Kapitel 1

# Einleitung

1

# 1

# 1 Einleitung

Die Informatik begegnet uns im Alltag ständig. Einmal natürlich als Rechenanlagen, die wir in Büros, Arztpraxen und zu Hause sehen. Zum anderen ist sie aber auch eingebettet in viele Alltagsgegenstände. Moderne Kameras, Autos, Waschmaschinen und Registrierkassen kommen nicht mehr ohne Programme aus. Gerade diese Allgegenwart bei unterschiedlichem Aussehen macht es Laien schwer, die Informatik als Fach von anderen Fächern zu unterscheiden. Wo ist die Grenze zum Maschinenbau, wenn moderne Maschinen Informatikanteile besitzen? Wo ist die Grenze zur Kommunikationswissenschaft, wenn wir Methoden der Informatik für die Telekommunikation einsetzen? Oder ist das Nachrichtentechnik? Was ist mit Computerspielen und animierten Trickfilmen? Im Sinne der Informatik sind all dies Anwendungsgebiete. Der Anteil der Informatik ist die formale Beschreibung von Vorgängen. Die Beschreibung abstrahiert von physikalischen Merkmalen. Wenn wir beispielsweise Briefwechsel beschreiben, abstrahieren wir die Tinte und das Papier weg. Wir konzentrieren uns auf die Eigenschaften, die wesentlich für einen Briefverkehr sind: die Kommunikationspartner müssen nicht am selben Ort sein und das Schreiben und Lesen muss nicht zur selben Zeit erfolgen. Außerdem finden wir wichtig, dass Sender und Empfänger mit ihren Adressen bekannt sind. Wir berücksichtigen, dass derselbe Text an mehrere Empfänger gehen kann. Diese Beschreibung enthält nur, was allen Briefwechseln gemeinsam ist. Das abstrakte Problem der raum-zeitunabhängigen Kommunikation kann nun durch verschiedene Techniken gelöst werden: Textmitteilungen bei Telefonen, elektronische Post, die zwischen Rechnern ausgetauscht wird, Aufzeichnungen gesprochener Sprache durch so genannte Anrufbeantworter. Schon dieses kleine Beispiel zeigt die Stärke der Abstraktion. Ein Problem wird einmal modelliert und kann unterschiedlich verfeinert werden. Dabei können auch Leistungen entwickelt werden, die in der ursprünglichen Vorlage nicht enthalten waren. Zum Beispiel kann man einen Brief auf Papier nicht hören, eine Nachricht der „voice mail“ schon. So erklärt sich die Vielfalt der Erscheinungsformen von Entwicklungen der Informatik, ohne dass ihr Profil ausfransen würde.

> „Grundsätzlich ist die Informatik jedoch die Wissenschaft der *Abstraktion* – das richtige Modell für ein Problem zu entwerfen und die angemessene mechanisierbare Technik zu ersinnen, um es zu lösen.“ [1]

Bis auf die „mechanisierbare Technik“ würden sich vielleicht mehrere Wissenschaften mit diesem Zitat identifizieren. Im Gegensatz zu Texten und mathematischen Formeln, die sonst oft zur Beschreibung gewählt werden, sind aber die Beschreibungen der Informatik selbst ausführbar. Eine Abstraktionshier-

archie führt von der allgemeinen Problembeschreibung zu Algorithmen, die in einer Programmiersprache realisiert und schließlich physikalisch ausgeführt werden. Die Datenstrukturen und Algorithmen sind ablauffähig, egal welches Gerät den Ablauf durchführt.
Sehen wir einmal, wie die Abstraktionsebenen sich in der Programmentwicklung wiederfinden. Zunächst finden wir gemeinsam mit einem Anwender bzw. einer Auftraggeberin eine *Aufgabenbeschreibung*. Diese abstrahiert von vielen konkreten Vorgängen, die das Programm realisieren soll. Anhand der Aufgabenbeschreibung wird ein *Modell* entwickelt, das die komplexe Aufgabe in Teile aufgliedert. Dieser erste Schritt wird auch als Programmierung im Großen bezeichnet.

---

**1.0.1** **Definition 1.0.1:** *Programmierung im Großen* Die Konzentration auf das Zusammenwirken von Teilen nennt man Programmierung im Großen, weil hier von der inneren Gestalt der Teile abstrahiert wird. Betrachtet wird das Außenverhalten von Teilen. Der englische Terminus ist *programming in the large*.

Bei der Programmierung im Großen werden Arbeitsabläufe betrachtet, eine Architektur für die Software entworfen, die Mengen von Aufgaben und Beziehungen zwischen ihnen angibt. Es werden also Teile mit ihren Zuständigkeiten und Kooperationen herausgearbeitet (siehe Abschnitt 2.2). Wir stellen sie uns je nach Programmierparadigma vor als Mengen von
- Klassen und Objekten (objektorientierte Programmierung),
- Funktionen (funktionale Programmierung) oder
- logischen Beziehungen zwischen Sachverhalten (logische Programmierung).

Für die abstrakten Teile müssen konkrete Umsetzungen gefunden werden. Manche Umsetzungen sind inzwischen Standard geworden. So hat man beispielsweise für geordnete Mengen (z.B. Teilnehmer an einem Wettbewerb, nach ihrer Startnummer geordnet) das Modell der Liste gefunden. Das Modell der Liste kann dann durch die Datenstruktur einer verketteten Liste realisiert werden. In dem Modell wird ebenfalls angegeben, was mit der Liste gemacht werden soll. In der Teilnehmerliste wollen wir einen Teilnehmer finden – hat er sich wirklich angemeldet? Das Wesentliche an der verketteten Liste ist also, dass wir darin suchen. Eine gegebene Programmiersprache erlaubt dann eine bestimmte Formulierung der Datenstruktur.

---

**1.0.2** **Definition 1.0.2:** *Programmierung im Kleinen* Die Ausformulierung von Teilen nennt man Programmierung im Kleinen, weil auf die Einzelheiten eines

Teils geachtet wird. Betrachtet wird die interne Realisierung eines Teils. Der englische Terminus ist *programming in the small*.

Bei der Programmierung im Kleinen haben wir immer drei Fragen:
- *Was* soll realisiert werden (eine Zahl, eine Liste, eine Menge, ...)?
- *Wie* soll es realisiert werden (durch eine mit 16 Bit dargestellte ganze Zahl, durch eine verkettete Liste, ...)?
- *Warum* ist die Realisierung vernünftig (weil sie nach nur endlich vielen Schritten bestimmt das Ergebnis ausgibt; weil sie meist nach nur 16 Sekunden das Ergebnis ausgibt)?

Was kann mit *vernünftig* gemeint sein?
- *Arbeitsablauf:* Das Programm wird in einer bestimmten Arbeitssituation von bestimmten Menschen oder anderen Maschinen genutzt. Wird der Arbeitsablauf durch das Programm besser und für die Menschen angenehmer? Welche Ziele der an dem Arbeitsablauf beteiligten Menschen werden in welchen Anteilen erfüllt, welche nicht? Mit derlei Fragen beschäftigt sich das Fach *Informatik und Gesellschaft*.
- *Wartbarkeit und Wiederverwendbarkeit:* Große Programme müssen gewartet werden. Zum einen, weil die Welt sich ändert, in der die Programme eingesetzt werden und die sie in Teilen abbilden. Zum anderen, weil man immer einen Fehler macht, etwas übersieht, wenn man ein Programm entwickelt. Deshalb ist die Wartung von Software ein wichtiges Thema der *Softwaretechnologie*. Bei wissensbasierten Systemen der *Künstlichen Intelligenz* ist die Wartung und Revidierbarkeit ein Forschungsschwerpunkt. Wiederverwendbarkeit bezeichnet die Möglichkeit, nicht immer wieder von vorn anfangen zu müssen, sondern Teile früherer Programme in den neuen Programmen verwenden zu können. Voraussetzung dafür ist, dass möglichst unabhängige Programmteile (Module) nur durch wohldefinierte Schnittstellen mit anderen Programmteilen verbunden sind, dass Annahmen, die bei der Programmierung gemacht wurden, auch dokumentiert sind, dass man sich beim Programmieren um Allgemeinheit bemüht, statt sehr spezielle Lösungen zu programmieren.
- *Effektivität:* Sind alle Fälle, die vorkommen können, abgedeckt? Dies ist die Frage nach der *Vollständigkeit*. Berechnen wir immer das richtige Ergebnis? Dies ist die Frage nach der *Korrektheit*. Das Spezialgebiet der Informatik, das sich mit diesen Fragen befasst ist die *Programmverifikation*.
- *Effizienz:* Innerhalb der *Komplexitätstheorie* wird die Schwierigkeit von Problemen untersucht. Dabei schätzen wir unter anderem den Aufwand

ab, der im schlimmsten Fall von dem Programm zu erbringen ist: wie lange wird der Rechner uns im schlimmsten Falle auf ein Ergebnis warten lassen? Zusätzlich zur analytischen Aufwandsabschätzung kommt in der praktischen Informatik die empirische Überprüfung durch systematische Experimente. Wie lang braucht das Programm unter bestimmten, systematisch variierten Umständen, bis es ein Ergebnis liefert?

Im Folgenden werden wir stets *wer, was, warum* bei jedem Thema behandeln, wobei hoffentlich die Abstraktionsebenen stets klar bleiben.