# 2 Grundmodell für Ziele, Inhalte und Lehrmethoden

## 2.1 Informatikdidaktische Orientierung für Lehrer und Schüler

Erfolgreicher Informatikunterricht beachtet bekannte Rahmenbedingungen aus dem Lernprozess und aus dem Prozess des fachlichen Arbeitens. In diesem Buch wird ein Lernprozess angestrebt, der vom aktiven Schüler ausgeht, der sich Informatik handlungsorientiert, problemorientiert, anwendungsorientiert und ganzheitlich aneignet und dabei vom Lehrer begleitet wird. Das erfordert Transparenz und Begründung des Lehr-Lern-Konzepts für den Schüler über einen längeren Zeitraum, z.B. für ein Halbjahr, mit Ausblick auf sinnvolle Fortsetzung und Einbettung in verschiedene Bildungsgänge der allgemeinbildenden Sekundarstufe, der beruflichen Bildung und der Weiterbildung. Mit dieser Kombination von Handlungsorientierung, Problemorientierung, Anwendungsorientierung und Ganzheitlichkeit des Lernens gelingt nicht nur die Betonung der Schüleraktivität, sondern diese aktive Rolle wird für den Schüler auch sichtbar. Der immer noch weit verbreitete Frontalunterricht, häufig verknüpft mit langen Lehrermonologen, entspricht der hier empfohlenen Vorgehensweise nicht. Unterschätzt wird meist, dass mehr Schüleraktivität einen deutlich höheren Vorbereitungsaufwand des Lehrers voraussetzt. Was so leicht aussieht bei Unterrichtsbeobachtungen, die den Lehrer als Impulsgeber im Hintergrund zeigen, funktioniert nur bei starker konzeptioneller Vorleistung. *Handlungsorientierung* wird vom Konstruktivismus abgeleitet (Vygotsky, 1978), d.h., der Schüler erwirbt Wissen und Können durch aktives Gestalten im Fach als Einzelperson und Gruppenmitglied. *Problemorientierung* stellt den Schüler vor Anforderungen, die er mit dem erworbenen Wissen und Können zunächst nicht vollständig überschauen kann, aber durch konsequentes Nachdenken und schrittweises Vorgehen schließlich bewältigt werden können und ihn zum Erschließen neuer Lernquellen motivieren soll. Typischerweise regt die Problemorientierung neue soziale Prozesse zur Kommunikation und Kooperation an. Das *ganzheitliche Lernen* soll viele Sinne und Kompetenzbereiche ansprechen, um die Nachhaltigkeit des Gelernten zu sichern. Neben den in der Schule dominierenden kognitiven Anforderungen beobachtet man immer wieder, mit wie viel Begeisterung (affektives Element) und körperlichem Einsatz am Rechner (psychomotorisches Element) Schüler Informatikprobleme bearbeiten. Die *Anwendungsorientie-*

S. Schubert, A. Schwill, *Didaktik der Informatik,*
DOI 10.1007/978-3-8274-2653-6_2, © Spektrum Akademischer Verlag Heidelberg 2011

*rung* ermöglicht es dem Schüler, die Einzelerkenntnisse zu systematisieren und im historischen und lebensweltlichen Kontext zu bewerten. Der Schüler versteht z.B., dass asymmetrische Verschlüsselungsverfahren aus der Notwendigkeit entstanden, viele Schlüsselpaare für schnelle symmetrische Verschlüsselungsverfahren auf einem unsicheren Kanal zu übertragen, und erlebt im historischen Kontext die Anwendung von Primzahlen. Die Bindung des Informatikunterrichts an das Fach wird im Kapitel 3 näher beschrieben.

Vorerst soll gelten, dass gute Verständlichkeit nicht zum Widerspruch zur fachlichen Korrektheit und Erweiterbarkeit des Gelernten führen darf. Informatikbegriffe und -konzepte bauen aufeinander auf, können also nicht in beliebigen Zusammenstellungen erlernt werden. In einem Graphen lässt sich für jeden Lerninhalt die Beziehung zwischen Vorwissen und neuen Erkenntnissen darstellen. Das Vorwissen wurde in der Regel in verschiedenen Unterrichtsfächern erworben, und umgekehrt liefert der Informatikunterricht wichtige Kompetenzen für das Lernen in anderen Fächern. Auf diese Beziehungen wird im Abschnitt 2.5 ausführlicher eingegangen. Handlungsorientierung und Problemorientierung zwingen Lehrer zur Auseinandersetzung mit der informatikspezifischen Vorgehensweise beim Lösen von Aufgaben. Diese Arbeitsweise beruht auf Empfehlungen (Heuristiken), die der Schüler verstehen und anwenden muss. Zum Verstehen benötigt man die geeignete Auswahl an Informatikprinzipien und -methoden, bleibt aber vollständig frei in der Wahl der Werkzeuge. Es existiert also keine Produktbindung in der informatischen Bildung. Im Mittelpunkt steht vielmehr die Vermittlung von Wirkprinzipien, die anhand von Systemen oder Produkten studiert werden und nicht umgekehrt. Es werden primär keine Bedienfertigkeiten vermittelt. Eine kleine Testfrage gibt Aufschluss: Ist das vermittelte informatische Prinzip auch ohne das Beispielsystem relevant, an dem es studiert wird? Die Informatikprinzipien und -methoden unterliegen allerdings einer historischen Bewertung und sind nicht zeitbeständig.

Man empfiehlt heute andere Problemlösestrategien als vor zehn oder zwanzig Jahren (Brauer/Brauer, 1995). Revisionen der Kompetenzbilder und Bildungspläne sind deshalb unverzichtbar (Humbert, 2003). Hier wird vor allem mit Basiskonzepten (Tab. 2.1) argumentiert, deren Bedeutung nicht vordergründig an die Wahl von Programmierstil (z.B. funktional, prozedural, objektorientiert) und Programmiersprache gebunden ist, z.B. der Einsatz virtueller Maschinen (vgl. Kapitel 9).

Gezeigt werden soll, dass Lehrer und Schüler ein informatikdidaktisches Grundmodell (Abb. 2.1) als Orientierungswissen verwenden können, das sowohl die Grundbildung als auch die Vertiefung besser organisieren und bewerten hilft (Schubert, 1991). Wo stehe ich im Bildungsprozess? Diese und folgende andere Fragen kann sich der Schüler selbst beantworten. Warum stimmen meine Lösungsvorschläge nicht mit denen der anderen Gruppenmitglieder

**Tabelle 2.1** Informatikprinzipien und -methoden des Problemlösungsprozesses

| Prinzip | Methode |
|---|---|
| Prozessmodellierung | Gliedern des Entwicklungsprozesses in Phasen, z.B.<br>- Spezifikation der Anforderungen<br>- Implementierung des Entwurfs |
| Einsatz virtueller Maschinen | Arbeiten in Projekten, z.B.<br>- Teststrategien<br>- projektbegleitendes Dokumentieren und Verwalten |
| strukturierte Zerlegung<br>- Modularisierung | - Top-down-Methode<br>- Bottom-up-Methode<br>- Geheimhaltung der Wirkungsweise<br>- Schichtung |
| - Hierarchisierung | - Klassifikation<br>- Schachtelung<br>- Vererbung |

überein? Wo liegen meine persönlichen Stärken und Schwächen? Was erwartet mich in der Informatikvertiefung?

Wünschenswert ist ebenfalls, dass Lehrer und Schüler gemeinsam die Bildungsinhalte mithilfe des Grundmodells so strukturieren, dass Kernkompetenzen und mögliche Erweiterungen klar getrennt werden. Der Bildungsbeitrag eines jeden Moduls kann so genau beschrieben und zertifiziert werden. Sehen wir uns das Grundmodell des Informatikunterrichts näher an. Es besteht inzwischen ein breiter Konsens, dass das Modellieren – Strukturieren und Beschreiben mit formalen Sprachen – in der Informatik Besonderheiten aufweist, die in keinem anderen Unterrichtsfach erlernt werden können, aber unverzichtbar sind für die aktive Mitgestaltung der Gesellschaft, die als Informations- oder Wissensgesellschaft bezeichnet wird (Breier et al., 2000) (vgl. Kapitel 5).

*Handlungsorientierung* konzentriert sich im Informatikunterricht auf die Bereitstellung von Ergebnissen, die nicht auf traditionellem Wege erzielt wurden. Der Schüler rechnet z.B. nicht aus, wie hoch die Jahresdurchschnittstemperatur ist. Er weiß, dass eine Wetterstation die täglichen Messwerte, niedrigste und höchste Tagestemperatur (Tmin, Tmax), in einer Datei bereitstellt, die der Rechner auswerten kann. Das funktioniert nicht ohne Lösungsplan, der bereits vorliegen kann oder konstruiert werden muss (Beispiel 2.1). Der Schüler erlebt, dass er für jede neue Aufgabenstellung prüfen muss, wie er einen Lösungsplan gewinnen kann. Er lernt, wie aufwendig es ist, einen einfachen Lösungsplan selbst zu

entwickeln. Es kann also lohnend sein, in der Literatur und in Programmbibliotheken nach einem Algorithmus für diese Art von Aufgaben zu suchen. Wir sprechen von *Aufgabenklasse* (vgl. Kapitel 5), um zu betonen, dass ein Plan mit einfachen Veränderungen viele ähnliche Aufgaben löst, hier z.B. die Ermittlung der durchschnittlichen Niederschlagsmenge, oder auch des Durchschnittsumsatzes oder des Durchschnittsalters. Der Schüler kann die erforderlichen Anpassungen eines Grundalgorithmus für eine Aufgabenklasse an eine konkrete Aufgabe nur vornehmen, wenn er Aufbau und Funktionsweise eines Lösungsplans prinzipiell verstanden hat. Das Lesen und Schreiben von Lösungsplänen mit einer formalen Beschreibungssprache (z.B. Pseudocode, Beispiel 2.1) und einer Programmiersprache sind deshalb unverzichtbar für den Informatikunterricht insgesamt. Sie müssen aber nicht gleichzeitig erlernt werden. Für eine Informatikeinführung in der Sekundarstufe I können graphische Schemata (eine Auswahl enthält Kapitel 11) als formale Beschreibungssprache zur Anwendung kommen. Die Aufgabenklassen sind so wählbar, dass programmierte Standardlösungen das Erlernen einer Programmiersprache in den ersten Jahrgangsstufen entbehrlich machen (Frey et al., 2001).

**Beispiel 2.1** Pseudocode

*Setze die Anfangswerte SdT:=0 und Anzahl:=0.*
*Wiederhole, solange nicht Dateiende erreicht wurde:*
    *Lies aus der Datei das Wertepaar Tmin, Tmax.*
    *Berechne die Tagesdurchschnittstemperatur dT:=(Tmin+Tmax)/2.*
    *Berechne die Summe der Tagesdurchschnittstemperaturen SdT:=SdT+dT.*
    *Aktualisiere die Anzahl der Wertepaare Anzahl:=Anzahl+1.*
*Berechne die Jahresdurchschnittstemperatur JdT:=SdT/Anzahl.*
*Gib das Ergebnis JdT aus.*

Wenn der Lösungsplan gefunden wurde, kann der Schüler mit ihm experimentieren – ihn implementieren und erproben –, bis die erzeugten Ergebnisse den anfangs formulierten Anforderungen entsprechen. Diese Phase der Ausbildung wird von vielen Schülern als interessant und abwechslungsreich beschrieben. Der Lösungsplan übernimmt die Funktion einer Hypothese, die im Anwendungsprozess auf ihre Tragfähigkeit bzw. Schwachstellen hin untersucht wird. Vernachlässigt wird im Informatikunterricht nicht selten, dass ähnlich wie im naturwissenschaftlichen Unterricht zu protokollieren ist, welche Ergebnisse die Experimente zeigten, und dass nach dem Experimentieren die Ergebnisse interpretiert und bewertet werden müssen (vgl. Kapitel 8).

Da meist kein formaler Nachweis die Korrektheit der Ergebnisse belegen kann, hilft in der Bewertungsphase der Technikeinsatz nicht weiter. Modellieren und Bewerten haben dem Informatikunterricht den Ruf eines schweren Faches eingebracht, vergleichbar zum Abstraktionsniveau der Mathematik. Das Grundmodell (Abb. 2.1) veranschaulicht, welche Schlüsselrolle diesen anspruchsvollen Phasen für den Erkenntnisprozess zukommt.

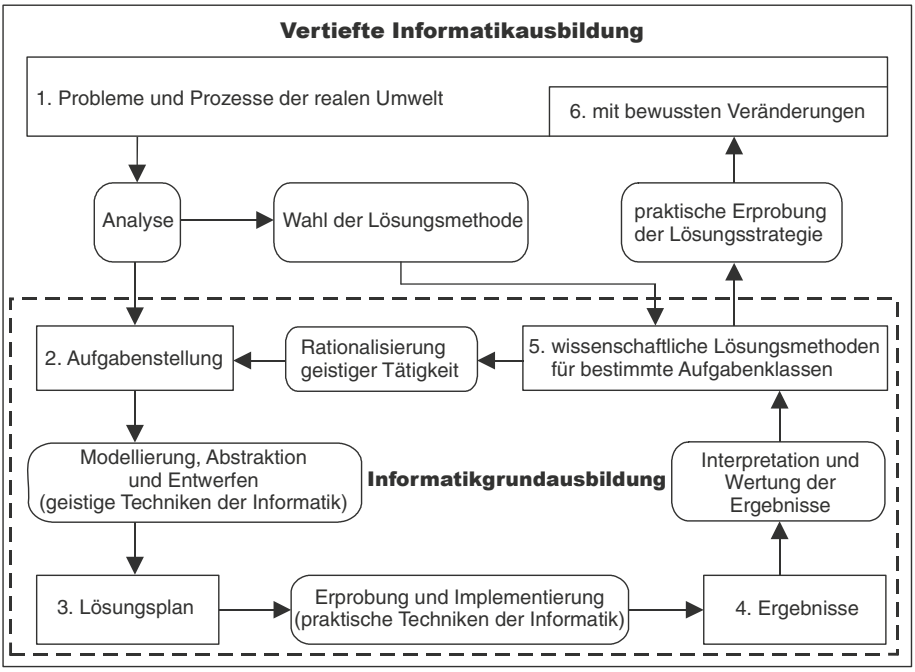


**Abbildung 2.1** Grundmodell für Informatikunterricht

Die eingezeichneten Pfeile zeigen den fachlogischen Ablauf. Sie sind nicht als Lernschritte zu interpretieren. Der Schüler kann mit jedem Lernmaterial, z.B. einer fertig vorliegenden informatischen Lösungsmethode für eine bestimmte Aufgabenklasse, beginnen und sich unterschiedlich tief mit einem Lerngegenstand, z.B. dem Verstehen der Rationalisierung geistiger Tätigkeit, auseinandersetzen. Gerade der Rationalisierungseffekt wird vom Schüler besonders gut verstanden, wenn die Lösungsmethode vorlag und nicht erst in vielen Stunden entwickelt werden musste.

Wie viel Unterrichtzeit benötigt eine solche Grundbildung? Darüber gehen die Erfahrungsberichte weit auseinander. Von Abwahl bedrohte Informatikkurse scheiterten nicht selten daran, dass zu viele Kompetenzen in zu kurzer Zeit ermöglicht werden sollten. Gesichert ist, dass Schüler nach einem Ausbildungsjahr den aktiven Rollen- und Sichtenwechsel zwischen den geistigen und praktischen Arbeitstechniken der Informatik erlernen können. Der Komplexitätsgrad der Beispiele (z.B. im Grund- und Leistungsfach) hängt dabei von bildungspolitischen Vorgaben ab. Das Grundmodell für Informatikunterricht bildet wie jedes Modell nicht alle Aspekte gut ab. So kann man nur indirekt erkennen, dass die angewendeten Informatiksysteme zum Lerngegenstand werden (vgl. Kapitel 9).

## 2.2 Kompetenzen und Unterrichtsziele

Der Schüler soll nach einem Jahr

- seinen Arbeitsplatzrechner als Beispiel einer Rechnerarchitektur, eines Betriebssystems, einer Sammlung von Anwendungssoftware verstehen und daraus Konsequenzen für sein Handeln ableiten können,
- seinen Arbeitsplatzrechner als Element eines Schul-Intranets, eines Weitverkehrsnetzes, eines sensiblen Sicherheitskonzeptes verstehen und daraus Konsequenzen für sein Handeln ableiten können.

Die Basiskonzepte von Rechnerarchitektur, Betriebssystemen, Rechnernetzen und Informationssicherheit gehören deshalb in jede Grundausbildung und müssen angemessen thematisiert werden (vgl. Kapitel 5). Gerade in diesen Bereichen fehlen handlungsorientierte Lernszenarios. Im Abschnitt 2.3 und im Kapitel 9 werden Ansätze dazu vorgestellt. Der Schüler eignet sich soziale und ethische Umgangsformen im Rahmen eines soziotechnischen Systems an. Nicht alle werden den gesellschaftlichen Normen und Wünschen entsprechen. Da die Verbindung kognitiver Prozesse mit sozialen und ethischen Prozessen Lebenszeit erfordert, sehen wir keine Möglichkeit, diese informatische Bildung und Erziehung stark zu komprimieren. Wir empfehlen deshalb ein zweites Ausbildungsjahr zur Vertiefung, d.h. Verknüpfung ausgewählter Ziele der Lebensumwelt mit den angeeigneten Basiskompetenzen. Die Methode des Projektunterrichts (vgl. Kapitel 14) bietet dem Schüler Gelegenheit zur Begegnung mit einer Folge von Informatikprojekten, die er einzeln und im Team gestaltet. Projektpartner außerhalb der Schule sind nach wie vor Einzelerfolge. Es fehlt die Breitenwirkung für alle Schüler an allen Schulen. Regionale Projektbörsen für jeden Schulverwaltungsbereich sind deshalb notwendig. Der Schüler soll durch persönliche Kontakte mit den Anwendern seiner Informatiklösung die Akzeptanz seiner Projektergebnisse kennenlernen. Lernmotivation und Selbstreflektion erhalten dadurch neue Impulse. Wir verwenden hier den *Kompetenzbegriff* der Kultusministerkonferenz:

„*Kompetenz* bezeichnet den Lernerfolg in Bezug auf den einzelnen Lernenden und seine Befähigung zu eigenverantwortlichem Handeln in beruflichen, gesellschaftlichen und privaten Situationen. Demgegenüber wird unter Qualifikation der Lernerfolg in Bezug auf die Verwertbarkeit, d.h. aus der Sicht der Nachfrage in beruflichen, gesellschaftlichen und privaten Situationen, verstanden“ (KMK, 2000, S. 9).

Im Anhang A werden die Definitionen für Handlung-, Fach-, Personal- und Sozialkompetenz aus dem KMK-Dokument aufgeführt.

Welcher Beitrag des Informatikunterrichts zur Bildung in der Wissensgesellschaft wird erwartet? Eine *neue Kulturtechnik* ist zu erlernen. Die ersten Thesen der „Erfurter Resolution“ lauten:

„Der Umgang mit Information ist neben den traditionellen Kulturtechniken des Lesens, Schreibens und Rechnens eine weitere unverzichtbare Grundkompetenz gewor-

den. Dies bedeutet, dass für den Erwerb dieser Kompetenz im Rahmen der aktuellen Schulentwicklung ein fester unterrichtlicher Ort vorgesehen werden muss – wie beispielsweise für schreiben und lesen. … Mit dem Schlagwort Medienerziehung ist die geforderte informatikspezifische Kompetenz nicht zu erreichen" (Erfurter Resolution, 1999).

Diese neue Kulturtechnik darf nicht mit dem Nutzungsaspekt verwechselt werden. Es geht vielmehr um sehr anspruchsvolle Gestaltungsfähigkeiten:

„Reduktion der Komplexität durch Strukturierung, Informatisches Modellieren (Objektorientierung), Interaktions- und Kommunikationsstrategien" (Erfurter Resolution, 1999).

Wenn Information als Rohstoff der Wertschöpfung eingesetzt wird, dann muss Bildung auch alle Schüler darüber aufklären, um eine Benachteiligung bei der Lebensgestaltung auszuschließen. Informationsverarbeitung war viele Jahre eine Spezialtätigkeit für wenige Berufsgruppen. Gegenwärtig fällt es schwer, eine Gruppe zu finden, die von Informationsverarbeitung nicht betroffen ist. „Betroffen sein" führt aber noch nicht zu selbstbestimmtem Handeln. Bildung kann die Kompetenzen fördern, die selbstbestimmtes Handeln im Kontext der Informationsverarbeitung ermöglichen. Dabei ist eine spezifische Ausprägung der informatischen Bildung für die unterschiedlichen Schularten und Bildungsphasen sehr sinnvoll. Je nach Neigung und Begabung empfehlen wir folgende Kompetenzbereiche:

1. Die Bewältigung von informatischen Alltagsanforderungen und die Befreiung von Aberglaube auf dem Gebiet der Informatik umfasst:
   - Beherrschung von Komplexität durch Strukturierung,
   - Informatisches Modellieren,
   - Interaktions- und Kommunikationsstrategien.
2. Der verantwortungsvolle Umgang mit Informatikanwendungen erfordert ein tiefes Verständnis soziotechnischer Systeme und kann deshalb gut mit beruflichen Tätigkeiten verbunden werden.
3. Die menschengerechte Gestaltung von Informatikanwendungen stellt hohe Anforderungen an Kommunikation und Kooperation.

Den ersten Kompetenzbereich benötigen alle Menschen für ihre selbstbestimmte Lebensgestaltung. Er muss im Rahmen der Allgemeinbildung angeeignet werden. Der zweite Kompetenzbereich baut auf dem ersten auf und kann von allen Menschen in individuellen Komplexitätsstufen und in unterschiedlichen Bildungsphasen (Allgemeinbildung, Berufsbildung, lebensbegleitendes Lernen) erworben werden. Der Komplexitätsgrad der Anwendungen und die damit verbundenen Bildungsanforderungen werden häufig unterschätzt.

„Im Zeitalter der Anwenderpakete wird damit das Programmieren nicht mehr nur als Werkzeug benötigt, sondern als Gedankengut, das den vernünftigen Einsatz der Werkzeuge ermöglicht, die von anderen erstellt wurden. Eine ähnliche Aussage gilt für jede Art von Allgemeinbildung" (Nievergelt, 1999, S. 368).

Der dritte Kompetenzbereich kann im Sinne einer Fortsetzung der Aufklärung aus dem ersten Bereich für alle Menschen in unterschiedlichen Bildungsphasen interessant sein, bleibt aber aufgrund der hohen Anforderungen der Prinzipien, Methoden und Werkzeuge ein Bereich, den sich nicht alle Menschen erschließen möchten.

Wir wollen nun den ersten Kompetenzbereich näher beschreiben. Handlungsorientierung bedeutet hier, dass der Schüler einen Tätigkeitszyklus (Abb. 2.2) erlernt, bei dem das Erzeugen und Interpretieren von Fehlern zu einem erfolgreichen Erkenntnisprozess gehört. Der Lehrer soll eine Lernumgebung bereitstellen, die die selbstorganisierte Aneignung des Tätigkeitszyklusses durch den Schüler fördert. Das vereinfachte Schema der Tätigkeiten soll dem Schüler vorliegen wie eine schrittweise zu erkundende Landkarte. Er muss selbst beobachten und bewerten, was er dazugelernt hat, was er beherrscht, wo seine Schwierigkeiten liegen.

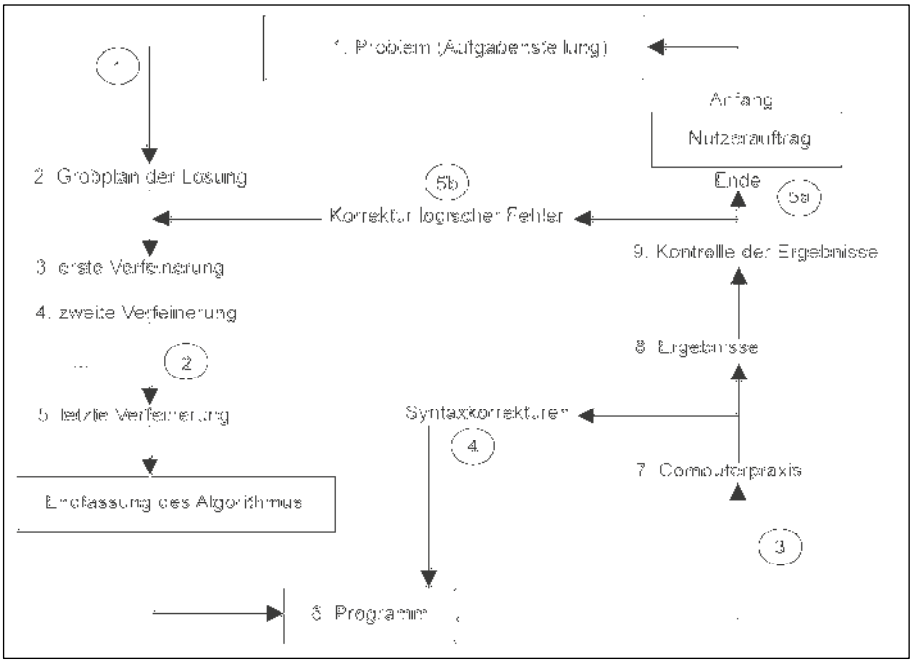


**Abbildung 2.2** Tätigkeitszyklus des Lernenden

Die Ermittlung der Jahresdurchschnittstemperatur stellt z.B. eine sinnvolle Anfangsaufgabe dar, weil gut verständlich ist, dass der Mensch einen Nutzen davon hat, solche Berechnungen künftig immer dem Rechner zu überlassen. Der einmalige Planungsaufwand steht also in guter Relation zum kontinuierlichen Vorteil der Rationalisierung geistiger Arbeit. Der Schüler kann diese Aufgabe aber ohne Vorkenntnisse nicht lösen. Er sieht, dass der Tätigkeitszyklus

die Vorgehensweise von einem Grobplan über mehrere Verfeinerungen bis zur Endfassung der Lösung empfiehlt. Vorerst ohne Rechnereinsatz muss er den Vorteil der Top-down-Methode in der Informatik verstehen. Dazu benötigt er Lernmaterial, das ihm einen Brückenschlag von Alltagserfahrungen, z.B. Reiseplanung, zur Informatik ermöglicht, z.B. Ermittlung der Jahresdurchschnittstemperatur. Der Schüler kann für eine Reiseplanung, z.B. Klassenfahrt nach Prag, beschreiben, wodurch sich der Grobplan von den Verfeinerungen unterscheidet. Er erkennt, dass der Grobplan die Teilaufgaben und Verantwortlichkeiten festlegt, ohne sich bereits mit der Realisierung der Teillösungen zu befassen. Analog geht der Informatiker bei der strukturierten Zerlegung in Module vor. Eine weitere Analogie zur Reiseplanung tritt auf. Wir wissen, wann wir am Ziel sein wollen bzw. müssen, und legen danach den Reisestart fest. Wir wissen, was der Rechner als Ergebnis liefern soll, und leiten daraus die Aktionen ab, die er ausführen muss.

**Beispiel 2.2** Grobplan

*Aktionen für die Ausgabe: JdT*
*Aktionen für die Verarbeitung:*
*- dT bereitstellen*
*- dT summieren*
*- JdT berechnen*
*Aktionen für die Eingabe: unklar*
*- Frage: In welcher Form liegen welche Eingabewerte vor?*
*- Antwort:*
*$Tmin_1$, $Tmax_1$, ... , $Tmin_n$, $Tmax_n$ liegen in einer Datei gespeichert vor.*

Für die Verfeinerung des Grobplans fehlt dem Schüler noch jede praktische Erfahrung. Wir empfehlen deshalb einen methodischen Wechsel zur Computerpraxis mit einer der Teilaufgaben. Wie erzeugt man eine Ausgabe mit dem Rechner? Statt des Wertes JdT, den der Rechner noch nicht ermitteln kann, wird die Zeichenkette „Jahresdurchschnittstemperatur“ zum Üben verwendet. Der Schüler erkennt, dass die Aktionen für die Ausgabe formal beschrieben werden müssen und vom Rechner erst ausgeführt werden, wenn eine spezielle Sprache, die Programmiersprache, zum Einsatz kommt. Schon hier lernt der Schüler, Syntax und Semantik einer formalen Sprache zu unterscheiden und auf die Bedeutung von Fehlern, Syntaxfehler und logische Fehler und deren Korrektur einzugehen. Den Prozess, sich Informatikwissen und -können anzueignen, soll der Schüler als systematisches Testen des Informatiksystems auffassen. Er hat dann den ersten Schritt zum Verständnis der interaktiven Arbeit mit dem Rechner bewältigt. Als nächster Schritt bietet sich an, sich davon zu überzeugen, was Dateien, Sammlungen gespeicherter Daten, wirklich enthalten. Eine solche Aufgabe erfordert die Aneignung von Datentypen und Datenstrukturen (vgl. Kapitel 6). Danach ist es sinnvoll, selbst Daten in Dateiform aufbewahren zu können. Die Lösung der Jahresdurchschnittstemperatur ist komplett,

wenn der Schüler verstanden hat, wie Rechner die Summenberechnung vornehmen, wenn es sich dabei nicht um den Sonderfall weniger, aufzählbarer Werte handelt. Der Schüler kann sein erstes Problem damit vollständig lösen. Der Tätigkeitszyklus begleitet ihn als Orientierungshilfe in seiner Spirale steigender Problemanforderungen, d.h., nach einfachen Aufgaben folgen kompliziertere, die nach ähnlicher Strategie gelöst werden können.

**Tabelle 2.2** Kontrollpunkte im Tätigkeitszyklus

| Geistig planende Tätigkeiten | Praktische Tätigkeiten |
|---|---|
| 1. Aufgabenanalyse zur Datengewinnung und Datenstrukturierung | 3. interaktive Arbeit am Rechner |
| 2. Lösungsplan durch strukturierte Zerlegung (Modularisierung, Hierarchisierung) | 4. Implementierung des Lösungsplans und Korrektur von Syntaxfehlern |
| 5. Erzeugen korrekter Ergebnisse | |
| 5a. Bewertung der Ergebnisse und Korrektur von logischen Fehlern | 5b. Erproben der Lösung (z.B. Setzen von Prüfpunkten, Nutzen von Trace-Komponenten) |

Die Lernerfolgskontrolle (Tab. 2.2) muss beide Bereiche, die geistig planenden und die praktischen Tätigkeiten, in gerechter Verteilung berücksichtigen. Schüler sind unterschiedlich begabt. Es motiviert sie, beide Bereiche zu erlernen, wenn sich die Leistungsmessung auch auf beide erstreckt. In der Vergangenheit galten rechnerbasierte Kontrollen als problematisch, da Zuverlässigkeit und Sicherheit dem Gleichbehandlungsgrundsatz nicht genügten. Diese Probleme sind überwunden. Zurzeit spielen eher organisatorische Hürden der Kursteilung und -beaufsichtigung eine Rolle, da jeder Schüler den individuellen Rechnerzugang in der Prüfung nutzen muss.

## 2.3 Auswahl und Klassifikation der Unterrichtsinhalte

Die Auseinandersetzung um Bildungskern und Linienführung des Schulfaches Informatik wird sehr heftig geführt, aber nicht unbedingt effizient. Als Hindernis hat sich eine Pseudotheorie der Informatik erwiesen, die das Entwickeln kleiner und größerer Programme zum Mittelpunkt des Schulfaches erklärte und sowohl die Lehrerfortbildung als auch die Lehrpläne vieler Bundesländer über Jahrzehnte beeinflusste. Einen Ausweg aus dieser Sackgasse bildet das Konzept der „*Fundamentalen Ideen der Informatik*“ (Schwill, 1993a) (vgl. Kapitel 3). Sie wurden exemplarisch aus dem Softwareentwicklungsprozess begründet und abgeleitet. Ihre prinzipielle Lehrbarkeit auf jedem Schulniveau wurde gezeigt. Da-

hinter verbirgt sich die Hypothese, dass die Softwareentwicklung das allgemeinbildende Kernstück der Fachwissenschaft Informatik darstellt und deshalb zur Strukturierung des gleichnamigen Schulfaches geeignet ist.

Einen zweiten möglichen Zugang zu Auswahl und Klassifikation der Unterrichtsinhalte bildet der neue Typ des Aufgabenlösens, der mit der Arbeitsteilung zwischen Mensch und Informatiksystem entstand und für die Automatisierung geistiger Arbeit in der Informations- oder Wissensgesellschaft typisch ist. Dabei erhalten Lösungsplanung und Mensch-Maschine-Interaktion einen deutlich höheren Stellenwert als die traditionelle Lösung einer Aufgabe, die automatisierbar wird. Der Mensch übernimmt es,

- Ziele zu definieren, Lösungsmodelle auszuwählen und Pläne aufzustellen, mit denen die Ziele erreicht werden,
- Systeme während der Ausführung der Pläne zu steuern,
- die Lösungsvarianten des Systems zu interpretieren, zu bewerten und zu verantworten.

Die Frage „Was ist Informatik?“ wird für den Schüler mit Leitlinien des Schulfaches beantwortbar. Solche Leitlinien erleichtern die Auswahl und Klassifikation der Unterrichtsinhalte, fördern deren Überschaubarkeit und betonen das Wesentliche. Sie leisten einen wichtigen Beitrag zur Lehrbarkeit des Faches. Leitlinien des Schulfaches

- erstrecken sich über mehrere Schuljahre,
- besitzen Schwerpunkte in einzelnen Schuljahren,
- ordnen den Stoff vom Einfachen zum Komplizierten (Spiralcurriculum),
- zeigen die Entwicklung und Beständigkeit grundlegender Prinzipien und Methoden im Fach, z.B. die „Fundamentalen Ideen der Informatik“.

Damit wird die Gefahr der Fehlinterpretation der Curricula reduziert. Die Leitlinien fördern die Vernetzung der Unterrichtsinhalte. Für jedes Element kann der Beitrag zu den Bildungszielen dargestellt werden. Die Sachlogik des Faches trägt zur Strukturierung des Lehr-Lern-Prozesses angemessen bei. Eine fachsystematische Abbildung der Wissenschaft auf die Allgemeinbildung ist nicht zu befürchten, da die Bildungsziele die Auswahl und Transformation der Unterrichtsinhalte prägen (vgl. Kapitel 10).

Wie kommt man zu Leitlinien für den Informatikunterricht? Die Frage „Was ist Informatik?“ lässt sich mit einfachen Teilfragen untersetzen und vollständig beantworten (Tab 2.3).

Zu jeder Teilfrage gibt eine Leitlinie die möglichen Antworten gestaffelt nach Jahrgangsstufen und dem damit verbundenen Vorwissen. Wir benötigen also mindestens drei Leitlinien: Pläne (I), Sprachen (II), Systeme (III). Pläne sollen

**Tabelle 2.3** Unterrichtsleitlinien

| Frage | Leitlinie |
|---|---|
| Was ist möglich?<br>Was wird in der Informatik gemacht? | I. Pläne |
| Wie kann man sich verständigen und etwas darstellen? | II. Sprachen |
| Wie funktionieren die verarbeitenden Informatiksysteme? | III. Systeme |

nicht auf Algorithmen reduziert werden, Sprachen nicht auf Programmiersprachen und Systeme nicht auf Anwendungs-Software.

### *I. Pläne*

Der Lerngegenstand umfasst menschliche Vorgehensweisen (Abb. 2.1 und Abb. 2.2) und die Berechenbarkeit von Algorithmen und heuristischen Funktionen. Hier wird geklärt, welche Eigenschaften Probleme besitzen müssen, um mit Informatiksystemen gelöst werden zu können. Außerdem wird begründet, dass eine hohe Komplexität der Probleme (selbst wenn die Probleme einfach zu beschreiben sind) deren Lösung verhindert. Die Bestimmung von Aufgabenklassen wird an Beispielen erläutert. Als Klassifikationsmerkmale kommen wiederum die Komplexität oder Analogien in der Lösungsstruktur in Frage. Die Begriffe *„berechenbar“*, *„entscheidbar“* und *„akzeptierbar“* werden von der intuitiven Einführung bis zur modellbasierten Definition schrittweise vertieft. Dadurch wird die Rationalisierung geistiger Arbeit verständlich, und die Beurteilung der gesellschaftlichen Anforderungen an die Informatik erhält ein fachliches Fundament. Die Transformation einer natürlichsprachlichen Problembeschreibung in eine abstrakte Darstellung fördert die Einordnung unbekannter Probleme in bekannte Klassen. Das Erkennen innerer Gesetzmäßigkeiten eines Problems erleichtert den Entwurf eines geeigneten Lösungsmodells. Grob- und Feinpläne können strukturiert und auf ihre Komplexität hin analysiert werden (vgl. Kapitel 6 und 7).

### *II. Sprachen*

Man spricht vom „Computerschock“, wenn ein Mensch mit den Reaktionen eines Informatiksystems konfrontiert wird, die von seinen Erwartungen abweichen und er die Orientierung in der Fülle möglicher Benutzungsaktionen verloren hat. Typische Fragen sind dann „Wo bin ich?“, „Wie kam ich hierher?“, „Was kann ich jetzt tun?“. Entgegen allen Behauptungen sind Informatiksysteme bisher überwiegend nicht intuitiv anwendbar und selbsterklärend. Im Gegenteil: Der ständig wachsende Funktionsumfang der Systeme lässt eine effiziente Anwendung nur mit gut überlegten Interaktionsstrategien zu. Der Schüler

lernt, die Menge aller Ikonen oder Menüeinträge einer Benutzungsoberfläche als spezielle formale Sprache zu interpretieren, mit der er sinnvolle Pläne für die Lösung einer Aufgabe mit Anwendungs-Software beschreiben kann.

Der Schüler muss zwischen Interaktion und Kommunikation im Kontext der Informatik unterscheiden lernen. *Interaktion* bezeichnet die Aktionen des Menschen bei der Anwendung eines Informatiksystems. Es wird von Mensch-Maschine-Interaktion oder Mensch-Computer-Interaktion gesprochen. Im Informatikunterricht treten zwei typische Formen auf, das Steuern einer Anwendung über die graphische Benutzungsoberfläche und das Entwerfen und Testen eines Plans mit einer Softwareentwicklungsumgebung. In jedem Fall gehört ein vertieftes Verständnis für Informationssicherheit und Datenschutz zum angemessenen und selbstbestimmten Handeln des Schülers. Unter *Kommunikation* verstehen wir hier den Austausch von Nachrichten unter Menschen. Zur netzbasierten Kommunikation mit Menschen kann der Informatikunterricht einen Bildungsbeitrag leisten, indem er über ethische, soziale und rechtliche Anforderungen aufklärt. Webbasierte Partner- und Gruppenarbeit wird nicht nur angewendet, sondern zum Thema des Informatikunterrichts, indem erklärt wird, warum und wie das möglich ist. Das elektronische Publizieren kann exemplarisch als eine wichtige Anwendung des Informationssystems „WWW" behandelt werden. Der Unterschied zwischen Dokumentenbeschreibungssprachen und Programmiersprachen lässt sich an diesem Beispiel gut verstehen.

Eine formale Sprache ermöglicht dem Schüler, einen natürlichsprachlichen Plan als Programm zu formulieren, den der Rechner ausführen kann. Der Schüler benötigt die Einführung in mindestens eine formale Sprache, deren Syntax, Semantik und Anwendung, um Probleme mit den Methoden der Informatik zu lösen und die Verarbeitung von Daten mittels Rechner zu verstehen. Der Vergleich einer Modellierungs- und einer Programmiersprache fördert das Verständnis für die menschliche Vorgehensweise beim Planen in der Informatik. Eine Klassifikation formaler Sprachen und der sie erzeugenden Grammatiken setzt nicht voraus, dass Vertreter unterschiedlicher Klassen vertieft werden müssen. Es existiert zurzeit keine ideale Programmiersprache für den Informatikunterricht. Eine Sprachenauswahl nach beruflichen Aspekten ist für die Allgemeinbildung abzulehnen (vgl. Kapitel 6, 7 und 11).

### *III. Systeme*

Der Schüler lernt, wie ein Informatiksystem prinzipiell aufgebaut ist und funktioniert. Es werden Repräsentanten wichtiger Systemklassen ausgewählt:

- Informationssystem, z.B. das „WWW",
- Rechnernetz und verteiltes System, z.B. das Schul-Intranet,
- Rechnerarchitektur für ein adäquates Rechnermodell,
- Betriebssysteme, z.B. Linux.

Es besteht keine Notwendigkeit alle Systemklassen in der gleichen Jahrgangsstufe vorzustellen. Sie sollten jedoch alle im Spiralcurriculum für den obligatorischen Informatikunterricht der Sekundarstufe I enthalten sein. Wurde früher im Informatikunterricht typischerweise mit Aufbau und Funktionsweise eines Einzelplatzrechners begonnen, findet heute die erste schulische Begegnung mit einem Rechner im Intranet der Schule statt. Informationssicherheit und Datenschutz stellen deutlich höhere Anforderungen an den Schüler. Eine altersgerechte Einführung in diese Thematik, die jahrgangsweise vertieft und erweitert wird, ist deshalb bereits zum Unterrichtsbeginn notwendig. Für die Schüler steht das Verstehen und Anwenden zeitbeständiger Wirkprinzipien von Informatiksystemen im Vordergrund. Jede Art von Produktschulung muss deshalb abgelehnt werden, da die Kurzlebigkeit des erworbenen Wissens den erforderlichen Transfer auf innovative Entwicklungen behindert. Wenn die Prinzipien verstanden wurden, sind Kriterien formulierbar, die die Auswahl des richtigen Systems, z.B. einer Anwendungs-Software, für die Lösung des jeweiligen Problems ermöglichen. Informatikunterricht ist nicht der Lernort für Benutzungsschulung an Anwendungs-Software.

Die Kriterien für die Gestaltung von Informatiksystemen werden angewendet, anfangs bei der Analyse und Bewertung fertiger Systeme, später auch bei der Anforderungsdefinition für eigene Entwicklungen. Automatenmodelle, z.B. das Modell des endlichen Automaten (Nievergelt, 1999) und des Kellerautomaten, unterstützen das Verständnis komplizierter Informatikanwendungen. Basiskonzepte, wie das der Prozesse, findet der Schüler in allen Informatiksystemen wieder (vgl. Kapitel 9).

## 2.4 Gestaltung und Bewertung typischer Unterrichtssituationen

Im Informatikunterricht kommen alle bekannten Sozialformen (z.B. Gruppenunterricht) und Handlungsmuster (z.B. Experiment) zur Anwendung. Hier interessiert uns der Begriff der Unterrichtssituation.

„Unterrichtsmethodische Handlungskompetenz von Lehrern und Schülern besteht in der Fähigkeit, in immer wieder neuen, nie genau vorhersehbaren *Unterrichtssituationen* zielorientiert, selbständig und unter Beachtung der institutionellen Rahmenbedingungen zu arbeiten, zu interagieren und sich zu verständigen“ (Meyer, 1994, S. 47).

Von allen möglichen Unterrichtssituationen konzentrieren wir uns auf jene, welche mit dem Konstruieren von Informatikbausteinen und -systemen verbunden sind, da hier eine Spezifik auf Schüler und Lehrer zukommt, die in anderen Unterrichtsfächern in dieser Form unbekannt ist. Wir wollen diese Spezifik im Aneigungsprozess als typische Unterrichtssituationen näher charakterisieren.

Aus dem Grundmodell für die Informatikausbildung ergeben sich folgende typischen Unterrichtssituationen, die der Lehrer mit den Schülern gestaltet:

a) Die Möglichkeiten des Rechnereinsatzes in einem Problemfeld sind zu prüfen und zu beurteilen.
b) Es ist eine konkrete, realisierbare Aufgabenstellung aus einem Problemfeld abzuleiten und zu beschreiben als Auftakt der Dokumentation, die alle Schritte begleitet.
c) Die Analyse der Aufgabenstellung ist bis zur Datengewinnung und Datenstrukturierung und zur Bestimmung der Aufgabenklasse zu führen.
d) Die Modellierung des Lösungsprinzips erfolgt bis zur Grobstruktur der Handlungen unter Benutzung der Grundalgorithmen, die in der Aufgabenklasse enthalten sind.
e) Ein Lösungsplan wird durch schrittweise Verfeinerung unter Anwendung von Strukturelementen entwickelt.
f) Ein Logiktest wird zur Überprüfung des Lösungsplanes eingesetzt.
g) Lösungsplan und Datenstrukturen werden in die Programmiersprache übertragen, oder das Programm wird mittels Codegenerator automatisch erzeugt.
h) Implementierung und Erprobung des Programms am Rechner führen mit den Korrekturschritten bis zur Erzeugung korrekter Ergebnisse, die interpretiert und bewertet werden müssen.
i) Die Informatiklösung ist insgesamt zu beurteilen und, wenn möglich, zu verbessern.
j) Die Dokumentation ist abzuschließen, und der entwickelte Informatikbaustein bzw. das Informatiksystem ist anzuwenden.

Man beachte, dass a), b), i) und j) der Vertiefung zuzuordnen sind, die die Schüler meist nur einmal als Abschluss ihrer Ausbildung in einer Projektarbeit kennenlernen und absolvieren. Die Grundausbildung wiederholt die Unterrichtssituationen c) bis h) mehrmals, um eine gewisse Sicherheit im elementaren Wissen und Können zu erzielen. Alle drei Leitlinien, Pläne, Sprachen und Systeme, tragen zur Aneignung des mit a) bis j) beschriebenen Problemlösens mit Methoden der Informatik bei (Tab. 2.4).

So bilden Einführung in die Entwurfsstrategie, Lösungstransfer und heuristische Vorgehensweise beim Lösen komplexer Übungen Schwerpunkte für Pläne. Bei der Implementierung mittels Programmiersprache wird die Anwendung formaler Sprachen gefördert. Die netzbasierte Gruppenarbeit erfordert ein Grundverständnis für Kommunikationssysteme. Wir konzentrieren uns zuerst auf die Verknüpfung der Leitlinien und betrachten den Lernprozess als Ganzes. In den folgenden Kapiteln werden die Leitlinien detaillierter untersetzt. Da Problemlösen nicht zeitunabhängig ist, müssen wir prüfen, welche Entwicklungen auf diesem Gebiet unterrichtsrelevant sind (vgl. Kapitel 4).

**Tabelle 2.4** Einführung in das Problemlösen der Informatik

| Stufe | Stoff | Schwerpunkt |
|---|---|---|
| 1 | komplexere Übungen<br>Informationssicherheit<br>Rechnernetz | heuristische Vorgehensweise<br>Schutzmöglichkeiten<br>Verteilungsmöglichkeiten |
| 3 | Hierarchisierung<br>Datenstruktur Liste<br>Betriebssystem | Lösungstransfer<br>Korrekturmöglichkeiten<br>Schichten-Modell |
| 2 | Zyklus, Alternative<br>Datenstruktur Datei<br>Rechnerarchitektur | Bewerten der Ergebnisse<br>Speichermöglichkeiten<br>Rechnermodell |
| 1 | Modularisierung, Datentypen<br>Informationssystem<br>Kommunikationssystem | Einführung in die Entwurfsstrategie<br>Prinzip der Rechneranwendung<br>netzbasierte Gruppenarbeit |

Das Entwerfen informatischer Modelle durch *strukturierte Zerlegung* (Modularisierung, Hierarchisierung, Orthogonalisierung) wurde als „Fundamentale Idee der Informatik" begründet (vgl. Kapitel 3). In den letzten Jahren wurden Aufgabenklassen im Informatikunterricht erprobt, die den Modellbildungsprozess betonen und von den Besonderheiten der Programmiersprachen trennen (Brinda, 2000). Die typischen Unterrichtssituationen a) bis j) besitzen einen Bildungswert, der weit über den Informatikunterricht hinausreicht (vgl. Kapitel 5). In ihnen erlernt der Schüler Techniken der geistigen Arbeit und einen besonderen Umgang mit Wissen, der für die aktive Teilnahme an der Informations- oder Wissensgesellschaft unverzichtbar ist und zugleich eine Schlüsselkompetenz für das lebensbegleitende Lernen darstellt. Diese Aspekte werden in den beiden folgenden Abschnitten ausführlicher vorgestellt, da sie den Schülern im Unterricht auch erklärt werden müssen.

### *Techniken der geistigen Arbeit*

Inzwischen hat sich Modellieren und Strukturieren im Informatikunterricht bewährt. Das komplizierte Wechselspiel von Modell, Algorithmus und System erfordert jedoch neue Techniken der geistigen Arbeit für die Wissensaufnahme, die Darstellung von Problemlösungen und die Systemerprobung (systematisches Testen und Verifikation von Lösungsabschnitten) (Abb. 2.3).

Stark vernachlässigt wurde bisher das Konzept des *Nichtdeterminismus*. Er spielt bei der Diskussion und Konstruktion von Algorithmen eine wichtige Rolle. Die Vorgehensweise „Erzeugen von möglichen Lösungshypothesen und ihre Über-

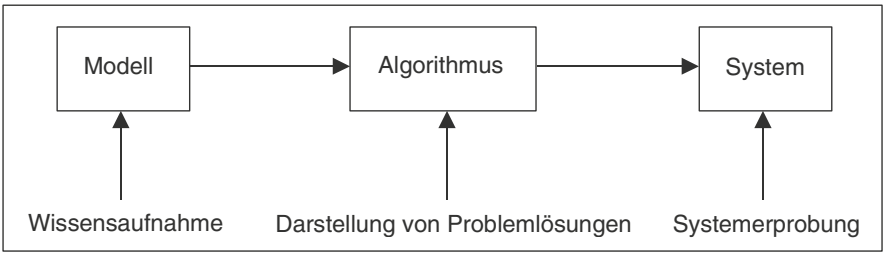


**Abbildung 2.3** Modellieren und Strukturieren

prüfung", z.B. beim Vier-Farben-Problem (vgl. Kapitel 4), ist eine Strategie, bei der nichtdeterministisches Denken zu interessanten Lösungen führt.

Zustandsgraphen, die Zerlegung in Teilproblemgraphen, die Beschreibung und Formalisierung einer Heuristik bilden einen natürlichen Zugang zu Datenstrukturen. Problemlösen heißt dann, Algorithmen kennenzulernen, die im Zustandsraum die Lösungssuche realisieren. Die Abhängigkeit zwischen der Wahl der Datenstruktur und der Reduzierung der Problemkomplexität kann auf diese Weise sehr gut veranschaulicht werden. Strategien für die Organisation der Suche ermöglichen die Diskussion darüber, welche Sprachen und Systeme für eine Problemlösung besonders geeignet sind.

Suchalgorithmen kommt heute im Informatikunterricht der Stellenwert zu, den die Sortieralgorithmen vorher einnahmen. Damit wird die fachdidaktische Frage, wie man Verständnis für die konzeptionellen Schichten moderner Informatikanwendungen gewinnt, ohne deren Entwicklung im Detail nachvollziehen zu müssen, wieder auf Grundalgorithmen und Datenstrukturen zurückgeführt, aber diesmal ohne die enge Bindung an die Programmiersprache.

Es besteht die Gefahr der Einengung auf zu schlichte Bildungsziele, da die komplexen Modellhintergründe zur Undurchsichtigkeit der Systeme führen. Informatikdidaktik stellt sich deshalb die Aufgabe, die Transparenz komplizierter Algorithmen und Datenstrukturen zu fördern, damit menschliche Verantwortung tatsächlich wahrgenommen werden kann. Dabei steht die Komplexitätsbewältigung im Vordergrund. Moderne Benutzungsoberflächen lösen dieses Bildungsproblem nicht. Das Schulfach Informatik kann von aktuellen Hardware- und Softwareentwicklungen Abstand gewinnen, wenn es sich an den Erfahrungen solcher Fächer wie Mathematik und Physik orientiert.

Wie die Mathematik besitzt die Informatik eigene Methoden für das Modellieren von Problemlösungen, die in vielen anderen Disziplinen angewendet werden können, als geistige Basistechnologie. Man spricht von Mathematisierung und Informatisierung, um diesen Grundlagencharakter von Mathematik und Informatik für viele andere Wissenschaften auszudrücken.

In Analogie zur Physik wendet die Informatik spezifische Möglichkeiten für anspruchsvolle Experimente an, um die Wirkprinzipien künstlicher Systeme zu

untersuchen. Hardware-Labore stehen eher den traditionellen Versuchsfeldern der Ingenieurwissenschaften nahe. Aber Software-Experimente im Sinne des explorativen Erkenntnisgewinns orientieren sich an Vorgehensweisen der Physik. Hypothesen über die Zusammenhänge der Parameter werden formuliert. Zur Überprüfung der Hypothese wird ein Experiment geplant, mit Software-Bausteinen aufgebaut und durchgeführt. Beobachtungen und Messungen sind zu protokollieren. Das Protokoll hilft, die Zusammenhänge der beteiligten Faktoren bzw. Komponenten zu klären. Mögliche Beobachtungs- oder Messfehler sind zu diskutieren. Auch diese Technologie der geistigen Arbeit beeinflusst zunehmend mehr Fachdisziplinen.

*Die pädagogische Doppelfunktion der Informatik, d.h. im Fach Informatik zugleich für andere Fächer lernen, verstärkt sich weiterhin.*

In der kurzen Geschichte des Schulfaches Informatik hat sich der Schwerpunkt von Sprach- und Systemdetails, also vielen Fakten, auf die menschlichen Vorgehensweisen beim informatischen Modellieren mit den Phasen Analyse, Spezifikation, Entwurf, Programmierung und Erprobung verlagert (vgl. Kapitel 6). Von exemplarischen Anwendungen sollte der Lernprozess zur theoretischen Durchdringung von Modellen führen, an denen, didaktisch vereinfacht, das Wesentliche komplizierter Algorithmen und Datenstrukturen erklärbar wird. So sind z.B. Informatiksysteme zur Untersuchung von Zusammenhängen bei Umweltschutzaufgaben durch solch hohe Komplexität gekennzeichnet. Nicht die Anwendungen, sondern die zugrundeliegenden Strukturen ermöglichen zukunftssicheres Allgemeinwissen. Die didaktische Vereinfachung darf die Vernetzung der Einflussgrößen nicht zerstören, kann aber sehr wohl einen Realitätsausschnitt zum Problemraum erklären, der als „abgeschlossene Welt" behandelt wird. Dabei kann eine Wissensbasis aufgestellt werden, die die Abhängigkeiten in Form von Fakten und Regeln enthält und auf Anfragen der Schüler Lösungen ableitet. Die Schüler können die Regeln mit Prioritäten versehen und experimentell deren Wirksamkeit überprüfen. Typischerweise sind den Regeln Nebenbedingungen zuzuordnen, um irreversible Prozesse auszuschließen. Die Schüler beobachten den Konflikt, wenn keine ideale Lösung des Problems mehr möglich ist. Im nächsten Schritt lernen sie die Bedingungen so zu verunschärfen, dass mit schwächeren Anforderungen an ausgewählte Kriterien eine Kompromisslösung erzielt wird.

### *Umgang mit Wissen*

In vielen Bereichen fehlt gut strukturiertes Wissen für die vollständig algorithmische Lösung von Aufgaben (z.B. in der medizinischen oder technischen Diagnostik). Das Informatikgrundwissen der Schüler reicht nicht aus, um den Einsatz des Rechners für solche Modellierungen zu verstehen. Obwohl sie täglich mit Wissen zu tun haben, wissen sie zu wenig über dessen komplizierte Eigenschaften (unvollständig, unsicher, zeitabhängig).

Die Idee, Wissen (eine Wissensbank) von der Problemlösekomponente (einem Inferenzmechanismus) getrennt aufzubauen, führt zu der Erkenntnis, dass nicht das konkrete Wissen, sondern dessen Darstellungsform die Problemlösekomponente beeinflusst. Das heuristische Vorgehen eines Fachmanns wird z.B. in Regeln formalisiert. Diese Regeln operieren auf Datenstrukturen. Die Reihenfolge, in der die Regeln verwendet werden, kann zur Problemlösung führen. Damit werden Methoden der Wissensrepräsentation eine Denkhilfe, um auch heuristisches Wissen zu strukturieren, zu formalisieren, zu klassifizieren und zu bewerten. Das aber gewinnt in der Allgemeinbildung grundlegende Bedeutung, da es die Möglichkeit zum selbständigen Weiterlernen und zum Lernen auf höherer Abstraktionsstufe, d.h. zum Lernen über das Lernen, fördert. Damit wird die These gestützt, dass Wissenserwerb, Wissensverarbeitung und die Methoden des Schlussfolgerns kognitiven Wert an sich besitzen (vgl. Kapitel 5). Problemlösungen führen die Schüler auf Techniken der Regelauswahl und Regelverarbeitung zurück. Die Vorwärts- bzw. Rückwärtsverkettung sind Strategien, die universell in vielen Disziplinen einsetzbar sind und deren Anwendung zum Unterrichtsthema wird (vgl. Kapitel 4). Die Diskussion darüber, wie eine Konfliktbeseitigung modelliert werden kann, führt wieder zu den Suchalgorithmen, z.B. auf Zustandsgraphen (s. auch Abschnitt 11.1).

Die Algorithmen und Datenstrukturen für die Wissensverarbeitung zeigen, dass diese Grundlinie des Informatikunterrichts weiterentwickelbar ist. Sie ermöglicht die Behandlung neuer Aufgabenklassen wie Klassifizieren, Planen, Entscheiden, Beraten, Lernen. Der Zugang zum Problemlösen mit Metaregeln, also Regeln zur Anwendung von Regeln, unterstreicht die Bedeutung dieser Techniken. Die Ableitung neuen Wissens kann transparent gemacht werden. Erklärungsalgorithmen, die ein Inferenzprotokoll anbieten, sog. „Wie-Erklärungen", zeigen den Schülern, welche Regeln mit welchen Nutzereingaben zum aktuellen Ergebnis führten. Sie werden aber bei umfangreichen Lösungsableitungen unübersichtlich. Bei Anforderung einer Eingabe können die Schüler einen anderen Erklärungsalgorithmus anfordern, eine „Warum-Erklärung". Es wird dann die augenblicklich verwendete Regel mit den bereits erfüllten Prämissen angezeigt. Unklar bleibt jedoch die Bedeutung der Objekte und Fakten und die Entwicklung und Tragweite der heuristischen Bewertung einer Situation. Das bildet aber gerade die Brücke vom Nichtwissen zum Wissen. Die Modellierung einer solchen Mensch-Maschine-Interaktion erlaubt es den Schülern, tiefere Einsichten in Missverständnisse, Fehlhandlungen und deren Ursachen zu gewinnen.

Die Übertragung des menschlichen Lernens aus Beispielen und Fehlern auf maschinelles Lernen kann an kleinen wissensbasierten Lösungen diskutiert werden. Lernalgorithmen werden auf diese Weise entmystifiziert.

Kriterien für Software-Qualität erhalten ein fachliches Fundament. Die Illusion vom naiven Benutzer, der mit Expertensystemen komplizierte Aufgaben löst, wird abgebaut zugunsten des Leitbildes vom kompetent entscheidenden Fach-

mann, der um die Leistungsgrenze des Modells weiß, Varianten prüfen und beurteilen kann.

## 2.5 Fachübergreifendes und fächerverbindendes Lernen

Jedes Unterrichtsfach leistet einen Beitrag zum fächerübergreifenden und fächerverbindenden Lernen. Deshalb ist es nicht so einfach zu verstehen, dass das Schulfach Informatik hier eine Sonderstellung einnimmt, die im Wesentlichen drei Gründe hat. Der erste Grund liegt in der pädagogischen Doppelfunktion der Anwendungsaufgaben des Informatikunterrichts. Dieser Grund hängt eng mit dem zweiten Grund zusammen, der Informatisierung aller Fachgebiete und damit verbunden auch aller Unterrichtsfächer (vgl. Abschnitt 2.4 und Kapitel 5).

Der dritte Grund hat historische Wurzeln im Scheitern der *„Informationstechnischen Grundbildung – ITG"* (vgl. Kapitel 1). Wir gehen darauf zuerst ein. Auf massiven Druck der Elternschaft und der Wirtschaft wurde über ein Fundamentum zur Informatik für alle Schüler beraten. 1984 beschloss die Bund-Länder-Kommission für Bildungsplanung und Forschungsförderung (BLK) das Rahmenkonzept für die „Informationstechnische Bildung in Schule und Ausbildung". Seitdem ist der Begriff ITG für das Fundamentum üblich, das für alle Schüler obligatorisch ist. Die ITG sollte einen Beitrag zur informatischen Bildung ausschließlich über fachübergreifendes und fächerverbindendes Lernen leisten. Ein Grund für das Scheitern dieser ITG lag darin, dass für die Schüler die informatischen Lernziele nicht deutlich herausgestellt wurden. Der Beschluss einer ITG bleibt aber gültig. Das Anwenden von Informatiksystemen und die gesellschaftliche Bedeutung der Informatik sollten von den Schülern in ca. 60 Unterrichtsstunden erlernt werden. Mögliche Themen waren z.B.: Verwaltung von Schülerdateien, Kalkulation einer Klassenfahrt, Wahlanalysen, Bewerbungsschreiben, Schülerzeitung, Simulation ökologischer Systeme, Auswertung von Schülerumfragen zu aktuellen Themen.

Es existierten verschiedene Möglichkeiten für die konkrete Realisierung (vgl. Kapitel 1). Inzwischen gibt es Informatiklehrpläne für die *Sekundarstufe I* in vielen Bundesländern, die weit über die Anwendungsschulung der ITG hinausgehen.

Sehen wir uns den ersten Grund näher an, die *pädagogische Doppelfunktion* der Anwendungsaufgaben des Informatikunterrichts (Becker, 1986). Im Informatikunterricht dominierte lange Zeit das fachübergreifende Lernen besonders beim Programmieren von kleinen Lösungen zu Aufgaben aus anderen Fächern. Dabei wurden sehr viele Aufgaben aus dem Mathematikunterricht gewählt. Die Datenmodellierung reduziert sich in solchen Fällen auf die Auswahl von Daten-

typen für Zahlen, und viele wichtige Modellierungsaspekte (vgl. Kapitel 6) kommen zu kurz. Die Schüler besitzen in der Informatikgrundausbildung noch nicht die fachlichen Voraussetzungen, um Problemstellungen der Informatik zu lösen. Deshalb wurde bei der Erkenntnisgewinnung im Fach Informatik meist von Problemstellungen aus anderen Unterrichtsfächern ausgegangen. Die Schüler lernten also im Idealfall für zwei Unterrichtsfächer gleichzeitig (Abb. 2.4). Deshalb spricht man von pädagogischer Doppelfunktion.

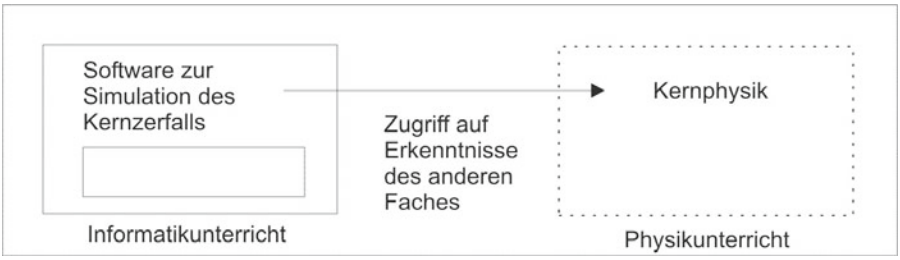


**Abbildung 2.4** Fachübergreifendes Lernen

Im Realfall waren sie häufig überfordert mit dieser Form des kontinuierlichen fachübergreifenden Lernens. Die erwarteten Vorkenntnisse aus dem anderen Unterrichtsfach erwiesen sich häufig als nicht anwendungsbereit. Dann musste die Informatikunterrichtszeit zur Wiederholung dieses Stoffes eingesetzt werden – eine Situation, die Lehrer und Schüler nicht motivierte. Inzwischen werden Informatiksysteme im Unterricht von Schülern so analysiert, modifiziert, gestaltet, dass der Schwerpunkt des Lernprozesses auf Inhalten der Informatik liegt. Als Anwendungsfall eignet sich eine Lebenswelterfahrung ebenso wie eine andere Wissenschaft. Das fachübergreifende Lernen nimmt immer noch angemessenen Raum ein, wird sich aber deutlich verlagern zu anderen Unterrichtsfächern, die auf die Vorkenntnisse aus dem Informatikunterricht zugreifen müssen.

Das Unterrichtsfach Informatik nimmt vor allem wegen des zweiten Grundes – der *Informatisierung aller Fachgebiete* und damit verbunden auch aller Unterrichtsfächer – eine besondere Stellung in Schulen ein. Das hat das Lernen bereits entscheidend verändert. Jedes andere Fach wendet Wissen und Können aus dem Informatikunterricht an. Das lässt sich nicht auf das Starten und Beenden von Informatiksystemen reduzieren. Das Gewinnen, Speichern, Verarbeiten und Interpretieren von Daten steht im Mittelpunkt. Deshalb kann jedes Fach einen fachübergreifenden Bezug zum Informatikunterricht herstellen. Häufig fehlt für diesen Ansatz noch eine solide Informatikgrundbildung für alle Lehrer.

Das fächerverbindende Lernen in Projektwochen oder Kombinationskursen, z.B. Musik und Informatik, lässt sich bedeutend leichter umsetzen (Abb. 2.5).

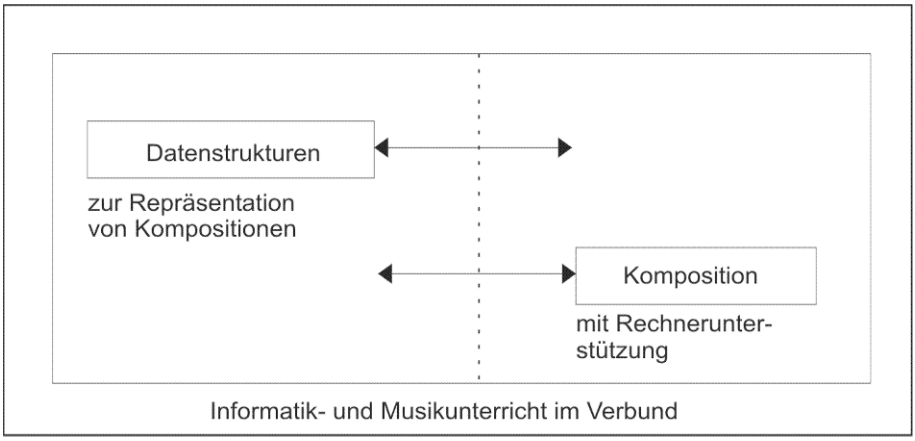


**Abbildung 2.5** Fächerverbindendes Lernen

Dazu sind zwei (oder ein) Lehrer mit passender Fächerkombination und hoher Motivation zu finden („teamteaching"). Die Gleichberechtigung der Informatik ist in diesem Verbund zu sichern, indem Prinzipien und Methoden des Faches zum Unterrichtsgegenstand in angemessener Breite und Tiefe werden. Eine Einschränkung auf Hardware- und Software-Dienstleistungen für andere Fächer widerspricht dem Grundsatz des fächerverbindenden Lernens.

http://www.springer.com/978-3-8274-2652-9

Didaktik der Informatik
Schubert, S.; Schwill, A.
2011, XII, 418 S. 111 Abb., Softcover
ISBN: 978-3-8274-2652-9