (Weber'sches Gesetz). In diesen Fällen normiert und logarithmiert man die Werte. Die Normierung schafft den willkürlichen Nullpunkt, die Logarithmierung (durch die gewählte Basis) einen willkürlichen Faktor. Ein bekanntes Beispiel sind Filmempfindlichkeiten. Die ASA-Skala (American Standards Association) gibt den Kehrwert der Lichtmenge an, die eine bestimmte Schwärzung des Films bewirkt, ein 25-ASA-Film benötigt 32-mal so viel Licht wie ein Film mit 800 ASA. Die deutsche DIN-Skala ist dagegen (ähnlich der Lautstärke-Skala in db) auf den Logarithmus bezogen, eine Zehnerpotenz sind zehn Stufen, d. h., zehn Stufen entspre-

2. Weitere interessante Informationen zu diesem Thema findet man unter »Zeitrechnung« in Wikipedia (o. J.)!

chen dem Faktor 10, drei Stufen entsprechen (wegen lg 2 = 0,3010) einer Verdoppelung der Empfindlichkeit. 25 ASA entsprechen 15 DIN, 800 ASA 30 DIN.

Im Software Engineering kommen solche Fälle aber kaum vor, man sieht sie gelegentlich bei der Größe von Programmen und bei den Kosten. Unentbehrlich ist die logarithmische Skala für Moore's Law (siehe Abschnitt 3.1).

Obwohl man auf der Intervall- und auf der Rationalskala den Mittelwert bilden darf, verwendet man stattdessen gern den Median, denn dieser ist unempfindlich gegen »Ausreißer«, also Werte, die weit über oder unter den anderen liegen und darum den Mittelwert drastisch beeinflussen. Das wird an einem einfachen Beispiel sichtbar:

Nehmen wir an, dass von 51 Studierenden je zehn ihr Studium nach 9, 10, 11, 12 und 13 Semestern abgeschlossen haben; einer hat 62 Semester gebraucht. Dann ist die mittlere Studiendauer $(10 \cdot (9+10+11+12+13)+62)/51 = 12$ Semester; der Median ist nur 11 Semester, denn das ist die Studiendauer des 26. von 51 Studierenden. Die Studiendauer des einzelnen Langzeitstudierende hat auf den Median keinen Einfluss.

Das folgende Beispiel aus dem Software Engineering zeigt den gleichen Effekt: Wenn wir Aussagen über die typische Größe der Module machen wollen, gehen wir von der Zahl der Code-Zeilen aus (Absolutskala). Wenn wir zehn Module mit je 100 LOC (lines of code) haben und eines mit 1200 LOC, dann erhalten wir (nach Übergang auf die Rationalskala) als mittlere Größe 200 LOC. Auch hier führt also ein »Ausreißer« zu einem falschen Eindruck. Ein klareres Bild bekommen wir, wenn wir stattdessen den Median einsetzen. Ordnen wir die elf Module nach ihrer Größe und schauen auf den mittleren (den sechsten), dann haben wir als Median 100 LOC.

Boxplots

Zur anschaulichen Darstellung einer Verteilung setzt man Boxplots ein; die Box umfasst die Quartilen, der Querstrich kennzeichnet den Median. Die beiden durch Striche begrenzten Bereiche oberhalb und unterhalb der Box werden als Whiskers (Fühler) bezeichnet. Ihre Definition ist nicht einheitlich; die Querstriche markieren entweder die Extremwerte oder die letzten Werte, die nicht (nach einem definierten Kriterium) als Ausreißer eingestuft werden. Die Ausreißer werden dann als einzelne Punkte markiert. Wie man im Beispiel (Abb. 1–8, Punktzahlen aus einer Prüfung) sieht, ist die Box umso höher, je stär-

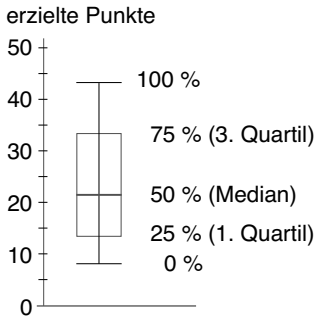


Abb. 1–8 *Boxplot mit Quartilen und Whiskers*

ker die Resultate streuen. Weder die ganz schwachen Kandidaten noch die »Überflieger« haben auf die Höhe und Lage der Box Einfluss.

Eine ausführlichere Beschreibung ist in Wikipedia unter »Boxplot« zu finden.

1.9.2 Rangliste und Ordinalskala

Werte auf einer Ordinalskala haben keine definierten Abstände oder Zwischenwerte. Darum ist es nicht sinnvoll, mit solchen Werten zu rechnen. Wenn beispielsweise die Läufer des Sportvereins »Bitsport« bei einem Langstreckenlauf die Plätze 18, 105, 109, 312 und 502 belegt haben (siehe Tab. 1–2), ist es absurd zu behaupten, sie lägen im Mittel auf Platz 209 oder gar auf Platz 209,2. Eine Rangordnung repräsentiert eine Ordinalskala (und nicht etwa eine Absolutskala!), darum können wir den Median ermitteln (hier 109), dürfen aber keine Mittelwerte berechnen. Dass der Mittelwert tatsächlich unsinnig ist, zeigt sich, wenn man neben der Rangliste auch die Zeiten der Bitsport-Läufer betrachtet.

Name	Rang	Zeit
Adler	18	2:36:12
Bärmann	105	2:43:17
Chow	109	2:50:55
Deckeler	312	2:55:40
Egli	502	3:32:27
Summe	1046	14:38:31
Durchschnitt (???)	209,2	2:55:42

Tab. 1–2 *Resultate bestimmter Teilnehmer in einem Laufwettbewerb*

Der Mittelwert der Laufzeiten ist 2:55:42. Wenn nun zufällig der Läufer auf Rang 313 das Ziel gerade mit dieser Zeit erreicht hat, könnte man behaupten, dass Bitsport im Mittel auf Rang 313 liegt. Aber auch das ist Unsinn. Lediglich die Aussage, dass der Median bei Rang 109 liegt, ist richtig; sie bleibt unverändert, wenn wir statt der Rangliste die Zeiten betrachten, die die Bitsport-Läufer erzielt haben.

Trotzdem wird immer wieder mit Werten auf Ordinalskalen gerechnet. Ist das zu rechtfertigen, oder sind die Resultate solcher Rechnungen einfach falsch? Wir sollten uns daran erinnern, dass wir es mit Modellen zu tun haben und dass darum nicht über richtig oder falsch, sondern über nützlich oder nicht nützlich zu entscheiden ist. Wenn eine Firma den Verein Bitsport unterstützt, hindert sie nichts daran, einen bestimmten Durchschnitt der Ranglistenplätze zu fordern. Erreicht sie dadurch ihr Ziel, mehrfach auf den Sportseiten der Zeitungen erwähnt zu werden, dann war die Vorgabe aus ihrer Sicht nützlich. Vielleicht hätte eine andere noch mehr gebracht (»Mindestens zwei unter den ersten hundert!«).

Wir haben also volle Freiheit, mit unseren Zahlen zu jonglieren. Wir müssen uns aber klar darüber sein, welche Aussagekraft die Resultate haben. Insbesondere dürfen wir nicht nach der Rechnung in den Ausgangsbereich zurückkehren und eine Seriosität vortäuschen, die nicht mehr gegeben ist.

1.9.3 Schulnoten

In der Schule werden Leistungen benotet. Auch die Schulnoten liegen auf einer Ordinalskala; nur wenn in einer Prüfung genau fünf Punkte erzielbar wären und jeder Punkt die Note um eins verbesserte, wäre es eine Rationalskala. Trotzdem werden die Noten der Prüfungen gemittelt. Welche Folgen hat das?

Nehmen wir an, dass bei einer Klassenarbeit in Mathematik maximal 20 Punkte erreichbar sind. Tabelle 1–3 gibt links die Mindestpunktzahl für jede Notenstufe, rechts die Resultate der Schüler Kain und Abel in fünf Klassenarbeiten wieder.

Note	Punkte min.
6	0
5	4
4	8
3	11
2	14
1	17

Punkte Kain	Noten Kain
4	5
8	4
14	2
4	5
4	5
Schnitt 6,8	Schnitt 4,2

Punkte Abel	Noten Abel
3	6
20	1
3	6
7	5
7	5
Schnitt 8,0	Schnitt 4,6

Tab. 1–3 *Notenskala und Ergebnisse von fünf Klassenarbeiten*

Wie man sieht, erreicht Kain mit durchschnittlich 6,8 Punkten einen Notendurchschnitt von 4,2, während Abel mit durchschnittlich 8 Punkten einen Notendurchschnitt von 4,6 erreicht. Bei üblicher Rundung hat Kain also im Zeugnis eine 4, Abel, der insgesamt sechs Punkte mehr erzielt hat, eine 5. (Wenn man eine feinere Abstufung der Noten verwendet, ändert das am grundsätzlichen Problem nichts.)

Das Ergebnis erscheint nicht fair, wenn man erwartet, dass das Zeugnis die Summe der Leistungen reflektiert. Welche anderen Möglichkeiten hätte der Lehrer?

Er könnte anstelle des Mittelwerts den Median der Punkte oder der Noten verwenden. Der Median der Punkte ändert nichts. Mit dem Median der Noten kämen beide Schüler auf eine 5 im Zeugnis. Dass Abel im Mittel der Punkte eine glatte 4 hat, kommt damit noch immer nicht zum Ausdruck.

In der hier geschilderten Situation könnte der Lehrer die durchschnittliche Punktzahl heranziehen, um die Zeugnisnote zu bilden. In anderen Fächern gibt es keine Punktbewertung, sodass diese Möglichkeit nicht besteht.

Nebenbedingungen, nach denen mindestens die Hälfte der Arbeiten »ausreichend« sein muss oder die allerletzte Arbeit (Schlussprüfung) nicht schlechter als 4 sein darf oder der Sohn des Ministers mindestens mit »gut« zu bewerten ist, kommen in keiner der möglichen Auswertungen zum Ausdruck.

Aus dieser Betrachtung können wir folgende Feststellungen ableiten:

- Wenn wir mit Schulnoten oder anderen Werten von Ordinalskalen rechnen, kommen wir oft zu überraschenden, gelegentlich zu unsinnigen Resultaten.
- Damit die Rechnung überhaupt ein sinnvolles Resultat liefern kann, muss gewährleistet sein, dass die einzelnen Prüfungen vergleichbare Beiträge zur Gesamtleistung liefern und sich Stärken und Schwächen ausgleichen können.
- Nebenbedingungen und nicht kompensierbare Schwächen lassen sich in die Rechnung nicht sinnvoll integrieren. Sie müssen *vor* der Rechnung ausgewertet werden.

1.9.4 Aufstellung einer Rangliste aus einer heterogenen Bewertung

In der Praxis geht es oft darum, aus einer Menge von Kandidaten (Wahlmöglichkeiten) den geeignetsten auszuwählen. Kandidaten können nicht nur Personen, sondern auch Gegenstände, Methoden, Reiseziele oder Algorithmen sein, um nur wenige Beispiele zu nennen. Das Problem der Auswahl besteht darin, dass es eine Reihe von Merkmalen der Kandidaten gibt, die die Auswahl beeinflussen. Diese Merkmale können nur benotet, also auf einer Ordinalskala angeordnet werden. Damit ist es zwar möglich, die Kandidaten hinsichtlich eines einzigen Merkmals zu vergleichen, aber nicht unter Berücksichtigung mehrerer Merkmale.

Eine Möglichkeit, das Problem zu lösen, besteht darin, die Noten in den verschiedenen Kategorien zu addieren und damit zu einer Gesamtnote zu verdichten. Diese erlaubt dann, die Kandidaten in einer Rangliste zu ordnen.

Bei diesem Vorgehen gibt es die folgenden Schwierigkeiten:

- Bedingungen, die unbedingt erfüllt sein müssen, können nicht ausgedrückt werden. Darum brauchen wir K.-o.-Kriterien. Ein Kandidat, der ein K.-o.-Kriterium nicht erfüllt, wird von der weiteren Betrachtung ausgeschlossen.
- Nicht alle Merkmale sind gleich wichtig. Darum können die Noten durch unterschiedliche Gewichtungen stärkeren oder schwächeren Einfluss bekommen. In vielen Fällen ist der Preis ein wichtiger Aspekt; dann müssen entweder alle anderen Aspekte durch einen Geldwert ausgedrückt werden, oder der Preis wird durch eine Note dargestellt.

Wenn wir also am Ende eine Gesamtnote bekommen wollen, müssen wir uns zunächst davon überzeugen, dass sich Stärken und Schwächen, die in einzelnen Noten ausgedrückt sind, kompensieren können. Ist diese Voraussetzung (wenn nötig nach entsprechender Gewichtung) gegeben, können wir anschließend die Einzelnoten zu einer Gesamtnote verdichten.

Beispiel: Wenn wir für eine Konferenz einen Redner einladen wollen, der den Eröffnungsvortrag halten soll, können wir verschiedene Aspekte (seine Berühmtheit, die Attraktivität seiner Themen, seine Vortragskunst, seine Honorarforderung) bewerten und gewichten, um auf diese Weise den geeignetsten Redner auszuwählen. Die Forderung, dass es sich um eine *lebende* Person handelt, ist aber ein K.-o.-Kriterium, also ein Kriterium, das unbedingt erfüllt sein muss; Alan Turing und Albert Einstein kommen auch dann nicht infrage, wenn sie mit Abstand am besten bewertet würden.

Sind wir nicht in der Lage, die einzelnen Aspekte gegeneinander aufzurechnen, so bleibt uns nur die Möglichkeit, Profile zu entwickeln, also Vektoren einzelner Bewertungen, und dann intuitiv eine Wahl zu treffen. Dabei können grafische Darstellungen hilfreich sein (z. B. das Kiviat-Diagramm in Abschnitt 14.3.2). Allerdings lässt sich dieses Verfahren nur anwenden, wenn die Zahl der Kandidaten überschaubar ist. Wenn 50 Studienplätzen 2000 Bewerber gegenüber stehen, kann man – nach Anwendung der K.-o.-Kriterien – nur auf eine Auswahl verzichten (Losverfahren) oder die Merkmale der Bewerber vollständig quantifizieren und daraus eine Gesamtnote bilden, die eine Reihung ermöglicht.

Wer sich entschließt, auf der Basis von Benotungen oder anderer Bewertungen auf Ordinalskalen zu rechnen, tut gut daran, seinen Ansatz mit extremen Werten durchzuspielen. Nur wenn im gesamten zulässigen Bereich plausible Resultate entstehen, kann das Verfahren mit der gebotenen Vorsicht angewendet werden.

Wir fassen das Vorgehen zusammen:

- Die relevanten Merkmale der Kandidaten werden identifiziert.
- Die unbedingt geforderten Eigenschaften werden durch K.-o.-Kriterien formuliert.
- Für alle übrigen Merkmale werden Notenskalen und Gewichtungen festgelegt.
- Echte oder erfundene normale und extreme Beispiele werden bewertet, die Ergebnisse werden auf Plausibilität geprüft.

Wenn die Resultate plausibel aussehen,

- werden alle Kandidaten ausgeschlossen, die ein K.-o.-Kriterium verletzen,
- wird für jeden verbliebenen Kandidaten die Gesamtnote ermittelt,
- werden die Kandidaten nach ihren Gesamtnoten geordnet.

Was passiert, wenn ein (früher gültiges) K.-o.-Kriterium nicht ernst genommen wird, zeigt der berühmte Schlussdialog aus »Some like it hot«:

Jerry: We cannot marry. I am a man!
Osgood: Nobody is perfect!

Billy Wilder, 1959