Begleitmaterial zur Vorlesung Einführung in die Didaktik der Informatik Wintersemenster 1998/99

Sigrid Schubert
Lehrstuhl Informatik XII
Universität Dortmund

25. März 1999

Inhaltsverzeichnis

1. Motivation und Grundbegriffe

Wenn im folgenden von Schülern, Lehrern, Studenten, Professoren, Informatikern usw. die Rede ist, so sind stets Mädchen und Jungen, Frauen und Männer gemeint. Didaktik bezeichnet die „Lehre vom Lehren und Lernen" bzw. die „Unterrichtslehre" [Drosdowski89], S. 342. Damit wird noch nicht deutlich, warum solche Studien zum Angebot z. B. eines Fachbereichs Informatik gehören. Man würde sie im Fächerkanon der Erziehungswissenschaft (Pädagogik) erwarten. Das bezeichnet Klafki als „bildungstheoretische Didaktik" [Klafki91], S. 9. In der allgemeinen (an kein spezielles Unterichtsfach gebundenen) Didaktik werden z. B. die Begriffe Kenntnisse, Fähigkeiten, Fertigkeiten und „Tugenden wie Selbstdiziplin, Konzentrationsfähikeit, Anstrengungsbereitschaft, Rücksichtnahme" [Klafki91], S. 74 diskutiert, um ihre historische Bedeutung und Entwicklung zu verstehen. Hier werden die „allgemein–didaktischen Prinzipien" [Klafki91], S. 87 ff. ebenso als bekannt vorausgesetzt wie andere Grundlagen von Erziehungswissenschaft und Psychologie, die zum Grundstudium von Lehramtstudiengängen gehören.

Geklärt werden soll, was unter Didaktik der Informatik zu verstehen ist. Jedes Unterrichtsfach hat eine eigene Fachdidaktik hervorgebracht, die Ziele, Inhalte und Methoden für die Lehr–Lern–Prozesse in diesem Fachgebiet begründet ableitet und variiert. Bevor eine geschlossene Theorie für ein Fach entwickelt werden konnte, fanden stets umfangreiche empirische Studien statt, die einzelne Unterrichtsbeispiele zu verbundenen Lehrplansequenzen verdichteten, Vorkenntnisse und Lernfortschritt diskutierten und auf die Besonderheiten verschiedener Lerngruppen eingingen. So entwickelte jede Fachdidaktik eine eigenständige Struktur, die Zusammenhänge einer Wissenschaft auf die Bildungsanforderungen einer Zielgruppe transformiert. Im Ergebnis entstehen Lehrpläne, Bildungskonzepte, Leistungskontrollen und Bildungsbausteine wie Aufgaben, Experimente, Denkmodelle, Anschauungsmaterialien. Sie liegen in Buchform, auf Datenträgern oder auch im Internet vor und werden durch Laborgeräte bei Bedarf ergänzt. Einige Fachdidaktiken gliederten sich in die Bereiche Primarstufe, Sekundarstufe I, Sekundarstufe II, berufliche Bildung, Hochschulbildung, Fort– und Weiterbildung. Andere Fachdidaktiken existieren nur für einen dieser Bereiche.

Hier wird Didaktik der Informatik als Teilgebiet der Wissenschaft Informatik vorgestellt, die alle Lehr–Lern–Prozesse der Informatik erforscht und lehrt mit dem Ziel der berufsqualifizierenden Vorbereitung auf ein Lehramt Informatik. Von einer geschlossenen Theorie kann vorerst nicht ausgegangen werden, aber das heuristische Wissen aus der Bildungstradition seit ca. 1960 sollte nicht unterschätzt werden. In den Landesinstituten der einzelnen Bundesländer und den neu entstandenen Arbeitsgruppen an den Universitäten findet eine systematische Lernprozessforschung statt. Die neuen Informationstechnologieberufe (IT–Berufe wie Informatikkaufmann/–frau, IT–Systemkaufmann/–frau, IT–System–Elektroniker/–in, Fachinformatiker/–in) erhöhten den Bedarf an Didaktik der Informatik für den Bereich der Berufsbildung sehr stark.

Es ist erstaunlich, welche Kluft zwischen der Entwicklung der Wissenschaft Informatik und der zugehörigen Lehrdisziplin in den zurückliegenden Jahren entstanden ist. Während die Fachwissenschaft eine deutliche Gliederung ihrer Teilbereiche in die Kerninformatik (Theoretische, Praktische und Technische Informatik) und in die Angewandte Informatik vornahm, fehlt eine Strukturierung der Lehrdisziplin. Viele Inhalte finden sich unabgestimmt in den verschiedenen Ebenen (Hochschulbildung, Berufsbildung, Allgemeinbildung) der Lehrdisziplin ohne aufeinander Bezug zu nehmen. Diese Unklarheiten führen dazu, dass die Hochschulausbildung nicht auf einem Fundament aufbauen kann, wie das in anderen Unterrichtsfächern üblich ist. Im Gegenteil wird oft die Teilnahme an einem unsystematischen Kurs als schädlich und „verbildend" abgelehnt. Das entstandene „Zerrbild der Wissenschaft Informatik" durch das gleichlautende Schulfach [Nievergelt91] oder die Informationstechnische Grundbildung (ITG) beschäftigt viele Expertengruppen, brachte aber bisher keine durchgängige Gliederung in die Lehrdisziplin. Rechnet man 1960 [CS93], S. 305 als Beginn der Fachwissenschaft, so geht auch der Anfang der Lehrdisziplin auf dem Hochschulniveau auf dieses Jahr zurück. Die EDV–Berufe können seit 1965

auf die Lehrdisziplin in der dualen Berufsbildung zurückblicken. Anfang der 70er Jahre fanden vorbereitende Unterrichtsversuche statt, die mit der Rahmenvereinbarung der Kultusministerkonferenz (KMK) Mitte der 70er Jahre zum Beginn der Lehrdisziplin in der Allgemeinbildung der Sekundarstufe II führten, die in Schüben weiterentwickelt wurde. 1984 und 1987 einigte sich die Bund–Länder–Kommission für Bildungsplanung und Forschungsförderung (BLK) [BLK84], [BLK87] auf ein Konzept zur „Informationstechnischen Bildung (ITB) in Schule und Ausbildung“, in dem erstmals die Informationstechnische Grundbildung (ITG) definiert wurde, die für alle Schüler der Sekundarstufe I im Pflichtbereich unterrichtet werden sollte. Das führte in vielen Bundesländern zu Modellversuchen und zum Anreichern der Lehrdisziplin in der Allgemeinbildung der Sekundarstufe I mit einer Vielzahl empirischer Erfahrungen.
Die 1969 gegründete Gesellschaft für Informatik trat seit 1976 regelmäßig mit Empfehlungen zu den Lehrinhalten der allgemeinbildenden Schulen und der Ausbildung der Lehrkräfte an die Öffentlichkeit:

- *1976 Empfehlungen zum Informatikunterricht in der Sekundarstufe II* [Brauer76]:
  Die Algorithmisierung dominiert, ergänzt vom funktionellen Aufbau einer Rechenanlage und den gesellschaftlichen Auswirkungen. Auch von Projektkursen (Software–Entwicklung) ist bereits die Rede.

- *1979 Empfehlungen zur Ausbildung von Informatik–Lehrkräften* [Claus79]:
  Im Mittelpunkt steht die Praxis des Programmierens bis hin zur Mitarbeit an einem komplexeren Projekt. Aber auch die theoretische Informatik (Automaten, Berechenbarkeit, Komplexität) wird eingeplant.

- *1986 Empfehlungen zum Informatikunterricht in der Sekundarstufe I* [Loos86]:
  Allen Schülern der Sekundarstufe I werden Informatik–Inhalte obligatorisch angeboten. Im Mittelpunkt stehen dabei Verständnis von Lösungsverfahren und deren Realisierung in Programmen, die Arbeitsweise des Computers und die Auswirkung der Informationstechnik auf die Gesellschaft.

- *1987 Empfehlungen zur Ausbildung von Informatik–Lehrkräften* [Arlt87]:
  Eine Informatikgrundausbildung wurde für alle Lehrer konzipiert. Die Ausbildung der Informatiklehrer konzentrierte sich auf den Software–Entwicklungsprozess und forderte die Mitarbeit an zwei Softwareprojekten. Insgesamt ist viel von Anwendungen und nichts mehr von den theoretischen Grundlagen zu lesen.

- *1993 Empfehlungen zum Informatikunterricht in der Sekundarstufe II* [SZ93]:
  Mit dem Ziel, fundiertes Grundwissen über Informatiksysteme zu vermitteln, werden ein breites Spektrum von Problemlösungsansätzen, Programmierparadigmenwechsel, Wissensrepräsentation und –verarbeitung, Parallelverarbeitung und verteilte Systeme berücksichtigt. Ferner werden die Rückwirkungen der Arbeit mit Computern auf den Menschen thematisiert, um Chancen und Risiken von Informatikanwendung für Gesellschaft und Umwelt zu erkennen. Es ist viel von neuen Sichtweisen die Rede, aber der Bezug zur Informatik bleibt oberflächlich. Kritiker sprechen von „technikorientierter Sozialkunde“.

- *1998 Empfehlungen zur Ausbildung von Informatiklehrkräften* (zur Zeit noch nicht veröffentlicht):
  Allgemeine Leitideen sind die Innovationsbereitschaft und –fähigkeit in der informatischen Bildung zu fördern, die Verbindung der fachlichen Bildung mit dem Erziehungsauftrag, die Stärkung des Praxisbezugs, sowie die Erprobung von neuen Lehr– und Lernformen.
  Bei der informatischen Bildung von Informatiklehrkräften können drei Bereiche unterschieden werden:

1. Informatiklehrer sollen so ausgebildet werden, daß das Fach Informatik in den Sekundarstufen I und II an allgemein– und berufsbildenden Schulen unterrichtet werden kann.
2. Alle Lehrkräfte sollen eine Zusatzausbildung für die Befähigung zur Vermittlung der Informationstechnischen Grundlagen (ITG) erhalten.
3. Alle Lehramtsstudenten sollen zum sachgerechten und kreativen Einsatz von Informations– und Kommunikationstechniken sowohl fachorientiert als auch fachübergreifend befähigt werden.

In dieser Vorlesung geht es um den Punkt 1, die Ausbildung von Informatiklehrern. Ziel der Ausbildung muss es sein, Informatik–Lehrern fachliche Kompetenz bei den Grundlagen der theoretischen, der praktischen und der technischen Informatik, softwaretechnische Kenntnisse, sowie Fertigkeiten und Erfahrungen zu vermitteln. Ferner werden ein sicherer Umgang mit ausgewählter Standardsoftware, sowie Kenntnisse über den Einsatz von Informations– und Kommunikationstechnik in wichtigen Anwendungsbereichen gefordert. Informatiklehrer sollen fachdidaktische Kompetenz unter Berücksichtigung eines vertieften Verständnisses der gesellschaftlichen Implikationen von Informations– und Kommunikationstechniken erlangen.

Aufgabe der Didaktik der Informatik ist die Bereitstellung von Kriterien und Vorgehensweisen bei der Beschreibung von Bildungszielen und den zu ihrer Realisierung erforderlichen Lehrinhalten und Lehrmethoden für mögliche Zielgruppen.

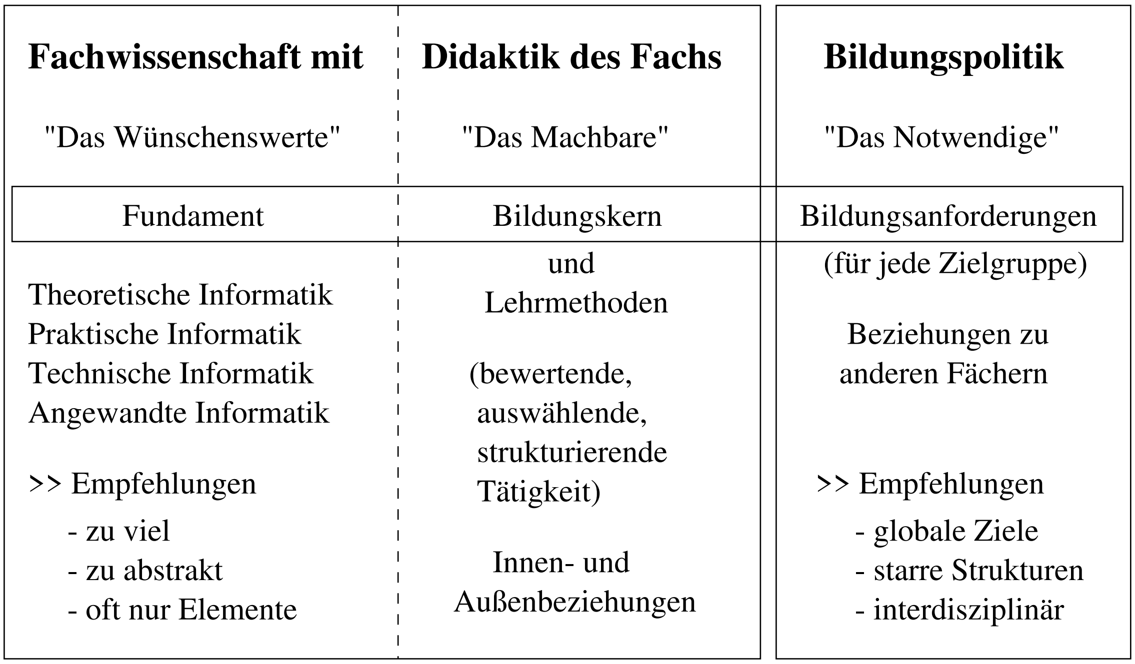


Abbildung 1: Stellung der Fachdidaktik

Aus der Stellung der Fachdidaktik (siehe Abbildung 1) wird das Konfliktpotential deutlich. Sie erhält Empfehlungen von zwei starken Gegenpolen, die sich kaum mit den Möglichkeiten der Lehrdisziplin befassen. Die Empfehlungen der Fachexperten berücksichtigen die Struktur der jeweiligen Wissenschaft und die sachlogischen Beziehungen im Stoff, bewerten aber häufig den Bildungswert ausgewählter Konzepte falsch, da Motivation und Vorkenntnisse der Zielgruppen unbekannt sind. Aktuelle Bildungspolitik leitet die Anforderungen aus empirischen Studien z. B.

in Berufsfeldern mit Prognosemöglichkeit ab, berückichtigt auch Motivation und Vorkenntnisse, bleibt aber häufig bei Empfehlungen stehen, für deren Umsetzung Personal, Mittel und ein geeigneter organisatorischer Rahmen fehlen. In besonderer Weise betrifft das die Didaktik der Informatik, die auf eine „Informationsgesellschaft" vorbereiten soll, ohne im Fächerkanon der Schulen verankert zu sein. Da informatische Bildungsziele und –inhalte ähnlich anspruchsvoll und kompliziert sind wie mathematische, stelle man sich einfach Mathematik als Wahlfach vor. Damit wird die niedrige Kursteilnehmerzahl ebenso verständlich wie der ständige Begründungszwang der engagierten Informatiklehrer zur Bedeutung und zum Bildungswert des Faches. Die Forderung nach einem Pflichtfach in der Sekundarstufe I bleibt bestehen.

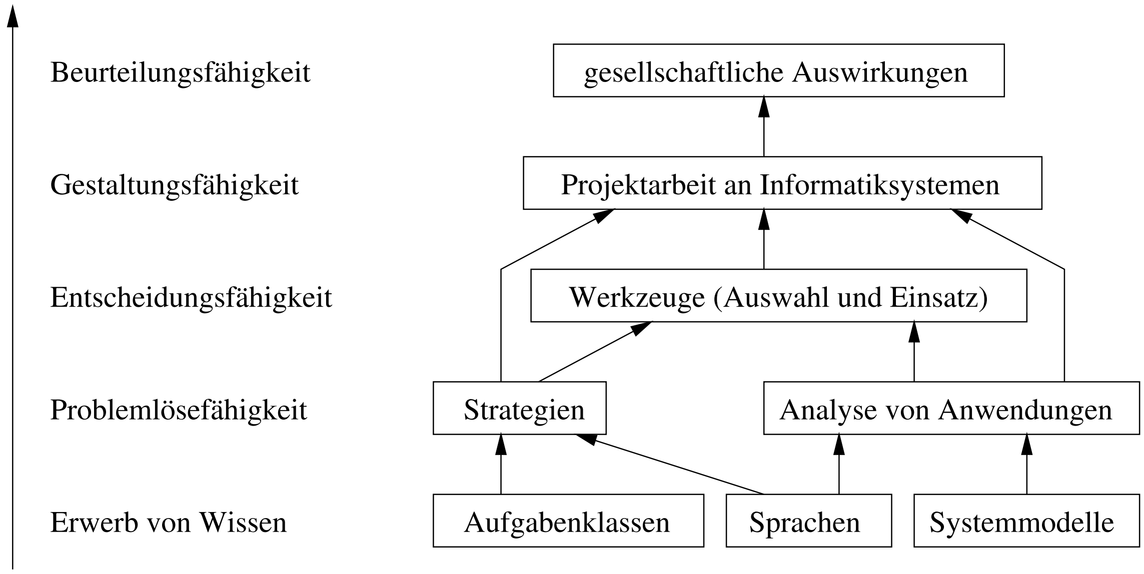


Abbildung 2: Bildungskern und Linienführung für Anfänger

Was kann die Didaktik der Informatik heute leisten? Die Hierarchie menschlicher Fähigkeiten kann so auf die Fachwissenschaft abgebildet werden, dass Bildungskern und Linienführung jeder Informatikausbildung theoretisch begründet werden können (siehe Abbildung 2).

Immer noch findet sich in Lehrplänen bzw. Rahmenrichtlinien die Forderung nach Thematisierung der gesellschaftlichen Auswirkungen der Informatik, ohne dass diese Beurteilungsfähikeit auf einer fachlichen Basis gebildet werden kann. Schon ein grober Überblick über Bildungskern und Linienführung für Anfänger (siehe Abbildung 2) zeigt, welche Minimalanforderungen dafür erfüllt werden müssen. Für den Wissenserwerb als Voraussetzung für Fähigkeiten sind Aufgabenklassen, Sprachkonzepte und Systemodelle wichtig. Das wird im zweiten Kapitel näher begründet. Die Von–Neumann–Architektur ist ein solches Systemmodell, aber auch ein Betriebssystem, ein Datenbanksystem, die Internetarchitektur oder das Intranet einer Schule gehören dazu. Auf Strategien zur Problemlösung wird im vierten Kapitel tiefer eingegangen. Mit Werkzeugauswahl und Systemgestaltung beginnt das zweite Kapitel. Die notwendige Vertiefung bringen das fünfte bis siebente und zehnte Kapitel. Das achte Kapitel möchte die Diskussion um ein noch fehlendes Gesamtkonzept der informatischen Bidlung anregen. Dazu liefert das neunte Kapitel Hintergrundinformationen zum Entwicklungstand dieser Didaktik. Im Anhang finden sich Absprachen für die Schulpraxis und Anregungen für die Selbststudien auf diesem Gebiet.

2. Grundmodell für Ziele, Inhalte und Lehrmethoden

2.1. Informatikprinzipien und –methoden

Eine Vielzahl von Informatikprinzipien und –methoden kann daraufhin untersucht werden, welche Bedeutung sie für eine Informatikgrundausbildung besitzen. Solche Prinzipien sind z.B.

- strukturiertes Programmieren, Modularisierung und Konkretisierung,
- algorithmisches Denken,
- Spezifikation der Anforderungen,
- Denken in Datenstrukturen,
- Idee der abstrakten Datentypen,
- Verifikation der erstellten Software,
- Einsicht in Syntax und Semantik,
- Effizienzsteigerung,
- Rekursion, Parallelität,
- virtuelle Maschinen für konkrete Aufgaben,
- Implementierungstechniken,
- Simulationstechniken und
- Arbeiten im Team (Komplexitätsbewältigung, Schnittstellendefinition, Anpassungs– und Wartungsfragen).

Um hieraus eine sinnvolle Auswahl zu treffen, ist es hilfreich, den Problemlösungsprozess der Informatik näher zu betrachten (siehe Tabelle 1) [Schubert88].

Prinzip	**Methode**
Prozessmodellierung	Zergliedern des Entwicklungsprozesses in Phasen – Spezifikation der Anforderungen – Implementierung des Entwurfs
Einsatz virtueller Maschinen	Arbeiten in Projekten – Verifikation der Lösung – Projektbegleitendes Dokumentieren und Verwalten
Strukturierte Programmierung	Hierarchisches Gliedern Top–Down–Entwurf mit schrittweiser Verfeinerung Einsatz fester Grundstrukturen für Handlungen und Daten Selbstdokumentierende Lösungsbeschreibung
Modularität	Modulprogrammierung – Schnittstellendefinition – Datenabstraktion – Geheimhaltung der Wirkungsweise

Tabelle 1: Informatikprinzipien und –methoden des Problemlösungsprozesses

2.2. Grundausbildung und Vertiefung

2.2.1. Motivation

„Informatik“ sollte nicht ständig neu begründet werden, aber die Entwicklung zum erfolgreichen Schulfach ist fortzusetzen. Dazu sind gute Erfahrungen zu bewahren. Aus erkanntem Mangel folgt die Suche nach Elementen, die den Lehrgegenstand bereichern, ohne ihn zu überladen. Dazu gehören in stärkerem Maße theoretische Grundlagen für praktisches Handeln. Die Techniken der geistigen Arbeit beschränken sich nicht auf die Anwendung einer Programmiersprache. Modelle suchen, deren Objekte und Strukturen bestimmen, ihre Leistungsfähigkeit bewerten, das alles besitzt einen hohen persönlichkeitsbildenden Wert in einer Zeit, in der qualifizierter Umgang mit Informationen den Zugang zu anderen Disziplinen beeinflusst. Informatikunterricht kann zeigen und erlebbar machen, dass zu verschiedenen Problemen die passenden Darstellungsformen für das Ausgangswissen und das abzuleitende Wissen (Lösung) auszuwählen sind, und nicht das Problem zurechtgebogen werden sollte. Der Umgang mit Wissen (Gewinnung, Darstellung, Verarbeitung) und dessen komplizierte Eigenschaften (unvollständig, unsicher, zeitabhängig) wird zur Verbindungslinie zwischen Grundmodellen und aktuellen Anwendungen. Donald E. Knuth sagt dazu [Knuth89]:

> „Was wir also verstehen lernen sollen sind Vorgehensweisen, nicht Softwarepakete! Ein gutes Programmierwerkzeug ist immer ein Modell, ein Muster, keine versiegelte Black Box, deren Inhalt unbekannt ist und bleibt.“

2.2.2. Das Missverständnis Programmierung

Informatikunterricht soll kein Programmierkurs sein. Warum eigentlich nicht? Hier gehen (berechtigte und unberechtigte) Vorwürfe eine interessante Verbindung ein. Problemlösen (Modellieren und Strukturieren) unter Anwendung von Informatikprinzipien und –methoden gilt als erstrebenswert. Die Programmiersprache soll im Hintergrund (Mittel zum Zweck) bleiben. Das aber ist Programmierung (nicht zu verwechseln mit Codierung).
Das Grundmodell der heute erfolgreichen Informatikausbildung beruht auf der Entwicklung guter (strukturierter) Lösungspläne (siehe Abbildung 3) und dem typischen Tätigkeitszyklus (siehe Abbildung 4) [Schubert88], [Schubert91]. Programmierung fördert solche Techniken der geistigen Arbeit (siehe Tabelle 2), die mit heuristischen Methoden von Realitätsausschnitten zu geeigneten Modellierungen, das heißt zu Algorithmen (Strukturen für dynamische Prozesse und Daten) führen. Bildungsziel sollten in stärkerem Maße die heuristischen Methoden der Strukturierung sein und weniger die algorithmischen Ergebnisse (in Form von Software). Software als vergegenständlichte Intelligenz erlaubt das Modellieren und Kombinieren von logischen Grundbausteinen (Sequenz, Zyklus, Alternative) mit ungewöhnlichen Freiheitsgraden. Die Möglichkeit zur Entwicklung von kreativen Problemlösungen ist ebenso gegeben wie die des schablonenhaften Nachvollziehens von Beispielen. Beim Lehrer liegt die Weichenstellung. Der persönlichkeitsbildende Wert der Informatik ist ihre spezifische Weise, Modelle aus Strukturen zu entwickeln und die experimentelle Manipulation mit diesen Modellen (unbewusstes Programmieren [Gorny91]), die dem Lernenden seine Denkfehler aufzeigen. Dazu sind die oben angeführten Verfahrenskenntnisse (siehe Abbildung 3) notwendig. Im Informatikunterricht sind diese Arbeitsweisen als Gegenstand des Lernens bewusst zu machen. Die Gefahr besteht sonst darin, dass Prinzipien und Methoden von der Faktenfülle (Systembesonderheiten, Werkzeugangebote) so überdeckt werden, dass ein Zerrbild entsteht. Tatsächlich kann mit dem Schwerpunkt auf der Programmiersprache ein Kurs entstehen, der bestenfalls vorgezogene Berufsbildung darstellt und die allgemeinbildenden Ziele verfehlt. Solches „Oberflächenwissen“ bringt Lehrer und Schüler in Konflikte, das heißt, dass Mängel im Bereich der Modellierung und Problemlösungsstrategie scheinbar mühelos mit Details der Hard– und Software–Produkte überdeckt werden können. Lehrer werden dann

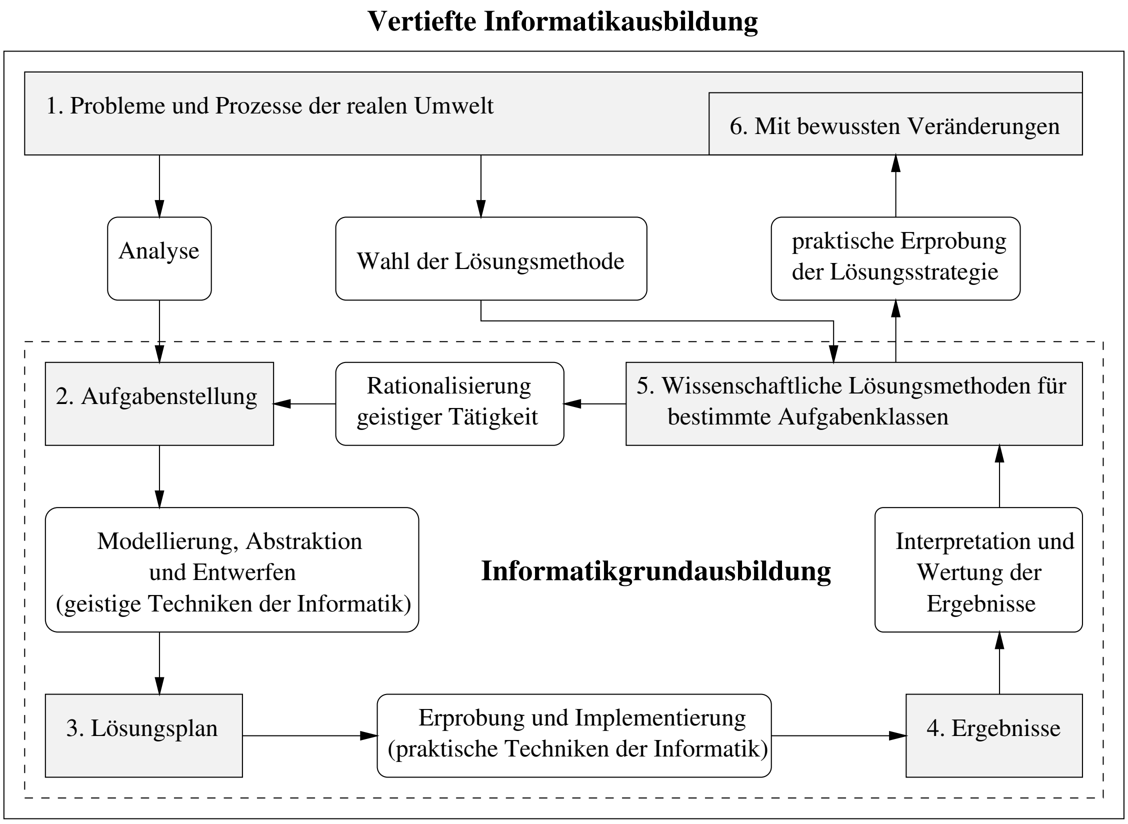


Abbildung 3: Grundmodell

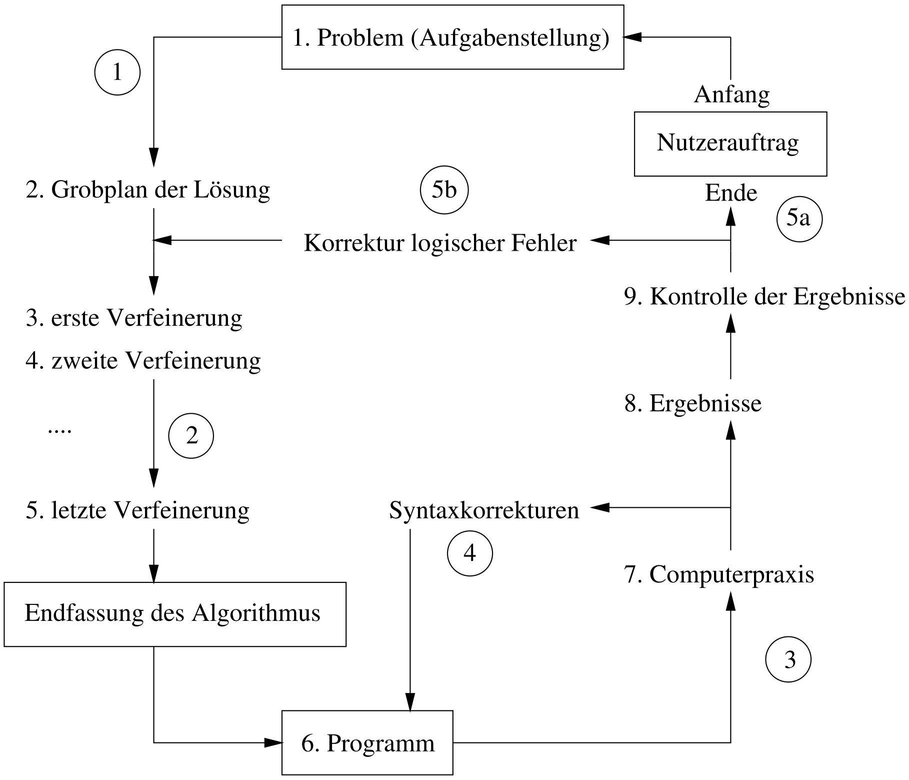


Abbildung 4: Tätigkeitszyklus

	Geistig planende Tätigkeit		Praktische Tätigkeit am Computer
1.	Aufgabenanalyse zur Datengewinnung und –strukturierung	3.	Steuerung des Computers
2.	Lösungsplan durch schrittweise Verfeinerung (Grobstufe, Feinstufen)	4.	Eingabe der Lösung und Korrektur von Formfehlern
	5. Realisierung konkreter Ergebnisse		
5a.	Wertung der Ergebnisse und Auffinden von Planungsfehlern	5b.	Korrektur von Planungsfehlern

Tabelle 2: Kontrollpunkte im Tätigkeitszyklus (siehe auch Abbildung 4)

an Produktkompetenz gemessen, die sie überfordert. Schüler mit viel Spezialwissen stehen dem Informatikgrundwissen (Tiefenwissen) skeptisch gegenüber. Sie können schon so viel. Außerdem hat sich gezeigt, dass die Programmiersprache doch nicht so nebensächlich ist. Sie steckt den Modellierungsspielraum ab [Löthe88]. Selbst bei sorgfältiger Auswahl bleiben Erfahrungen der Art, dass zu jeder Aufgabenklasse die besonders geeignete Sprache eingesetzt werden soll, auf der Strecke. Es geht deshalb nicht so sehr darum, etwas ganz anderes im Informatikunterricht zu lehren, sondern auf den guten Lehrerfahrungen aufzubauen und Schwächen zu überwinden. Dazu gehört der Mangel an theoretischen Grundlagen für die Schulinformatik. Es wird mit erstaunlicher Naivität etwas für praktisch gehalten, nur weil es die Theorie meidet.

2.2.3. Beständigkeit durch Theorie

Worin besteht die Theorie der Informatik? Einen Zugang bildet die Verbindung von Algorithmen, Sprachen und Maschinenmodellen (siehe Tabelle 3), die in drei Ebenen der Leistungsfähigkeit diskutiert werden kann, um die prinzipiellen Möglichkeiten der Informatikanwendung abzustecken und Verifikationsmethoden vorzubereiten.
Für diesen abstrakten Lehrgegenstand bietet sich der Computer als Unterrichtsmittel förmlich an, der mit seinen Geschwindigkeits– und Grafikmerkmalen die kognitiven Grenzen traditioneller Lehrprozesse verschieben kann. Unklar ist noch, wie die Theorie in den Informatikunterricht einzubeziehen ist. Als Leistungskurs kommt sie nach der Grundausbildung zu spät und muss überflüssig (weil lange Zeit entbehrlich) wirken. Die Schüler sollten sie zur Absicherung ihrer Problemlöseideen nutzen können und im Anfangsstadium ihrer Ausbildung bereits gezeigt bekommen, dass auch einfach beschreibbare Probleme am Computer nicht gelöst werden können. Parallel zum eigenen Anwenden der Informatik sind die Grenzen des Machbaren theoretisch zu begründen. Aus intuitivem Wissen sollte Klarheit über die Beschränkungen in Raum und Zeit werden. Wie kompliziert eine Realisierung ist, kann der Theorie entnommen werden. Der zentrale Begriff Sprachklasse kommt aus der Einengung auf eine Programmiersprache heraus [Claus90]. Programmiersprachen wiederum werden in ihrer Entwicklung und ihrem Aufbau transparent. Nichtdeterministische Denkweisen bereichern die Lösungssuche. Offen ist das Problem, wie man für den Umgang mit Modellen der Theorie kognitives Verständnis entwickeln kann, ohne eine Vielzahl von konstruktiven Beweisen und Transformationsmechanismen mit den Schülern wiederholt nachvollziehen zu müssen. Die didaktische Vereinfachung für die Klassenstufen 9 und 10 bildet die Voraussetzung für einen von der Theorie begleiteten Informatikunterricht. Gute fachdidaktische Ansätze findet man bei Minipascalprogrammen [Stetter88] und der Verbindung von Maschinenmodellen und Programmierung [GLSS92]. Zu jeder Sprachklasse (siehe Tabelle 3) können interessante Anwendungen, wie z.B. Texteditoren, neuronale Netze, Programmier-

Ziele

Algorithmen	**Sprachen**	**Maschinenmodelle**
Was wird gemacht?	Wie kann man das aufschreiben?	Wie funktionieren die verarbeitenden Geräte?
	obere Ebene	
Nichtentscheidbarkeit Akzeptanz Entscheidbarkeit	Rekursiv aufzählbare Sprachen	Turing–Maschinen
	mittlere Ebene	
Äquivalenz von KfG und KA	Kontextfreie Sprachen Kontextfreie Grammatiken (KfG)	Kellerautomaten (KA)
	untere Ebene	
Äquivalenz von RA, NEA und DEA	Reguläre Sprachen (Mengen) Reguläre Ausdrücke (RA)	Endliche Automaten (EA) Nichtdeterministische EA (NEA) Deterministische EA (DEA)

Tabelle 3: Theorie

	Modell	**Anwendung**
1.	Reguläre Ausdrücke und Endliche Automaten	• neuronale Netze • Schaltkreise • Compiler (lexikalischer Analyser) • Texteditoren • Mustererkennung
2.	Kontextfreie Grammatik und Kellerautomaten	• Spezifikation von Programmiersprachen • Compiler (Entwicklung von Parsern)
3.	Allgemeine Grammatiken und Turing–Maschinen	• Nachweis der Existenz von nichtberechenbaren Funktionen • Ermitteln der Zeit– und Raumkomplexität

Tabelle 4: Anwendungsbeispiele

sprachen und deren Compiler, in die Unterrichtsdiskussion einbezogen und deren Möglichkeiten und Grenzen durch die Eigenschaften der zugrunde liegenden Modelle verstanden werden (siehe Tabelle 4). Zum Beispiel kann die Substitution von Zeichenketten mit Texteditoren auf das Modell des nichtdeterministischen endlichen Automaten zurückgeführt werden. Die Bedeutung des Nichtdeterminismus (Sprachen, Berechenbarkeit) wurde bisher im Informatikunterricht zu wenig beachtet.

2.2.4. Logik als Werkzeug

Traditioneller Informatikunterricht zeigt den Nachholbedarf bei der Entwicklung des logischen Denkens. Bedingungen, sowohl die vorauszusetzenden als auch die resultierenden, werden unzureichend erkannt. Daran scheitert oft das Planen von Handlungen, die sich auf bestimmte Objekte beziehen. Mit Prolog als „virtueller Maschine“ lernen die Schüler ein System kennen, das für sie ungewohnte Eigenschaften und Möglichkeiten hat. Sie erfahren, welche Problemsituationen damit in Angriff genommen werden können und was die Maschine bewirkt. Die logische Programmierung stellt eine Problemlösephilosophie bis hin zu effektiven Werkzeugen bereit, die es den Schülern ermöglicht, Objekte und deren Eigenschaften zu verknüpfen und in der Muttersprache aufzuschreiben. Sie unterliegen kaum syntaktischen Restriktionen, wohl aber streng logischen. Sie definieren Relationen zwischen Objekten eines Problemraumes, lernen Fakten und Regeln aufzustellen und ergänzen mit ihrem Wissen über das Problem (Prolog–Programm) den vorgefertigten Lösungssuche–Algorithmus, den das Prolog–System automatisch zur Nutzung bereitstellt. Die Schüler müssen sich weniger als in der Pascalwelt um Abläufe, deren Konstruktion und Steuerung, kümmern. Sie können sich in stärkerem Maße auf die Beschreibung des Problems konzentrieren. Dadurch können sie relativ schnell und ausdrucksstark komplizierte Probleme bewältigen. Sie müssen nicht gleichzeitig mit der Komplexität der Lösung und der Komplexität der Herstellung der Software kämpfen. Logische Programmierung fördert eine Vielzahl von Techniken der Modellierung und Konstruktion von Lösungen (siehe Tabelle 5), die zur Prädikatenlogik der 1. Stufe gehören, und führt zur Wissensverabeitung. Die Schüler lernen verstehen, dass das Prolog–System ihre Aussagen nicht überprüfen kann, sondern kritiklos verarbeitet. Falsche Eintragungen im Prolog–Programm führen zu falschen Lösungsvorschlägen des Systems, die durch logisch korrekte Schlussfolgerungen entstehen. Solche anspruchsvollen Methoden wie die Rekursion sind mit Prolog sehr einfach zu realisieren, da der strukturierte Term ebenfalls rekursiv definiert ist. Die Anwendung von Listen und Bäumen führt deshalb zu eleganten und kurzen Lösungen. Schwierigkeiten gibt es beim Verständnis für die logische Va-

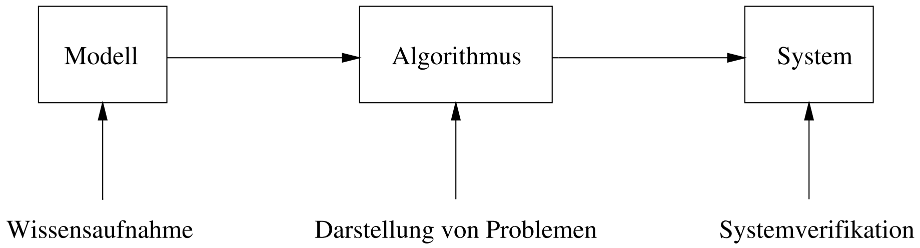
Problemstellung	**Prolog**
Universum von Objekten Eigenschaften der Objekte	Faktenbasis
Beziehungen zwischen den Objekten Prädikate (Relationen) Nebenbedingungen (Constraints)	Regeln
Verkleinern des Suchraums Vollständige Durchmusterung	Suchalgorithmen Tiefensuche mit Backtracking
Implikation (Modus Ponens)	Ableitungspfade
Logische Formeln	Variablenbindung

Tabelle 5: Logische Programmierung

riable. Die Techniken der Unifikation und des Backtrackings besitzen Leistungsstärke, sind aber unbedingt durch die Diskussion von Modellen (Prozedurmodell, Ableitungsbäume) transparent zu machen. Erst wenn die Schüler den Beweisverlauf (virtuelle Maschine) verstanden haben, können sie ihre Wissensbasis wirkungsvoll strukturieren. Den für Anfänger häufig langwierigen Prozess von der Aneignung der Grundkenntnisse bis zu eigenständigen Programmen kann der Lehrer eindrucksvoll abkürzen. Die Verbindung zur Theorie ergibt sich bei den Problemen der Terminierung (links–rekursive Regeln, Zyklen) und der Heuristik über die Reihenfolge der Ziele (Plazierung von Tests und Terminierungsbedingungen). Der Ableitungsverlauf sollte mit der Tracekomponente (evtl. graphisches Trace dazu) experimentell erkundet werden. Dabei liegt mit der Anzahl der Reduktionsschritte in der Ableitung ein Maß für die Lösungskomplexität vor, das die Schüler gut nachvollziehen können. Es sollte davon abgegangen werden, die Programmiersprachen gegeneinander zu stellen. Die Schüler benötigen verschiedene Sprachwerkzeuge, um deren Vorzüge und Nachteile in Abhängigkeit vom zu lösenden Problem zu verstehen. Offen ist die Frage, in welchem Umfang (Breite und Tiefe) man über alternative Sprachkonzepte aufklären muss, um exemplarisches Verständnis zu entwickeln. Auf Fertigkeiten wird man weitgehend verzichten müssen. Die Einsichten dominieren. Die motivierende Wirkung, die aus der Erprobung eigener Lösungen und dem Experimentieren am Computer resultiert, darf nicht verlorengehen.

2.2.5. Techniken der geistigen Arbeit

Die Linienführung „Modellieren und Strukturieren" hat sich im Informatikunterricht bewährt (siehe Abbildung 5). Das komplizierte Wechselspiel von Modell, Algorithmus und System er-



Lösungsansätze:
- Erzeugen von Lösungshypothesen und deren Testen
- Zustandsgraphen, Zerlegung in Teilproblemgraphen
- Beschreibung und Formalisierung einer Heuristik

Möglichkeiten:
- natürlicher Zugang zu Datenstrukturen
- Problemlösen: Algorithmen, die im Zustandsraum die Lösungssuche realisieren
- Abhängigkeit zwischen Wahl der Datenstruktur und Reduzierung der Problemkomplexität
- Verständnis für die konzeptionellen Schichten moderner Informatikanwendungen
- Fachdidaktik organisiert das "menschliche Fenster" (Komplexitätsbewältigung)

Abbildung 5: Modellieren und Strukturieren

fordert jedoch neue Techniken der geistigen Arbeit für die Wissensaufnahme, die Darstellung von Problemen und die Systemverifikation. Stark vernachlässigt wurde bisher das Konzept des Nichtdeterminismus. Mit der Theorie der Informatik kann man diese Denkweise entwickeln. Sie sollte dann bei der Diskussion und Konstruktion von Algorithmen eine strategische Rolle spielen. Die Vorgehensweise „Erzeugen von möglichen Lösungshypothesen und deren Testen" (z. B. Vier–Farben–Problem) ist eine Strategie, bei der nichtdeterministisches Denken zu interessanten Lösungen führt [SS86]. Zustandsgraphen, die Zerlegung in Teilproblemgraphen, die Beschreibung und Formalisierung einer Heuristik bilden einen natürlichen Zugang zu Datenstrukturen, die durchaus noch nicht programmiersprachengerecht sein müssen. Problemlösen heißt dann, Algorithmen kennenzulernen, die im Zustandraum die Lösungssuche realisieren. Die Abhängigkeit zwischen der Wahl der Datenstruktur und der Reduzierung der Problemkomplexität kann auf diese Weise sehr gut veranschaulicht werden (siehe hier zu zwei verschiedene Modellierungen des 8–Damen–Problems, Tabelle 6). Strategien für die Organisation der Suche ermöglichen

	1. Möglichkeit	**2. Möglichkeit**
Darstellung des Schachbretts	Tabelle brett(i,j)	Vektor zeile(i)=j mit Bedeutung: Dame in der j–ten Diagonalen Vorteil: nur noch Diagonalen und Vertikalen testen
Aufwand	2^{64} Zustände	2^{27} Zustände
Lösung	Beliebige Konfiguration auswählen, testen und ändern, wenn kein Erfolg erzielt wurde	<u>induktiv vorgehen:</u> • erste Dame korrekt platzieren • Absuchen des Bretts nach freien Plätzen <u>Vorteile:</u> • Problemlösung mit Constraints • Brett reduziert • fast lineare Suche

Tabelle 6: Zwei Modellierungen des 8–Damen–Problems

die Diskussion darüber, welche Sprachen und Werkzeuge für ein Problem besonders geeignet sind. Diese Algorithmen könnten die Sortierverfahren aus dem Informatikunterricht verdrängen. Damit wird die fachdidaktische Frage, wie man Verständnis für die konzeptionellen Schichten moderner Informatikanwendungen gewinnt, ohne deren Entwicklung nachvollziehen zu müssen, wieder auf Grundalgorithmen und Datenstrukturen zurückgeführt ohne die enge Bindung an die Programmiersprache.
Die Gefahr der Einengung von Kurszielen auf den Computer und seine Bedienung besteht, da die komplexen Modellhintergründe zur Undurchsichtigkeit der Systeme führen. Fachdidaktik stellt sich die Aufgabe, das „menschliche Fenster“ zu organisieren, das heißt die Transparenz komplizierter Daten– und Algorithmenstrukturen so zu erhöhen, dass menschliche Verantwortung tatsächlich wahrgenommen werden kann. Dabei steht die Komplexitätsbewältigung [Claus91] im Vordergrund. Moderne Benutzungsoberflächen lösen dieses Bildungsproblem nicht.

Das Schulfach Informatik kann vom aktuellen Hard– und Software–Markt Abstand gewinnen, indem es sich an den Erfahrungen solcher Fächer wie Mathematik und Physik orientiert. Informatik besitzt eigene Mittel und Methoden für das Modellieren von Problemlösungen und spezifische Möglichkeiten für anspruchsvolle Experimente. Diese Technologie der geistigen Arbeit wiederum beeinflusst zunehmend mehr Fachdisziplinen, das heißt die pädagogische Doppelfunktion der Informatik (im Fach Informatik zugleich für andere Fächer lernen) verstärkt sich noch.
In der kurzen Geschichte des Schulfaches Informatik hat sich der Schwerpunkt von den Systemkenntnissen (viele Fakten) auf die Verfahrenskenntnisse (Methoden für Modellierung, Strukturierung, Erprobung) verlagert. Von exemplarischen Anwendungen kann der Weg zur theoretischen

Untersetzung von Modellausschnitten führen, an denen didaktisch vereinfacht, das Wesentliche komplizierter Algorithmen und Datenstrukturen erklärbar wird. Nicht die Anwendungsarten, sondern die Strukturtypen ermöglichen zukunftssicheres Allgemeinwissen. Zum Beispiel Umweltprobleme sind durch solch hohe Komplexität gekennzeichnet. 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. Nach der Einführung in Prolog 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 versuchen sie die Bedingungen zu verunschärfen, d. h. mit schwächeren Forderungen an bestimmte Eigenschaften eine Kompromisslösung zu erreichen.

2.2.6. Der Umgang mit Wissen

In vielen Bereichen fehlt gut strukturiertes Wissen für die vollständig algorithmische Lösung von Aufgaben (z. B. in der Diagnostik). Das Informatikgrundwissen der Schüler reicht nicht aus, um den Einsatz des Computers für solche Modellierungen zu verstehen. Obwohl sie täglich mit Wissen zu tun haben, wissen sie zu wenig über dessen komplizierte Eigenschaften (unvollständig, unsicher, zeitabhängig). Besonders interessant scheint für die Allgemeinbildung die folgende spiralförmige Entwicklung zu sein, die im Bereich der Wissensverarbeitung beobachtet wurde:

1. Zuerst liegen Aufgaben vor, die vollständig intuitiv, also heuristisch, gelöst werden.

2. Mit dem Eindringen in die Probleme entsteht ein gesichertes theoretisches Umfeld.

3. Daraus werden neue Heuristiken auf höherer Ebene abgeleitet.

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 bilden dann die Datenstruktur. Die Reihenfolge, in der die Regeln verwendet werden, führt zur Problemlösung. Damit werden Methoden der Wissensrepräsentation ein Denkwerkzeug, 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 Abstraktionstufe (Lernen über das Lernen) fördert. Damit wird die These gestützt, dass Wissenserwerb, Wissensverarbeitung und die Methoden des Schlussfolgerns kognitiven Wert in sich selbst tragen [Gorny88].
Problemlösung führt die Schüler zu Techniken der Regelauswahl und –verarbeitung. Die Vorwärts– bzw. Rückwärtsverkettung sind Strategien, die universell (in vielen Disziplinen) einsetzbar sind, deren Spezifik zum Unterrichtsthema wird. Die Diskussion darüber, wie Konfliktbeseitigung modelliert werden kann (Varianten), führt wieder zu den Suchtechniken (Zustandsgraphen). Die Algorithmen und Datenstrukturen für die Wissensverarbeitung zeigen, dass diese Grundlinie des Informatikunterrichts weiterentwickelbar ist. Sie ermöglicht die Behandlung solcher 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 (Distanz des Menschen wegnehmen [Peschke91]). Erklärungsalgorithmen sind heute noch sehr unvollkommen. Sie können ein Inferenzprotokoll anbieten („Wie–Erklärungen“), das den Schülern zeigt, welche Regeln mit welchen Nutzereingaben zu dem aktuellen Ergebnis

führten. Das wird sehr schnell unübersichtlich.
Bei Anforderung einer Eingabe können die Schüler eine „Warum–Erklärung“ erhalten, das heißt, dass die augenblicklich verwendete Regel und die bereits erfüllten Prämissen angezeigt werden. Unklar bleibt 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 eines solchen Mensch–System–Dialoges erlaubt es den Schülern, tiefe Einsichten in Missverständnisse, Fehlhandlungen und deren Ursachen zu gewinnen. Die Übertragung des menschlichen Lernens aus Beispielen und aus Fehlern auf maschinelles Lernen kann an kleinen wissensbasierten Systemen diskutiert werden. Lernalgorithmen werden entmystifizierbar. Kriterien für Software–Qualität erhalten ein fachliches Fundament. Die Illusion vom naiven Benutzer, der mit Expertensystemen komplizierte Aufgaben löst, kann abgebaut werden zugunsten des Leitbildes vom kompetent entscheidenden Fachmann, der um die Leistungsgrenze des Modells weiß, Varianten prüfen und beurteilen kann.

2.3. Typische Unterrichtssituationen

Folgt man dem Grundmodell für die Informatikausbildung, dann ergeben sich folgende typischen Unterrichtssituationen, die der Lehrer zu organisieren hat:

1. Die Möglichkeiten des Computereinsatzes in einem Problemfeld sind zu prüfen und zu beurteilen.
2. Es ist eine konkrete, realisierbare Aufgabenstellung aus einem Problemfeld abzuleiten und zu beschreiben als Auftakt der Dokumentation, die alle Schritte begleitet.
3. Die Analyse der Aufgabenstellung ist bis zur Datengewinnung und –strukturierung und zur Bestimmung der Aufgabenklasse zu führen.
4. Die Modellierung des Lösungsprinzips erfolgt bis zur Grobstruktur der Handlungen unter Benutzung der Grundalgorithmen, die in der Aufgabenklasse enthalten sind.
5. Der Lösungsplan wird durch schrittweise Verfeinerung von der Grobstufe bis zur letzten Feinstufe unter Anwendung der Strukturelemente entwickelt.
6. Der Logiktest wird zur Überprüfung des Lösungsplanes eingesetzt.
7. Lösungsplan und Datenstrukturen werden in die Programmiersprache übertragen.
8. Implementierung und Erprobung des Programms am Computersystem führen mit den Korrekturschritten bis zur Erzeugung korrekter Ergebnisse, die interpretiert und bewertet werden müssen.
9. Die Computerlösung ist insgesamt zu beurteilen und zu verbessern.
10. Die Dokumentation ist abzuschließen, und das Softwareprodukt zur Nutzung zu übergeben.

Dabei ist zu beachten, dass 1., 2., 9. und 10. der Vertiefung zuzuordnen sind, die die Schüler meist nur einmal als Abschluss ihrer Ausbildung kennenlernen und absolvieren. Die Grundausbildung dagegen wiederholt den Abschnitt 3. bis 8. mehrmals, um eine gewisse Sicherheit im elementaren Wissen und Können zu erzielen.

Linie "Problemlösen mit Mitteln und Methoden der Informatik"

4. Stufe	***Schwerpunkte***
Komplexere Übungen Dateiverarbeitung Rechnerarchitekturen und Netze	Programmierheuristik Datensicherheit und Datenschutz Einzelcomputer und Verbund
3. Stufe	
Unterprogrammtechnik Datenstrukturen Verbund, Datei Betriebssystem	Lösungsübertragung Mehrfachnutzung von Bausteinen Dienstleistungen, Prozesse
2. Stufe	
Alternativen, Iterationszyklen Datenstruktur Feld Computermodell	Wertung der Ergebnisse Korrekturmöglichkeiten Aufbau und Wirkungsweise
1. Stufe	
Sequenz, Zählzyklus, Datentypen Prinzip der Computernutzung	Problemanalyse, Implementierung Einführung in die Entwurfsstrategie

Abbildung 6: Leitlinie für den Informatikunterricht

3. Theoretische Fundierung der Schulinformatik

3.1. Motivation

Lehrer mit langjähriger Erfahrung zum Informatikunterricht in der Sekundarstufe I und II klagen über die fehlenden zeitbeständigen Unterrichtsinhalte [Peschke89]. Diese Beständigkeit kann durch stärkere Orientierung an den theoretischen Grundlagen erzielt werden [Schubert91]. Deshalb möchte ich einen Weg aus der unbefriedigenden Situation weisen und für mehr Theorieverständnis plädieren. Tatsächlich ist die Mehrzahl der existierenden Lehrpläne so interpretierbar, dass die theoretischen Grundlagen der Informatik sehr vernachlässigt werden. Als extremes Beispiel sei der sächsische Lehrplan angeführt, der für den Abschluss der Klasse 12 [SMK92a] plötzlich die Behandlung des Halteproblems vorsieht, um die Möglichkeiten und Grenzen der Informatik darzustellen. Für die Schüler entsteht nach vier Kurshalbjahren so kurz vor dem Abitur der Eindruck, dass dieses Wissen bisher offenbar nicht notwendig war, also auch insgesamt keine wesentliche Rolle spielen kann. Gerne werden Zeitgründe von den Lehrern angegeben, die sie auf diesen komplizierten Lehrabschnitt verzichten lassen. Die wahren Gründe liegen in der unzureichenden Vorbereitung der Lehrer auf das Unterrichten solcher theoretischer Grundlagen [Claussen93b] und in der gerade erst begonnenen Transformation der Fachwissenschaft in die zugehörige Lehrdisziplin Informatik, die eine wissenschaftliche Strukturierung der Schulinformatik ermöglicht (siehe Abbildung 7) [Schubert95b]. Das Lehrgebiet Theoretische Informatik hat bei

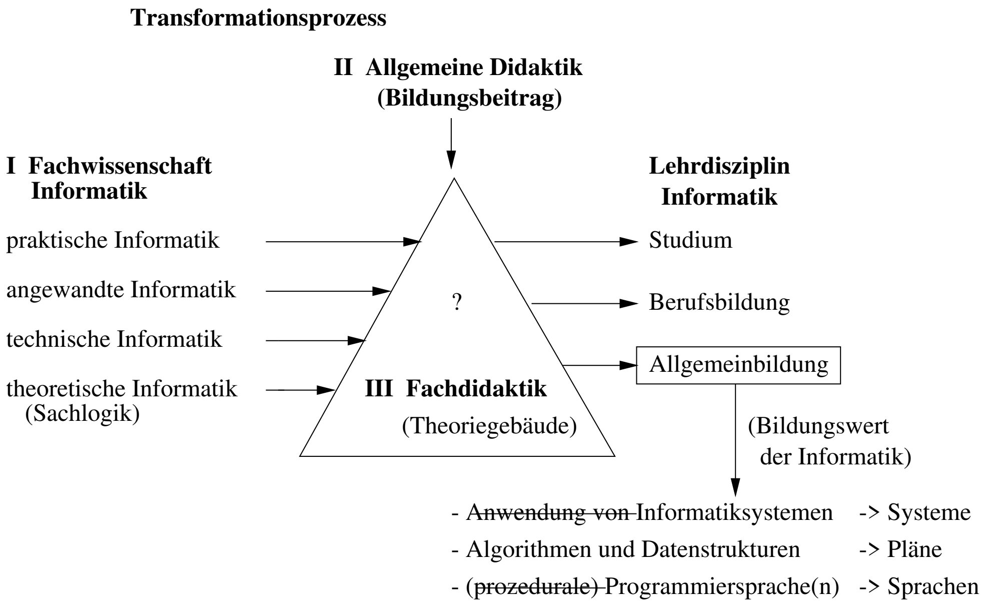


Abbildung 7: Rolle der Fachdidaktik

den Lehramtsstudenten den Ruf, besonders kompliziert zu sein. Es ist deshalb notwendig, die Nützlichkeit dieses Faches für den Beruf des Informatiklehrers aufzuzeigen, um Motivation und Ausdauer für die Beschäftigung mit diesem anspruchsvollem Inhalt zu entwickeln. Dazu können viele Verweise auf andere Lehrgebiete genutzt werden.

Das Schulfach Informatik leistet einen wichtigen Beitrag zur Allgemeinbildung. Zur Begründung wird Klafki [Klafki91], S. 49ff. herangezogen, der die Ziele der Allgemeinbildung diskutiert:

- als Bildung für alle zur Selbstbestimmungs–, Mitbestimmungs– und Solidaritätsfähigkeit,
- als kritische Auseinandersetzung mit einem neu zu durchdenkenden Gefüge des Allgemeinen als des uns alle Angehenden,
- als Bildung aller uns heute erkennbaren humanen Fähigkeitsdimensionen des Menschen.

In der Informationsgesellschaft kann die persönliche Verantwortung nur in Verbindung mit Systemtransparenz wirklich wahrgenommen werden. Diese Systeme der Informatik sind aber nicht selbsterklärend, wie häufig irrtümlich behauptet wird. Eine Informatikgrundbildung muss deshalb dieses Systemverständnis schrittweise entwickeln. Außerdem findet mit der zunehmenden Anwendung der Informations– und Kommunikationssysteme eine deutliche Veränderung der menschlichen Kommunikation (z. B. durch Telekommunikation im Intranet oder Internet) statt. Die unvorbereitete Teilnahme daran birgt persönliche Risiken, da Basismechanismen der Informationssicherheit nicht beherrscht werden. Elektronische Zahlungsmittel und digitale Signatur bringen den Benutzer in Rechtsbeziehungen, denen er nicht ausgeliefert sein darf. Informatikgrundbildung eröffnet neue Perspektiven für lebenslanges Lernen [Schubert97] mit den Vorzügen der Flexibilität bezüglich Ort und Zeit.

In den Lehrplandiskussionen der vergangenen zehn Jahre spielte die Beschreibung des Beitrages der Informatik zur Allgemeinbildung von Bussmann und Heymann [BH87] eine wichtige Rolle, wobei ihre Schwerpunkte mit konkreten Informatikinhalten verknüpft wurden:

- Vorbereitung auf zukünftige Lebenssituationen,
- Stiftung kultureller Kohärenz,
- Aufbau eines Weltbildes,
- Anleitung zum kritischen Vernunftgebrauch,
- Entfaltung eines verantwortlichen Umgangs mit den zu erwerbenden Kompetenzen (Verantwortungs–Postulat),
- Stärkung der Schüler-Ichs.

Die Veränderung von Berufsbildern und die historische Entwicklung der Informatik wurden im Unterricht stärker thematisiert. Die Forderung nach Systemtransparenz und Gestaltungskompetenz rückte den Softwareentwicklungsprozess in den Mittelpunkt des Informatikgrundkurses. Der breite Interpretationsspielraum der globalen Ziele brachte sehr unterschiedliche Lehrplankonzepte hervor (siehe Kapitel 9). Der ursprüngliche Rahmen von „Algorithmen und Datenstrukturen“ erschien vielen Lehrern zu eng geworden. Der Suchprozess nach geeigneten Strukturelementen des Informatikunterrichts wurde durch Peschke [Peschke89] eingeleitet und hält bis heute an.

3.2. Berufliche Anforderungen

Die beruflichen Anforderungen, die sachlogischen Zusammenhänge des Faches und dessen typische Tätigkeitsstrukturen sowie die Erfordernisse eines Studienprozesses stellen Auswahlprinzipien mit Filterfunktion für den Lehrstoff und die Bestimmung des Grades seiner Beherrschung dar.
Der Lehrstoff lässt sich in Komponenten gliedern:

- Definitionen, Sätze, Beweisideen,
- Verfahren (Algorithmen und Heuristiken),
- Bewertungskriterien,
- Modelle,
- Strukturen und
- Perspektiven.

Jede didaktisch gültige Auswahl aus diesem Spektrum liefert die erforderliche Fachkompetenz einer bestimmten Zielgruppe.
Der Grad der Aneignung kann unterteilt werden in Werte wie informiert sein, kennen, beherrschen, weiterentwickeln können. Daraus ergibt sich der Aktionsradius des Lernenden, der zeigt, wie er mit dem erworbenen Wissen und Können umgehen kann. Zu dieser Fachkompetenz gehören außerdem Methodenkompetenz (z.B. allgemeine Problemlösefähigkeit, Metawissen über das Lernen) [Anderson96] und Sozialkompetenz (z.B. allgemeine Kommunikationsfähigkeit, Orientierung an Werten und Normen). Die beruflichen Anforderungen ermöglichen die Beantwortung der Frage: „Was benötigt man von der Theorie in der Praxis?" Folgendes Vorgehen hat sich bewährt. Den Ausgangspunkt bildet eine Prognose (Analyse) der zukünftigen Anforderungen an Schüler, die dann zu den Zielen der Allgemeinbildung zusammengefasst werden (Synthese) [Klafki91], [BH87]. Daraus wiederum und aus den Möglichkeiten der Wissenschaft Informatik kann der Beitrag des Schulfaches Informatik abgeleitet werden. Dessen Ziele, Inhalte und Lehrmethoden lassen die klare Abgrenzung der beruflichen Anforderungen an Informatiklehrer zu. Aber hier stockt der Entwicklungsprozess, da die Transformation der Wissenschaftsdisziplin Informatik (Fach) in die Schulinformatik (Schulfach) erst allmählich Konturen zeigt. Deshalb wird im dritten Abschnitt darauf begründend eingegangen. An dieser Stelle wird aus dem Vergleich mit anderen Schulfächern die These gewonnen, dass das Beherrschen der theoretischen Grundlagen, das Kennen von Anwendungsbeispielen und das Weiterentwickeln der Schulinformatik bei den beruflichen Anforderungen im Vordergrund stehen. Dabei bildet die Verbindung der theoretischen Grundlagen mit erläuternden Anwendungsbeispielen den Schwerpunkt der Lehrtätigkeit. Damit wird auch verständlich, warum Schulwissen die größere Beständigkeit besitzt im Vergleich zum Hochschulwissen und zum beruflichen Fachwissen. Nach diesen analysierenden Vorüberlegungen wird es möglich, die Hierarchie der Grobziele für die Lehrerausbildung zu bestimmen.
Die Lehrbefähigung resultiert aus:

Fachwissenschaft	und	Fachdidaktik
mit dem Ziel		mit dem Ziel
der **Transparenz von Informatikprozessen**		der Entwicklung von
basierend auf		**Lehrsequenzen** und
Plänen, Sprachen, Systemen		**Lehrbeispielen**.

Dabei führt die inhaltliche Linie von elementaren Lösungsbausteinen bis zu berufsspezifischen Problemlösungen. Für den Studienprozess ist die selbständige Anwendung der wissenschaftlichen Methoden charakteristisch.

3.3. Ansätze

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 Pascalprogramme zum Mittelpunkt des Schulfaches erklärt [KMK91]. Einen Ausweg aus dieser Sackgasse zeigte Schwill [Schwill93], indem er exemplarisch aus dem Software–Entwicklungsprozess die fundamentalen Ideen begründet ableitete und deren Lehrbarkeit auf jedem Schulniveau zeigte. Dahinter verbirgt sich die Hypothese, dass die Software–Entwicklung das allgemeinbildende Kernstück der Fachwissenschaft Informatik darstellt und deshalb zur Strukturierung des gleichnamigen Schulfaches geeignet ist. Ein solcher Nachweis wurde bisher nicht erbracht.
Einen möglichen Zugang zur theoretischen Fundierung der Schulinformatik bildet der neue Typ des Aufgabenlösens (Arbeitsteilung zwischen Mensch und Informatiksystem). Dabei erhalten Lösungsplanung und Mensch–Maschine–Interaktion (mit Darstellungs–, Bedeutungs– und Handlungsebene) einen deutlich höheren Stellenwert, da die traditionelle Ausführung automatisierbar wird. Der Mensch übernimmt es, Systeme, Pläne und Parameter problemgerecht auszuwählen

und die Lösungsvarianten des Systems zu interpretieren, zu bewerten und zu verantworten. Zwischen Anfangs– und Zielzustand dieses Prozesses liegt eine mehr oder weniger komplexe Mensch–Maschine–Interaktion, um

- Zielvorstellungen zu Plänen zu verdichten,
- Lösungsmodelle auszuwählen,
- Systeme zu steuern und
- Interpretation und Verifikation der Ergebnisse zu sichern.

Die Frage „Was ist und was kann Informatik?“ wird dann in drei inhaltlichen Strukturlinien, den **Leitlinien des Schulfaches**, beantwortet. Solche Leitlinien kennzeichnen die Struktur des Inhalts, fördern die Überschaubarkeit und betonen das Wesentliche. Sie leisten einen wichtigen Beitrag zur *Lehrbarkeit* des Faches, die sich auf folgende Basiskonzepte stützt:

- Ziel–Inhalt–Methoden–Relation,
- Anwendbarkeit und Handlungsorientierung,
- ganzheitlicher Unterricht: Beitrag des Faches zu Methoden– und Sozialkompetenz entfalten,
- Strukturierung der Inhalte als Orientierung für Lehrende und Lernende: *Leitlinien des Schulfaches*
  - erstrecken sich über mehrere Schuljahre,
  - besitzen Schwerpunkte in einzelnen Schuljahren,
  - ordnen den Stoff vom Einfachen zum Komplizierten,
  - zeigen die Beständigkeit grundlegender Methoden.

Damit wird die Gefahr der abschweifenden Interpretation von Bildungszielen reduziert. Die Elemente des Stoffes untersetzen die Ziele für Wissen und Können entsprechend der bestehenden Innenbeziehungen. Die Sachlogik des Faches wird zur Strukturierung des Lehr–Lern–Prozesses eingesetzt.

3.4. Pläne

Was ist möglich? Was wird gemacht?
Lösungsmodelle – Berechenbarkeit – Algorithmen und Heuristiken
Hier wird geklärt, welche Eigenschaften Probleme besitzen müssen, um mit Informatiksystemen gelöst zu werden. Außerdem wird begründet, dass die Komplexität auch einfach beschreibbarer Probleme 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 naiven Einführung bis zur modellbasierten Definition systematisch aufgebaut. Dadurch wird die Rationalisierung geistiger Arbeit begründbar, und die Beurteilung der Auswirkungen erhält ein fachliches Fundament. Die Transformation einer Problembeschreibung in eine andere kann demonstriert werden, um die Einordnung unbekannter Probleme in bekannte Klassen zu ermöglichen. Das Aufzeigen innerer Gesetzmäßigkeiten eines Problems führt zum geeigneten Lösungsmodell. Grob– und Feinpläne können strukturiert, exemplarisch verifiziert und auf ihre Komplexität hin analysiert werden:

- *Lösungsmodelle – Berechenbarkeit*
  - Eigenschaften der Probleme (Problemklassen),
  - Beurteilung der Komplexität,
- *Algorithmen*
  - Automatisierung geistiger Arbeit,
  - Verifikation der Ergebnisse und der Verfahren,
- *Heuristiken*
  - Wissen über den Problemlösungsprozess,
  - Erkennen von Analogien von Strukturkonzepten.

3.5. Interaktion – Kommunikation – Sprachen

Wie kann man sich verständigen und etwas darstellen?
Sprachen und Grammatiken (Sprachklassen)
Die exemplarische Behandlung von Entwurfs–, Programmier– und Interaktionssprachen gehört in diese Linie, um Objekte erzeugen, verbinden und verwalten zu können. Protokolle sind zu analysieren mit dem Ziel, Verständnis für typische Strukturen und Situationen der Mensch–Maschine–Interaktion zu entwickeln:

- *Interaktion mit Informatiksystemen*
  - Automatisierung geistiger Tätigkeiten,
  - Syntax und Semantik formaler Sprachen,
  - Sprachklassen (Erzeugung, Unterscheidung),
  - Pragmatik (Interaktionspläne),
  - Interpretation und Verantwortung der Ergebnisse.
- *Kommunikation und Lernen über das Netz*
  - Veränderung der menschlichen Kommunikation,
  - lokale Informations– und Kommunikationssysteme,
  - Lernform computergestützte Gruppenarbeit,
  - Internet–Anwendungen,
  - Anforderungen an weltweite Informationsstrukturen.

Syntax und Semantik sind ebenso wie der Begriff der Grammatik im Kontext der Informatik zu behandeln. Leichtere Aufgaben, wie das Bereitstellen von Parametern, das Beantworten von Systemfragen, wechseln sich ab mit anspruchsvollen Aufgaben wie dem Interpretieren, Vergleichen und Beurteilen von Ergebnissen in Abhängigkeit von der jeweiligen Zielvorstellung.

3.6. Systeme

Wie funktionieren die verarbeitenden Informatiksysteme?
Hard- und Softwaremodelle (Werkzeuge)
Die Modellierung von Umweltausschnitten in Informatiksystemen und die daraus resultierenden Werkzeuge können bis zu den Basisoperationen betrachtet werden. Wenn diese Prinzipien verstanden wurden, sind Kriterien formulierbar, die die Auswahl des richtigen Systems für die Lösung des jeweiligen Problems ermöglichen. Die Automatenmodelle und speziell das Zustandsraummodell unterstützen die Transparenz komplexer Systeme sowohl im Hardware– als auch im Software–Bereich. Das Halteproblem kann mit Hilfe des allgemeinen Prinzips der Diagonalisierung nachgewiesen werden. Das Übersetzen von Plänen in Sequenzen von Basisoperationen wird als typische Systemeigenschaft erkannt. Auch einfache Nutzungsaufgaben, wie das Steuern der ausgewählten Funktionen, die Aktualisierung und Speicherung von Informationen, gehören in diese Linie. Die Vernetzung von Systemen und die Erfordernisse der Informationssicherheit erweitern die Grundvorstellungen:

- Wirkprinzipien,
- Modellierung von Umweltausschnitten,
- Auswahlkriterien für die Anwendung,
- Anforderungskriterien für die Gestaltung,
- Vernetzung von Systemen,
- Informationssicherheit.

3.7. Bildungskonsequenzen für Informatik–Lehrkräfte

Folgt man den vorgestellten Überlegungen zum Berufsbild (bzw. Schulfach), dann zeigt sich, dass Diplominformatiker und Informatiklehrer sich besonders in den Ausbildungszielen zur Programmierungs– und Software–Technik unterscheiden, aber in den theoretischen Grundlagen übereinstimmen. Das Lehramt erfordert keine Systementwickler, auch keine professionellen Programmierer, sehr wohl aber Experten, die Transparenz in Informatikprozesse bringen können. Der Nachteil liegt in der stärkeren Trennung von Diplom– und Lehramtsstudiengang und der damit verbundenen höheren Lehrbelastung. Als Vorteil kann die damit verbundene Weiterentwicklung der Lehrdisziplin angesehen werden und deren zielgerichtete Studierbarkeit. Gerade die Kompliziertheit der theoretischen Grundlagen der Informatik wurde von didaktisch begabten Autoren als Herausforderung betrachtet, hervorragende Lehrbücher zu verfassen, z.B. [Reischuk90], [LP81], [Wagner94]. Damit steht eine Vielzahl von vergegenständlichten Lehrvarianten zur Verfügung, die mit Modellvorstellungen und Beispielsammlungen ausgestattet sind. Teile davon wurden von Schulbuchautoren aufgenommen, z.B. [GLSS92], wobei sofort die unberechtigte Kritik auftauchte, die Hochschulinformatik solle in die Schule gebracht werden. Zur Zeit läuft ein stürmischer Prozess ab, in dem sich die einzelnen Zweige der Lehrdisziplin Informatik für Hochschulen, für berufliche Schulen und für allgemeinbildende Schulen deutlicher herauskristallisieren. Noch ist die Standortbestimmung vieler Inhalte umstritten. Zu dem Zeitpunkt, da eine verbindliche Vorbildung zur Informatik in der allgemeinbildenden Schule absolviert wird, erfolgt eine Entlastung der nachgelagerten Bildungsgänge (Hochschulen und berufliche Schulen). Sie übernehmen dann nicht mehr die Aufgabe, in das Grundwissen zur Informatik einzuführen, sondern bauen auf diesem auf. Also werden logischerweise Inhalte, die heute ausschließlich in Studium und Berufbildung anzutreffen sind, in das Schulfach einmünden. Der wesentliche Irrtum besteht darin, dass angenommen wird, man könnte diese Inhalte mit den für andere Ziel– und Altersgruppen bewährten Lehrmethoden verbinden. Baumann sagt dazu sehr deutlich: „Die

Behandlung von Themen der theoretischen Informatik darf nicht nach Art der Hochschule geschehen, ...“ [Baumann93], S. 13. Also sind die Inhalte, die dem allgemeinbildenden Kern der Informatik zugeordnet werden, so zu strukturieren, dass sie die Ziele des Schulfaches umsetzen und einige allgemeine Anforderungen erfüllen, wie Zugang zur Wissenschaft erschließen, wesentliche Grundlagen vermitteln (Begriffe, Prinzipien, Methoden), bereits im Ausbildungsprozess anwendbar sein. Die Befähigung dazu muss zwar die Fachdidaktikausbildung absichern, aber die wiederum baut auf den fachlichen Studienkomponenten auf. Die Begründungslinie ist damit soweit vorangeschritten, dass konkrete Überlegungen zu den Theorie–Inhalten angeschlossen werden können, die alle die im zweiten und dritten Abschnitt genannten Anforderungen an Informatiklehrer untersetzen. Da diese Absolventen Experten für die Lehrdisziplin werden, sollte neben der formalen Auswahl der Studieninhalte ebenso die zugehörige Hochschullehrmethode betrachtet werden. Denn gerade in diesem Punkt bestehen gravierende Unterschiede zwischen Diplomstudiengang und Lehramtsstudiengang. Die bisher versuchte gemeinsame Grundvorlesung zur Theoretischen Informatik führte nicht dazu, dass das wichtige berufliche Ziel eines Lehramtsstudienganges „für die später zu unterrichtenden Schüler Transparenz in Informatikprozesse zu bringen“ ausreichend unterstützt wurde. Es ist nicht zu erwarten, dass die angehenden Lehrer das, was sie selbst nicht umfassend verstanden haben, an ihre künftigen Schüler vermitteln können. Die isolierte Darstellung der Theoretischen Informatik mit einem umfangreichen Anteil sehr komplizierter Beweise lässt vor allem die Studenten scheitern, die nicht die Mathematik vertieft studieren. Eine solche Einengung ist bei der sonst üblichen freien Kombination der Lehramtsfächer nicht zu begründen.

Hier wird die Neukonzipierung folgender Theorie–Inhalte zu einer eigenständigen Vorlesung für das Lehramt vorgeschlagen, in die nur ausgewählte Beweise aufgenommen werden, die dafür aber bereits gezielte Hinweise auf die darauf aufbauenden Informatiksysteme (z.B. Spezifikation von Programmiersprachen, Compiler, Schaltkreise, neuronale Netze, Betriebssysteme ...) liefern:

- Theorie formaler Sprachen und Berechenbarkeit,
- Verifikations– und Komplexitätstheorie,
- Grundlagen der Mensch–Maschine–Interaktion,

So wird es möglich, speziell in den Übungen und Seminaren in die Palette der Beispiele solche mit angebahntem Anwendungsbezug aufzunehmen und die Mächtigkeit der theoretischen Konzepte zu demonstrieren. An diesem Sachverhalt zweifeln übrigens auch viele Diplomstudenten. Gerade in einer Informatikausbildung sollte die Beispielauswahl nicht von den Möglichkeiten, die Tafel und Kreide oder die Projektorfolie bieten, begrenzt sein. Informatiksysteme in der Ausbildung gestatten es, durch ein Praktikum zum Experimentieren mit theoretischen Modellen (Automaten, Grammatiken, Komplexitätsklassen, Verifikationsverfahren, Kommunikationsstrukturen ...) zu ermutigen und auf diese Weise den abstrakten Lehrstoff anschaulich und begreifbar (im wahrsten Sinne des Wortes) zu gestalten.
Die Theorie–Inhalte erlauben es in weitaus besserer Weise, als das bisher versucht wurde, **Wissen zu strukturieren**. Der Vergleich zur Mathematik und deren Schulfach liegt nahe. Dort bilden die Zahlenbereiche und deren Erweiterung eine Grundsäule, die in der Informatik der Einordnung von Problemen in Komplexitätsklassen und deren hierarchischer Erweiterung zukommt. Dafür ist es nicht ausreichend, diese Inhalte zu lehren, ohne auf deren Bedeutung im Gesamtsystem des Fachwissens tiefer einzugehen (Wissenschaftstheoretische Reflexion):

- wissenschaftliche Standortbestimmung,
- Innen– und Außenbeziehungen der Fachdisziplin
  - Stellung der Teilgebiete zueinander,

- Wechselwirkung mit anderen Wissenschaften,

- Stellenwert einer Informatikgrundausbildung
  - Bedeutung informatiktypischer Denk– und Arbeitsweisen,
  - Strukturieren von Wissen (Metawissen),
  - alternative Konzepte,

- gesellschaftliche Auswirkungen
  - Kriterien für Informatiksysteme,
  - Erkennen von Konflikten und Aushandeln von Kompromissen.

Man könnte das als eine Wissensebene (Metawissen) über den theoretischen Grundlagen auffassen, die für die Informatik eine wissenschaftliche Standortbestimmung liefert. Hier sind die Innen– und Außenbeziehungen der Fachdisziplin zu erläutern. Zu vermuten ist, dass damit der Stellenwert einer Informatikgrundbildung wesentlich präziser formulierbar wird.

Die fachspezifische Vorgehensweise

- Objekte, deren Eigenschaften und Beziehungen zu definieren,
- diese Strukturen als Modell einer „abgeschlossenen Welt“ aufzufassen und
- aus diesen Strukturen durch Formalisierung von Aktionen das potentiell enthaltene Wissen abzuleiten

beeinflusst den Umgang mit Informationen und Wissen in anderen Wissenschaftsdiziplinen beachtlich. Die Schlussfolgerung daraus ist, dass für den Lehramtsstudiengang nach der Einführung in die theoretischen Grundlagen (oder parallel dazu?) eine **wissenschaftstheoretische Reflexion** dieser Inhalte erforderlich wird. In dieser Ausbildungsphase kann die Bedeutung informatiktypischer Denk– und Arbeitsweisen eine fundierte Begründung erfahren, die wiederum zur Vertiefung des gesamten Theorieverständnisses führt. Das, was man für Lehrprozesse allgemein unter ganzheitlicher Ausbildung versteht [HT94], kommt im Lehramtsstudiengang nicht genügend zur Wirkung. Besonders die Niveaustufen Entscheidungsfähigkeit, technische und soziale Gestaltungsfähigkeit, Bewertungsfähigkeit, Erkennen von Konflikten und Aushandeln von Kompromissen finden in der theoretischen Grundausbildung zu wenig Förderung. Sie wurden bisher von der Ausbildung in Programmierungs– und Software–Technik mit der zugehörigen Projektentwicklung erwartet. Diese Lehrgebiete binden bisher den Hauptteil der Studienzeit. Gerade in diesem Punkt findet zur Zeit ein Umdenken statt. Eingeleitet wurde dieser Prozess von der Erfahrung, dass das vertiefte Beschäftigen mit einem Programmierkonzept nicht die erwartete Erweiterung dieses Wissens und Könnens auf alternative Konzepte liefert. Das wiederum kann die Theorie formaler Sprachen (Kalküle, Nichtdeterminismus, Parallelität …) leisten. Nievergelt [Nievergelt91] spricht vom „Zerrbild der Informatik (einige Details von vorübergehender Bedeutung erscheinen trügerisch wichtig, während grundlegende Ideen unerwähnt und unerkannt bleiben)“, das die Schüler zu sehen bekommen, wenn den Lehrern die Übersicht und die Perspektive des Fachgebietes nicht erschlossen wird.

Aus der Theorie der Informatik wird folgender Bildungswert abgeleitet:

- **Fachkompetenz** (Beherrschung der theoretischen Grundlagen, sachlogische Zusammenhänge des Faches):
  - Definitionen, Sätze, Beweise,

  - Modelle,
  - Perspektiven,

- **Methodenkompetenz** (allgemeine Problemlösefähigkeit, Metawissen über das Lernen):
  - Kennen von Anwendungsbeispielen (typische Tätigkeiten),
  - Verfahren (Algorithmen und Heuristiken),
  - Strukturen (für Handlungen, für Wissen),

- **Sozialkompetenz** (allgemeine Kommunikationsfähigkeit, Orientierung an Werten und Normen):
  - Bewertungskriterien,
  - Möglichkeiten und Grenzen,
  - gesellschaftliche Auswirkungen,
  - Verständnis für Sprachen,
  - Mensch–Maschine–Interaktion,
  - Telekommunikation.

Wie sieht nun die Zukunft der Lehramtsstudiengänge Informatik aus? Es gibt eine optimistische und eine pessimistische Variante. Bedauerlich wäre, wenn das Schulfach Informatik nicht den Durchbruch zum eigenen Theoriekern findet. Dann steht zu befürchten, dass immer weniger Lehrer sich auf das Wagnis einer Anwendungsfolge ohne Fundament einlassen. Dazu erübrigen sich weitere Erörterungen. Interessanter ist die wissenschaftliche Begleitung des Schulfaches und der Lehrerausbildung dann, wenn an das theoretische Niveau vergleichbarer Schulfächer angeschlossen werden soll. Auch wenn man der These zustimmt, dass Informatiklehrer auf Dauer nicht bestehen können ohne fundierte Kenntnisse auf theoretischem Gebiet, bleiben noch viele Fragen offen. Welche Visionen haben eine Chance realisiert zu werden? Hier ist zu beachten, dass die Allgemeinbildung, vergleichbar einem sehr sensiblen System, nicht durch hektischen Veränderungen vorangebracht wird, sondern durch eine Folge wohldurchdachter kleiner Schritte, deren Auswirkungen überschaubar und steuerbar bleiben. Den Anfang könnte eine Investition in die Ausbildung des Lehrernachwuchses darstellen. An dieser Aufgabe kann sich die Hochschullehrdisziplin der Informatik in der vorgeschlagenen Weise weiter profilieren. Parallel zu dieser Phase sind Forschungsprojekte zur theoretischen Fundierung des Schulfaches erforderlich, um mögliche Thesen und Varianten auf ihre Wirksamkeit zu prüfen und empirisches Material zu gewinnen. Außerdem können von den wissenschaftstheoretischen und philosophischen Untersuchungen zur Informatik Ergebnisse erwartet werden, die für die Klärungsprozesse innerhalb der Lehrdisziplin ausgesprochen wichtig sind.

4. Problemlösen in der Informatik

Modellvorstellungen

Problemlösen wird in fast jedem Unterrichtsfach gefordert, aber in keinem bisher fundiert thematisiert. Das Fach Informatik eignet sich in besonderer Weise dafür, da eine Vielzahl anderer Fächer die Problemlösungsmethoden der Informatik als Vorkenntnisse benötigen. In den 80er Jahren befürchteten Menschen eine „Digitalisierung der Sinne“, da Missverständnisse bezüglich der kreativen Anforderung des informatischen Modellierens und der rechnerinternen Umsetzung dieses Modells bestanden. Irrtümlich nahm man an, der kreative Spielraum des Menschen würde durch Algorithmen und Datenstrukturen stark eingeschränkt. Gezeigt hat sich, dass der Entwurf von Algorithmen und Datenstrukturen eine anspruchsvolle geistige Tätigkeit ist. Die Konzepte dafür wandelten sich von maschinennahen zu problemorientierten Beschreibungen, von unstrukturierten zu strukturierten Entwürfen, von prozeduralen zu objektorientierten Modellen. Unklar ist heute, ob die objektorientierte Programmierung eine Weiterentwicklung des Prinzips der strukturierten Programmierung darstellt oder unter diesem Dach das zur Zeit erfolgreichste Paradigma bildet. Für die informatische Bildung hat das wichtige Konsequenzen. Zum Verständnis soll dieses kleine Kapitel über Erkenntnisse aus dem Bereich der kognitiven Psychologie beitragen.

Prozedurales Wissen bezieht sich auf die Art, wie kognitive Prozesse beim Problemlösen ausgeführt werden. Der Suchraum und die potentiellen Schritte werden charakterisiert. Die Frage, welche Schritte wirklich unternommen werden, liefert Prinzipien, die den Suchprozess steuern (Algorithmen, Heuristiken). Mit Heuristiken kann man einen Suchraum einschränken. Das Modell vom Problemraum (Zustandsraummodell) wurde von Allen Newell und Herbert Simon entwickelt. Verschiedene Wissenszustände, die ein Problemlösender erreichen kann, definieren einen Problem– oder Zustandsraum. Gesucht ist eine Transformation (Problemlöser), die den unerwünschten Anfangszustandes z_0 (Problem) in den erwünschten Zielzustand z_n (Problemlösung) überführt (siehe Abbildung 8). Diese Transformation entspricht einer Sequenz von Regeln (Ope-

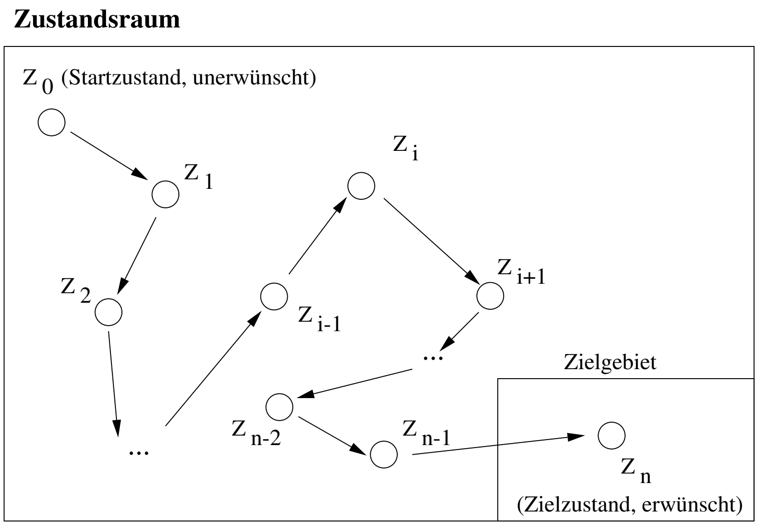


Abbildung 8: Zustandsraummodell

ratoren), die schrittweise z_0 in z_n umwandelt. Dabei sind gewöhnlich Zwischenzustände erforderlich. Objekte, Attribute, Werte und Überführungsfunktion sind zu ermitteln. Solche Operatoren zu finden, die die gewünschte Überführung herbeiführen, kann ein sehr komplizierter Prozess sein. Problemlösen wird als zielgerichtetes Verhalten definiert (Zerlegung in Teilaufgaben bis Operator einen Anfangszustand in einen Zielzustand überführt). Der Problemlösende muss einen angemessenen Weg durch das Labyrinth von Zuständen finden (Suchprozesse). Suchbäume

(Zustands–Handlungs–Bäume) können hilfreich sein.

Allgemeine Strategien und Methoden des Problemlösens

Zum Problemlösen sind heuristische Methoden erforderlich. Heuristik ist die Lehre, Wissenschaft von den Verfahren, Probleme zu lösen, d.h. die Lehre von den Methoden zur Auffindung neuer wissenschaftlicher Erkenntnisse. Für jede Problemklasse stehen spezielles Wissen und spezielle heuristische Techniken zur Verfügung. Variable, kognitive Strukturen, Wissen und eine Heuristik zur Problemklasse bilden den Problemlöser (die Problemlösung).
Problemlösungsprozesse

- verlaufen nicht algorithmisch,
- besitzen kein strenges Schema,
- weisen häufig Versuch–Irrtum–Strategie in mehreren Zyklen auf,
- zeigen die große Bedeutung von Fehlern (Fehler sind positiv zu bewerten) und
- erfordern Strategien:
  1. Die Versuch–Irrtum–Strategie dürfte aus persönlichem Erleben jedem bekannt sein.
  2. Regelbasierte Systeme stellen einen allgemeinen Formalismus zum Problemlösen dar mit verschiedenen und wechselnden Bearbeitungsrichtungen. Dazu stehen bereichsübergreifende und bereichsspezifische Regeln der Form *„Wenn Bedingungen erfüllt, dann Handlungen ausführen.“* zur Verfügung. Man unterscheidet:
     a) *Regelverkettung vorwärts (vom Anfang zum Ziel)*
        Regeln *„Wenn … dann …“* sind datengetrieben, Schlüsse werden aus Fakten gezogen.
     b) *Regelverkettung rückwärts (vom Ziel zum Anfang)*,
        Regeln *„Dann … wenn …“* sind zielgetrieben, von Folgerungen wird auf die Bedingungen geschlossen.
        Diese Rückwärtssuche findet man bei mathematischen Beweisen oder Reiseplanungen. Ein Zielkonflikt erfordert die Teilziele neu zu strukturieren. Im Plan werden Abhängigkeiten zwischen den Teilzielen ermittelt und Fakten gesucht, um Ziele durch Teilziele zu ersetzen.
  3. Problemlösen durch Analogien wird in der Informatik erfolgreich angewendet. Beim Analogieschluss wählt man eine erfolgreiche Struktur als Leitfaden. Der Schlüssel zum Erfolg liegt in der (operatorgerechten) Repräsentation des Problems und der Verfügbarkeit des Wissens.
  4. Deduktion (Herleitung des Besonderem aus dem Allgemeinen)
  5. Induktion (Herleitung von allgemeinen Regeln aus Einzelfällen)
  6. Generalisieren
  7. Konkretisieren

Die folgenden Methoden eignen sich für unterschiedliche Problemklassen.

Die Methode der Unterschiedsreduktion
Differenzierte Ähnlichkeitsmuster sind zur Auswahl der Teilziele so zu nutzen, dass der Abstand zum Ziel abnimmt. Es besteht die Gefahr irreführender Ähnlichkeit. Am Beispiel des Bauern,

der mit Ziege, Kohlkopf und Wolf einen Fluss in einem Kahn mit maximal zwei Ladungskomponenten überqueren möchte, wird das deutlich.

Die Mittel–Ziel–Analyse
Die Objekte mit ihren Unterschieden sind vorgegeben. Operatoren verändern die Merkmale, um Unterschiede zu beseitigen. Eingaben sind so zu modifizieren, dass Operatoren anwendbar werden. Schwer beeinflussbare Unterschiede sind möglich. Diese müssen ersetzt werden durch andere Unterschiede. So kann es zu zeitweiligen Entfernungen vom Ziel kommen.

Repräsentation
Von der Bedeutung konkreter Repräsentationen hängt die Wahl der Operatoren ab. Funktionale Gebundenheit (Fixierung) hemmt Objekte in der gewohnten Umwelt auf neue Art zu repräsentieren. Eine Lösung kann blockiert werden, wenn man die Werkzeuge nur einseitig nutzen kann oder möchte. Das psychologische Experiment zum „Kerzenproblem“ zeigte, dass bei Vorgabe einer Schachtel mit Reißnägeln, einer Kerze und Zündhölzern die Befestigung der Kerze an einer Tür leichter gemeistert wurde, wenn die Reißnägel nicht mehr in der Schachtel ausgeteilt wurden. Die neue Funktion der Schachtel als Kerzensockel wurde dann deutlich schneller erkannt. Für die Formulierung von Aufgaben ist das eine wichtige Erkenntnis. Aber auch die Gestaltung von Lernprozessen insgesamt sollte auf Fixierungen untersucht werden.

Einstellungseffekte
Die Automatisierung der Denkvorgänge (Operatorsequenz im Gedächtnis) und der Einfluss allgemeiner semantischer Faktoren führen dazu, dass einige Wissensstrukturen (deklarative oder prozedurale) auf Kosten anderer besser zugänglich sind.

Diese Methoden sind alle den prinzipiellen Barrieren beim Problemlösen ausgesetzt:

1. Operatoren bekannt, aber ihre Reihenfolge (Sequenz) nicht (z.B. Schach),
2. neue Operatoren erforderlich (z.B. Technologie),
3. Zielzustand dynamisch (wird erst während des Problemlösungsprozesses präzisiert, z.B. Reparaturauftrag).

Das Gedächtnis spielt bei der Überwindung der Barrieren eine entscheidende Rolle. Erforderlich ist Wissen über den Problembereich und über heuristische Techniken (allgemeine heuristische Techniken (Analogieschluss, Induktion, Deduktion, Abstraktion) und spezielle heuristische Techniken, die problemspezifisch sind).

Spezifika der Informatik

<u>1. Schritt</u>
Von der (unscharfen) Problemdefinition durch Lösungshypothese zum Lösungsmodell (prinzipielle Lösbarkeit klären)

Dazu gehören die Definition und Vorplanung und die Konstruktion (Spezifikation und Grobentwurf). Algorithmisch beschreibbare Problemlösungen sind:

- Definierbarkeit (exakt bis ungenau),
- Komplexität (klein bis groß),
- Lösungsbeschreibung (mathematisch, partiell formal),
- Lösungswissen (bekannt, partiell bekannt).

Ein Weltausschnitt von unstrukturierten Ausgangsdaten wird unterlegt mit einem Strukturkonzept, das das zu untersuchende System (automatentheoretische, graphentheoretische) zerlegt und gliedert. Dabei sind Varianten möglich (unterschiedliche Paradigmen). Außerdem werden Modelle genutzt (physikalische, graphische, mathematische, Simulationsmodelle) (siehe Abbildung 9). Zu den Prinzipien und Methoden zählen Hierarchisierung, Modularisierung und Top–down–

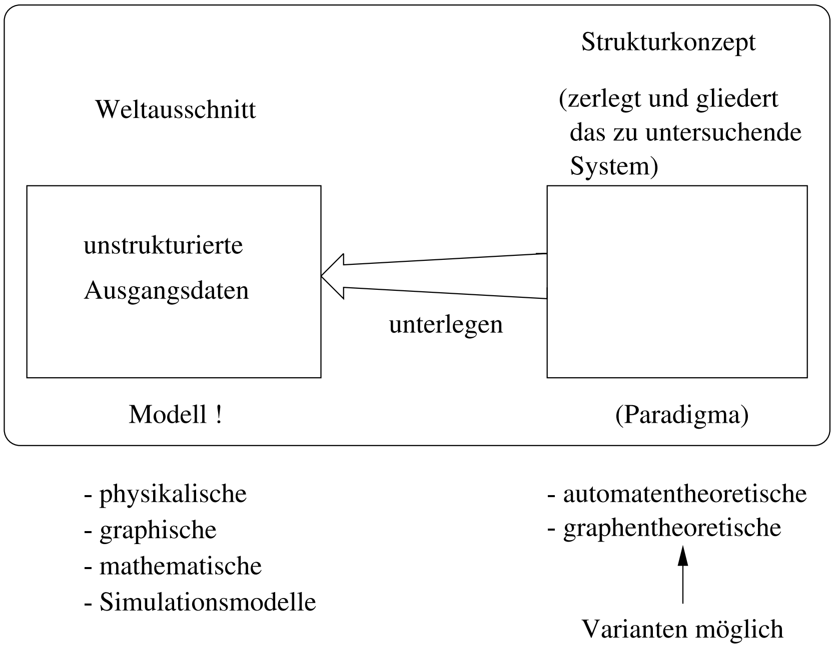


Abbildung 9: Unterlegung eines Weltausschnitts mit einem Strukturkonzept

Entwurf. Die kognitiven Leistungen umfassen Entdecken (Wiederentdecken), kreatives Denken, begriffliches Denken und Variieren von Strukturkonzepten. Die Versuch–Irrtum–Strategie dominiert bei Anfängern. Dieser schöpferische Akt ist nicht algorithmierbar.

2. Schritt

Vom Lösungsmodell zur Software für Klasse von Problemen (abstrakte, virtuelle Maschine)

Dazu gehören die Feinentwürfe und die Implementierung auf der realen Basismaschine. Diese Konstruktion des Programms (verschiedene Programme für ein Modell) erfordert solche Prinzipien und Methoden wie Top–down–Entwurf, Prozedurkonzept, schrittweise Verfeinerung, strukturierte Programmierung. Es stehen Programmentwicklungssysteme für verschiedene Programmiersprachen zur Verfügung, die Teamarbeit und Fehlersuche unterstützen. Die kognitiven Leistungen erfordern hochspezialisierte Heuristiken und große Disziplin. Ein wesentlicher Vorteil liegt in der Anwendung (Modifikation) fertiger Strukturkonzepte (Analogien erkennen und nutzen).

Die prinzipiellen Eigenschaften von Wissen erschweren die Entwicklung von Informatiklösungen:

- Redundanz (Regel ist Spezialfall einer anderen Regel),
- Widersprüche,
- Unsauberkeiten (Sackgassen, Unerfüllbarkeit),
- Unvollständigkeit,

- Unsicherheit (Maß des Vertrauens),
- Unpräzision (unscharfe Aussagen; z.B. „jung“).

Im Informatikunterricht erleichtert folgende Reduzierung der Anforderungen den Lernerfolg in der Grundausbildung:

1. Reduzierung des Anwendungsprozesses
   Grundausbildung, Tätigkeitszyklus, Kontrollpunkte, Typische Situationen,
2. Komplexitätsreduzierung der zu lösenden Aufgabe
   Einsatz von Grundstrukturen und Grundalgorithmen,
   Folge: Sequenz, Zyklus, Alternative, Kombination der Grundstrukturen, Datenstruktur Feld, Unterprogrammtechnik, Datenstruktur Rekord, Verknüpfung der Strukturen für Aktionen und Daten,
3. Reduzierung der Planungsentscheidungen durch Empfehlung einer Verfahrensheuristik (Lernen aus Beispielen)
   a) Idee, Zerlegung, Analogie, Konstruktion,
   b) Eingrenzen des Suchraumes,
   c) Abhängigkeit zwischen Wahl der Datenstruktur und der Reduzierung der Problemkomplexität,
   d) Verständnis für die konzeptionellen Schichten moderner Informatikanwendungen,
   e) Kreativität und Disziplin.

Die Fachdidaktik organisiert das „menschliche Fenster“, d.h. sie ermöglicht Komplexitätsbewältigung und Transparenz.

In der Informatik haben sich graphische Darstellungen (z.B. Graphen, Bäume) zur Darstellung des zu formalisierenden Wissens durchgesetzt. Der Und-Oder-Baum (siehe Abbildung 10) kann Handlungsstrategien veranschaulichen, die für Entscheidungsprozesse charakteristisch sind.

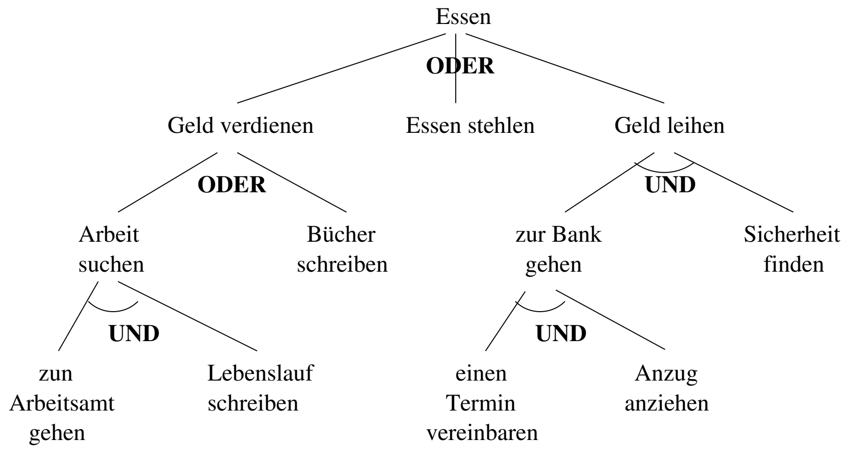


Abbildung 10: Beispiel für einen Und-Oder-Baum

Strategiespiele können so modelliert werden.

Das konstruktive Element der Informatiklösung bringt stark motivierende Wirkung in den Lernprozess. Der Schüler hat fast „grenzenlose“ Modellierungsmöglichkeiten. Erlernt werden soll:

1. von der vage formulierten Problemstellung zur präzisierten Aufgabenstellung durch Konkretisierung zu gelangen,

2. durch Analyse der Aufgabenstellung das Problem in Teilprobleme zu zerlegen, da diese leichter überschaubar und lösbar sein können,

3. den Prozess des Sammelns und Klassifizierens von Information bis zur Abstrahierung von realen Objekten zu abstrakten Datenstrukturen (heute) und Wissensstrukturen (künftig) zu vollziehen,

4. entsprechend den modellierten Datenstrukturen und dem vermuteten Ablaufmodell die Zuordnung zur Aufgabenklasse erkennen und

5. die statische Darstellung der dynamischen Abläufe mit Grundstrukturen realisieren.

Die Tatsache, dass stets mehrere originelle und korrekte Lösungen gefunden werden können, die nicht mit der des Lehrers übereinstimmen müssen, erhöht bei kreativen Schülern die Freude am Experimentieren.

5. Informatisches Modellieren

Informatiksysteme bestehen nach Rechenberg [Rechenberg94], S. 62 aus Schichten, die alles Geheimnisvolle ausschließen (siehe Abbildung 11). Die Wiederholbarkeit gleicher Aktionsfolgen ist

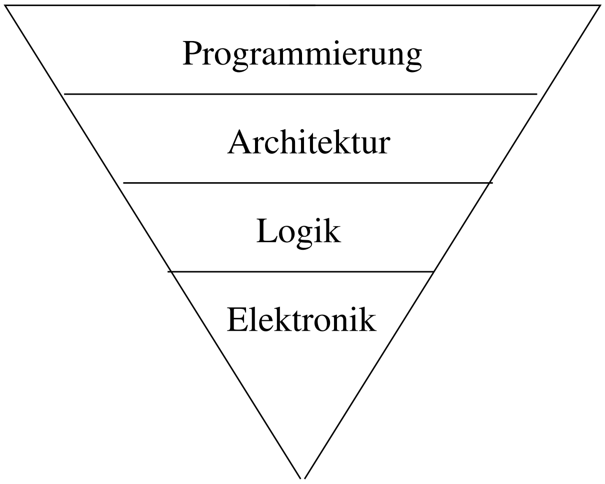


Abbildung 11: Abstraktionsschichten von Informatiksystemen

prinzipiell gegeben. Überraschungen treten auf, wenn die hochkomplexen Systeme Zustände annehmen, die vom Menschen in der Planungsphase nicht bedacht wurden. Solche Fehlerzustände lassen sich unterteilen in einfache und schwere. Einfache Fehler lassen sich gut rekonstruieren, sind der Analyse leichter zugänglich und damit schneller behebbar. Schwere Fehler treten scheinbar zufällig auf. In Wirklichkeit sind sie an seltene Konstellationen im System gebunden, dadurch kaum analysierbar und entsprechend schwer zu beheben. Genau diese Alltagserfahrung verleiht Computern in den Augen von Anfängern etwas Mystisches. Aufklärung darüber kann eine Analyse des informatischen Modellierens bringen. Informatiker konstruieren Beschreibungen (Programme, Software), die einen Aktionsraum abgrenzen sollen. Bei Bedarf wird die Beschreibung angewendet, d.h. sie steuert Prozesse der realen Welt. Diese Aktionen sollen nicht dem Zufall überlassen sein, sondern menschliche Ziele in die Tat umsetzen. Das kann nur gelingen, wenn die reale Welt so genau wie möglich im Informatiksystem beschrieben (abgebildet) wird. Genau hier liegt das Problem. Informatiksysteme können prinzipiell nur einen Weltausschnitt aufnehmen. Diese künstlichen Ausschnitte bilden ein Modell. Immer hat der Mensch (der Entwickler) entscheiden müssen, worauf es ihm im Falle einer konkreten Anwendung ankommt und worauf er verzichten möchte, glaubt verzichten zu können. Sind die Entscheidugen gut geglückt, bemerkt der Anwender kaum, dass er mit einem Modell auf Lebenswirklichkeit einwirkt. Er ist zufrieden, wenn er seine Aufgaben lösen, seine Ziele erreichen kann. Im ungünstigsten Fall erweist sich das Modell als unbrauchbar. Das Informatiksystem kann nicht sinnvoll eingesetzt werden. Natürlich sind solche Modelle veränderbar und für eine Anwendung verbesserbar. Immer aber bleiben es Modelle mit allen Vor– und Nachteilen, die ein Modell prinzipiell hat. Das Fördern dieses Grundverständnisses von Informatiksystemen ist eines der wesentlichsten Bildungsziele der informatischen Bildung. „In der Entwurfsphase entwicklen die Programmierer ein Modell des Gesamtsystems, das, umgesetzt in ein Programm, die Anforderungen erfüllt.“ [CS93], S. 659. Erlernen kann man das informatische Modellieren am besten handlungsorientiert. Das Modell in der Informatik wird durch eine Spezifikation konstruiert. Die Spezifikation fasst alle Systemeigenschaften zusammen, bevor das reale System exisitiert und auf seine Funktionstüchtigkeit überprüft werden kann. Möglich ist eine Unterteilung in Problem–, Anforderungs– und Entwurfspezifikation. Die Korrektheit eines Systems wird durch Vergleich mit der zugehörigen Spezifikation ermittelt. Die Zusammenhänge der realen Welt können nicht vollständig zur Überprüfung herangezogen werden, da sie auch nicht vollständig einbeziehbar sind. Anwenderforderungen finden also nur Berücksichtigung, wenn sie in die Spezifikation aufgenommen wurden. Erschwert wird dieser Einfluss bei der Systementwicklung für eine Klasse von Aufgaben, deren Bearbeiter nicht persönlich am Enwicklungsprozess beteiligt sind. Entsprechend unzufrieden sind die Anwender mit Lösungen, die große Aufgabenbereiche abdecken sollen. Häufig wird ein Funktionsumfang mit dem System gekauft, der überwiegend ungenutzt bleibt. Im Gegenzug fehlen individuelle Arbeitshilfen, die sich Anwender für ihr Spezialgebiet wünschen. Vorerst interessiert der Übergang vom Ausschnitt aus der Wirklichkeit zum Modell. Nicht die Kenntnis von Programiersprachen steht am Anfang, sondern die Kenntnis von Strukturkonzepten (siehe Abbildung 12). Am Beispiel des Hypertextes kann der Anfänger gut verstehen, wie hilfreich

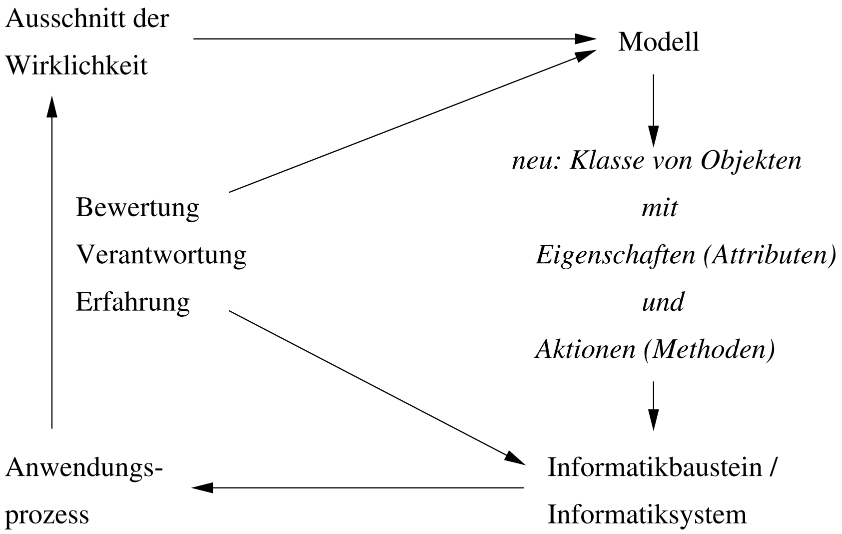


Abbildung 12: Informatisches Modellieren

oder lähmend der Strukturrahmen wirkt. Die lineare Anordnung von Informationselementen (Texten, Bildern, Tönen) entspricht nicht den menschlichen Bedürfnissen nach Selbstbestimmung im Handeln und nach Kreativität mit der zugehörigen Spontanität. Angeblich erzwingt das Buch lineares Lesen. Das kann nicht stimmen. Das Überblättern von Informationen ist in beide Richtungen möglich. Ein gewünschter Rückkehrpunkt muss mit Lesezeichen markiert werden. Beim Hypertextprinzip können digitalisierte Informationselemente (z. B. Textseiten) mit Steuerinformationen verknüpft werden. Der Graph liegt als Strukurkonzept vor, die Knoten sind die Informationselemente. Die Kanten sind die Präsentationsanweisungen. Es ist also möglich, jede Informationseinheit mit jeder anderen zu vebinden (siehe Abbildung 13). In diesem Labyrinth verliert der Leser leicht die Übersicht, wertvolle Zeit und gegebenenfalls auch wichtige Informationen. Sinnvoll ist hier eine informatische Modellierung, die sich an den Bedürfnissen der Anwender orientiert. Für ein Lehrmaterial bildet sich die Analogie zum bewährten Lehrbuch an (siehe Abbildung 14). Typischerweise beginnt eine Anwendung mit der Startseite, die nicht zu vermeiden ist. Über die Fortsetzung entscheidet der Anwender individuell. Für den Leser sind alle Pfade im Graphen erlaubt. Der Übergang muss aber erst aktiviert werden, er wird nicht erzwungen. Vergessene Pfade kann der Leser nicht selbst einrichten. Er hat mehr Service aber weniger Spielraum als im Umgang mit dem Buch. Der Autor übernimmt die Informationsgestaltung und konstruiert die möglichen Pfade zwischen ihnen. Kennt er die Gewohnheiten und Wünsche seiner Leserschaft schlecht, wird er seine Struktur falsch planen. Neu ist die unmittelbare Überleitung in Materialien anderer Autoren. In solchen Fällen wird kaum mit einer Rückführung zu rechnen sein. Die Anzeigeprogramme (Browser) halten einen Steuerrahmen bereit, der solche Lücken überbrückt und die Rückkehr ermöglicht. Sparsam sollte der Wechsel von der lokalen zur globalen Dokumentenebene erfolgen, da der direkte Netzzugang dafür erforderlich ist. Der Leser setzt Zeit und Geld dafür ein. Bevor er einen solchen Schritt geht, sollte er darüber informiert werden und den voraussichtlichen Nutzen erfahren.

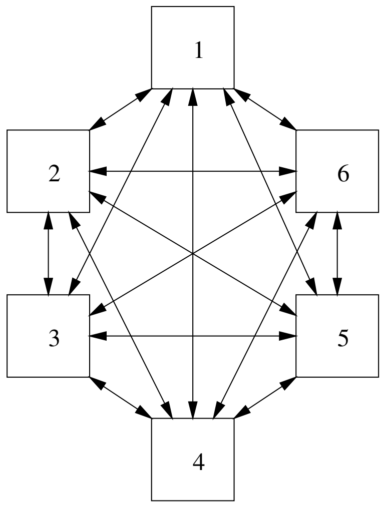


Abbildung 13: Hypertextprinzip: unstrukturiert

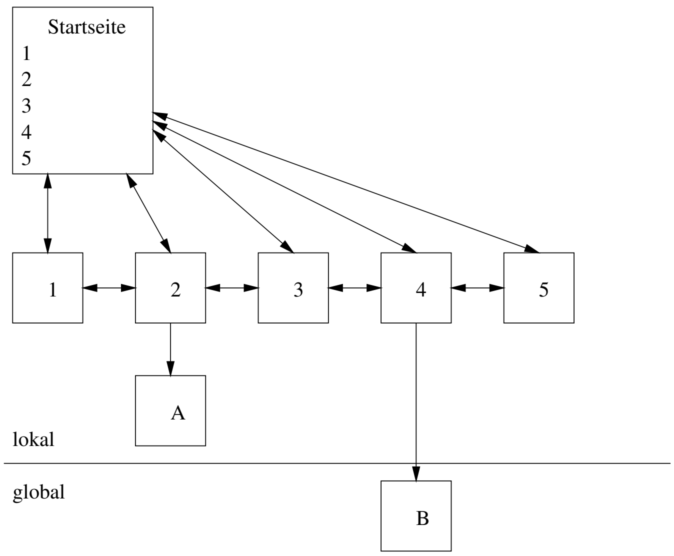


Abbildung 14: Hypertextprinzip: strukturiert

Am Beispiel einer Netzpräsentation wird deutlich, dass die Anwendungssituation die Kriterien für die Strukturierung liefert (siehe Abbildung 15). Die Startseite, hier Begrüßungsseite, kann

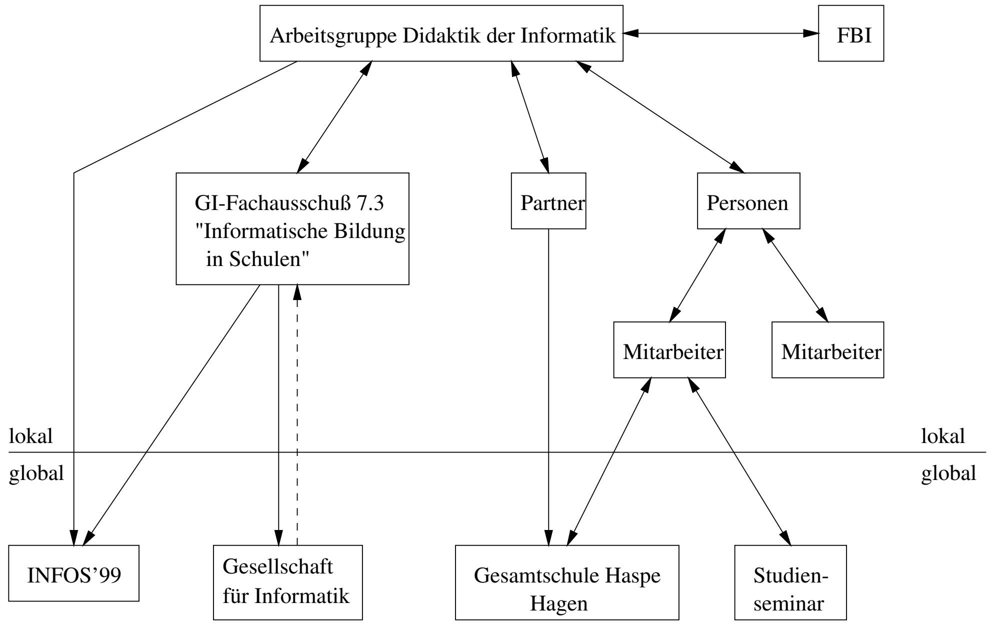
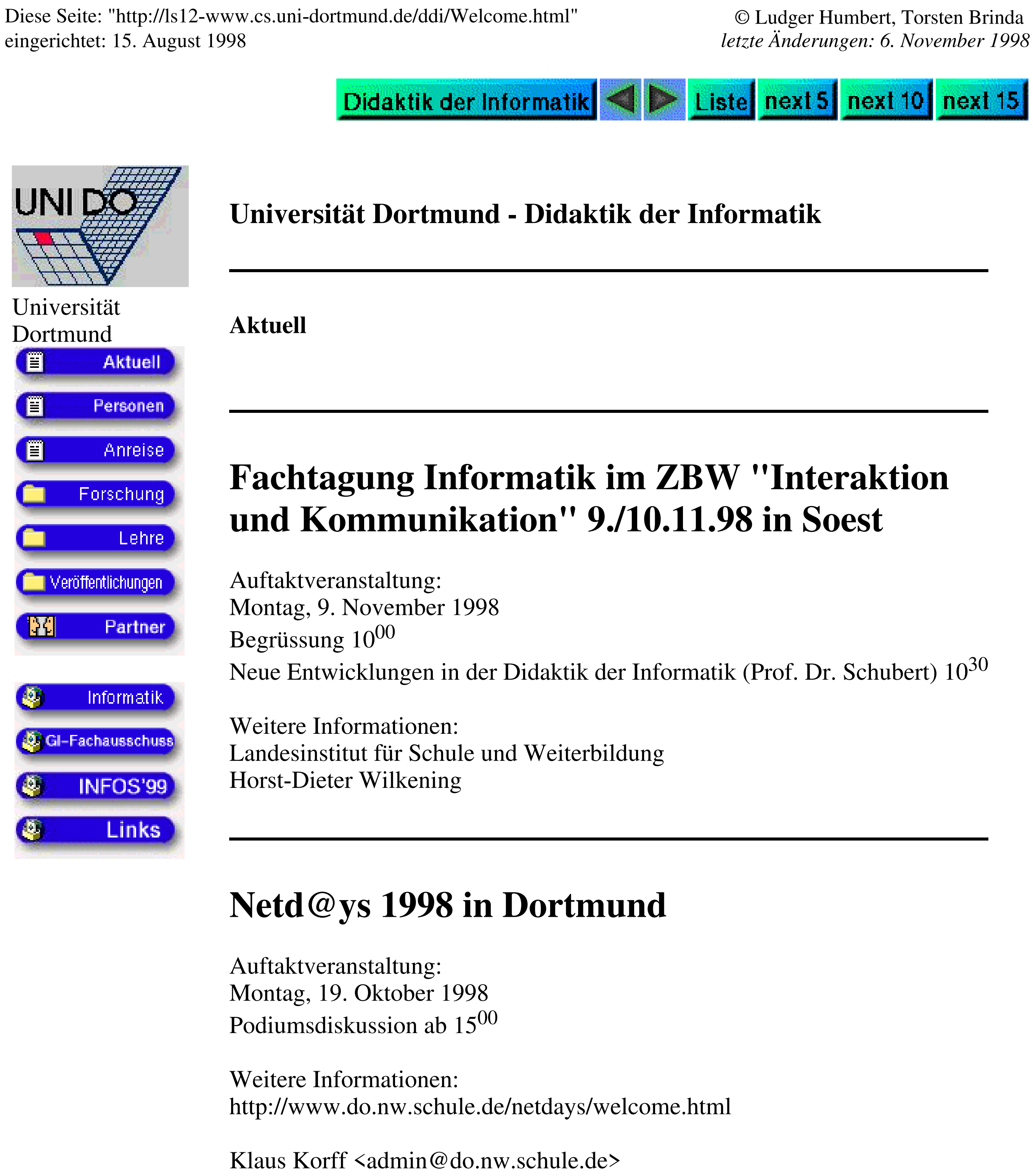


Abbildung 15: Teil des Entwurfs einer Netzstruktur

durchaus aktuell wechselnde Schwerpunkte anzeigen (siehe Abbildung 16). Das bringt den Leser schnell zu den neuen Informationen. Gliederungsrahmen analog zu dem Inhaltsverzeichnis setzten sich durch. Mit Verweisen kann Zusammenarbeit gut vorgestellt werden. Auf Grund der hohen Netzbelastung sind Bilder sparsam zu verwenden. Skripten werden typischerweise von den Lesern als Datei abgeholt und nicht am Bildschrim gelesen.

Die Medienverbindung von Text und Grafik (siehe Abbildung 17) erfordert bereits Sicherheit im Umgang mit unterschiedlichen Aufbereitungswerkzeugen und Dateiformaten. Die Dateiattribute sind für Kenner wichtige Parameter für eine mögliche Kombination im Gesamtdokument. Neu ist die Gestaltung von Ringstrukturen, die thematisch verbunden sind. So könnten z.B. alle Informatikdidaktikgruppen einen solchen Verbund bilden.



Diese Seite: "http://ls12-www.cs.uni-dortmund.de/ddi/Welcome.html"
eingerichtet: 15. August 1998

© Ludger Humbert, Torsten Brinda
letzte Änderungen: 6. November 1998

Didaktik der Informatik | Liste | next 5 | next 10 | next 15

UNI DO
Universität Dortmund

Aktuell
Personen
Anreise
Forschung
Lehre
Veröffentlichungen
Partner

Informatik
GI-Fachausschuss
INFOS'99
Links

Universität Dortmund - Didaktik der Informatik

Aktuell

Fachtagung Informatik im ZBW "Interaktion und Kommunikation" 9./10.11.98 in Soest

Auftaktveranstaltung:
Montag, 9. November 1998
Begrüssung 10^{00}
Neue Entwicklungen in der Didaktik der Informatik (Prof. Dr. Schubert) 10^{30}

Weitere Informationen:
Landesinstitut für Schule und Weiterbildung
Horst-Dieter Wilkening

Netd@ys 1998 in Dortmund

Auftaktveranstaltung:
Montag, 19. Oktober 1998
Podiumsdiskussion ab 15^{00}

Weitere Informationen:
http://www.do.nw.schule.de/netdays/welcome.html

Klaus Korff <admin@do.nw.schule.de>

Abbildung 16: Die HTML–Seite in der Browsersicht

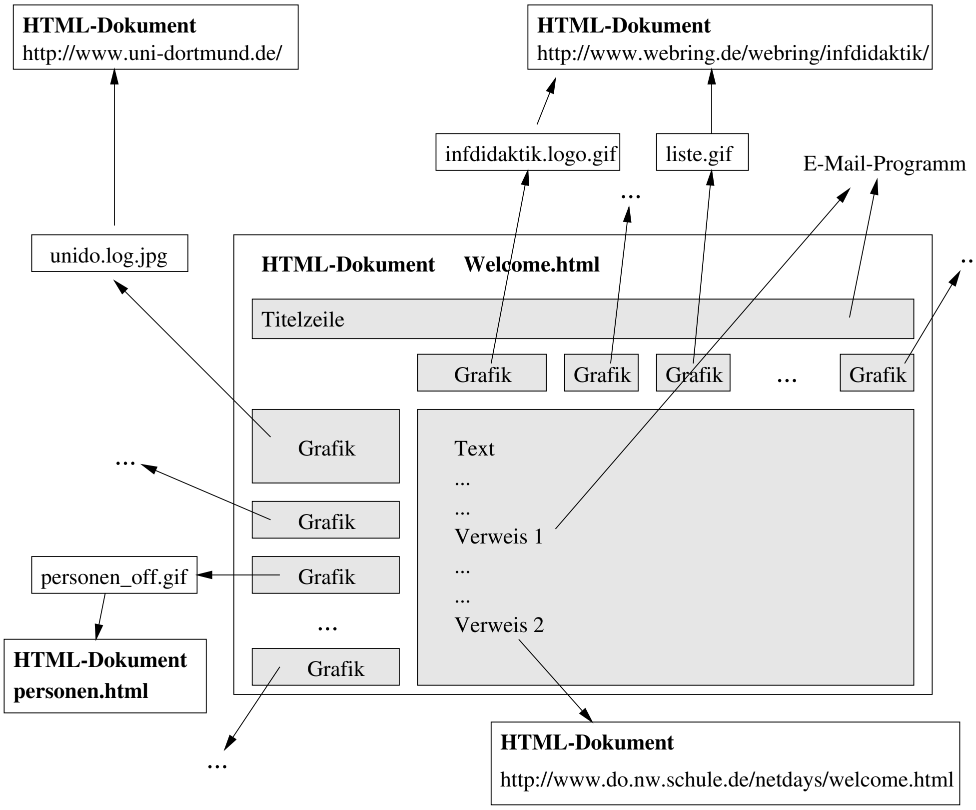


Abbildung 17: Medienverbindung

Mit HTML (<u>H</u>yper<u>t</u>ext <u>M</u>arkup <u>L</u>anguage) steht eine formale Sprache zur Verfügung, die die Steuerkommandos zur Strukturbeschreibung bereitstellt. Ein Interpreter für diese Sprache erzeugt die gewünschte Darstellung. Der Quelltext ist durchaus nicht angenehm zu schreiben, deshalb setzen sich Editoren dafür gut durch:

```
<HTML>
<HEAD><TITLE>Universitaet Dortmund: Didaktik der Informatik
        </TITLE>
        ...
</HEAD>

<BODY...>
<A HREF="personen.html" ><img src="personen_off.gif"
                        alt="Personen" border="0"></A>


<!- eigentlicher Text ->
<TABLE CELLSPACING="20" BORDER="0">
        <TR><TD><H3>Universit&auml;t Dortmund-Didaktik der Informatik</H3>
             <HR>
             <H4>Aktuell</H4>
             </TD>
        </TR>

        <TR><TD><H2>Fachtagung Informatik im ZBW "Interaktion und
             Kommunikation" 9./10.11.98 in Soest</H2>
             Auftaktveranstaltung:<BR>
```

...
<A HREF="mailto:horst-dieter.wilkeninguni-bielefeld.de"
>Horst-Dieter Wilkening</A>
</TD>
</TR>
...
</TABLE>
</BODY>
</HTML>

Diese sehr eingeschränkte formale Sprache eignet sich für genau diese spezielle Aufgabenklasse. Mit Programmiersprachen kann sie deshalb nicht verglichen werden. Um die Programmiersprachen entbrennt immer wieder neuer Streit. Die Frage nach der ungeeignetsten ist leichter zu beantworten als die Umkehrung. Mehrsprachigkeit gehört zum Informatikunterricht. Die verschiedenen Paradigmen sind prinzipiell gleich leistungsstark, eignen sich aber für Aufgabenklassen unterschiedlich gut. Diese Erkenntnis und die Fähigkeit zum begründeten Vergleich besitzen einen höheren Bildungswert als die Vertiefung in einem einzigen Paradigma (siehe Abbildung 18). Auf Grund der Bedeutung objektorientierter Denkweisen werden diese im 6. Kapitel ausführ-

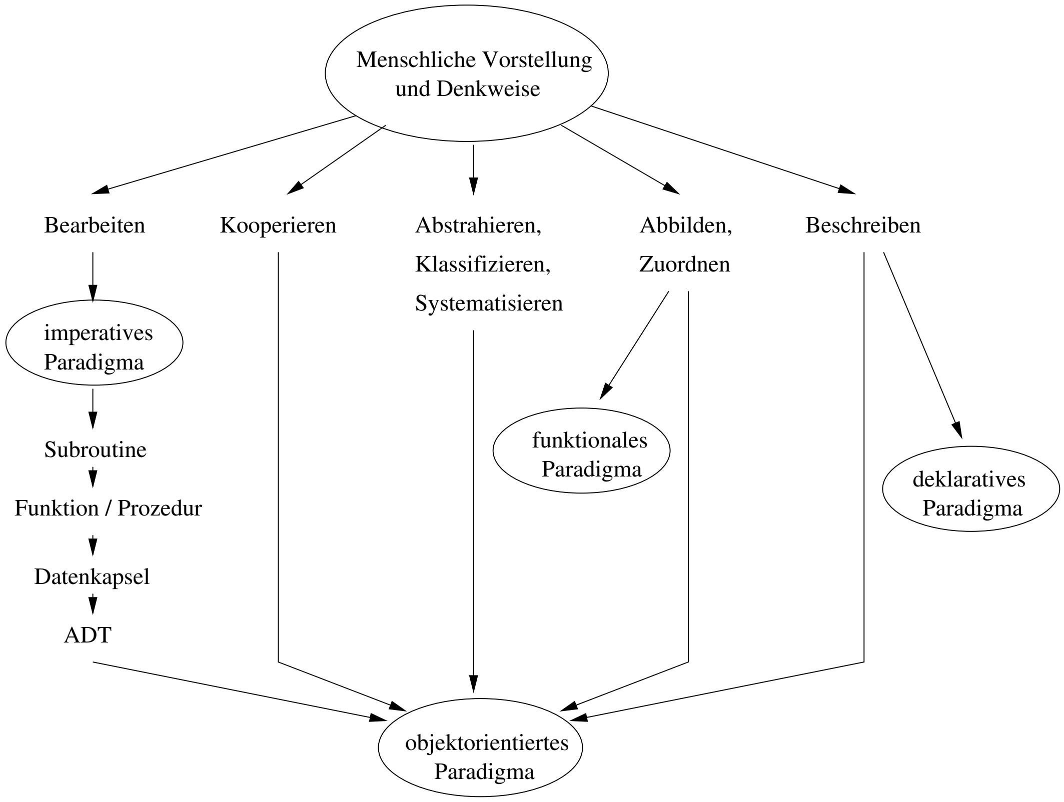


Abbildung 18: Übersicht über die Programmierparadigmen, [Müller92], S. 159

lich vorgestellt. Das imperative Paradigma wird als bekannt vorausgesetzt. Das funktionale und das deklarative (prädikative) Modellieren eignen sich sehr gut als Alternativkonzept. So wird z. B. Rekursion beim deklarativen Paradigma sehr natürlich empfunden, da rekursive Datenstrukturen vorhanden sind. Rekursive Regeln beschreiben Lösungsmodelle sehr prägnant und beeindruckend. Auf das informatische Modellieren im deklarativen Paradigma soll auführlicher eingegangen werden, da es besonders gut geeignet ist, Grenzen des gewählten Weltausschnittes zu veranschaulichen. Einteilung der konvexen Vierecke (siehe Abbildung 19) könnte ein Anfängerbeispiel sein. Objektorientierte Denkweisen lassen die Knoten des Netzes als Objekte erkennen

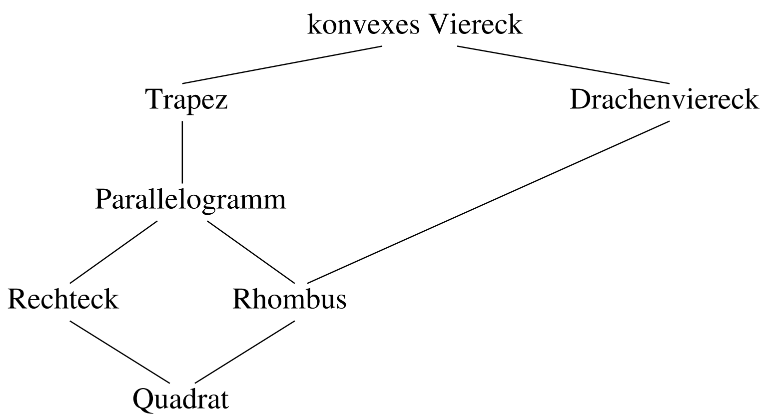


Abbildung 19: Modellierungsgraph

(z.B. Rhombus) und die Kanten als Relationen zwischen den Objekten (z.B. Rhombus ist ein spezielles Drachenviereck). Ziel der Modellierung ist eine Abbildung der Objekte mit ihren Relationen in einem Informationssystem. Der Weltausschnitt wird durch den Bereich der konvexen Vierecke begrenzt. Die Beschreibung aller Kanten führt zu einer Sammlung von Fakten, über denen einfache Anfragen möglich sind. Damit liegt ein Prolog–Programm vor:

Faktenbasis:

ist_ein(trapez,konvexes_Viereck).
ist_ein(drachenviereck,konvexes_Viereck).
ist_ein(parallelogramm,trapez).
ist_ein(rechteck,parallelogramm).
ist_ein(rhombus,parallelogramm).
ist_ein(rhombus,drachenviereck).
ist_ein(quadrat,rechteck).
ist_ein(quadrat,rhombus).

Wurde der 1. Fakt vergessen, liefert die Anfrage:

?-ist_ein(trapez,konvexes_Viereck).

die Antwort „nein“, obwohl die Relation in der realen Welt gültig ist. Der Prolog–Interpreter bringt „nein“ zum Ausdruck: „Diese Relation wurde nicht in das Modell aufgenommen.“ Ohne Verständnis für die Lösungsversuche der virtuellen Prologmaschine werden mit Sicherheit gravierende Modellierungsfehler auftreten. Eine korrekte Faktenbasis liefert im vorliegenden Beispiel keine Relation, die sich über Zwischenstufen ergibt. Da ein Quadrat ein spezieller Rhombus ist, und ein Rhombus ein spezielles Drachenviereck, so ist ein Quadrat auch ein spezielles Drachenviereck. Solche zusammengesetzten Relationen lassen sich mit rekursiven Regeln modellieren:

> X gehört zu Klasse Y, wenn sich eine Klasse Z finden läßt, zu der X gehört, und die zu Y gehört.

Rekursive Regeln:

gehoert_zu:-ist_ein(X,Y).
gehoert_zu:-ist_ein(X,Z), gehoert_zu(Z,Y).

An diesem Beispiel lässt sich eine typische Modellierungsheuristik erlernen:

1. Der Rekursionsabbruch muss vor dem Rekursionsaufruf erreichbar sein.

2. Die Teilziele im Körper rekursiver Regeln sind so anzuordnen, dass der Rekursionsaufruf so lange wie möglich aufgeschoben wird.

Sie folgt nicht aus der Prädikatenlogik 1. Stufe, sondern aus der Regelverarbeitung mit einer virtuellen Maschine (siehe Abbildung 20). Für Anfänger gelingt die Veranschaulichung der Ak-

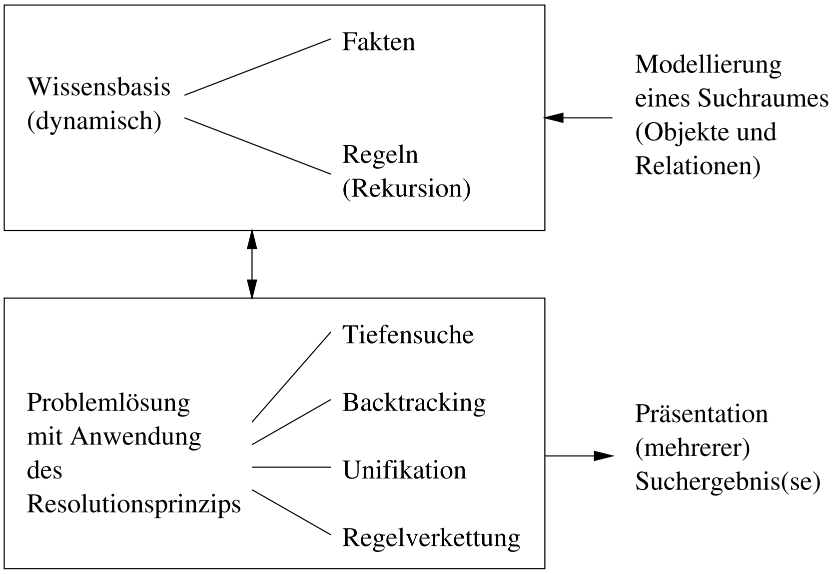


Abbildung 20: Regelsuche mit einer virtuellen Prologmaschine

tionen in einem Beweisbaum, um Wissensbasen (siehe Prolog–Programm auf S. 37) auf ihre Vollständigkeit und Korrektheit zu überprüfen. Aus dem Ableiten der Ziellisten (siehe Abbildung 21) kann die geeignete Heuristik für die Modellierung gefunden werden. Modellierungen im deklarativen Paradigma werden erschwert durch einen prozeduralen Anteil des Konzeptes, ohne den Ein– und Ausgabeströme und Fenstertechnik nicht bereitgestellt werden könnten. Dieser Einsatz von Prädikaten mit Nebeneffekten ist ein notwendiger Kompromiss an die Rechnerarchitektur.

Unter dem Begriff „Maschinelles Lernen“ können selbstmodifizierbare Wissensbasen betrachtet werden, die nach erfolgreicher Modellanwendung neue Fakten erzeugen, mit denen der Anwendungserfolg (bzw. –misserfolg) gemerkt und der künftige Suchprozess gesteuert wird. Die Flüchtigkeit von logischen Variablen liegt in ihrer Verbindung zum Objekt, nicht zur Speicherzelle. Für die informatische Bildung ist die Erkenntnis wertvoll, **dass das System die Aussagen nicht bewerten kann**, sondern nach festem Kalkül verknüpft. Die Strukturierung von Wissen steht im Vordergrund. Von der eigenen Wahl adäquater Sprachelemente für Prädikate wird die Modellierung stark beeinflusst. Diese unmittelbare Verbindung zwischen Modell und System wird in vielen Paradigmen von der Beschreibungssprache vorbestimmt. Prolog lässt in der Syntax viel Freiraum, wenn die Semantik korrekt abgebildet wird.

Eine Reihe von Hypothesen spricht für Veränderungen des informatischen Modellierens im Bereich der Allgemeinbildung, von denen noch keine empirisch untersucht wurde.

Hypothese 1:

> Menschliche Vorstellungs– und Denkweisen finden sich unterschiedlich ausgeprägt in den Programmierparadigmen (imperativ, funktional, deklarativ, objektorientiert) wieder. Dabei nimmt die objektorientierte Modellierung das breiteste Spektrum auf, wird also menschlichen Denkweisen am ehesten gerecht.

Diese Hypothese brachte Müller bereits 1992 [Müller92] in die Diskussion. Sie beeinflusste die Lehrplanentwicklungen und GI–Empfehlungen dieser Jahre, konnte aber von vielen Lehrern noch

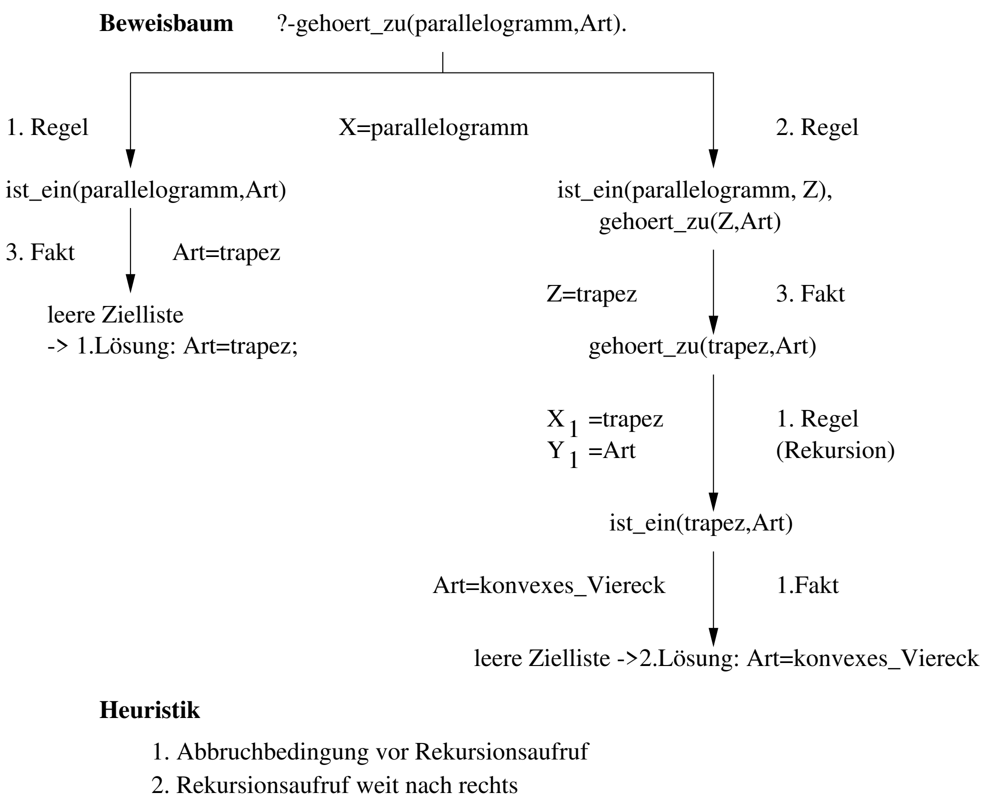


Abbildung 21: Beweisbaum, [Schubert92], S. 179

nicht erprobt werden, da notwendige Fortbildungen dazu fehlten. Eine durchaus interessante Gegenhypothese ist die Bindung der Programmierparadigmen an bestimmte Aufgabenklassen. So lassen sich z.B. Berechnungen schwer in Prolog modellieren. Der imperative und funktionale Lösungsansatz stellen geeignete Strukturen für Algorithmen und Daten dieser Art zur Verfügung. Berechnungen als Klassenhierarchien mit Vererbungsmechanismen zu modellieren ist keinesfalls naheliegend. Bisher erwies sich jede Beschränkung auf ein Paradigma als zu enge Basis für die Modellierung unterschiedlicher Anwendungsbeispiele.

Hypothese 2:

> Informatiksysteme lassen sich klassifizieren als simulierende, registrierende, regelnde, autonome Systeme, die stets über Objekte, deren Eigenschaften und Interaktionen verstanden werden können. Man kann so objektorientierte Modellierungskompetenz ohne objektorientierte Progammierung fördern.

Mit dieser Hypothese begründeten Crutzen und Hein [CH95] 1995 eine objektorientierte Informatikdidaktik, die intensive Diskussionen auslöste und starken Widerspruch weckte. Der Versuch, alle Architekturen objektorientiert zu analysieren, steht im Wiederspruch zu Modellen der Rechnerarchitektur, die mit dem imperativen Konzept sehr anschaulich beschrieben werden konnten. Viele Lehrer suchen die Verbindung erfolgreicher Lehr–Lern–Konzepte mit neuen Entwicklungen ohne starke Verwerfungen im Bildungprozess. Die Verallgemeinerung einer Sichtweise erwies sich im Informatikunterricht bisher stets als Sackgasse, aus der die Umkehr nicht leicht fiel. Interessant ist die Überlegung, dass Modellierungstechnik nicht an ein Programmierparadigma gekoppelt sein muss. Für Beispiele der deklarativen Programmierung kann ein objektorientiertes Modell eine sehr nützliche Basis bilden. Die Verbindung von imperativen Erfahrungen für das objektorientierte Modellieren betonte Wirth 1992 [Wirth92]. Es liegen erfolgreiche empirische Studien zu diesem Vorgehen im Informatikunterricht vor [FS97].

Hypothese 3:

> Objektorientierte Modellierung fällt Anfängern aus psychologischen Gründen am leichtesten, da menschliche Wissenspräsentation über Kategorien/Schemata erfolgt, die eine starke Analogie dazu aufweisen, z. B.:
>
> - Variablen als Wahrnehmungsattribute,
> - Methoden als Funktionsattribute,
> - Vererbung als relationale Attribute.

Diese Hypothese wurde von Schwill [Schwill95] 1995 aus einem Vergleich des objektorientierten Paradigmas mit dem deklarativen gewonnen. Überzeugend gelang ihm der Nachweis der Modellierungskomplikationen im deklarativen Paradigma. Der Schritt vom unstrukturierten Weltausschnitt zum Formalisieren in Hornklauseln aus der Prädikatenlogik gelingt dann leicht, wenn bereits formale Darstellungen wie Hierarchien und Netzwerke vorliegen. Für andere Aufgabenklassen ist der prädikative Ansatz einer Lösung durchaus schwer nachzuvollziehen. Die Entwicklung in der Informatik räumt objektorientierten Denkweisen inzwischen eine so herausgehobene Bedeutung ein, dass es sinnvoll erscheint, diese in einem gesonderten Kapitel zu untersuchen (siehe Kapitel 6). Vorerst bleibt der Vorbehalt, dass nicht alle Aufgaben leicht objektorientiert zu modellieren sind. Das Vorliegen von Klassenhierarchien, aus denen Objektinstanzen abgeleitet werden können, bringt wieder eine Formalisierung, die im Alltag erst erkundet werden muss. Sie ist nicht offensichtlicher als imperative oder deklarative Modellierungen.

6. Objektorientierte Denkweisen

6.1. Prinzip der objektorientierten Programmierung (OOP)

Der Begriff OOP wird mit verschiedener Bedeutung und Tragweite versehen. Das führt zu ähnlichen Missverständnissen, wie sie vom Prinzip der strukturierten Programmierung ausgingen [Schubert91]. Dem versucht man in der Literatur mit einer Fülle ähnlicher Begriffe für einzelne Tätigkeiten zu begegnen, z.B.:

- objektorientierter Entwurf (OOE),
- objektorientiertes Design (OOD),
- objektorientierte Modellierung (OOM),
- objektorientierte Programmierung (OOP).

Hier wird die Sichtweise aus dem Schülerduden Informatik übernommen [CS97], S. 382:

> „Daher lässt sich die objektorientierte Programmierung zusammen mit der strukturierten Programmierung, als deren Fortsetzung sie aufgefasst werden kann, eher in die Klasse der Software–Enticklungsmethoden als in die Klasse der programmiersprachlichen Denkschemata einordnen.“

Zur Unterscheidung vom Programmierkonzept, das stets an konkrete Sprachen gebunden ist, wird im folgenden von objektorientiertem Modellieren (OOM) gesprochen, wenn das Prinzip betont werden soll. OOM erfordert eine Analyse des unstrukturierten Weltausschnittes mit dem Ziel:

1. Objekte zu erkennen und zu beschreiben,
2. den Objekten Eigenschaften zuzuordnen,
3. die Objekte auf Grund ihrer Eigenschaften zu klassifizieren,
4. die entstandenen Klassen und Teilklassen in einer Hierarchie anzuordnen,
5. möglichst viele gemeinsame Objekteigenschaften in abstrakte Klassenbeschreibungen zu verlagern,
6. Operatoren zu bestimmen, die Objekte aus Klassen erzeugen und diese Objekte in der gewünschten Weise beeinflussen.

Am Beispiel der Studierenden im Lehramtstudiengang Informatik für die Sekundarstufe II der Universtät Dortmund wird deutlich, dass die OOM keineswegs genau ein Ergebnis bringt. Die Entscheidung darüber, welche Eigenschaften für die Lösung wichtig sind, hängt vom Anwendungskontext ab. Im Falle einer Studienberatung erscheint die Adresse für die Einladung sinnvoll. Es wird dann davon ausgegangen, dass jeder Studierende eine Postadresse besitzt. Gleiches kann von einer E–Mail–Adresse nicht vorausgesetzt werden. Wichtig kann eine Unterscheidung nach der Studienphase wie Grundstudium, Hauptstudium, Prüfungsphase sein, denn an einer Einladung zum jährlichen Mentorentreffen sind typischerweise Studierende des Hauptstudiums interessiert, die einen Mentor für ihr Blockpraktikum finden wollen. Ein solcher Einladungsbaustein benötigt aktualisierte Daten, die auf Studienfortschritte ebenso eingestellt sind wie auf Veränderungen in der individuellen Berufsplanung. Eine OOM (siehe Abbildung 22) bleibt

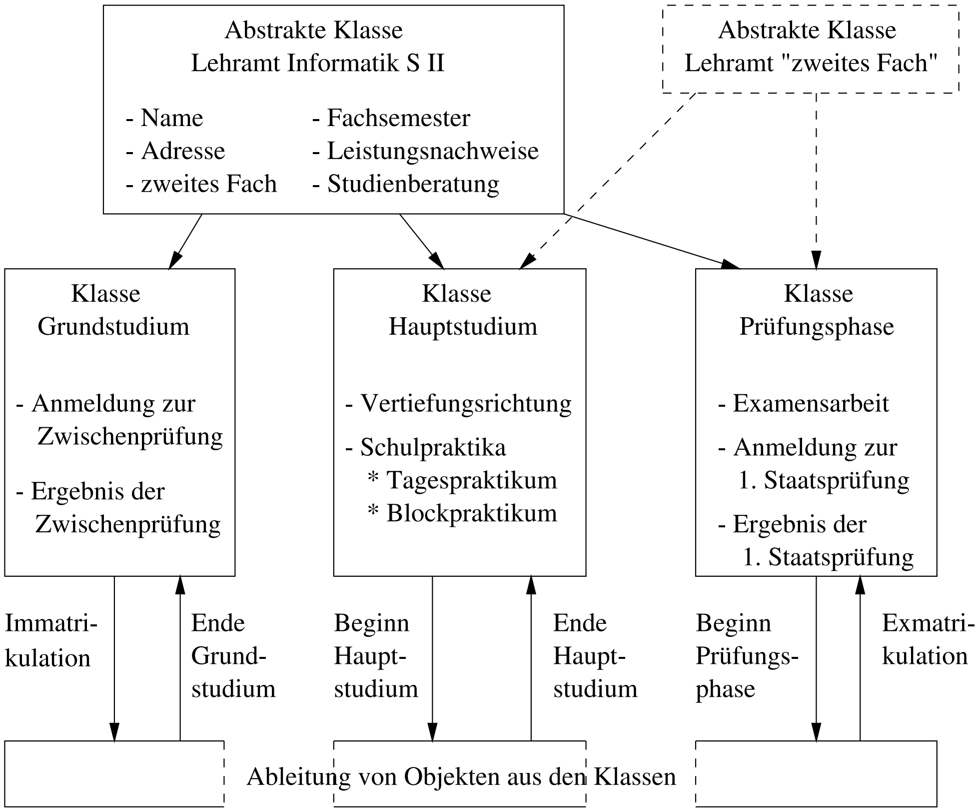


Abbildung 22: Objektorientiertes Modellieren einer Studienberatung

vorerst noch sprachenunabhängig. Ohne Sprachkenntnis bleibt jedoch unklar, welche Informationen in das Modell aufzunehmen sind. Der Vorteil eines solchen Modells liegt in der individuellen Einladung zur Studienberatung immer dann, wenn ein kritischer Studienabschnitt bevor steht. Ebenso wie OOM kommt imperatives, funktionales, deklaratives Modellieren dafür in Frage. In der Informatik werden nach wie vor alle Paradigmen eingesetzt. Für jede Aufgabenklassse kann ein spezielles Paradigma im Vordergrund stehen.

> "Es gibt Problemstellungen, deren Umsetzung geradezu nach objektorientierter Programmierung verlangen, z. B. im Bereich der Simulation oder bei der Entwicklung graphischer Benutzeroberflächen, und andere, bei denen das nicht der Fall ist. Insofern kann tatsächlich der Fall eintreten, dass objektorientierter Entwurf und Programmierung das Problem nicht löst und daher nicht 'funktioniert'." [Claussen93a], S. 2–3.

Wenn das Prinzip der OOP das Prinzip der strukturierten Programmierung ablöst, so ist vorstellbar, dass OOM nicht in das objektorientierte Paradigma führen muss. Allerdings war strukturierte Programmierung sehr eng mit dem imperativen Paradigma verbunden. Dem deklarativen und funktionalen Paradigma wurde die Methode des strukturierten Wachstums zugeordnet. Die Transparenz großer Wissensbasen aus Fakten– und Regelmenge ging dabei leicht verloren. Erklärungskomponenten sollten die Ableitungspfade verständlicher gestalten. Sie erwiesen sich als unzureichend für komplexe Systeme.
Ob die objektorientierten Denkweisen den lehrmethodisch günstigsten Zugang zum Verständnis der Informatik bilden, bleibt vorerst offen. Im Kapitel 5 wurden interessante Hypothesen vorgestellt, deren empirische Überprüfung noch fehlt. Die neuen Lehrplanentwicklungen nahmen OOM als obligatorische Erweiterung oder mögliche Alternative auf. Verwechselt wird teilweise die Bedeutung des OOM und des Modellierens verteilter Systeme (siehe Kapitel 7). Falsche Schwerpunkte bringt ein neuer Sprachenstreit in die Diskussion. Die Suche nach der günstigsten

Programmiersprache verdeckt die viel wichtigere Wahl geeigneter Modellierungstechniken.

6.2. Vorstufe des objektorientierten Modellierens

Eine erste Stufe des objektorientierten Denkens kann für Anfänger beim Modellieren von Lösungen mit typischen Anwendungssystemen entwickelt werden. Bundesweit existieren Phasen der informatischen Bildung, die das Problemlösen mit Informatiksystemen zum Gegenstand haben, nicht das Gestalten von Informatiksystemen. Typischerweise finden solche Lernprozesse in der Sekundarstufe I statt, mit Verlagerungstendenz zur Primarstufe. Sie stehen unter der Kritik, die geistigen Techniken der Informatik zu vernachlässigen und Benutzungsfähigkeit überzubetonen. Deshalb wurde bereits 1992 [Bosler92], S. 40, eine Vorstufe des Objektorientierten Modellierens empfohlen. Am Beispiel der Textverarbeitung (siehe Abbildung 23) wird eine Glückwunsch-

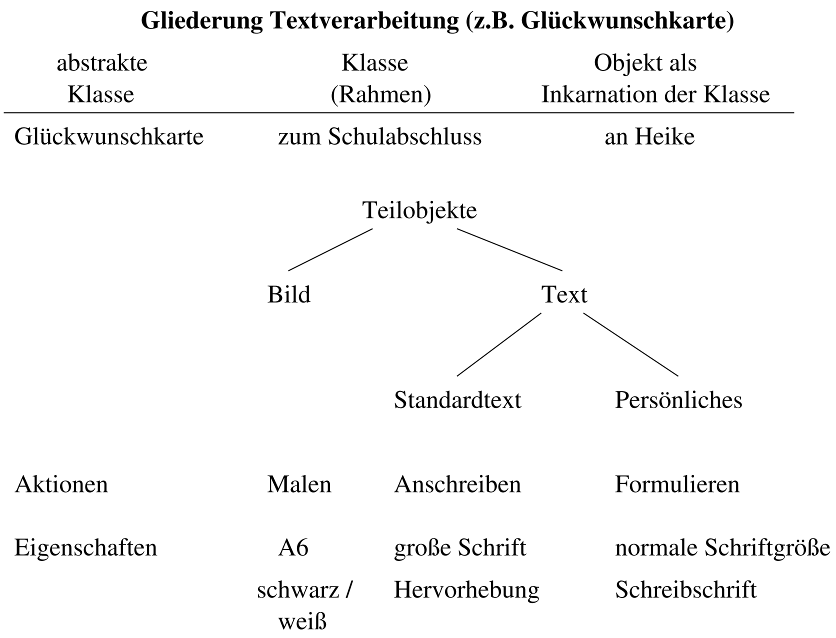


Abbildung 23: Textverarbeitung

karte als Prozess der Inkarnation (inkarnieren=körperliche Gestalt annehmen) gestaltet. Die abstrakte Klasse Glückwunschkarte kann nicht verschickt werden. Eine Unterklasse bilden Karten zum Schulabschluss. Dieser Grund zum Gratulieren steckt den Gestaltungsrahmen ab. Erst der künftige Empfänger ermöglicht die konkrete Realisierung mit dem ganz persönlichen Text und dem Senden an eine Adresse. Die Problemlösung erfolgt über eine praktische Strukturierung der Ziele (Oberklasse, Unterklasse, Objekt), der Aktionen (Zeichnen, Formulieren) und der Eigenschaften. Falsch ist die Verwechslung mit dem objektorientierten Paradigma. Es entstehen keine Objekte, die über Botschaften zur Ausführung von Aktionen angeregt werden. Der Lerneffekt dieser Lehrmethode besteht im Strukturieren der eigenen Arbeitsweise. Typische Tätigkeiten der Textverarbeitung werden unabhängig vom konkreten System zu einer Handlungsorientierung verbunden. Im Vordergrund stehen die Handlungsziele. Nachgeordnet wird die Oberflächengestaltung von Benutzungsmöglichkeiten. Ein weiterer Vorteil besteht in der Übertragung auf andere Anwendungsaufgaben, wie Tabellenkalkulation (siehe Abbildung 24) und Datenbanken (siehe Abbildung 25). Es fällt den Schülern leichter, die neuen Aktionen in einem vertrauten Handlungsrahmen zu erkunden und anzuwenden. In der Tabellenkalkulation fördert die Analogie von Klasse und Objekt die Sicht auf den Wechsel von der Formelebene zur Werteebene (Ergebnis der Formelanwendung). Die Datenbankmodellierung kommt der OOM schon sehr nahe. Hier wird ein Strukturrahmen definiert, der mit konkreten Datensätzen gefüllt wird. Die Klasse entspricht tatsächlich der geistigen Vorwegnahme, der Vision eines Objektes, das noch nicht existiert. In einer leeren Schülerbibliothek sind keine Buchausleihen möglich.

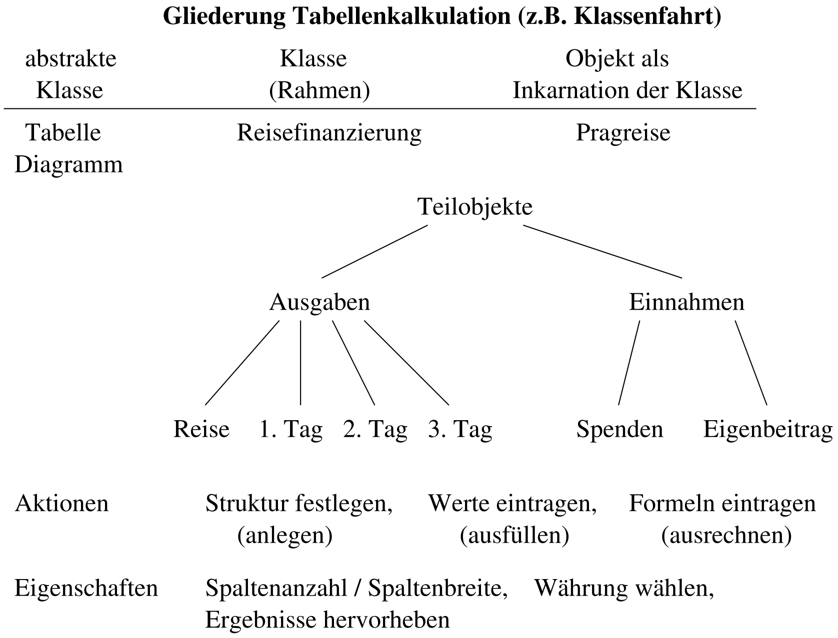


Abbildung 24: Tabellenkalkulation

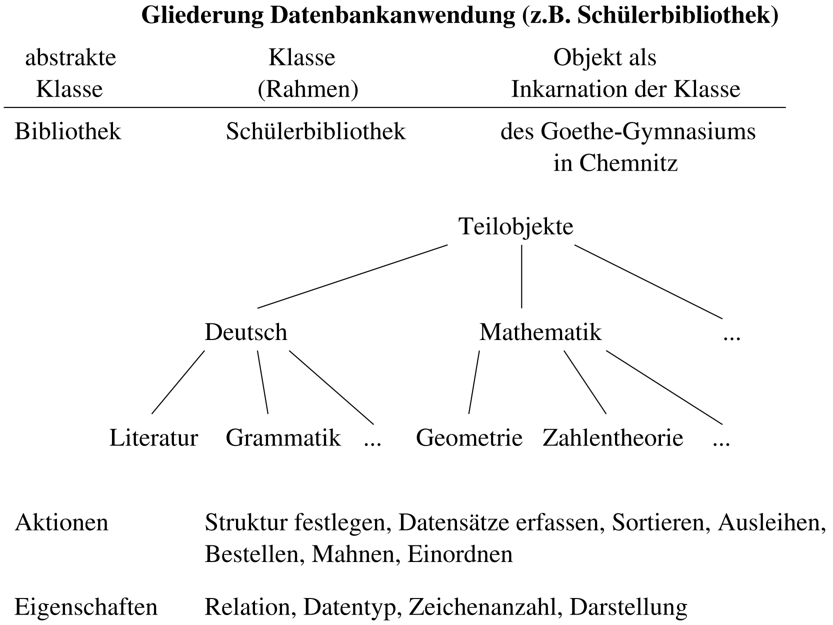


Abbildung 25: Datenbanken

Die Grundbegriffe Klasse und Objekt strukturieren die Anwendungsaufgaben und zeigen deren Komplexität auf. Mit den Aktionen und Eigenschaften wird die Arbeitssystematik verbessert. Schülertätigkeiten orientieren sich dadurch an Grundfunktionen und deren Modifikationen. Offen ist die Frage der Lernzielabgrenzung zwischen informatischer Grundbildung und den Anwendungsfächern. Wieviel Informatik benötigt ein Schüler vor dem Einsatz von Informatiksystemen in anderen Fächern? Im Kapitel 8 wird darauf tiefer eingegangen. Gezeigt hat sich, dass keine gesonderte Benutzungsschulung für diese Werkzeuge erforderlich ist. Ohne Verständnis für Daten (einschließlich Modellierung und Speicherung) entwickeln sich leicht chaotische Arbeitsweisen. Ohne Vorstellung von virtuellen Maschinen (der Informatik) stellen sich typische Erwartungs–

und Planungsfehler ein.

6.3. Entwicklung des objektorientierten Paradigmas

Die These von der Weiterentwicklung der strukturierten Programmierung wird nachvollziehbar, wenn man die Vorstufen der OOP näher betrachtet. Ihre Thematisierung im Informatikunterricht entspricht einer Linie mit steigender Abstraktion und beantwortet die Frage nach den zeitbeständigen Inhalten. Zur Komplexitätsbewältigung und der damit verbundenen Fehlerreduzierung wurde die funktionale Abstraktion als Grundkonzept zur Beschreibung benutzerdefinierter Operationen entwickelt. Die Schüler lernen diese Möglichkeit zur Konstruktion wiederverwendbarer Lösungsbausteine im imperativen Paradigma in Form von Prozeduren und Funktionen kennen. Die Anwendung scheitert nicht selten am Verständnis des Parameterkonzeptes. Die Entscheidung für Wert– und Referenzparameter wird dann ohne Wissen um die damit verbundenen Konsequenzen getroffen. Als Lernhilfe wurde das Modell eines Hauses mit verschiedenen Räumen vorgestellt, an deren Türschildern man die aktuellen Parameterbelegungen ablesen kann, die jeweils im Rauminneren erzeugt wurden. Das erwies sich bei Referenzparametern eher als Hürde im Lernprozess. Informatiklehrbücher wurden von Lehrern in der Vergangenheit nicht selten mit der Begründung abgelehnt, dass darin Denkmodelle als didaktische Vereinfachung vorgestellt wurden, die gegen fachliche Prinzipien und Methoden verstießen. Auf Abstraktion haben Fachwissenschaft und Lehrdisziplin eine unterschiedliche Sichtweise. Die Abstraktion vereinfacht Entwicklungsprozesse im Fach generell, wenn sie richtig verstanden wird. Aber gerade damit hat die Lehrdisziplin große Schwierigkeiten. Wissen und Können entstehen in einem Lehr–Lern–Prozess, der von sehr anschaulichen Einzelbeispielen beginnend die Abstraktionsstufen entwickeln lässt. Die Konzentration auf das Wesentliche erfordert Verständnis für die Auswahl genau dieser Kriterien. Da Kapselung ein wichtiges Prinzip der Informatik ist, lohnt es, den Entwicklungsweg transparent zu machen und die Etappen zu beurteilen. Die funktionale Abstraktion realisiert gekapselte Daten und Operationsfolgen, aber lokal sichtbare Daten existieren nicht über die Ausführungszeit der Operationen hinaus. Eine Datenkapsel kann als Verbesserung des Konzeptes verstanden werden, da sie auch die funktionale Abstraktion kapselt. Die „Units“ in Pascal und die Module in Modula stellen Datenkapseln zur Verfügung. Als nachteilig erwies sich, dass nur genau ein Exemplar einer solchen Struktur beschreibbar ist. Es fehlt also ein Konzept, das den Export eines Datentyps ermöglicht. Mit dem abstrakten Datentyp (ADT) wurde genau das möglich. Obwohl die Vorteile offenkundig sind:

- beliegig viele Exemplare (Instanzen) vom Typ ableitbar,
- hoher Grad der Abstraktion für benutzerdefinierte Datentypen erreicht,

bleibt der Nachteil, dass keine Ableitungen von Typen aus einem Basistyp möglich sind.

Im Informatikleistungskurs erwies sich das Konzept des abstrakten Datentyps als erlernbar und anwendbar. Da keine Vorkenntnisse zum Fach Informatik aus der Sekundarstufe I erwartet werden können, entschied man sich in vielen Bundesländern für genau ein Paradigma der Informatik, das imperative (prozedurale) Paradigma. In diesem wurde das erforderliche Abstraktionsniveau systematisch bis zu ADT und dynamischen Datenstrukturen über die möglichen sechs Kurshalbjahre entwickelt. Bei der Euphorie über das hohe fachliche Niveau in einem Teilgebiet der Informatik wurde übersehen, dass zunehmend weniger Schüler bereit waren, ein Schulfach zu **wählen**, in dem es sehr schwer ist, Spitzenleistungen zu erzielen. Die Abiturstrategie zeigte in vielen Bundesländern, dass die Informatik in der Sekundarstufe II in hartem Wettbewerb mit attraktiveren Fächern bestehen muss. Das objektorientierte Konzept bot sich förmlich an. Die Nachteile des ADT werden überwunden durch:

- Vererbung von Schnittstelle und Funktionalität des Basistyps,

- Modifikationsmöglichkeiten des abgeleiteten Typs.

Außerdem versprachen die Experten des Faches große Erleichterungen in der Software–Entwicklung durch Anwendung des objektorientierten Paradigmas. Inzwischen existieren in der Sekundarstufe II der Bundesländer unterschiedliche Empfehlungen zur OOP. Die Machbarkeit wurde sowohl für den Ersatz des imperativen Paradigmas durch OOP als auch für die Verbindung beider Paradigmen nachgewiesen [Czischke97], [Thüringen98].

Wenn die folgende These als korrekt angenommen wird:

> „**These 2.1**
> Objektorientierter Entwurf ist die Entwicklung von Softwaresystemen als strukturierte Sammlung von Implementierungen abstrakter Datentypen.“ [Claussen93a], S. 15,

dann wird OOP keineswegs leichter erlernbar sein, als das Basiskonzept ADT. Die beobachtete bessere Motivationslage kann also durchaus andere Ursachen haben. Z. B. kann die Begeisterung der Lehrer für das neue Paradigma einen wichtigen Einfluss auf ihren Unterricht ausüben. Was nach einem erfolgreichen Start in die OOM in den folgenden Kursen darauf aufbauen soll, wird noch heftig diskutiert. Die erforderlichen Fähigkeiten, Methoden mit prozeduralen Strukturen auszugestalten, werden nun im Kontext OOM erlernt. Der historische Zugang zu OOP in der oben diskutierten Form bleibt den Schülern verborgen. Das muss kein Nachteil sein. Eine Evaluierung alternativer Lernprozesse steht noch aus. Sie wird erschwert durch die geringen Schülerzahlen im Informatikunterricht.

6.4. Objektorientiertes Modellieren

Im Kapitel 4 wurde das Zustandsraummodell als Grundvorstellung für das Problemlösen diskutiert (siehe Abbildung 8 auf S. 25). Fasst man einen Knoten des Graphen als Objekt auf, das sich in einem Zustand z_n befindet, bis eine Botschaft b_1 es zur Veränderung zwingt, also in einen Zustand z_{n+1} überführt, so liegt der OOM das Modell des endlichen Automaten zugrunde. Der Programmablauf wird durch den Austausch von Botschaften gesteuert (siehe Abbildung 26). Die Berechnung kann parallel ausgeführt werden, da die Objekte ihre Botschaften unabhängig

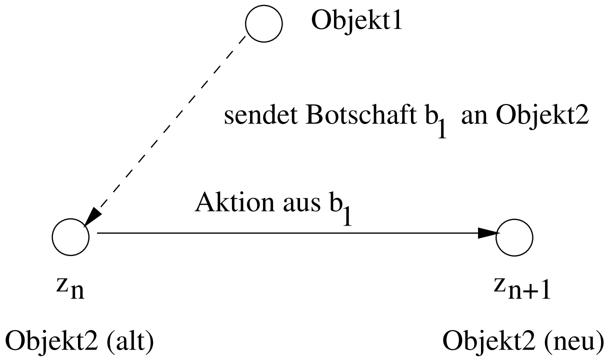


Abbildung 26: Programmschritt im OOM

voneinander versenden können. Die Reihenfolge ergibt sich aus der Anordnung der Botschaften im Programmtext. Für die Modellierungsphase wurden von Blooch, Jacobson und Rumbaugh [BJR96] grafische Hilfsmittel in Form einer einheitlichen Modellierungssprache UML (unified modeling language) entwickelt, die sich sehr gut zur Veranschaulichung von Ereignissteuerung (siehe Abbildung 27 und Zustandwechsel (siehe Abbildung 28) eignen.

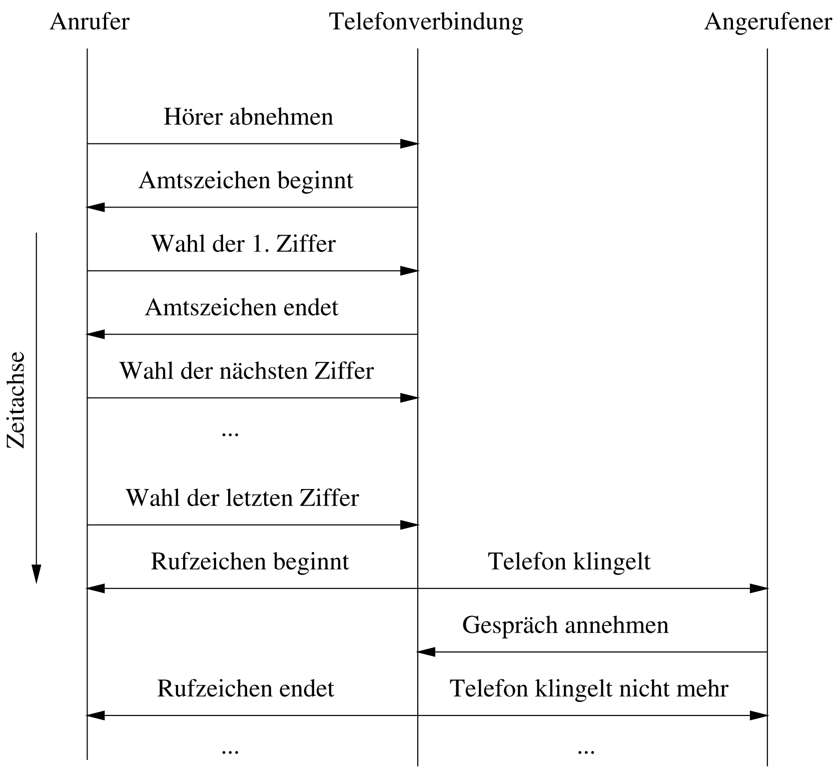


Abbildung 27: Ereignissteuerung [BR95], S. 25

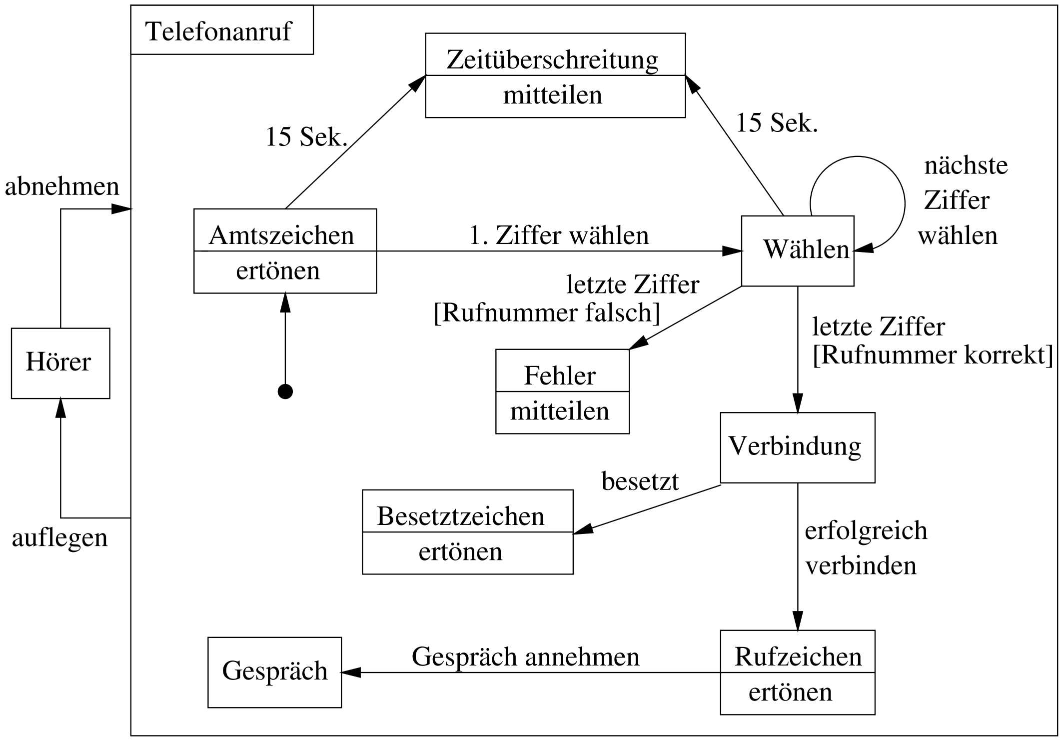


Abbildung 28: Zustandswechsel [BR95], S. 37

Eine Reihe neuer Grundbegriffe ist zu behandeln:

1. Objekte (abgeschlossene und autonome Bausteine) und Verfahren (Operationen, Methoden) müssen definiert werden. Die Autonomie der Objekte bedingt Nebenläufigkeit. Das Objekt entspricht der Variablen.

2. Vererbungsmechanismus:
   Objektarten können einander Eigenschaften (Angaben) „vererben". Man findet diese Möglichkeiten z. B. in Smalltalk, Eiffel, Turbo Pascal, Oberon.

3. Botschaften:
   sind die Kommunikationseinheiten zwischen den Objekten (siehe Abbildung 29). Objekte senden einander Mitteilungen, die die Veränderung der Daten bewirken.

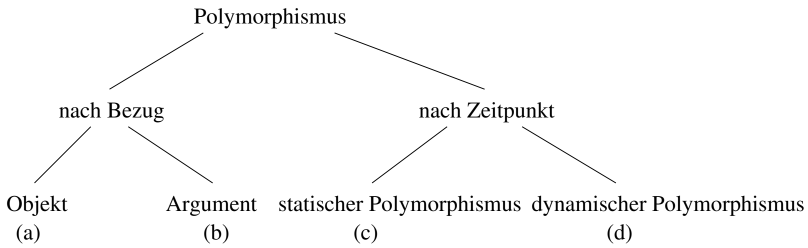
Botschaft: Interaktionseinheit zwischen den Objekten

Adressat	Selektor	Argumente
Zielobjekt	Operation (Methode)	Parameter für Operation

Abbildung 29: Botschaft

4. Klassen:
   Die Klasse entspricht dem ADT. Jedes Objekt entsteht durch Inkarnation aus einer Klasse (Objektart). Eine Klasse beschreibt Instanzvariablen (gekapselte Datenstruktur in den Objekten) und Methoden (die über den Variablen definierten Operationen).

5. Polymorphismus (viel Erscheinungsformen, siehe Abbildung 30):



(a) Variablen können verschiedene Typen annehmen.

(b) Funktionen können für verschiedene Parametersätze mit gleichem Namen definiert sein.

(c) gilt für (a) und (b).

(d) Operationen können sich auf Instanzen verschiedener Klassen zur Laufzeit beziehen.

Abbildung 30: Arten des Polymorphismus

Deutlich wird, dass es nicht ausreicht, anschauliche Klassenhierarchien zu skizzieren, um OOM zu thematisieren. Das kann für das Erlernen von Mehrfachvererbung aber durchaus hilfreich sein (siehe Abbildung 31).

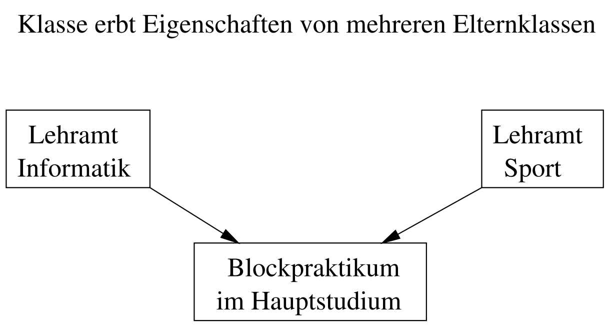


Abbildung 31: Mehrfachvererbung

Die Manipulation mit rechteckigen Gebilden soll automatisiert werden [Müller92]. Gegeben sind die kartesischen Koordinaten Punkt (Ursprung), Länge und Breite. Die Kanten sollen parallel zu den Koordinatenachsen liegen. Dann sind drei Operationen erforderlich (siehe Abbildung 32):

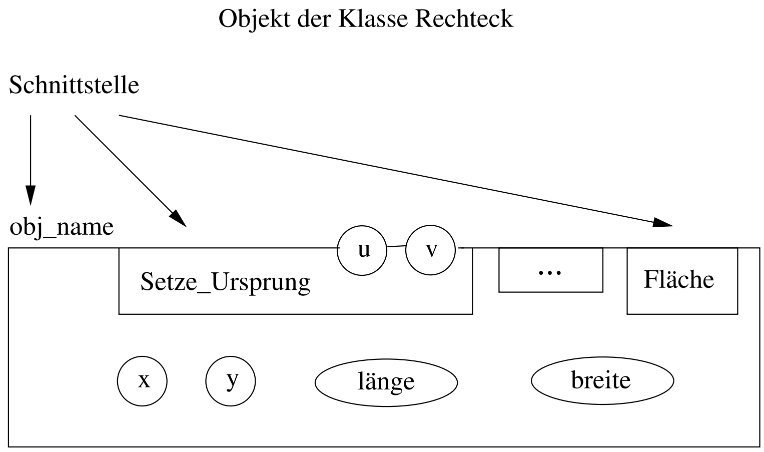


Abbildung 32: Beispiel zur Beschreibung von Rechtecken

1. (neu) Festlegen von Ursprung, Länge und Breite,
2. Zeichnen des Rechtecks,
3. Angabe des Flächeninhalts.

Das führt zur Klassendefinition:

```
type Rechteck = objekt
   constructor Init(u, v, l, b : real);
   procedure SetzeUrsprung(u, v : real);
   procedure SetzeLaenge(l : real);
   procedure SetzeBreite(b : real);
   procedure Zeichne;
   function Flaeche : real;
private
    x, y : real;
    laenge : real;
    breite : real;
end;
```

Die Methode „Init" bildet den sogenannten Konstruktor. Der Konstruktor wird benutzt, um die Objekte der entsprechenden Klasse zu initialisieren. Die Instanzvariablen werden mit „private" nach außen verborgen.

$constructor\ Rechteck.Init(u, v, l, b\ :\ real);$
$begin\quad x := u;\quad y := v;\quad laenge := l;\quad breite := b\quad end;$
$procedure\ Rechteck.SetzeUrsprung(u, v\ :\ real);$
$begin\quad x := u;\quad y := v\quad end;$

„Klassenname.Methodenname" ist erforderlich, da gleiche Methodennamen in unterschiedlichen Klassen verwendet werden können (Objekt–Polymorphismus). Von diesem neu definierten Datentyp können wir Objekte erzeugen (wie Variablenvereinbarungen):

$var\quad r1\ :\ Rechteck;$

Der Aufruf des Konstruktors erfolgt mit:

$r1.Init(1, 1, -2, 3);$

Damit wird die Botschaft „Init" mit den angegebenen Parametern an das Objekt „r1" gerichtet (siehe Abbildung 33). Da eine reale Klasse stets eine Vision, eine geistige Vorwegnahme von

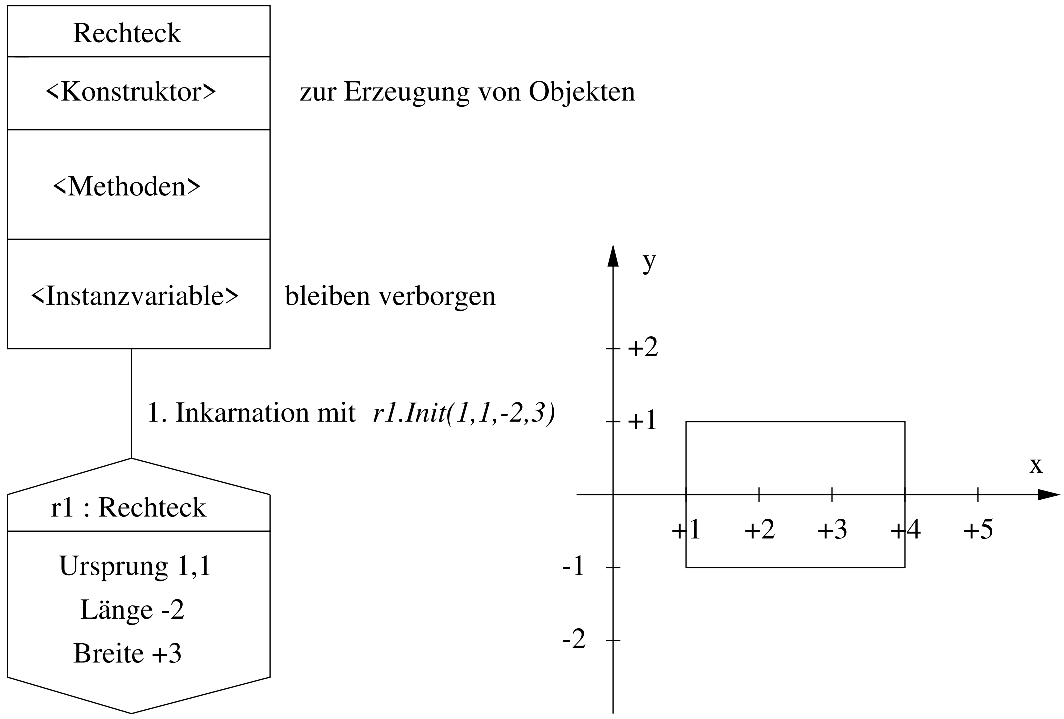


Abbildung 33: Inkarnation eines Objektes

realen Objekten ist (analog zum Bauplan), kann mit Klassen nicht manipuliert werden. Erst wenn durch Inkarnation Objekte abgeleitet wurden, können diese Beispielrechtecke gezeichnet werden.

$r1.SetzeLaenge(5);$
$r1.SetzeBreite(3);$

```
procedure Rechteck.SetzeLaenge(l : real);
begin   laenge := l   end;

procedure Rechteck.SetzeBreite(b : real);
begin   breite := b   end;
```

Quadrate werden durch Vererbung als spezielle Rechtecke modelliert:

```
type Quadrat = objekt(Rechteck)
   constructor Init(u, v, l : real);
   procedure SetzeLaenge(l : real);
   procedure SetzeBreite(b : real);
end;
```

Die Besonderheiten werden aus der Vererbung herausgenommen (programming by difference):

```
constructor Quadrat.Init(u, v, l,  : real);
begin     Rechteck.Init(u, v, l, l)     end;

procedure Quadrat.SetzeLaenge(l : real);
begin   Rechteck.SetzeLaenge(l);
        Rechteck.SetzeBreite(l)
end;

procedure Quadrat.SetzeBreite(b : real);
begin   SetzeLaenge(b)   end;

var   r : Rechteck;
      q : Quadrat;
begin   r.Init(1, 1, 2, 3);
        q.Init(−1, −1, 2);
        r.SetzeBreite(5);
        q.SetzeBreite(5)
end.
```

Gleichlautende Botschaften (siehe Abbildungen 34 und 35) werden individuell interpretiert (sta-

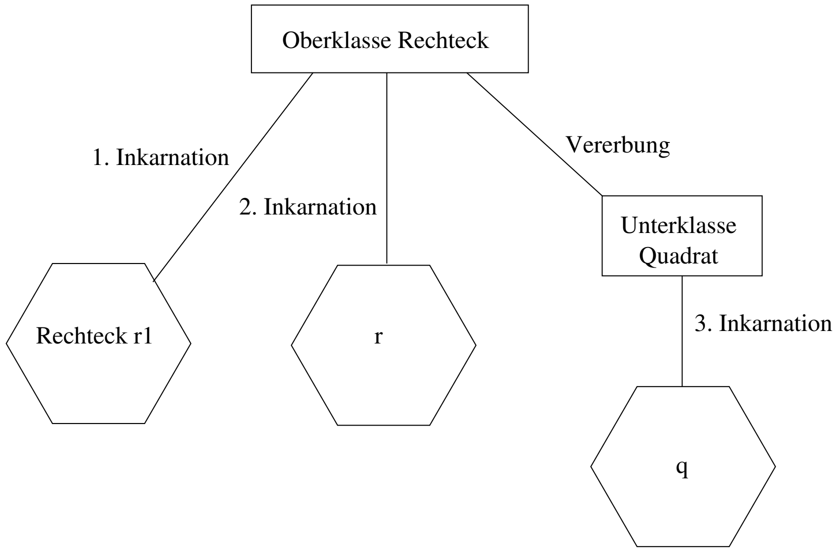


Abbildung 34: Statischer Objekt–Polymorphismus

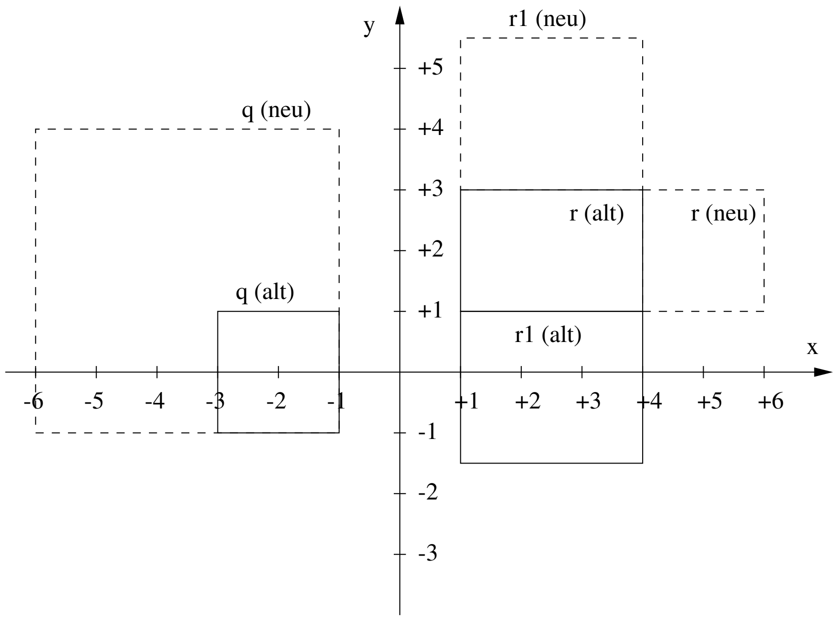


Abbildung 35: Wirkung des statischen Objekt–Polymorphismus

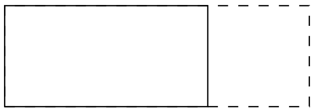
tischer Objekt–Polymorphismus), z. B.

$r1.SetzeLaenge(5);$
$r1.SetzeBreite(3);$

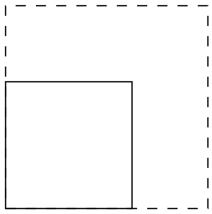
Mit virtuellen Methoden kann die Entscheidung ihrer Anwendung auf Oberklasse oder abgeleitete Unterklasse zur Ausführungszeit der Methode getroffen werden (siehe Abbildung 36). Da der Zeiger immer genau auf ein Objekt zeigt (siehe Abbildung 37), ist die Methode stets ausführbar.

Zeiger auf Objekte der Klasse Rechteck: *var pointer.^Rechteck;*
gleichlautende Botschaft: *pointer^.SetzeBreite(10);*

1. Fall
Rechteck:



2. Fall
Quadrat:



Entscheidung zur Laufzeit:

procedure SetzeLaenge(l:real)virtual;
procedure SetzeBreite(b:real)virtual;

Abbildung 36: Virtuelle Methode

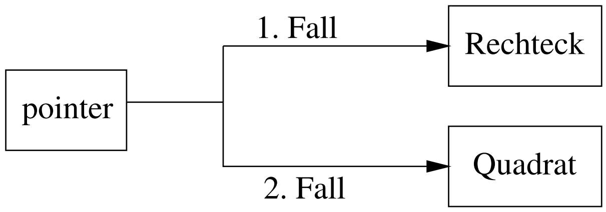


Abbildung 37: Dynamischer Objekt–Polymorphismus

7. Interaktion und Kommunikation

7.1. Begriffsbildung

Lange Zeit wurde in der Informatik von Mensch–Maschine–Kommunikation [Charwat94] und von Datenkommunikation gesprochen. Zunehmend setzen sich in der deutschsprachigen Fachliteratur die Begriffe Mensch–Maschine– bzw. Mensch–Computer–Interaktion und Datenaustausch durch. Damit soll deutlich gemacht werden, dass der Kommunikationsbegriff menschliche Partner voraussetzt. Die Arbeit des Menschen mit Informationssystemen (vereinfacht Computern) zeigt neue Wechselbeziehungen, die man mit dem Begriff Interaktion davon unterscheiden möchte. In der englischen Literatur werden diese Begriffe häufig synonym verwendet [Halsall95]. Informatiklehrer sollten diese Sprachprobleme im Unterricht thematisieren. Eine Lösung des Problems ist mit dem oben genannten Kompromiss noch nicht gelungen, denn laut Duden [Drosdowski89], S. 773 ist „Interaktion aufeinander bezogenes Handeln zweier oder mehrerer Personen, Wechselbeziehung zwischen Handlungspartnern". „Sprachliche Kommunikation ist die wichtigste Form menschlicher Interaktion". Es kann hineingedeutet werden, dass es Interaktion mit Handlungspartnern geben kann, die keine Menschen sind. Aber ob das der Weg zum Verständnis der Arbeit mit Informationssystemen wird, bleibt unklar. In diesem Text wird Interaktion für die neue Wechselbeziehung zwischen Mensch und Informatiksystem gewählt und Kommunikation bleibt im traditionellen Bereich [Drosdowski89], S. 865: „Verständigung untereinander, zwischenmenschlicher Verkehr besonders mit Hilfe von Sprache, Zeichen". Wir benötigen beide Begriffe. In der Entwicklung der Informatik nimmt die Interaktion den historisch längeren Zeitraum ein. Vom ersten Computer (synonym Rechner, besser Informatiksystem) an bestand die Notwendigkeit, diese Systeme zu steuern mittels Interaktion. Die freie Programmierbarkeit des Computers stellte diese Maschinen auf ein besonderes Steuerniveau. Die ursprüngliche Trennung in solche Berufsbilder wie Systemanalytiker, Programmierer und Bediener (Operator) verschwand mit der Einführung der Arbeitsplatzcomputer weitgehend. Systemanalytiker entwarfen die Algorithmen und Datenstrukturen für einen Anwendungsauftrag. Programmierer erzeugten daraus Programmcode in der gewünschten Programmiersprache und implementierten diesen bis zur funktionstüchtigen Software. Bediener großer Rechenanlagen führten die fertigen Anwendungen im Routinebetrieb aus. Mit dem neuen Berufsbild des Informatikers wurde die Arbeitsteilung so umgesetzt, dass die Anwender aller Berufsgruppen mit der von Informatikern geschaffenen Software selbständig ihre Aufgaben lösen. Die Interaktion mit Informatiksystemen wird deshalb nach Entwicklern und Anwendern unterteilt. Die Systemkomponente für diese Interaktion wird Benutzungsschnittstelle genannt. Graphische Benutzungsschnittstellen mit Fenstertechnik, Menüführung und Bildern (Ikonen) haben sich durchgesetzt. Die Bilder stellen ein Brücke zum traditionellen Arbeitsprozess her, z. B. Ordner, Dokument, Papierkorb, Koffer. Entwicklungswerkzeuge und Anwenderysteme unterscheiden sich kaum noch in den Benutzungsschnittstellen, so verschieden die Fuktionalität sein mag. Die Darstellung der Interaktionsmöglichkeiten hat eine weitgehende Vereinheitlichung erfahren. Die Interaktion selbst wird bestimmt von der zu lösenden Aufgabenklasse und den Ein– und Ausgabemöglichkeiten. Die Kommunikation mittels Informatiksystem (Telekommunikation) gehört zu den neuen Entwicklungen in der Informatik [Krüger95]. Die Glasfaserübertragung und die Satellitentechnik brachten den Wechsel von der analogen zur digitalen Telekommunikation (Telematik für Telekommunikation und Informatik). Danach entstand der Bedarf neuer Software für traditionelle Kommunikationsdienstleistungen. Die E–Mail wurde anfangs zu einer Alternative der Postkarte. Nachdem Verschlüsselungsmechanismen einsetzbar wurden, konnte die Vertraulichkeit des Briefes nachgebildet werden. Seriöse Geschäftsbeziehungen konnten erst mit der rechtskräftigen Verbindlichkeit der digitalen Signatur über ein Netzwerk wie das Internet aufgebaut werden. Das „Gesetz zur Regelung der Rahmenbedingungen für Informations– und Kommunikationssdienste (IuKDG)" vom 13. Juni 1997 [Bundestag97] brachte die erforderliche Rechtssicherheit für die veränderten Kommunikationsmöglichkeiten.

7.2. Dialogsysteme als Unterrichtswerkzeug

Informatikunterricht ohne Computernutzung ist wie Schwimmunterricht ohne Wasser im Schwimmbecken möglich, aber nicht empfehlenswert. Der Computer wird in diesem Text meist als Informatiksystem bezeichnet, um das Wechselspiel von Hard– und Softwaresystemen zu betonen. Mit der Nutzung von Informatiksystemen entstanden Deutungsmuster wie Automat, Werkzeug, Partner, Denkzeug. Sie sollten helfen, die neuen Tätigkeiten des Menschen in vorhandene Denkstrukturen einzuordnen, führten aber auch zu falschen Erwartungshaltungen. Die Werkzeugsicht brachte eine Unterschätzung der freien Programmierbarkeit. Die Partnersicht überschätzte die Interaktionsmöglichkeiten auf der Basis formaler Sprachen. Ob das Kunstwort Denkzeug eine nützliche Metapher bildet, bleibt abzuwarten. Der Dialogbetrieb [CS97], S. 55 ist die typische Betriebsart in der Schule. Der Mensch erteilt seine Aufträge meist über Tastatur und Maus. Das Informatiksystem bearbeitet diese Aufträge und liefert die Ergebnisse wahlweise über Bildschirm und / oder Drucker an den Menschen (siehe Abbildung 38).

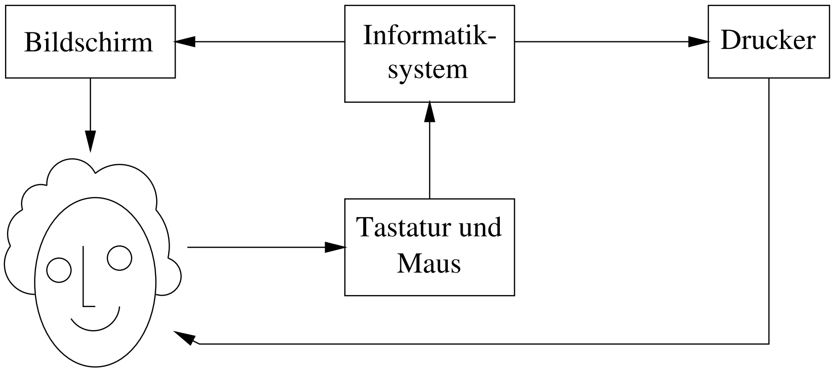


Abbildung 38: Dialogsystem

Aus dieser Interaktion mit dem Informatiksystem wurden Begriffe wie interaktive Systeme und Dialogsyteme abgeleitet. Anfangs standen Einzelcomputer im Informatiklabor. Heute werden meist lokale Netzwerke genutzt. Das Informatiklabor der Schule ist die Lehr–Lernumgebung für dieses Fach und kann den Anforderungen nur gerecht werden, wenn Gestaltungsregeln beachtet werden (siehe Abbildung 39).

Besonders hinderlich ist der Bildschirm als Barriere zwischen den Menschen. Deshalb empfiehlt sich eine Gestaltung des Informatiklabors mit zentralem Kommunikationbereich und dezentralen Interaktionsbereichen. Der Demonstrationsbereich des Lehres bleibt an der traditionellen Spitze. Wesentlich ist die Verbindung der beiden Arten von Schülerarbeitsplätzen. Eine Stuhldrehung muss ausreichen, um vom zentralen Gruppenbereich zum dezentralen Experimentierplatz zu wechseln. Wenn das ohne Zeitverlust möglich wird, kann der Informatiklehrer variantenreichere und in sich abgestimmtere Unterrichtssequenzen gestalten. Bisher muss der Wechsel zwischen den planenden (und diskutierenden) Phasen und den realisierenden und experimentellen Phasen sehr sparsam eingesetzt werden, da der Zeit– und Konzentrationsverlust zu hoch ist. Unverzichtbar ist der benachbarte Vorbereitungsraum mit Lehrmaterialentwicklungsplatz und Lehrmaterialsammlung (Hard– und Software, Lehrbücher, Arbeitsblätter, Videos). Hier kann die ungestörte Arbeit der Lehrer außerhalb des Unterrichts stattfinden. Servicepersonal stört den Unterricht nicht, wenn es im Vorbereitungsraum Wartungs– und Entwicklungsarbeiten am Netzwerk ausführt.

Für das lokale Netz des Informatiklabors haben meist Informatiklehrer die Netzwerkadministration übernommen, ohne dass die Frage der Anrechnung auf das Arbeitspensum geklärt wurde. Mit der Administration des Schulnetzes muss ein Informatiker bertraut werden. Im Berufsfeld des Lehrers ist dafür kein Platz. Es ziehen sich bereits Lehrer aus dem Fach Informatik zurück, weil sie nicht zum Schulpostmeister für die Verwaltung von ca. 1000 E-Mail-Adressen der Schüler

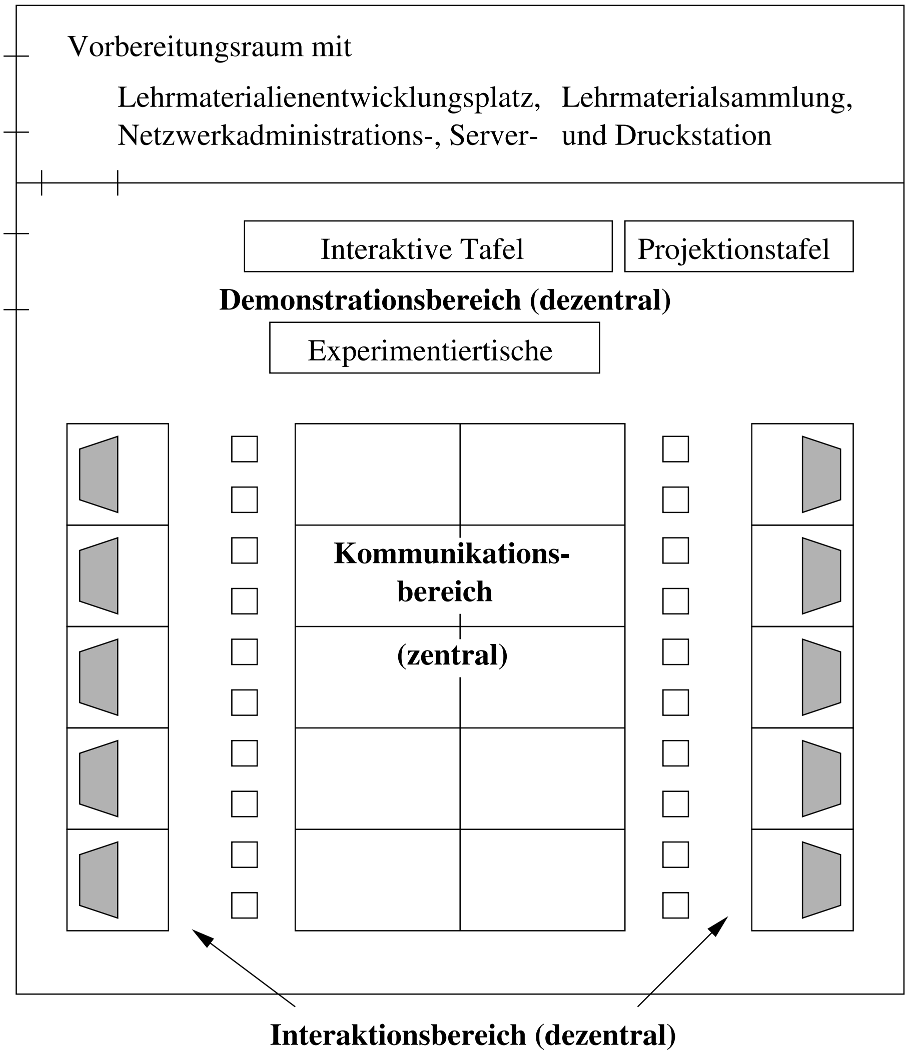


Abbildung 39: Informatiklabor

und für den daraus resultierenden Postversand werden wollen.

Die Einsatzumgebung von Informatiksystemen entwickelt sich in der Schule vom Einzelplatzsystem über das lokale Netz des Informatiklabors zum Intranet der Schule (siehe Abbildung 40)

Es ist kein Internetanschluss erforderlich, um ein leistungsstarkes Kommunikationsnetz für Schulen auszugestalten. Der Internetanschluss stand vielen Schulen zuerst an einem Einzelplatzsystem zur Verfügung. Meist blieb es danach den Lehrern über lassen, das lokale Labornetz mit dem weltweiten Netz zu verbinden. Eine Netzinfrastruktur, die auf die Bedürfnisse der Schule abgestimmt wurde, fehlt heute noch weitgehend. Dabei handelt es sich um eine interdisziplinäre Aufgabe, die nur von Informatikern und Lehrern gemeinsam gelöst werden kann. Den Schülern blieben die Probleme der komplexen und störanfälligen Netzwerke nicht verborgen. Das funktionstüchtige Dialogsystem als Unterrichtswerkzeug ist nicht an den Internetzugang gebunden. Das lokale Netz bringt tatsächlich Kooperationsmöglichkeiten und Organisationsvorteile, die der Einzelplatz nicht bereitstellen kann. Die Schülertätigkeiten werden deshalb für ein vernetztes Dialogsystem betrachtet. Typischerweise beginnt Informatikunterricht mit dem Erlernen der Anmeldung im Labornetz. Tastatur und Maus sind als Eingabegeräte neu und gewöhnungsbedürftig. Am Bildschirm wird eine Benutzungsoberfläche sichtbar, die keineswegs selbsterklärend ist. Die Erleichterung der individuellen Arbeit zeigt sich für Anfänger lange Zeit nicht. Stress und Frustration sind häufig zu beobachten. Ein kognitives Modell vom Informa-

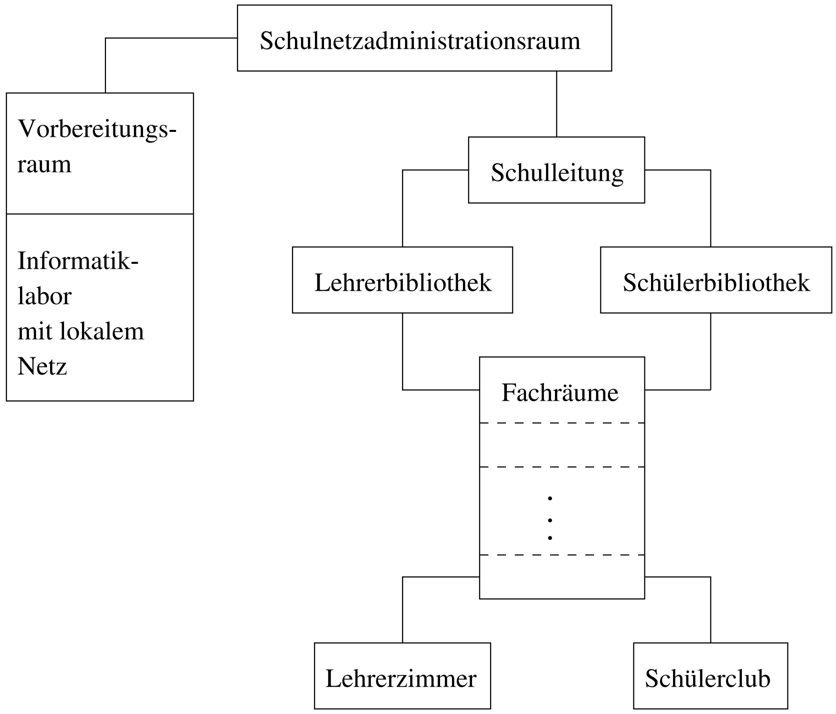


Abbildung 40: Schulnetzwerk

tiksystem (Computer) bringt Orientierungswissen in die anfängliche Versuch–Irrtum–Strategie. Die Schüler durchlaufen Stufen der Dialogarbeit. Sie können:

1. Ausgaben des Systems richtig interpretieren,
2. Ausgaben des Systems richtig vorhersehen,
3. Eingaben zum Steuern des Systems so planen, dass die gewünschten Ausgaben geliefert werden,
4. Ergebnisse im Anwendungskontext bewerten und verantwortungsbewusst einsetzen.

Die logische Speicherstruktur der Verzeichnisse für Systeme, Lehrerdaten und Schülerdaten muss thematisiert werden. Daraus leitet sich die erforderliche Informationssicherheit ab, deren Basismechanismen Verschlüsselung, Authentifizieren und Zugriffsschutz schrittweise zu erlernen sind. Da es keine absolut sicheren Systeme geben kann, sind die Schüler für den sorgfältigen Umgang mit dem Informatiklabor zu sensibilisieren und in das tägliche Aufräumen im Netz einzubeziehen. Es ist ein Beitrag zum selbstbestimmten Lernen, erzeugt Wissen über die ihnen anvertrauten Werte und Achtung vor der Arbeitsleistung der Lehrer– und Schülergruppe, die diese Lernumgebung entwickelte. Eine kontrollierbare Laborordnung darf nicht fehlen. Sie wird mit traditionellen Belehrungen und Sanktionen verbunden. Die Konfrontation zwischen Schülern auf der einen Seite und Lehrer mit System auf der anderen Seite würde das kreative Potential der Schüler in sehr ungünstiger Weise aktivieren. Hier hilft Transparenz der Vorschriften und des Managementaufwandes und Verteilung der Dienstleistungslasten, die in Vor– und Nachbereitung der Laborexperimente auftreten. Was Lehrer und Schüler gemeinsam entwickeln, wird auch gemeinsam geschützt. Neu nachgedacht werden muss über die Organisation des Informatikunterrichtes. Eine kurze Pause reicht nicht aus, um die notwendigen Systemumstellungen für die Lehr–Lern–Experimente vorzunehmen. Der Informatiklehrer kann seine Experimentaltechnik auch nicht auf separaten Rolltischen vorbereiten und in den Laborraum fahren. Die Doppelstunde erweist sich als sinnvoller Rahmen für Planen und Experimentieren (Implementieren). Die traditionelle 45–Minuten–Stunde produziert Stress bei der Dialogarbeit, die den Lernprozess stark behindert.

7.3. Informations– und Kommunikationssysteme (IuK-Systeme) als Unterrichtsthema

Beim Erschließen eines neuen Unterrichtsthemas, z. B. IuK–Systeme, treten typische Fehler auf. Meist werden Einzelheiten stark überbetont, da die sachlogische Struktur des Themas unzureichend erkannt wurde und die Frage nach dem allgemeinbildenden Fundamentum nicht beantwortet werden konnte. Deshalb erscheint eine exemplarische Diskussion zum didaktischen Vorgehen hilfreich.

Motivation durch Ziele

Nur wenn die Frage nach den Bildungszielen die Einführung des Themas erzwingt, lohnt sich der hohe Aufbereitungsaufwand für den Informatikunterricht. Eine kritische Sicht ist angebracht, da die stürmische Entwicklung der Fachwissenschaft Informatik viele Ergebnisse hervorbringt, die keinesfalls zur Weiterentwicklung des Schulfaches beitragen. Im Falle der IuK-Systeme muss der Bildungswert sehr hoch eingeschätzt werden, da sie folgende Ziele für Schüler ermöglichen:

- Sie beherrschen die neuen Kommunikations– und Kooperationsformen der Informationsgesellschaft.
- Sie können einen Informationsraum strukturieren und in solchen Strukturen navigieren, um
  - auf solche Informationen zuzugreifen,
  - Informationen zu verteilen.
- Sie verstehen die Wirkprinzipien von Rechnernetzen und verteilten Systemen, der Basistechnologie neuer gesellschaftlicher Entwicklungen.

Zur Bedeutung dieser Bildungsziele besteht heute hohe Übereinstimmung in der Gesellschaft. Zur Erreichung dieser Ziele gehen die Meinungen weit auseinander. Sie reichen vom Glauben an die naive Anwendung bis zur Forderung nach eigenverantwortlicher Systementwicklung. Hier wird eine systematische Einführung empfohlen, die von beiden Extremwerten weit entfernt ist. Bei klarer Zielstellung steht der Lehrer vor der Notwendigkeit, den Inhalt zu strukturieren und stark zu filtern.

Schwerpunkte für den Inhalt

Mit folgenden Themen kann eine systematische Einführung in die Grundlagen der IuK–Systeme realisiert werden:

- historische Entwicklung,
- Bewertungssystematik,
- technische Grundlagen,
- ethische und rechtliche Regelungen,
- ausgewählte Anwendungen, wie
  - E–Mail,

  - Informationssysteme,
  - Softwarearchive,
  - Diskussionsgruppen (News),
  - Nutzung entfernter Rechner,

- Informationssicherheit,

- Kooperationstechniken in vernetzten Systemen, wie

  - E–Mail–Listen zur abgeschlossenen Gruppendiskussion,
  - Diskussionsgruppen (News) für die offene Diskussion,
  - gemeinsame Dokumentenbearbeitung (asynchron und synchron),
  - Videokonferenzen.

Aus diesen Schwerpunkten lassen sich zielgruppengerechte Unterrichtssequenzen ableiten, die die Vorkenntnisse der Schüler angemessen berücksichtigen. Für Anfänger empfiehlt sich der Zugang über die Anwendungen, deren informatische Grundlagen schrittweise aufgedeckt werden. An Literatur zum Thema mangelt es nicht. Eine didaktisch sinnvolle Auswahl fällt schwer. Der Autor dieses Skriptes hat gute Erfahrungen gesammelt mit einer Kombination von Schülerduden Informatik [CS97], Vorlesungskript Rechnernetze der TU Chemnitz (siehe Anhang I: Internetempfehlungen), dem LOGIN–Heft 5/1997 zum Thema „Programmieren weltweit“, zwei gut gegliederten Büchern [MW95], [SBGK94] und dem IuKD–Gesetz [Bundestag97]. Eine Folge von Lexikonstichwörtern zum Thema „IuK–Systeme“ der Zeitschrift „Computer + Unterricht“ veröffentlicht, um Lehrern den Überblick zu erleichtern:

- Bildungsserver, Heft 30/1998,
- Computerunterstützte Gruppenarbeit, Heft 25/1997,
- Digitale Unterschrift, Heft 21/1996,
- Diskussionsgruppen (News), Heft 30/1998,
- Elektronische Zeitung, Heft 28/1997,
- Internet–Protokoll (TCP/IP), Heft 29/1998,
- Internet–Recherche, Heft 28/1997,
- Kommunikationsnetze, Heft 29/1998,
- Netz–Software (Java), Heft 31/1998,
- Netzwerkmanagement, Heft 27/1997,
- Telekonferenz, Heft 25/1997,
- Verteilte Systeme, Heft 27/1997,
- Visuelle Programmierung, Heft 31/1998.

Die Vernetzung der Grundbegriffe und Wirkprinzipien bleibt eine Aufgabe, die der Lehrer selbst lösen muss. Anregungen dazu liefern die Überlegungen zur Staffelung vom Einfachen zum Komplizierten und die Berücksichtigung der inneren Zusammenhänge des Stoffes.

Grundbegriffe und Wirkprinzipien

Am Anfang stehen dann die verteilten Systeme mit Betrachtungen zu Funktionalität, Komponenten und Client–Server–Modell. Darauf kann mit dem Internet–Protokoll aufgebaut werden, indem Vorschriften, Bestandteile, Adressen und Zustellung thematisiert werden. Die Basismechanismen der Informationssicherheit führten zur Weiterentwicklung des Internet–Protokolls. Verschlüsselung, Authentifizierung und Zugriffsschutz müssen behandelt werden, wenn ein Verständnis zu Grundbedrohungen und deren Abwehr entwickelt werden soll. Die computergestützte Gruppenarbeit in der Schule besitzt eine inhaltliche und eine lehrmethodische Dimension. Sie soll nicht nur Unterrichtsthema sein, sondern als Voraussetzung für das Lernen in allen Unterrichtsfächern angeeignet werden. Die asynchronen und synchronen Phasen eignen sich in Verbindung mit Projektarbeit sehr gut zur Vorbereitung auf selbstbestimmtes Lernen und späteres Studieren. Aktuelle Anwendungen wie digitale Signatur und elektronische Zahlungsmittel sind deshalb eine so wichtige Ergänzung, weil sie die gesellschaftlichen Auswirkungen der IuK–Systeme deutlich machen und auf künftige Lebenssituationen vorbereiten. Eine Einführung in die Grundbegriffe gelingt am Beispiel der historischen Entwicklung (siehe Tabelle 7 und Abbildung 41). Anfänger zeigen sich überrascht davon, dass die Paketvermittlung im Datennetz 1969

1969	Beginn der **Paketvermittlung** im Datennetz
1972	**Telnet** und **FTP** mit **Protokollen**; **E–Mail** etwas später
1973	Erfindung Ethernet LAN
1977	Entwicklung von **TCP** (Transmission Control Protocol) und **IP** (Internet Protocol)
1983	Unix–Variante mit TCP/IP; Internet ca. 600 **Hosts**
1988	**Backbone** des Internet wird auf 1,5 Mbit/s umgestellt; ca. 56.000 Hosts
1990	**Informationssuche** mit Archie, WAIS, Gopher; Backbone mit 45 Mbit/s; Hostanzahl ca. 300.000
1993	**WWW** mit Mosaic; Hostanzahl > 2.000.000

Tabelle 7: Grundbegriffe am Beispiel der historischen Entwicklung

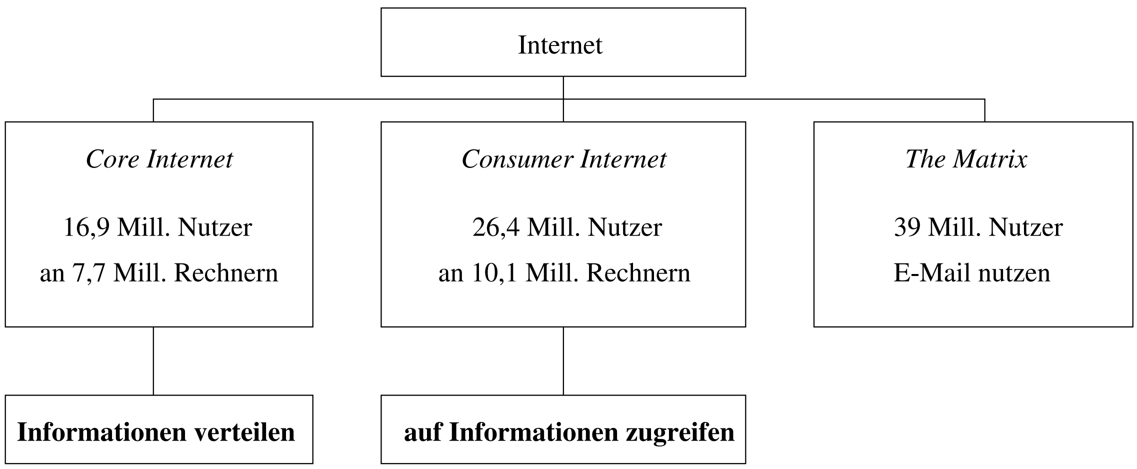


Abbildung 41: Einteilung des Internet (Stand 1995)

entwickelt wurde. Dieses Wirkprinzip wird für den gewaltigen Datenaustausch im Internet heute erfolgreich eingesetzt. Die Einteilung des Internet relativiert die riesigen Teilnehmerzahlen etwas, da nur ein Bruchteil der Anwender tatsächlich Informationen verteilen, also elektronisch publizieren und präsentieren kann. Vor Absolutzahlen sei gewarnt, da sie sich sehr schnell ändern.

Eine Bewertungssystematik unterstützt den Lehr–Lern–Prozess, da sie es erlaubt von Einzelheiten zu Gemeinsamkeiten der Phänomene vorzudringen. IuK–Systeme können eingeteilt werden nach:

- Anwesenheit der Kommunikationspartner (siehe Tabelle 8),
- Kommunikationsart (siehe Tabelle 9),
- Funktionalität des Netzes (siehe Tabelle 10).

nicht gleichzeitig	**gleichzeitig**
E–Mail Informationssysteme Diskussionsgruppen	Telefongespräch Telekonferenz gemeinsame Dokumentenbearbeitung (CSCW)

Tabelle 8: Anwesenheit der Kommunikationspartner

Individualkommunikation	**allgemeine Kommunikation**
z. B. E–Mail • Grafiken, Bilder, Audio, Video, Programme • Vorteile: – Weiterverarbeitbarkeit – Geschwindigkeit – Preis–Leistungs–Verhältnis	z. B. Informationssysteme • Informationen mit Verweisen • automatische Suche • verteilte Speicherung • Interaktion

Tabelle 9: Kommunikationsart

Übertragung ohne Zusatzfunktion	**Übertragung mit Zusatzfunktion**	**Zusatzfunktionen zur Unterstützung von Anwendungen**
Datenstrom zwischen zwei Punkten z. B. ISDN (Integrated Services Digital Network)	z. B. Paketvermittlung im Internet	z. B. Postämter für E–Mail

Tabelle 10: Funktionalität des Netzes

Suchverfahren

Eine wesentliche Forderung im Rahmen der Allgemeinbildung besteht darin, dass die Schüler die Wissensbestände des Internets durch gezieltes Suchen erschließen können. Die Versuch–Irrtum-Strategie versagt hierbei vollständig. Systematische Suchverfahren sind zu erlernen. Den Schülern muß bewußt werden, wer auf Informationen verweist, indem er Zeiger setzt, der führt eine Wertung durch. Sie lernen, daß für das Auffinden relevanter Informationen Adressen benötigt werden, URL (uniform resource locator) genannt. Genau die fehlen aber meist oder stimmen nicht mehr. Eine Navigationshierarchie ermöglicht die Orientierung in großen Informationsmengen. Im ungünstigsten Fall wird eine automatische Suche nach Stichwörtern gestartet. Das entspricht bei traditioneller Suche der Schlagwortkartei. Dabei wird zwischen lokaler und globaler Suche unterschieden. Bei lokaler Suche wird das Dokument auf dem eigenen Server gesucht. Für die lokale Suche können lineare Suche, binäre Suche und Hash–Verfahren angewendet werden. Bei den formalen Anfragesprachen kommt es wie immer in der Informatik auf den Zusammenhang von Syntax, Semantik und Pragmatik an. Schüler mit Schwächen im Umgang mit logischen Ausdrücken benötigen besondere Unterstützung. Das Abstraktionsniveau sollte nicht unterschätzt werden. Es handelt sich um einen weit verbreiteten Irrtum, dass eine automatische Suche auch automatisch die gewünschten Informationen liefert. Die Schüler lernen, daß bei der linearen Suche eine Suchmaske (Suchfenster) über den gesamten Text verschoben wird, bis Mustergleichheit gefunden wird. Sie erlernen eine Syntax zur Formulierung von Suchanfragen. Das Gegenstück dazu ist ein permutierter Index analog zum Index von Fachbüchern. Dazu kann dann die binäre Suche mit der Halbierung des Suchraumes eingesetzt werden oder das Hash–Verfahren, bei dem der Hash–Wert des Suchbegriffs zum Suchen verwendet wird. Wenn die Logik der lokalen Suche beherrscht wird, können die globalen Suchverfahren mittels thematischer Verzeichnisse (Kataloge) und Suchmaschinen (Web–Roboter) darauf aufbauen.

Bei der Behandlung globaler Suchsysteme muss thematisiert werden, wie Suchmaschinen zu ihren Metainformationen kommen:

1. Der Informationsanbieter informiert. Das kann zum Eintrag in thematische Verzeichnisse, analog zu Katalogen, genutzt werden, die mehrere Ebenen einer Hierarchie bilden. Der Suchende benötigt Einsicht in die logische Struktur.

2. Web-Roboter führen die automatische Suche nach Web–Servern, Dokumenten und weiteren URLs durch. Da sie große Datenbanken aufbauen, führt das häufig zu Komplikationen aufgrund des Ressourcenverbrauches (Netzbandbreite, Serverlast). Es existieren Vorschriften für Suchsysteme, z. B. wird der Zutritt für bestimmte Bereiche des Servers nicht erlaubt.

Der Suchende wird von der Informationsfülle überfordert, wenn er die Eingrenzung des Suchraumes durch Anfragesprachen unzureichend beherrscht. Erschwert wird die Ausbildung, da die Anfragesprachen nicht einheitlich sind. Es besteht die Möglichkeit mit Proximity–Operatoren den Abstand zwischen Suchbegriffen festzulegen. Mit Qualifikatoren kann die Suche auf Dokumententeile, z. B. die Überschrift eingeschränkt werden. Eine wichtige Frage bildet die Bewertung der Informationsangebote. Eine automatische Bewertungsfunktion, die Häufigkeit und Ort des Suchbegriffes als Kriterien ansetzt, scheitert oft. Es besteht die Gefahr, dass unseriöse Autoren mit unzutreffenden Schlagwörtern auf sich aufmerksam machen wollen. Eine fachliche Bewertung wie bei Buch oder Fachzeitschrift ist wünschenswert. Zum Teil findet man einen Gutachterservice, der eine Bestenauswahl vornimmt. Bei der eigenen Informationsverteilung soll erkannt werden,

- welche Bedeutung der Dokumententitel und die Überschriften für die Suche haben,
- wie eine Zusammenfassung aller wichtigen Begriffe die Suche unterstützt,
- wie sich Suchhinweise im Kopf der Hypertextdokumente auswirken.

Modellvorstellungen

Hier bieten sich vor allem die technischen Grundlagen mit Schichtenmodell, Vermittlungsverfahren, Internet–Architektur und Systembeispiele an. Statt des umfangreichen OSI–Referenzmodells sollte das Schichtenmodell der Internet–Technologie mit seinen vier Schichten Subnetz–Schicht für die physische Übertragung (elektrisch oder optisch), Internet–Schicht mit Protokoll, Transportschicht mit Protokoll und Anwendungsschicht mit Protokoll eingeführt werden (siehe Abbildung 42).

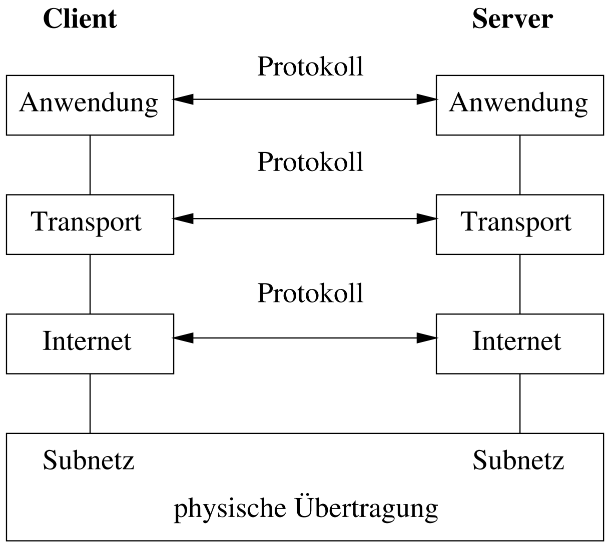


Abbildung 42: Schichtmodell 1 der Internet–Technologie

Das Internet–Protokoll (TCP/IP transmission control protocol / internet protocol) fördert das Verständnis der Wirkprinzipien von IuK–Systemen. Es ist außerdem die Grundlage für die selbständige Orientierung im Adressraum globaler Netzwerke. Protokolle sind als Satz von Vorschriften und logische Beziehungen zu thematisieren. Das Internet als Zwischenstation wird erst verständlich, wenn die Anforderung, Datenströme in die richtige Richtung zu lenken, diskutiert wurde (siehe Abbildung 43). Eine wesentliche Ergänzung bildet der sichere Transport über unsichere Übertragungswege mit Prüfsummen, Quittungsmechanismen und Wiederholungen.

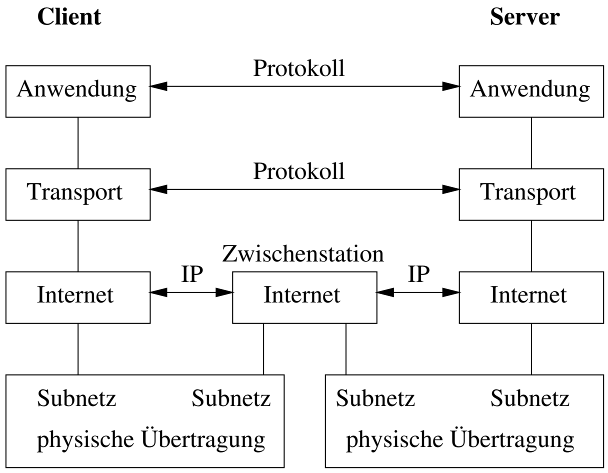


Abbildung 43: Schichtmodell 2 der Internet–Technologie

Kanalvermittlung und der damit verbundene typische Besetztfall sind vom Telefon her bekannt. Nachrichtenvermittlung blockiert kurze Nachrichten, wenn lange gesendet werden. So wird verständlich, warum sich die Paketvermittlung in IuK–Systemen durchsetzte (siehe Abbildung 44). Das Zerlegen der IP–Adressen in Empfänger– und Absenderadresse wurde bereits mit den Protokollbetrachtungen im Schichtmodell vorbereitet. Jetzt wird daran angeknüpft, um das Vermittlungsverfahren zu analysieren.

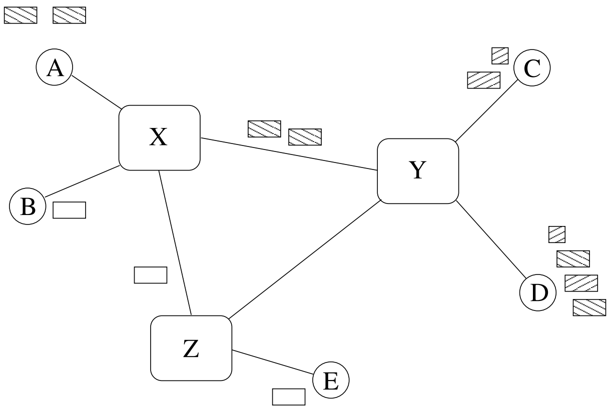


Abbildung 44: Paketvermittlung 1 im Datennetz

Die Betrachtung eines Leitungsausfalls zeigt die Robustheit des Verfahren (siehe Abbildung 45), das alternative Wege finden, Datenpakete zwischenspeichern und bei Bedarf beim Empfänger so umsortieren kann, dass die richtige Reihenfolge der Information rekonstruiert wird. Die Struktur eines IP–Pakets wird untersucht. Deutlich soll für die Schüler dabei werden, dass die Protokollköpfe als Verwaltungsdaten fungieren.

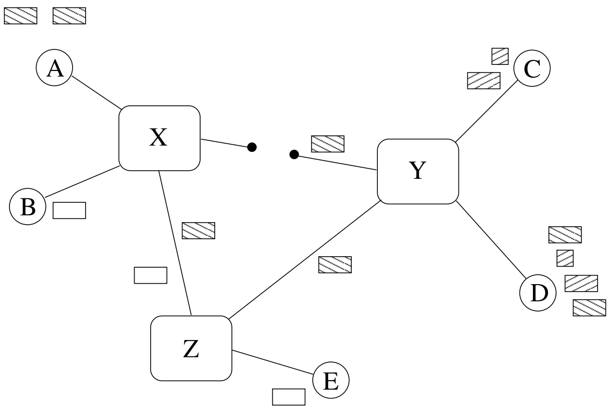


Abbildung 45: Paketvermittlung 2 im Datennetz

Mit diesen sehr gut zu veranschaulichenden Wirkprinzipien der IuK–Systemen wird ein selbstbestimmter Umgang mit diesen Technologien ermöglicht. Ein Ausschnitt der Internet–Architektur (siehe Abbildung 46) verbindet ausgewählte Anwendungen, die für fachübergreifende Unterrichtsprojekte große Bedeutung erlangten. Die Protokolle können an konkreten Anwendungen wie dem Senden und Empfangen von E–Mail bei Bedarf vertieft werden. Am Beispiel des UDP (user datagram protocol) wird der Verzicht auf Fehlerbehandlung und die daraus resultierenden Konsequenzen darstellbar.

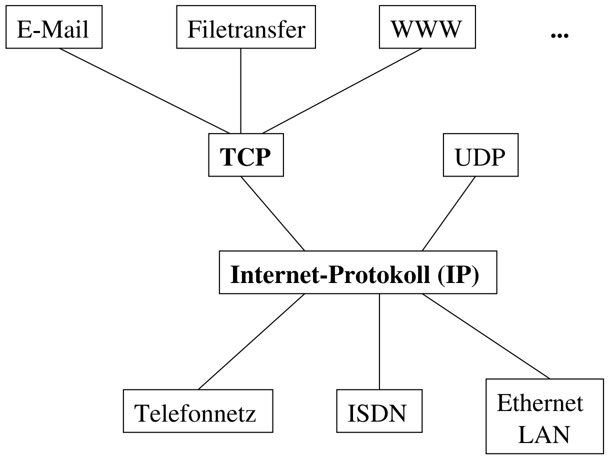


Abbildung 46: Ein Ausschnitt der Internet–Architektur

Die informatische Bildung liefert mit dem Verstehen der WWW–Architektur Grundlagen für alle anderen Fächer , indem z. B. das typische Missverständnis, WWW sei identisch mit dem Internet, ausgeräumt werden kann (siehe Abbildung 47). Das Grundprinzip verteilter Systeme (Client–Server–Architektur) wird eingeführt. Es findet sich heute in allen Intranet–Entwicklungen der Schulen wieder. Das HTTP (hypertext transfer protocol) begegnet den Anwendern in allen Internet–Adressangaben. Hier kann viel Unverständnis beseitigt werden durch Anaogiebetrachtung zu bereits behandelten Protokollen. Mit der Wirkung der Proxy–Cache–Technik werden wichtige Entscheidungen zu Auswahl und Verweildauer von globalen Dokumenten im lokalen Speicherbereich verständlich.

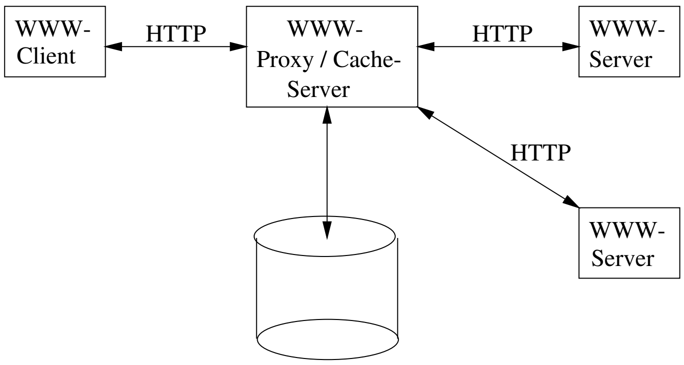


Abbildung 47: WWW–Architektur

Probleme

Zur Zeit wird bundesweit viel Kreativität in die Gestaltung guter Unterrichtsbeispiele investiert. Die Fachzeitschrift LOGIN, die Bildungsserver und die Fachtagung INFOS'99 (Informatik und Schule) machen solche Sammlungen zugänglich, werden aber noch viel zuwenig beachtet von Lehrern und Studierenden. Auch hier lohnt eine Suche vor der Neuentwicklung!

Um beständige Grundlagen auszuwählen, benötigt man einen tiefen Einblick in dieses Teilgebiet der Informatik, das sich zur Zeit sehr innovativ entwickelt. Der Übergang zum Internet–Protokoll der Version 6 steht bevor. Dabei geht es nicht nur um eine Vergrößerung des Adressraumes, sondern auch um die Einführung wichtiger Sicherheitskonzepte.

Ethische und rechtliche Regelungen spielen in diesem Unterrichtsgebiet eine große Rolle. Wenn sich Informatiklehrer dafür unzureichend ausgebildet fühlen, gehen die fachlichen Bezüge im Lernprozess verloren. Man kann sicher in anderen Fächern sehr allgemein darüber sprechen, aber kaum Verständnis für die komplexen Zusammenhänge damit entwickeln. Die Wirkprinzipien der

Internet–Architektur bilden die kognitive Basis für das Verstehen der komplizierten Umsetzung der Gesetzgebung.

Das Unterrichtsthema ist an die Existenz eines Schul–Intranets gebunden. Die dafür erforderliche Zeit wird heute den Informatiklehrern bei voller Lehrbelastung abverlangt. Dafür setzen amerikanische Schulen bereits erfolgreich technisches Personal ein [Diener98]. Eine Minimalforderung besteht in Entlastungsstunden für die Netzadministratoren der Schulen [NPS98]. Es gilt Schulaufsichtsbehörden und Schulträger mit dem Problem vertraut zu machen und an zukünftigen Lösungen mitzuwirken. Die Gesellschaft für Informatik (GI) gründet eine Fachgruppe, die Informatiklehrer dabei unterstützen kann.

8. Gesamtkonzept der informatischen Bildung

8.1. Informatische Bildung

Informatische Bildung stellt eine intensive Begegnung (Mensch–Maschine–Interaktion) mit Informatiksystemen (Zusammenwirken von Hardware und Software) dar. Dabei sind grundsätzliche Fragen zu beantworten [Schubert95a], S. 97:

Motivation

- Fragen nach den Zielen dieser Technikgestaltung:
  Warum wird das gemacht?

Linien im Curriculum

- Fragen nach den Plänen (Algorithmen, Heuristiken):
  Was ist möglich? Was wird gemacht?

- Fragen nach der Mensch–Maschine–Interaktion:
  Wie kann man sich verständigen und etwas darstellen?

- Fragen nach den Systemen:
  Wie funktionieren die verarbeitenden Informatiksysteme?

In diese natürliche Struktur der Informatik lassen sich alle Grundbegriffe (Definitionen), Prinzipien (Sätze, Erkenntnisse, Basismechanismen), Methoden (Verfahren) und Werkzeuge einordnen. Der Bereich zu den Zielen der Informatik nimmt eine übergeordnete Stellung zu den drei anderen ein, die als Spalten einer Inhaltsmatrix verstanden werden können. Die Zeilen dieser Matrix können aus allgemeinen Anforderungen an eine Systematisierung von Lehrinhalten abgeleitet werden:

		I Pläne	II Sprachen	III Systeme
a)	theoretische Grundlagen			
b)	historische Entwicklung		Komponenten zum	
c)	Wirkungsweise, Funktionsumfang		aktuellen Stand	
d)	Verallgemeinerungen, Spezialfälle		des Schulfaches	
e)	Entwicklungsmethodik		auswählen	
f)	Anwendungsbeispiele			
g)	gesellschaftliche Anforderungen			

Aus den Überlegungen zu Kapitel 2 und 3 lässt sich folgender Vorschlag für ein Gesamtkonzept der informatischen Bildung in den Sekundarstufen I und II ableiten (siehe Abbildung 48).

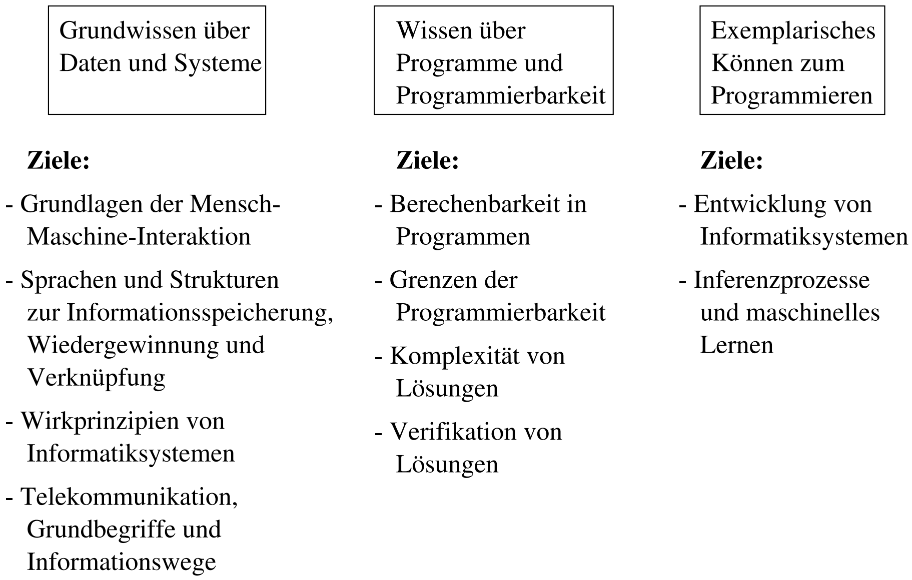


Abbildung 48: Gesamtkonzept

8.2. Grundwissen über Daten und Systeme

Der Abschnitt Daten und Systeme ist als verpflichtende Einführung für alle Schüler konzipiert. Das hat die Konsequenz, dass er in der Sekundarstufe I aller Schularten verankert sein muss. Der folgende Strukturierungsvorschlag kann alternativ auch mit der Einführung in die Telekommunikation beginnen. Das wird heute schon an den Schulen praktiziert, die über ein funktionstüchtiges Intranet verfügen.

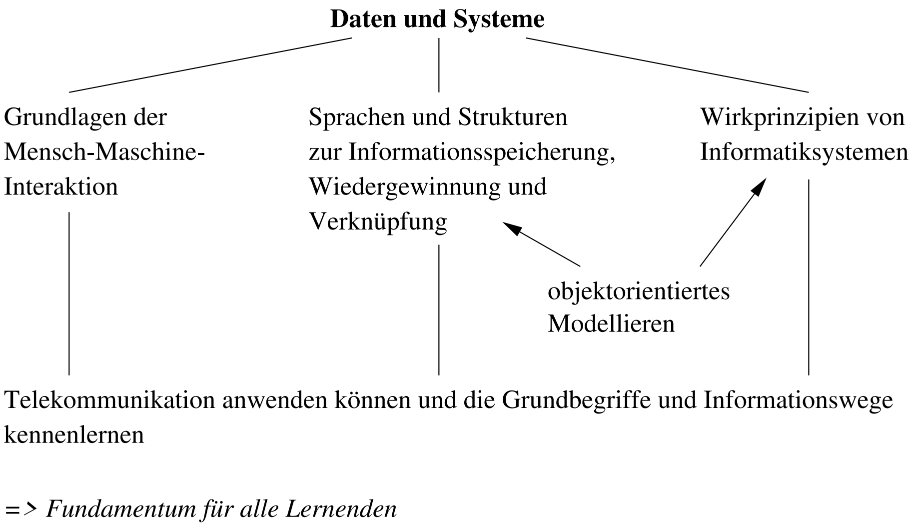


Abbildung 49: Daten und Systeme

8.2.1. Ziel: Grundlagen der Mensch–Maschine–Interaktion kennenlernen und anwenden können

Gesprochen wird vom Computer als „Denkzeug“. Die Bedeutung dieser symbolverarbeitenden Maschine mit universellen Einsatzbereichen soll verstanden werden, um selbstverantwortliche Lebensgestaltung zu ermöglichen [KP93].
Einen besonders günstigen Zugang zur Informatik bildet der neue Typ des Aufgabenlösens

(Arbeitsteilung zwischen Mensch und Informatiksystem). Dabei erhalten Lösungsplanung und Mensch–Maschine–Interaktion einen deutlich höheren Stellenwert, da die traditionelle Ausführung automatisierbar wird. Der Mensch übernimmt es, Systeme, Pläne und Parameter problemgerecht auszuwählen und die Lösungsvarianten des Systems zu interpretieren, zu bewerten und zu verantworten. Zwischen Anfangs– und Zielzustand dieses Prozesses liegt eine mehr oder weniger komplexe Mensch–Maschine–Interaktion, um:

- Zielvorstellungen zu Plänen zu verdichten,
- Lösungsmodelle auszuwählen,
- Systeme zu steuern,
- Interpretation und Verifikation der Ergebnisse zu sichern.

Um vernetzendes Denken und Handeln (verteilt, interaktiv, nichtsequentiell) zu fördern, werden folgende Inhalte dem Ziel zugeordnet:

- Benutzungsschnittstellen,
- Dialogsprachen,
- Dialogplan (Funktionsauswahl, objektorientierter Entwurf).

Als Vorgehensweise wird empfohlen:

- Beschreibung von Objekten, deren Verknüpfung mit Attributen und Aktionen;
- Klassifikation von Aufgaben durch Abstrahieren;
- Systematisieren von Dialogplänen

Folgende Schülerrollen sind im Unterricht zu ermöglichen [Bosler92]:

- Bediener: Geräte bedienen und Daten eingeben,
- Benutzer: Anwendungen gestalten,
- Betroffener: Folgen kennen bzw. erfahren,
- Gestalter: selbständig algorithmisierbare Probleme lösen.

8.2.2. Ziel: Sprachen und Strukturen zur Informationsspeicherung, Wiedergewinnung und Verknüpfung einsetzen können

Die Grundprinzipien von Datenbanken und die damit verbundenen Anforderungen an den Schutz personenbezogener Daten ermöglichen tiefere Einsichten in die gesellschaftlichen Auswirkungen der Informatik. Daraus resultieren Anforderungen an die Technikgestaltung und die Datensicherheit. Die Gefahr des Verlustes oder Verfälschens der großen Datenbestände einer Datenbank wird untersucht. Es werden Mechanismen vorgestellt, die dem entgegengewirken. Die Datensicherheit ermöglicht die Realisierung der juristischen Festlegungen zum Datenschutz in der Gesellschaft. Aber erst die Rechnernetze und damit verbundene Sicherheitspannen führten zu der Erkenntnis, dass die Sicherheitsaspekte von fundamentaler Bedeutung für jede Systementwicklung in der Informatik sind. Offene Fragen zu zukunftsorientierten Anwendungen, wie elektronische Unterschriften und elektronische Zahlungsmittel runden den Überblick ab.
Zum Inhalt gehört:

- Aufbau und Dienstleistungen einer Datenbank,
- Datenmodellierung (logischer Entwurf),
- Analyse, Bewertung und Modifikation von Datenbankstrukturen,
- Grundbegriffe und Gesetze des Datenschutzes,
- Sicherheitsanforderungen: Vertraulichkeit, Integrität, Verfügbarkeit, Anonymität, Originalität, Verbindlichkeit,
- Gefährdung durch Viren und Gegenmaßnahmen,
- Aufbau und Funktionsweise von Kryptosysteme zum Verschlüsseln, Authentifizieren und für den Zugriffsschutz.

Als Vorgehensweise wird empfohlen:

- Erkunden der Datenschutzbestimmungen und der Maßnahmen zu ihrer Realisierung an der Schule;
- Thematisieren der persönlichen Betroffenheit;
- Vergleich von Datentupeln auf ihre Verknüpfbarkeit;
- Wirksamkeit von Anfragen;
- Experimentieren mit einfachen Kryptosystemen und Analysieren der zugrunde liegenden Algorithmen.

8.2.3. Ziel: Wirkprinzipien von Informatiksystemen verstehen und erläutern können

Die Modellierung von Umweltausschnitten in Informatiksystemen und die daraus resultierenden Werkzeuge können bis zu den Basisoperationen betrachtet werden. Wenn diese Prinzipien verstanden wurden, sind Kriterien formulierbar, die die Auswahl des richtigen Systems für die Lösung des jeweiligen Problems ermöglichen. Die Automatenmodelle und speziell das Zustandsraummodell unterstützen die Transparenz komplexer Systeme sowohl im Hardware– als auch im Software–Bereich. Das Übersetzen von Plänen in Sequenzen von Basisoperationen wird als typische Systemeigenschaft erkannt. Auch einfache Nutzungsaufgaben, wie das Steuern der ausgewählten Funktionen, die Aktualisierung und Speicherung von Informationen, gehören in diese Linie. Die Vernetzung von Systemen und die Erfordernisse der Informationssicherheit erweitern die Grundvorstellungen. Die fachspezifische Vorgehensweise:

- Objekte, deren Eigenschaften und Beziehungen zu definieren,
- diese Strukturen als Modell einer „abgeschlossenen Welt“ aufzufassen und
- aus diesen Strukturen durch Formalisierung von Aktionen das potentiell enthaltene Wissen abzuleiten

beeinflusst den Umgang mit Informationen und Wissen in anderen Wissenschaftsdiziplinen beachtlich.
Die Informatik ändert ihre Forschungsziele und Denkgewohnheiten. Brauer formulierte [BB95]: „Computersystem als Gruppe gleichrangiger, selbständiger Akteure, die bestimmte Aufgaben erledigen und dazu miteinander und mit der Umgebung interagieren.“ Die Lern– und Anpassungsfähigkeit der Akteure spielt dann eine wesendliche Rolle. Folgende Inhalte sind dem Ziel zuzuordnen:

- Programmsteuerung und das Konzept des logischen Speicherplatzes,
- Hard– und Software als formale Systeme,
- Schaltwerke und Maschinensprachen,
- Aufbau und Funktionsweise von Computern,
- Prozesse und Ressourcen (prozeduraler Entwurf),
- Aufbau und Dienstleistungen eines Betriebssystems,
- Leistungsklassen und ihre Merkmale,

Als Vorgehensweise wird empfohlen:

- Modelle für Grundarchitekturen entwickeln,
- Weg vom Baustein zum komplexen System aufzeigen,
- Beispiele für Rechner-, Betriebs-, Anwendungssysteme.

8.2.4. Ziel: Telekommunikation anwenden können und die Grundbegriffe und Informationswege kennenlernen

Die weltweite Informationspräsentation und Informationsgewinnung in Computernetzen setzt Kenntnisse und Fertigkeiten voraus, die bereits in der Schule vorbereitet werden müssen. Mit der Bildungsinitiative Informatik und Telekommunikation wird es möglich, Schulen an das Netz anzuschließen und das weltweite Netz (WWW) auf die Bedürfnisse der Schulen vorzubereiten. Telekommunikation als Medium, Werkzeug und Thema des Unterrichts fördert fächerübergreifendes und projektorientiertes Lernen. Solche Etappen wie:

- Entwicklung eines schulspezifischen Informationssystems [Meyer95],
- Meinungsaustausch und Ergebnispublikation durch Netzkontakte [Rauch95],
- Schülertätigkeiten zur Auswahl und Aufbereitung von Dokumenten für den Unterricht,

helfen, Strategien zu entwicklen, die es ermöglichen, in den Schulen angemessen auf die neuen Ausbildungsanforderungen zu reagieren. Es entstehen neue Arbeitsweisen bei der Vorbereitung auf den Unterricht. Das Netz wird als Wissensquelle und Diskussionspodium erschlossen. Das Verständnis für Datenschutz und Informationssicherheit wird vertieft. Fragen des Jugendschutzes stellen sich neu. Die Verarbeitung von Informationen und Wissen durch vernetzte Computersyteme wird mit folgende Inhalten untersetzt:

- Einzelplatzsystem,
- lokale Netze,
- globale Netze,
- Navigation in Netzwerken,
- Dienste des Internet,
- Protokolle.

Als Vorgehensweise wird empfohlen:

- computerunterstützte Gruppenarbeit;
- Schulnetz zum Informationsaustausch;
- elektronische Zeitschriften kennenlernen;
- E-Mail anwenden;
- Transfer von Shareware;
- Fragen der Informationssicherheit ableiten.

8.3. Wissen über Programme und Programmierbarkeit

8.3.1. Problemlösung

In der Informatik existieren alternative Konzepte, die zur Lösung von Aufgaben unterschiedlich gut geeignet sind:

- logische Programmierung:
  - Problembeschreibung aus Fakten und Regeln,
  - Lösungssuche im Zustandsraum,
  - Wissensbasen, Datenbankabfragen,
- prozedurale Programmierung:
  - Lösungsbeschreibung als Prozedurennetzwerk,
  - Ein– und Ausgabeprozeduren,
  - Berechnungen, Zeichenkettentransformationen,
- funktionale Programmierung:
  - Lösungsbeschreibung als Geflecht von Funktionen,
  - Grafiken, Formelmanipulation,
- objektorientierte Programmierung:
  - Problembeschreibung als Klassenhierarchien,
  - Einbeziehung der prozeduralen Programmierung,
  - Konfigurationsaufgaben, Benutzungsschnittstellen, Grafikanwendungen.

Dabei kann es nicht darum gehen, ein Konzept zu bevorzugen und die anderen zu vernachlässigen, sondern Verständnis für die Vielfalt der Denk– und Arbeitsweisen der Informatik zu vermitteln. Die bisherige Beschränkung auf solche Algorithmen, die sich problemlos in eine prozedurale Programmiersprache übertragen lassen, und die Befrachtung mit entbehrlichem Detailstoff zur vorhandenen Programmierumgebung und dem Schulcomputer überzeugten die Lernenden nicht.

Dosch wies auf einer Schultagung 1992 auf die Möglichkeit zur Verbindung verschiedener Programmierparadigmen mit den Ebenen der Programmentwicklung hin (siehe Abbildung 50). Das Vorgehen im Informatikunterricht soll einen breiteren Ansatz entwickeln:

- Analyse der Aufgabenstellung,

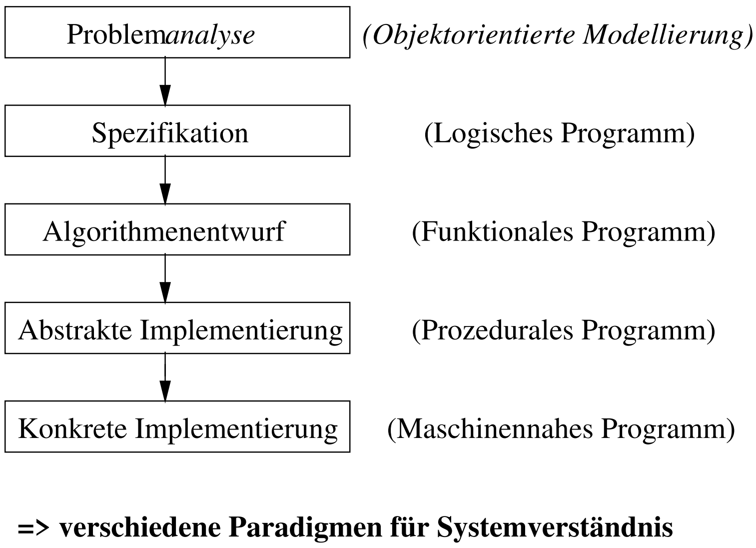


Abbildung 50: Ebenen der Programmentwicklung [Keidel93], S.13

- Wahl des geeigneten Programmierparadigmas,
- Modellierung der Lösung und Übertragung in die zugehörige Sprache,
- Variantenvergleich zwischen Problemlösungen in verschiedenen Programmierparadigmen.

Hier wird vorgeschlagen, diesen Ansatz mit parallelen Linien eines spiralförmigen Curriculums so zu strukturieren, dass das Wissen über die Programmierbarkeit deutlich in den Vordergrund gestellt wird und nur exemplarisch mit dem eigenen Programmieren der Schüler verbunden werden sollte.

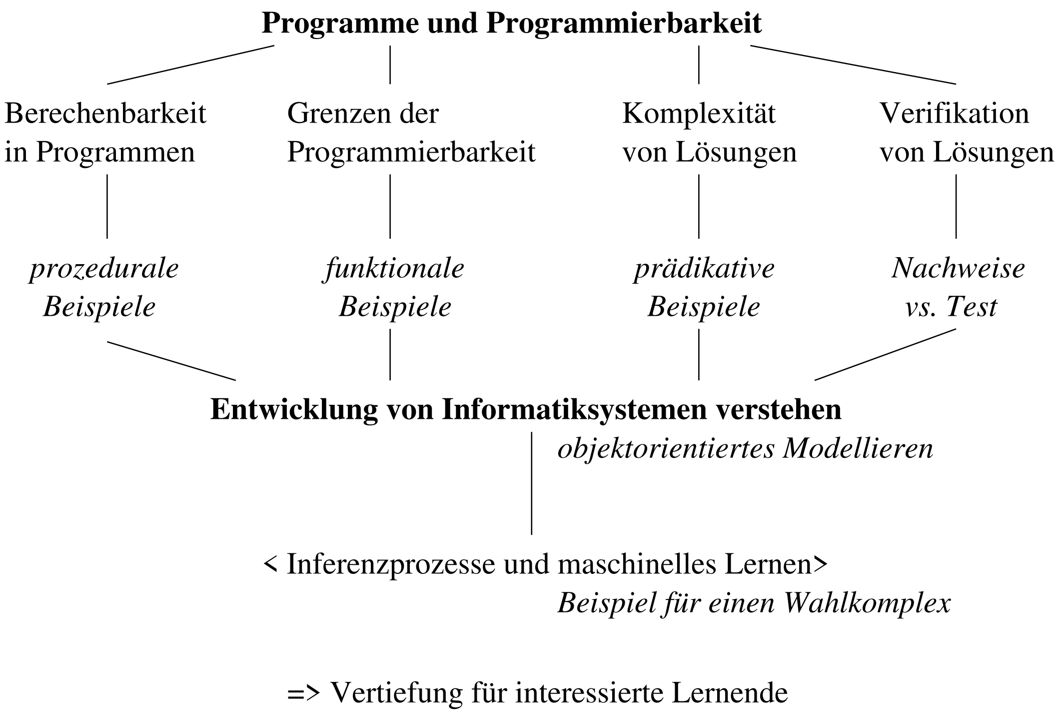


Abbildung 51: Programmierbarkeit

8.3.2. Ziel: Berechenbarkeit in Programmen verstehen und anwenden können

Die exemplarische Behandlung von Entwurfs–, Programmier– und Dialogsprachen gehört in diese Linie, um Objekte erzeugen, verbinden und verwalten zu können. Interaktionsprotokolle sind zu analysieren mit dem Ziel, Verständnis für typische Strukturen und Situationen der Mensch–Maschine–Interaktion zu entwickeln. Syntax und Semantik sind ebenso wie der Begriff der Grammatik im Kontext der Informatik zu behandeln. Leichtere Aufgaben, wie das Bereitstellen von Parametern, das Beantworten von Systemfragen, wechseln sich ab mit anspruchsvollen Aufgaben wie dem Interpretieren, Vergleichen und Beurteilen von Ergebnissen in Abhängigkeit von der jeweiligen Zielvorstellung.
Folgende Inhalte sind dem Ziel zuzuordnen:

- formale Sprachen,
- deren Grammatiken,
- Maschinenmodelle.

Als Vorgehensweise wird empfohlen:

- mit prozedurale Beispielprogrammen (fertigen Lösungen) verbinden;
- Weg vom Theorie–Element zur Anwendung.

8.3.3. Ziel: Grenzen der Programmierbarkeit erkunden

Die Begriffe „berechenbar", „entscheidbar" und „akzeptierbar" werden von der naiven Einführung bis zur modellbasierten Definition systematisch aufgebaut. Dadurch wird die Rationalisierung geistiger Arbeit begründbar, und die Beurteilung der Auswirkungen erhält ein fachliches Fundament. Die Transformation einer Problembeschreibung in eine andere kann demonstriert werden, um die Einordnung unbekannter Probleme in bekannte Klassen zu ermöglichen. Das Aufzeigen innerer Gesetzmäßigkeiten eines Problems führt zum geeigneten Lösungsmodell. Folgende Inhalte sind dem Ziel zuzuordnen:

- Entscheidbarkeit,
- Akzeptanz,
- Unentscheidbarkeit (Halteproblem).

Als Vorgehensweise wird empfohlen, an prozeduralen und funktionalen Lösungen die Modelle der Berechenbarkeit zu vergleichen.

8.3.4. Ziel: Bestimmen der Komplexität von Lösungen

Hier wird geklärt, welche Eigenschaften Probleme besitzen müssen, um mit Informatiksystemen gelöst zu werden. Außerdem wird begründet, dass die Komplexität auch einfach beschreibbarer Probleme deren Lösung verhindert. Die Bestimmung von Aufgabenklassen wird an Beispielen erläutert. Als Klassifikationsmerkmale kommen wiederum die Komplexität oder Analogien in der Lösungsstruktur in Frage. Grob– und Feinpläne können strukturiert, exemplarisch verifiziert und auf ihre Komplexität hin analysiert werden.
Folgende Inhalte sind dem Ziel zuzuordnen:

- Reduzierung der Komplexität: Von besonderem Interesse sind Komplexitätsschranken, z.B.:
  - konstante Komplexität,
  - logarithmische Komplexität,
  - lineare Komplexität,
  - polynomiale Komplexität,
  - exponentielle Komplexität.
- Klassifizieren von Problemen:
  - Die Klasse P: In diese Klasse gehören alle Probleme, die deterministisch lösbar sind mit polynomial–beschränktem Zeitaufwand. Einfacher gesagt sind das alle praktisch lösbaren Aufgaben.
  - Die Klasse NP: In diese Klasse gehören alle Probleme, die nichtdeterministisch lösbar sind (mit Raten) mit polynomial–beschränktem Zeitaufwand. Einfacher gesagt sind das alle praktisch unlösbaren Aufgaben, denn das Raten kann nur durch vollstäandiges Durchmustern realisiert werden. Es existieren aber Informatiksysteme dafür, die Heuristiken enthalten, d.h. die Lösung wird nicht für jedem Fall gefunden.
- Transformieren von Problemen.

Als Vorgehensweise wird empfohlen:

- objektorientierte und prädikative Lösungen analysieren;
- Bestimmen der Komplexität zuerst an konkreten Beispielen;
- Übergang zu Komplexitätsklassen.

8.3.5. Ziel: Verifikation von Lösungen kennenlernen

Grundbegriffe
Gesucht sind Verfahren, mit denen man die Korrektheit eines Informatiksystems oder bescheidener ausgedrückt eines Programms beweisen kann. Die Analogie dazu ist der Beweis eines mathematischen Satzes.
Korrektheit liegt vor, wenn aus der Eingabespezifikation tatsächlich die und nur die gewünschte Ausgabespezifikation folgt.
Daraus lassen sich zwei Teilaufgaben bilden:

- Nachweis der korrekten Berechnung aller Ausgabewerte,
- Nachweis der Terminierung für alle zulässigen Eingabewerte.

Die Programmverifikation ist schwieriger und fehleranfälliger als das Programmieren selbst.
Inhalte und Methode

- Konsistenz:
  Das System selbst und die von ihm verarbeiteten Informationen sollen keine Widersprüche enthalten. Weltausschnitt (Modell) und Realität sollten bezüglich der zu lösenden Aufgaben gut harmonieren.

- Vollständigkeit:
  Die Vollständigkeit einer Spezifikation ist nachzuweisen. Es tritt auch Überspezifikation auf, wenn an die Software unnötig einschränkende Anforderungen gestellt werden.

- Gerechtigkeit (fairness):
  Die Zuteilung von Betriebsmitteln erfolgt so, dass mehrere Informatiksysteme parallel oder verzahnt (zeitlich ineinander gefügt) aktiv werden können.

- Softwaretechnik:
  Die Korrektheit großer Informatiksysteme soll mit besonderen Prinzipien, Methoden und Werkzeugen für ihre Entwicklung gesichert werden. Dahinter verbirgt sich die Hypothese, dass die richtige Arbeitweise, in diesem Fall eine ingenieurmäßige, Fehler von vornherein verhindern kann. Zu diesem Zweck wird der Entwicklungsprozess in Phasen unterteilt:

  - Problemanalyse und Anforderungsdefinition,
  - Entwurf und Programmentwicklung,
  - Programmierung und Implementierung,
  - Wartung.

  Ziel ist die Qualitätssicherung.

- Test – Suche nach Fehlern:
  Testen beweist nicht die Korrektheit, sondern nur die Anwesenheit oder Abwesenheit bestimmter Fehler. Das Testen wird nicht überflüssig durch eine Verifikation, da zwischen Quelltext und Maschinenprogramm durchaus semantische Unterschiede bestehen können.

- Terminierung:
  Terminierung bedingter Schleifen (Iterationen) untersuchen durch formale Methode:

  - Wählen einer Funktion (Heuristik erforderlich),
  - Reduktion des Funktionswertes je Schleifendurchlauf zeigen (streng monotones Fallen der Funktion).

- Partielle Korrektheit:
  Methode:

  - Zerlegung des Programms in überschaubare Teile,
  - Verwendung von Regeln der axiomatischen Semantik (Prädikatensematik): Prädikate als Vorbedingung und Nachbedingung (Zusicherungen) ,
  - Konstruktion einer Schleifeninvarianten.

Als Vorgehensweise wird empfohlen:

- Beispielprogramme zum Experimentieren;

- Formalisierungsmöglichkeiten für kritische Lösungsabschnitte.

8.4. Exemplarisches Können zum Programmieren

8.4.1. Ziel: Entwicklung von Informatiksystemen verstehen

Menschengerechte Technikentwicklung
Das Informatiksystem soll seine Reaktionen nach den Denk– und Arbeitsweisen der Benutzer richten. Dazu müssen bereits bei der Entwicklung des Systems bekannt sein:

- der Benutzerkreis (Fähigkeiten, Fertigkeiten, Nutzungsart),
- die Arbeitsaufgaben (Inhalte, Komplexität, Struktur).

So können Funktionalität und Benutzungsoberfläche genau spezifiziert werden. Der Nutzer soll sich auf seine Aufgabe konzentrieren können. Das Werzeug „Informatiksystem“ soll dabei in den Hintergrund treten und keine zusätzliche Belastung darstellen. Die Lernenden vertiefen die Anforderungen an die Benutzungsfreundlichkeit von Informatiksystemen und erfahren, wie diese zu realisieren sind:

- Problemangemessenheit: Dazu gehört, dass die Funktionalität geprüft wird.
- Dialogflexibilität: Die Steuerung und Aktivität sollte prinzipiell beim Nutzer liegen. Das System hat angemessene Rückmeldungen bereitzustellen.
  - Menüs können auch sehr lästig sein. Deshalb sollte der Nutzer Voreinstellungen und Sprünge vornehmen können.
  - Der Zeitbedarf ist zu überprüfen und
  - die Einbettung in den Gesamtprozess der Anwendung.
- Selbsterklärungsfähigkeit (Transparenz): Die Verständlichkeit ist zu überprüfen. Mit Farben können aktive von inaktiven Fenstern unterschieden werden. Jeder Zustandwechsel sollte von einem Farbwechsel begleitet sein.
- Zuverlässigkeit:
  - Fehlertoleranz: Das System soll auch dann noch korrekt arbeiten, wenn Komponenten fehlerhaft sind.
  - Der Nutzer kann Fehler rückgängig machen.
  - Warnungen.
- Hilfen
  - Hilfe–Funktionen: Informationen über den Modus (Zustand) des Systems,
  - aussagekräftige Fehlermeldungen,
  - Korrekturvorschläge.
- Erwartungskonformität: Sie wird verbessert, wenn einheitliche Syntax für anwendungsunabhängige Funktionen eingesetzt wird.
- Vorteile der Anwendung: Ein Vergleich mit der traditionellen Arbeitweise lässt Vor– und Nachteile erkennen. Untersuchungen zur Effizienz der Lösungsalgorithmen gehören in dazu.

Inhalte und Methoden
Die fachspezifische Vorgehensweise bei der Entwicklung von Informatiksystemen kann aus dem allgemeinen Problemlösungsprozess abgeleitet werden. Verschiedene Wissenszustände, die ein

Problemlösender erreichen kann, definieren einen Problem– oder Zustandsraum. Problemlösen ist dann die Zerlegung in Teilaufgaben bis Operatoren einen Anfangszustand in einen Zielzustand überführen. Der Problemlösende muss einen angemessenen Weg durch das Labyrinth von Zuständen finden (Suchprozesse). Mit Heuristiken kann solch ein Suchraum eingeschränkt werden. Die Teilziele sind so zu wählen, dass der Abstand zum Ziel abnimmt. Alle Objekte werden nach ihrer Funktion klassifiziert. Problemlösen umfasst dann [Anderson96]:

- Objekte mit Unterschieden erfassen.
- Operatoren wählen, um Merkmale zu verändern und damit Unterschiede zu beseitigen.
- Eingaben so modifizieren, dass Operatoren anwendbar werden.
- Schwer beeinflussbare Unterschiede durch Kombinationen von leichter veränderbaren ersetzen.

Bereichsübergreifende und bereichsspezifische Regeln werden auf diese Weise zur Anwendung gebracht. Von der konkreten Repräsentation der Objekte hängt die Wahl der Operatoren ab. Sprachen und Sprachklassen entscheiden ganz wesentlich über die Qualität der Informatiksysteme. Zu jeder Problemklasse lässt sich eine besonders geeignete Darstellungsform bestimmen. In der Ausbildung ist deshalb großer Wert auf die Wahl des geeigneten Werkzeuges zu legen.
Bei der Entwicklung von Informatiksystemen treten zwei typische Tätigkeitsabschnitte auf:

1. Von der Problemdefinition (unscharf) gelangt man durch Lösungshypothese zum Lösungsmodell (prinzipielle Lösbarkeit klären). Dazu gehören die Definition und Vorplanung und die Konstruktion (Spezifikation und Grobentwurf). Ein Weltausschnitt von unstrukturierten Ausgangsdaten wird mit unterlegt mit einem Strukturkonzept, das das zu untersuchende System (automatentheoretische, graphentheoretische) zerlegt und gliedert. Dabei sind Varianten möglich (Entscheidung für ein Paradigma). Außerdem werden Modelle genutzt (physikalische, graphische, mathematische, Simulationsmodelle).
   Prinzipien und Methoden: Hierarchisierung, Modularisierung, Top–down–Entwurf.
   Kognitive Leistungen: Entdecken (Wiederentdecken), kreatives Denken, begriffliches Denken, Variieren von Strukturkonzepten, Versuch–Irrtum–Strategie. Der schöpferischer Akt ist nicht algorithmierbar.

2. Vom Lösungsmodell erfolgt der Übergang zum Informatiksystem für eine Klasse von Problemen (abstrakte, virtuelle Maschine). Dazu gehören die Feinentwürfe und die Implementierung auf der realen Basismaschine. Bei dieser Konstruktion des Systems sind verschiedene Varianten möglich (verschiedene Hard– und Software–Lösungen für ein Modell).
   Prinzipien und Methoden: Top–down–Entwurf, Modulkonzept, schrittweise Verfeinerung, Vorteil fertiger Strukturkonzepte. Es stehen Entwicklungsumgebungen für verschiedene Problemklassen und Sprachen zur Verfügung, die Teamarbeit und Fehlersuche unterstützen.
   Kognitive Leistungen: hochspezialisierte Heuristiken und große Disziplin

Für den Lernenden spielt die Reduzierung des Anwendungsprozesses, die Komplexitätsreduzierung der zu lösenden Aufgabe und die Einführung in die Verfahrensheuristik (Lernen aus Beispielen) eine entscheidende Rolle. Damit wird das Verständnis für eine Reihe fundamentaler Ideen der Informatik vertieft [Schwill93] (Algorithmisierung, strukturierte Zerlegung, Sprache). Eine sinnvolle Verallgemeinerung für das fachdidaktische Vorgehen ist:

- Beschreibung des Problems,
- Komplexitätsreduzierung,
- Auswahl und Bewertung der Modelle,
- Transparenz der Lösung sichern,
- Effektivität untersuchen,
- Verifikation durchführen,
- Gestaltungsprinzipien diskutieren,
- Bewertung der Systeme vornehmen,
- Konsequenzen der Anwendung aufzeigen.

Dieser Ausbildungsabschnitt fördert interdisziplinäres Arbeiten, Teamfähigkeit und Projektarbeit [Ambros92].
In jeder Ausbildungsphase können auf höherem Niveau neue Erfahrungen und Erkenntnisse vom Lernenden gewonnen werden zu:

- Darstellung von Wissen durch geeignete Sprachen,
- Wissen strukturieren,
- Informatiksysteme erkunden:
  - Lösungskonzepte für Problemklassen,
  - Modellierung - objektorientiert,
  - Spezifikation - logisch,
  - Entwurf - funktional,
  - Implementation - prozedural,
  - Ausführung - maschinennah.

8.4.2. Ziel: Inferenzprozesse und maschinelles Lernen kennenlernen als Beispiel für einen Wahlkomplex

Folgende Inhalte sind dem Ziel zuzuordnen:

- Eigenschaften von Wissen:
  - Redundanz (Regel ist Spezialfall einer anderen Regel),
  - Widersprüche,
  - Unsauberkeiten (Sackgassen, Unerfüllbarkeit),
  - Unvollständigkeit,
  - Unsicherheit (Maß des Vertrauens),

  - Unpräzision (unscharfe Aussagen; z. B. „jung"),

- Wissenserwerb und Strukturierung,
- Regelauswahlstrategien (Suchbaum, Suchgraph):
  - blinde Suche,
  - Tiefensuche,
  - Breitensuche,
  - Heuristische Suchverfahren.
- Überprüfung und Aktualisierung,
- Gernerieren von Erklärungen.

Als Vorgehensweise wird empfohlen, ein wissensbasierte Systeme vorzustellen, zu analysieren und zu modifizieren (z. B. Sprachverarbeitung, Robotik).

9. Didaktische Konzepte

9.1. Entwicklung

9.1.1. Entstehung des Informatikunterrichtes

Bis heute spüren wir die Auswirkungen des fehlenden Gesamtkonzepts der informatischen Bildung. Berufliche Schulen und Hochschulen beginnen im Fach Informatik ohne gesicherte Vorkenntnisse. Eine Ausnahme zeichnet sich in Sachsen ab, da das Fach „Angewandte Informatik“ in allen Mittelschulen (Schulen für Real– und Hauptschulabschluss) von den Jahrgangstufen 7 bis 10 verpflichtend unterrichtet wird. Schwierigkeiten bereiten den beruflichen Schulen deshalb hier die Absolventen der Regelgymnasien, denen dieses Wissen und Können fehlt. Hochschulen differenzieren ihre Studienprogramme vorerst nicht, d.h. Absolventen ohne Informatikgrundkurs und solche mit Leistungskurs einschließlich Informatikabitur erfahren die gleiche Ausbildung. Diese Wiederholungszeiten schaden den Studierenden im internationalen Wettbewerb um zügige Studienabschlüsse. Abhilfe kann nur ein Gesamtkonzept bringen, das die Bildungsziele für die Sekundarstufen I und II verbindlich festlegt.

Das Informatikbildungsangebot konnte dort zuerst Fuß fassen, wo eine bildungspolitische Umgestaltung Freiräume für das neue Fachgebiet ermöglichte. Da es gute Gründe für die Aufnahme in die Allgemeinbildung gab (z.B. Rolle der Informatik in anderen Wissenschaftsdisziplinen) und der internationale Anschluss nicht verpasst werden sollte, wurde die Entwicklung der gymnasialen Oberstufe genutzt.

Mitte der 70er Jahre beschloss die Ständige Konferenz der Kultusminister (KMK) in einer Rahmenvereinbarung, das Fach Informatik in der gymnasialen Oberstufe einzuführen mit folgenden Schwerpunkten:

- Algorithmen und Datenstrukturen (leider kam es häufig zur Umsetzung als Programmiersprachenkurs),
- Projektunterricht zur Entwicklung eines Software–Projektes.

Schüler der Sekundarstufe II konnten nun eine Informatikausbildung wählen. Es gab Stimmen, die es ungerecht fanden, dass damit nur in der gymnasialen Oberstufe eine solche Möglichkeit geboten wurde. Da das neue Fach gut aufgenommen wurde, fanden sich viele Lehrer bereit, eine Fortbildung oder einen autodidaktischen Einstieg zu nutzen, um das Fach zu unterrichten. Eine fundierte Lehrerausbildung besitzt nach wie vor Seltenheitswert. Mit der Kritik am unzureichenden Informatikunterricht zogen sich die Schüler und Lehrer systematisch zurück. Befördert wurde dieser Prozess ebenso von den sich ständig verschlechternden Wahlmöglichkeiten für das Fach. Informatik erwies sich als schweres Fachgebiet, in dem die Abiturpunkte nicht einfach zu erwerben sind. Es kann keine Naturwissenschaft ersetzen, bringt also in der Pflichtbelegung keinen Vorteil. So verkümmerte ein innovatives Fach aus mehreren Gründen: fehlende Lehrerbildung, schlechte Wahlbedingungen und Unklarheit über die konzeptionelle Ausgestaltung.

Ab 1989 wurde davon gesprochen, dass der Informatikunterricht in der Krise steckt, da man in den oben aufgeführten Schwerpunkten keinen sinnvollen Beitrag zur Allgemeinbildung mehr sah [Peschke89]. Das Argument: „Wir brauchen keine Nation von Programmierern.“, war immer häufiger zu hören.

9.1.2. Ein Fundamentum für alle Schüler

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“ [BLK84]. Seitdem ist der Begriff „Informationstechnische Grundbildung (ITG)“ für dieses Fundamentum festgeschrieben, das für alle Schüler obligatorisch ist. Es existieren viele Möglichkeiten für die konkrete Realisierung. Die Ausbildung:

- findet in den Jahrgangsstufen 7 bis 9 statt,
- umfasst 40 bis 90 Stunden und
- wird im Verteilungsmodell oder Blockmodell organisiert [Bosler92].

Das Verteilungsmodell hat ein Leitfach, das mit unterstützenden Unterrichtsfächern kooperiert, die die Inhalte der ITG unter sich aufteilen und aufeinander Bezug nehmen. Die betroffenen Fächer müssen also Zeit für die ITG abgeben, d.h. auf traditionellen Stoff verzichten. Das Blockmodell stellt in einer oder mehreren Jahrgangsstufen ein Stundenvolumen im Block für die ITG zur Verfügung. In beiden Fällen wird die ITG von dafür ausgewählten Lehrern anderer Fächer unterrichtet, die dafür in einer sehr kurzen Weiterbildung vorbereitet wurden. Kritiker vermerken, dass bundesweit Vollzug gemeldet wird, aber oft keine ITG stattfindet.

Die erste Ausnahme bildete Sachsen mit dem Fach Angewandte Informatik in den Klassenstufen 7 bis 10 an Mittelschulen, die zum Real– und Hauptschulabschluss führen [SMK97].

9.1.3. Die Lücke in der Sekundarstufe I

1987 erschienen die von der BLK fortgeschriebenen Empfehlungen in Form eines „Gesamtkonzeptes für die informationstechnische Bildung“ [BLK87]. Dabei wurden die Problemlösungsmethoden der Informatik in den Mittelpunkt der Ausbildung gestellt. Es war der Versuch, mit einer Bildungslinie von der ITG zum Informatikunterricht in der SII die Lücke in der SI zu schließen. Dieser Versuch ist bis heute nicht bundesweit gelungen. Die SII–Sicht von der Programmierumgebung nimmt meist keinen Bezug auf die ITG, die mit Standardsoftware Aufgaben löst.

Es ist aber Bewegung in den SI–Bereich gekommen. Begonnen hatte es mit dem Wahlpflichtfach in Berlin und einem Informatikblock im Fach Mathematik in Bayern und Baden–Württemberg. Sachsen brachte das erste Pflichtfach 1992 mit Zielen und Inhalten, die im ITG–Bereich blieben bis zur Neuorientierung 1997. Das Wahlfach wird in allen Bundesländern angeboten, wenn Lehrer zur Verfügung stehen und das Schülerinteresse wecken.

9.2. Informationstechnische Grundbildung (ITG)

9.2.1. Ziele

Die Schüler sollen in ca. 60 Stunden als aktive Benutzer ausgebildet werden:

1. Aufarbeitung und Einordnung der individuellen Erfahrungen mit Informationstechniken,
2. Vermittlung von Grundstrukturen und Grundbegriffen, die für die Informationstechniken von Bedeutung sind,
3. Einführung in die Handhabung des Computers und dessen Peripherie,
4. Vermittlung von Kenntnissen über die Einsatzmöglichkeiten und die Kontrolle der Informationstechniken,
5. Einführung in die Darstellung von Problemlösungen in algorithmischer Form,

6. Gewinnung eines Einblicks in die Entwicklung der elektronischen Datenverarbeitung,
7. Schaffung des Bewusstseins für die sozialen und wirtschaftlichen Auswirkungen, die mit der Verbreitung der Mikroelektronik verbunden sind,
8. Darstellung der Chancen und Risiken der Informationstechniken sowie Aufbau eines rationalen Verhältnisses zu diesen,
9. Einführung in die Probleme des Persönlichkeits– und Datenschutzes.

Daraus werden vier Lernbereiche abgeleitet:

- Anwendungsbereich,
- gesellschaftlicher Bereich,
- algorithmischer Bereich,
- technischer Bereich.

9.2.2. Inhalte

Das Konstruieren und Benutzen von DV–Systemen und die gesellschaftliche Bedeutung der Informatik wurden ursprünglich als Ziel genannt mit folgenden Inhalten:

- Verwaltung von Schülerdateien,
- Kalkulation einer Klassenfahrt,
- Abrechnung eines Schulfestes,
- Computerkassen im Supermarkt,
- Wahlanalysen,
- Bewerbungsschreiben,
- Schülerzeitung,
- Simulation ökologischer Systeme,
- Informationstechnik in der U–Bahn,
- Auswertung von Schülerumfragen zu aktuellen Themen.

Sachsen ist derzeit das einzige Bundesland mit einem eigenen Unterrichtsfach für die ITG. An der Mittelschule gibt es das Fach Angewandte Informatik, das ab 1992 im technischen, sprachlichen und sozial–/hauswirtschaftlichen Profil in den Jahrgangsstufen 9 und 10 (60 Stunden) und im wirtschaftlichen Profil von 7 bis 10 (120 Stunden) unterrichtet wurde [SMK92b]. Im Gymnasium findet eine sehr reduzierte Variante der ITG statt im Fach Informatik der Jahrgangsstufe 7 mit nur 30 Stunden [SMK92a].

Stets wird Textverarbeitung gelehrt, dazu kommt:

- an der Mittelschule von 1992–97 im:
  - technischen Profil: Prozessdatenverarbeitung,
  - sprachlichen Profil: Desktop Publishing,
  - sozial–/hauswirtschaftlichen Profil: Tabellenkalkulation,
  - wirtschaftlichen Profil:
    - ▷ Datenbanken,
    - ▷ Tabellenkalkulation,
    - ▷ Desktop Publishing,
- am Gymnasium: Problemlösen mit Algorithmen und Datenstrukturen und rekursiver Arbeitsweise (4 Stunden).

Auf den neuen Orientierungsrahmen [SMK97] für die sächsische Mittelschule wird im Abschnitt 9.3.2 der Sekundarstufen–I–Konzepte näher eingegangen.

9.2.3. Methode

Die ITG nutzt eine Abbildung von Realität durch Vereinfachung und Veranschaulichung. Beispiele für Realitätsausschnitte und *Unterrichtsthemen* in Hessen waren z.B.:

- Umweltbelastung – *Lärm*,
- Bürokommunikation/Textverarbeitung – *Presseagentur*,
- Computerspiele und Video – *Neue Medien*,
- Prozesssteuerung/Fertigung – *Schritt für Schritt*,
- Transportwesen – *Telefracht*,
- Datenschutz – *Datenbanken und Datenschutz*,
- Homebanking/Dienstleistungen – *Die Bank im Wohnzimmer*,
- Supermarkt/Handel – *Warenhaus*.

Folgende Schülerrollen wurden beschrieben [Peschke90]:

- **Bediener:** Geräte bedienen und Daten eingeben,
- **Benutzer:** Anwendungen gestalten,
- **Betroffener:** Folgen kennen bzw. erfahren,
- **Gestalter:** selbständig algorithmisierbare Probleme lösen.

Der Informatikunterricht orientiert sich sehr stark auf den Gestalter. Die Rollen Betroffener und Benutzer sind von geringerer Bedeutung und der Bediener spielt angeblich eine untergeordnete Rolle. In der ITG soll der Benutzer im Mittelpunkt stehen. Der Gestalter hat angeblich kaum einen Stellenwert. Bediener und Betroffener sind wichtig.

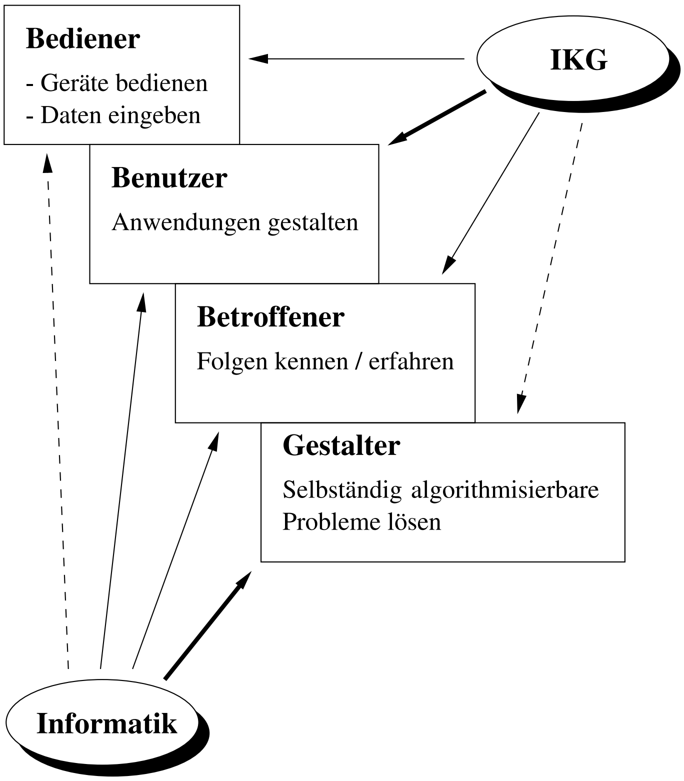


Abbildung 52: „Schülerrollen“ beim Umgang mit Computertechnik [Peschke90], S. 32

9.2.4. Beispiele

(a) Die Berliner ITG–Ausbildung gestaltet unter anderem das Beispiel „Im Reisebüro“ in 11 Unterrichtsstunden:

- 1. Stunde mit dem Thema *Im Reisebüro I; Btx.* und folgenden Inhalten:
  - Tätigkeiten zusammenstellen,
  - Btx–System,
- 2. Stunde mit dem Thema *Im Reisebüro II; Fahrpläne; Buchen im Reisebüro.* und folgenden Inhalten:
  - das EVA–Prinzip beim Buchen,
  - Algorithmus,
  - Wie funktioniert Btx?
- 3. Stunde mit dem Thema *Das Autobus–Platzbuchungssystem BUS I.* und folgenden Inhalten:
  - Funktionsüberblick,
  - Bedienung,
- 4. Stunde mit dem Thema *Das Autobus–Platzbuchungssystem BUS II.* und folgenden Inhalten:
  - Buchungsliste,
  - Stornieren,
  - Rekonstruktion der Arbeitsvorgänge,
- 5. Stunde mit dem Thema *Algorithmen I; Programm I.* und folgenden Inhalten:
  - Pascalprogramm zu „Einnahmen“ (Entwurf und Analyse des fertigen Programms),

  - Programm nutzen,

- 6. Stunde mit dem Thema *Buchen über Btx.* und folgendem Inhalt:
  - professionelles Buchungssystem kennenlernen,
- 7. und 8. Stunde mit dem Thema *BUS III; Programm und Dateien.* und folgenden Inhalten:
  - neue Busse aufnehmen,
  - Fahrtenplanung,
  - Inhaltsverzeichnis einer Diskette,
  - Quellprogramm,
  - compiliertes Programm,
  - BUS–Dateien,
- 9. Stunde mit dem Thema *Datenschutz.* und folgenden Inhalten:
  - personenbezogene Daten (Wer speichert sie?),
  - Gefahr der Verknüpfung,
  - Datenschutzgesetz,
- 10. Stunde mit dem Thema *Reiseverkehrskaufmann(frau).* und folgendem Inhalt:
  - Berufsbild,
- 11. Stunde mit dem Thema *Informationen des Arbeitamtes mit Hilfe von Btx einholen.* und folgenden Inhalten:
  - Aufsuchen von Btx–Seiten,
  - Informationen über Arbeitsämter.

(b) In Nordrhein–Westfalen ([Bosler92], S. 105ff) hieß die ITG Informations– und Kommunikationstechnologische Grundbildung (IKG) und wurde bis 1998 mit drei Lernfeldern untersetzt:

- Prozessdatenverarbeitung,
- Textverarbeitung, Dateiverwaltung, Kalkulation,
- Modellbildung und Simulation.

IKG wurde im achten Schülerjahrgang unterrichtet [AKHK+90]. 1998 wurden neue Richtlinien entwickelt, die die IKG in die Medienerziehung integrieren.

9.2.5. Probleme

Gesprochen wird vom Computer als „Denkzeug“. Die Bedeutung dieser symbolverarbeitenden Maschine mit universellen Einsatzbereichen soll verstanden werden, um selbstverantwortliche Lebensgestaltung zu ermöglichen.
Es besteht die Gefahr der Verengung der ITG auf den Computer und seine Bedienung. Die Undurchsichtigkeit der Systeme fördert eine solche Oberflächenkompetenz.

9.3. Informatik in der Sekundarstufe I an Beispielen

9.3.1. Gymnasium Nordrhein-Westfalen

Informatik wird als Wahlpflichtfach in den Jahrgangsstufen 9 und 10 mit drei Wochenstunden unterrichtet nach einem Lehrplan von 1993 [KMNW93].

a) Ziele

Werkzeugfunktionen von Informatiksystemen sollen mit wissenschaftlichen Arbeitsweisen so verbunden werden, dass die Lernenden Planen, Konstruieren, Anwenden und Kontrollieren im Fach verstehen und das Betroffensein erleben.

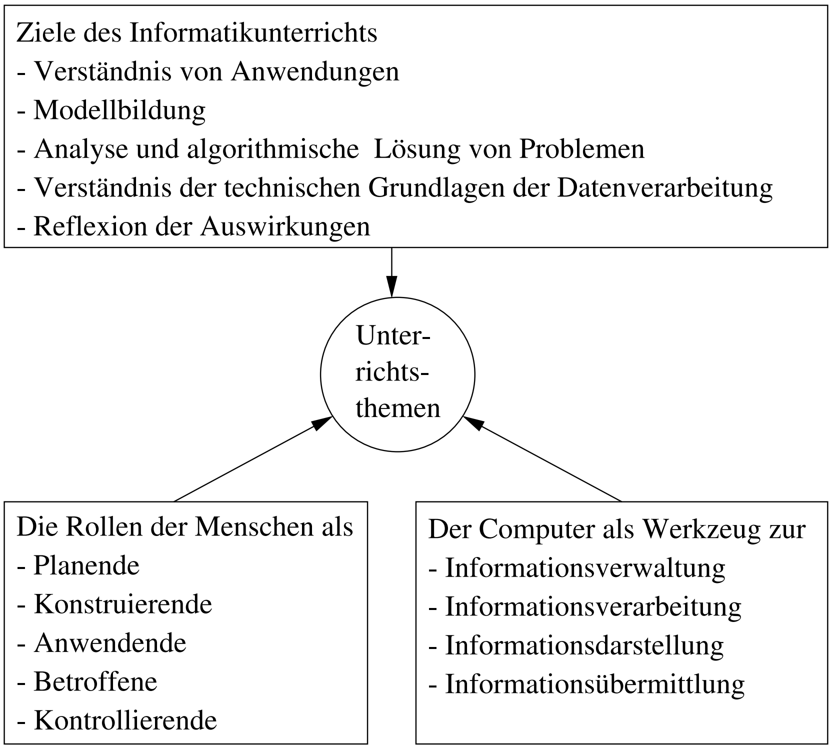


Abbildung 53: Lehrplanintention [KMNW93], S. 35

b) Inhalte

Für jedes Schulhalbjahr werden obligatorische und fakultative Inhalte ausführlich aufgezählt mit Beispielen untersetzt.

- 9.1: Umgang mit Software
- 9.2: Funktionsweise von Software
- 10.1: Funktionsweise von Hardware, Prozessdatenverarbeitung,
- 10.2: Softwareprojekte

c) Besonderheiten

Informations– und Kommunikationssysteme werden als einer von sechs Bereichen ausgewiesen, die im Unterricht abgedeckt werden sollen.

Objektorientierte Modellierung kann in 9.2 thematisiert werden mit:

- „Datenkapselung,
- Anbindung von Methoden an Objekte,
- Elementare Vererbung,
- Elementares Nachrichtenkonzept“ [KMNW93], S. 44.

Der Lehrplan bleibt an dieser Stelle erstaunlich unkonkret und knapp, so dass von einer Ermutigung der Lehrer nicht ausgegangen werden kann. In 10.2 soll es in einer komplexeren Aufgabenstellung in Teamarbeit vertieft werden:

- „Allgemeines Vererbungskonzept: Objektklassenhierarchien,
- Allgemeiner Nachrichtenaustausch : Ereignissteuerung“ [KMNW93], S. 48.

Da gleichzeitig auch noch Simulation in 10.2 gelehrt und angewendet werden soll, bleibt die Realisierbarkeit mit empirischen Studien zu untersuchen. Wissen über Programme, Programmierbarkeit und das übergeordnete Informatische Modellieren lassen sich (vergleiche Kapitel 8) in diesem Rahmen sehr vernünftig thematisieren. Fähigkeiten oder gar Fertigkeiten sind unklar.

9.3.2. Mittelschule Sachsen

Seit dem Schuljahr 1997/98 besteht die Möglichkeit in allen Profilen von Klassenstufe 7 bis 10 Angewandte Informatik mit einer Wochenstunde als Pflichtfach zu unterrichten [SMK97].

a) Ziele

Die Trennung von ITG und Vertiefung SI fällt schwer, aber hier soll die Schule nach Zielen, die über den Umgang mit Anwendungssystemen hinausgehen, erfolgen:

„Die Schüler

7: erlangen erste Einblicke in die historische Entwicklung der Informatik,

8: entwickeln prinzipielles Verständnis für die Funktion und das Zusammenwirken der Hardwarekomponenten sowie deren Beeinflussung durch Software,

9: können Modellieren und Aufstellen von Algorithmen,

10: lernen Grenzen der verwendeten Informatiksysteme kennen“ [SMK97], S. 6ff.

b) Inhalte

Die inhaltlichen Schwerpunkte sind sehr knapp gehalten. Wieder werden Themen ausgewiesen, die über Nutzungskompetenz hinausgehen:

7: Historische Entwicklung,

8: Rechnerarchitektur und Betriebssysteme (8 Std.), Verwendung logischer Operatoren, Datenschutz und Datensicherheit,

9: Steuern mittels Computer, Netzwerke und Kommunikationssysteme (5 Std.), 10: gesellschaftliche und historische Aspekte der Informatik, Variante Grafik, Variante Steuern und Progammieren

c) Besonderheiten

Da der Orientierungsrahmen 1997 abgeschlossen wurde, finden sich solche aktuellen Bildungsanforderungen wie „Netzwerke und Kommunikationssysteme“ und „Hypermediasysteme“. Das ist sehr zukunftsweisend. Die Umsetzung der Informatikthemen wird aber mit zu knappen Stundenzahlen sehr unwahrscheinlich. Viel Unterrichtszeit kann für die Informatikziele und –inhalte dadurch gewonnnen werden, dass man die typischen Anwendungssysteme in den Fächern einsetzt, denen sie besonders nahestehen:

7: Arbeit mit Texten (15 Stunden + 4 Stunden) im Fach Deutsch,

9: Arbeiten mit Tabellen (12 Stunden) im Fach Mathematik,

10: Komplexe Anwendung (10 Stunden)
Variante Desktop Publishing im Fach Deutsch,
Variante Musik im Fach Musik,
Variante Hypermediasysteme fächerverbindend Deutsch, Fremdsprachen, Naturwissenschaften, Kunst.

Die dadurch gewonnenen Freiräume werden unbedingt benötigt, um die ausgewiesenen Informatikthemen bis zu Wissens– und Könnenszielen bei den Schülern auszugestalten. Es wird dann sogar möglich, die Informatikwirkprinzipien der „Hypermediasysteme“ in das Fach aufzunehmen.

9.4. Informatik in der Sekundarstufe II

9.4.1. Aufgaben und Empfehlungen

Die BLK legte folgende Aufgaben fest [BLK87]:

1. Behandlung der Wirkungsweise, Leistungsfähigkeit und Leistungsgrenzen von Rechnern,
2. Vermittlung von Problemlösungsmethoden,
3. Vermittlung von Kenntnissen bestimmter Programmiersprachen,
4. Behandlung des strukturierten Programmierens und der Datenstrukturen,
5. Einsatz von Rechnern für Berechnungen, für die Erstellung von Grafiken und für die Simulation von Verfahren,
6. Erörterung von Prozesssteuerung durch Mikroprozessoren.

1976 gab die Gesellschaft für Informatik (GI) die ersten Empfehlungen für den Informatikunterricht in der Sekundarstufe II heraus [Brauer76]:

1. systematisches Finden algorithmischer Problemlösungen,
2. Formulierung algorithmischer Problemlösungen als Programm,
3. Vertiefung durch Anwendung auf praxisorientierte Probleme,
4. Erkennen der Auswirkung der DV auf die Gesellschaft,

5. Vertiefung und Erarbeitung theoretischer und technischer Grundlagen der Informatik.

1993 veröffentlichte die Gesellschaft für Informatik neue Empfehlungen [SZ93]. Gefordert wird fundiertes Wissen über Informatiksysteme:

- breites Spektrum von Problemlösungsansätzen,
- Programmierparadigmenwechsel,
- Wissensrepräsentation und –verarbeitung,
- Parallelverarbeitung,
- verteilte Systeme.

Die Chancen und Risiken der Informatikanwendung sind auf zwei Ebenen zu lehren:

- Mensch–Computer,
- Informatiksysteme, Gesellschaft und Umwelt.

Seit 1981 existieren „Einheitliche Prüfungsanforderungen für die Abiturprüfung im Fach Informatik“, kurz EPA genannt, die von der KMK beschlossen werden. Die gültige EPA wurde 1989 beschlossen und 1991 veröffentlicht [KMK91]. Zu Fragen der Gleichwertigkeit von allgemeiner und beruflicher Bildung kann in [KMK94] nachgelesen werden.

9.4.2. Beispiele

(a) Sachsen

Sachsen ermöglicht den Grundkurs Informatik im Wahlbereich der Jahrgangsstufe 11 und 12 mit folgendem Inhalt [SMK92a]:

Kl.	**Inhalte**	**Std.**
11	Problemlösung mit einfachen und komplexen Datenstrukturen:	
	Einführung in die Programmierung:	(60)
	Grundstrukturen, Datentypen, Datenstrukturen Feld und Verbund, Unterprogrammtechnik	
oder	Komplexe Problemlösungen:	(60)
	Datenstrukturen Feld und Verbund	12
	Unterprogrammtechnik	14
(oder	Datenstruktur Datei und typische Dateioperationen	14
in 12)	Projektarbeit	16
12.1	Datenbank– und Informationssysteme	30
oder	Prozessdatenverarbeitung	30
oder	Computergrafik	30
12.2	Datenschutz und Datensicherheit	8
	Geschichte der Informatik	2
	Algorithmentheorie:	14
	Berechenbarkeit (Halteproblem), Komplexität, Verifikation	

(b) Nordrhein–Westfalen

In Nordrhein–Westfalen wurden 1982 Richtlinien für Informatik für die gymnasiale Oberstufe veröffentlicht [MSNW96]. 1991 wurden vorläufige Richtlinien für Leistungskurse im Fach Informatik für die Gymnasiale Oberstufe herausgegeben [KMNW91]. Zur Zeit findet noch die Diskussion der neuen Richtlinien statt, die den Verbänden zur Stellungnahme vorliegen. Die ursprünglichen 1982 veröffentlichten Richtlinien orientierten sich maßgeblich in Zielsetzung und Aufbau an den GI–Empfehlungen von 1976. Die Organsiation erfolgt in der gymnasialen Oberstufe in der Form, dass die Schüler Informatik vom 11. bis zu 13. Jahrgang durchgängig belegen können. Nach jedem Schulhalbjahr kann Informatik abgewählt werden.

Die inhaltliche Gliederung

11/I:	Einführung in die Informatik,
11/II:	Algorithmik I: Methoden des systematischen Problemlösens,
12/I:	Struktur und Arbeitsweise einer Datenverarbeitungsanlage,
12/II:	Algorithmik II: Datenstrukturen,
13:	Probleme aus der praktischen Anwendung der Datenverarbeitung

ist festgelegt und darf nur nach Genehmigung durch die obere Schulaufsichtsbehörde geändert werden. Die vorläufigen Richtlinien für Leistungskurse im Fach Informatik hingegen weisen eine geänderte Struktur auf. In der Einleitung wird ausdrücklich formuliert:

> „... dem erfahrenen Fachlehrer eröffnen die vorliegenden Regelungen sehr weite Freiräume für methodische Entscheidungen und die Auswahl und Anordnung der Unterrichtsinhalte. Er wird die Möglichkeit begrüssen, selbständig thematische Schwerpunkte zu setzen, stoffliche Ergänzungen auszuwählen und neue Anwendungsgebiete zu erproben, um so einem Unterricht Aktualität und fachlichen Anspruch zu sichern.“ [KMNW91], S. 1.

Auf S. 22 f wird deutlich, dass diese Richtlinien auch für die Grundkurse genutzt werden können.

Inhaltlich:

- Lernbereich I (Algorithmik):
  algorithmische Strategien, Standardalgorithmen zum Suchen und Sortieren sowie Aussagen über Algorithmen nur exemplarisch (gegenüber dem Leistungskurs),

- Lernbereich II (Datenstrukturen):
  Anzahl der zu behandelnden Datenstrukturen einschränken,
  Während in einer Leistungskurssequenz beim Problemlösen die Phasen von der Modellbildung bis zur Repräsentation und Implementation verschiedener Datenstrukturen häufiger durchlaufen werden, wird der Grundkurs dies nur exemplarisch tun und bei vielen Datenstrukturen nach einer funktionalen Spezifikation auf entsprechende vorhandene Werkzeuge zurückgreifen und diese zur Problemlösung einsetzen.

- Lernbereich III:
  Schwerpunkte der Grundkurssequenz – exemplarische Behandlung der von–Neumann–Maschine und der Implementation von Algorithmen und Datenstrukturen auf Maschinenebene,

  > „Auf die Behandlung einer weiteren Programmiersprache muss im Grundkurs ebenso wie auf eine Vertiefung im Bereich der theoretischen Informatik in der Regel verzichtet werden.“ [KMNW91], S. 22

- Lernbereich IV:
  Realisierung, Probleme und Auswirkungen der praktischen Datenverarbeitung,

  > „... so sind im Rahmen einer Grundkurssequenz ... Einschränkungen hinsichtlich der Komplexität der zu behandelnden praktischen Probleme vorzunehmen. Beim Prozess der Modellbildung wird das ursprüngliche Problem weitergehenderen Reduktionen zu unterwerfen sein als im Rahmen einer Behandlung im Leistungskurs.“ [KMNW91], S. 23

Beispiel für eine Kurssequenz

Voraussetzung: Schüler in der Jahrgangsstufe 11.1

- beherrschen alle Kontrollstrukturen,
- kennen die elementaren Datentypen,
- kennen die Grundlagen des Prinzips der schrittweisen Verfeinerung und der
- Strukturierung von Algorithmen durch Prozeduren.

11.2	1.	eindimensionale Felder
	2.	Prozeduren und Funktionen
	3.	Strukturierte Objekte
12.1	4.	Standardalgorithmen
	5.	Projekt
	6.	Dateien
	7.	Dynamische Strukturen
12.2	8.	Vom Programm zur Maschine
	9.	Betriebssysteme
	10.	Programmiersprachenvergleich
13.1	11.	Anwendungen in grösseren Systemen
13.2	12.	Strukturierte Wiederholung

(c) Thüringen

Thüringen erprobt im Leistungsfach Informatik ein sehr zukunftsweisendes Konzept. Der Lehrplan stützt sich erstmals auf Leitlinien [Thüringen98]:

- Umgang mit Information,
- Wirkprinzipien von Informatiksystemen,
- Problemlösen mit Informatiksystemen,
- Auswirkungen der Informatik auf Individuum und Gesellschaft.

Nach Jahrgangstufen ergibt sich dabei folgende Inhaltsstruktur:

- *Themenbereiche im Leistungsfach Informatik Jahrgangstufe 11*

1	Kommunikation in Netzen	18 Std.
2	Bearbeiten von Problemen mit PASCAL oder OBERON	40 Std.
3	Iteration, Rekursion und Backtracking	25 Std.
4	Sortieren und Suchen	15 Std.
5	Listen und Bäume	20 Std.
6	Realisation und Anwendung von abstrakten Datentypen	25 Std.
7	Projektarbeit I	25 Std.

- *Themenbereiche im Leistungsfach Informatik Jahrgangstufe 12*

8	Möglichkeiten und Grenzen des Einsatzes von Informatiksystemen	25 Std.
9	Logik–orientiertes Programmieren	40 Std.
10.1	Einblick in die Technische Informatik	25 Std.
	oder:	
10.2	Einblick in formale Sprachen	25 Std.
11	Projektarbeit II	40 Std.
12	Prüfungsvorbereitung	20 Std.

9.4.3. Lehrmethoden

a) Erfahrungen mit Lehrmethoden für die informatische Bildung

Die ersten drei Lehrmethoden wurden nach [Buse80] eingeteilt in:

- hardwareorientierter Ansatz:
  Die Wirkprinzipien der Hardware konnten in der Gründungsphase der Informatik tätigkeitsorientiert gelehrt werden, indem man schrittweise mit den Schülern einen Computer baute.
  - Lehrmethode für den Informatikunterricht ab 1970,
  - heute in aktuellen Lehrplnen der beruflichen Ausbildung,
  - Kritik: wurde zu einseitig realisiert,
  - Ablehnung durch die Mathematiklehrer,
  - wichtige Komponente, die heute unterschätzt wird.
- algorithmenorientierter Ansatz:
  Er entstand durch die GI–Empfehlungen von 1976 und dominierte lange Zeit (bis 1993) alle existierenden Lehrpläne. Die enge Verbindung zur Mathematik führt zum Implementieren bekannter Algorithmen (meist aus der Mathematik) im Rahmen des Informatikunterrichts.
  - Kritik: Sprache ohne Modellieren, Einseitigkeit und unbegründete Tiefe
- anwendungsorientierter Ansatz:
  Die Arbeitsweise der Software–Technologie wurde zum didaktischen Modell für Planung und Durchführung des Informatikunterrichts empfohlen. In stärkster Ausprägung trat dieser Ansatz nur im früheren Berliner Lehrplan auf. Er wurde am häufigsten missverstanden.
  - anfangs überzogene Erwartungen an Software–Projektunterricht,
  - gilt in dieser Form als gescheitert,
  - später reduziert auf Aufgabenlösen mit Anwendungssoftware,

  - Begriff wird heute wieder positiv belegt für lebensnahen Informatikunterricht.

- systemorientierter Ansatz [Lehmann93]:
  Der neue Berliner Lehrplan hat das Ziel, vom Programmieren im „Kleinen" zum Programmieren im „Großen" (Systementwicklung) überzugehen. Wohlmodellierte und transparente Systeme werden mit Schülern im Anfangsunterricht analysiert, um die Informatikprinzipien und –methoden so gut zu erlernen, dass Bausteine modifiziert werden können und später selbständige Systementwicklung möglich ist.

  - Kritik:
    - ▷ eingeschränktes Modifizieren von Bausteinen ohne Modellieren,
    - ▷ Zweifel an der Erlernbarkeit der Gestaltungskompetenz,
  - Mangel an geeigneten Systemen.

Eine Besinnung auf die „Fundamentale Ideen der Informatik" soll helfen, tragfähige Unterrichtslinien aufzuzeigen. „Was nicht prinzipiell auch einem Volksschüler vermittelt werden kann, kann keine fundamentale Idee sein." [Fischer84]. Die bisherige Unterrichtslinie „Problemlösen mit Mitteln und Methoden der Informatik" erwies sich als zu grob, um den Informatikstoff zu strukturieren. Peschke forderte deshalb 1989 dazu auf, den Bildungskern der Informatik für die Schule zu suchen [Peschke89]. Seitdem sind eine Reihe von Anregungen in der Erprobung:

- Nievergelt 1991 [Nievergelt91]:
  Algorithmen und Datenstrukturen veralten nicht. Sie sind und bleiben das klassiche Thema der Informatik. Sie müssen aber anders als bisher gelehrt werde.

- Schubert 1991 [Schubert91]:
  Eine sinnvolle Verallgemeinerung für das fachdidaktische Vorgehen ist:

  - Beschreibung des Problems,
  - Komplexitätsreduzierung,
  - Auswahl und Bewertung der Modelle,
  - Transparenz der Lösung sichern,
  - Effektivität untersuchen,
  - Verifikation durchführen,
  - Gestaltungsprinzipien diskutieren,
  - Bewertung der Systeme vornehmen,
  - Konsequenzen der Anwendung aufzeigen.

- Wirth 1992 [Wirth92]:
  Er stellt die Bedeutung des prozeduralen Programmierenkonzeptes für die objektorientierte Programmierung dar.

- Baumann 1993 [Baumann93]:
  Der Begriff der Berechenbarkeit der Informatik muss thematisiert werden. Dazu gehört systemorientierter Unterricht, der Problemlösen als Systementwicklung lehrt.

- Schwill 1993 [Schwill93], [CS93]:
  Er veröffentliche folgende fundamentale Ideen der Informatik im Duden Informatik mit dem Ziel, damit den Informatikunterricht zu strukturieren:

  - Algorithmisierung:
    - ▷ Entwurfsparadigmen,

    - ▷ Programmierkonzepte,
    - ▷ Ablauf (Prozess, Nebenläufigkeit, Prozessor),
    - ▷ Evaluation (Verifikation, Komplexität),
  - strukturierte Zerlegung:
    - ▷ Modularisierung (Methoden, Hilfsmittel),
    - ▷ Hierarchisierung (Darstellung, Realisierung),
    - ▷ Orthogonalisierung (Emulation),
  - Sprache
    - ▷ Syntax (Erkennen, Erzeugen),
    - ▷ Semantik (Konsistenz, Vollständigkeit, Transformation).

Dieser Vorschlag für fundamentale Ideen findet bisher wenig Zustimmung und Umsetzung im Informatikunterricht, da er einen sehr eingeschränkten Bereich der Informatik in die Auswahl einbezieht. Aspekte der technischen und angewandten Informatik fehlen darin.

In der Informatik existieren alternative Konzepte, die zur Lösung von Aufgaben unterschiedlich gut geeignet sind (siehe S. 72). Sie ermöglichen eine neue Vorgehensweise im Informatikunterricht:

- Analyse der Aufgabenstellung,
- Wahl des geeigneten Programmierparadigmas,
- Modellierung der Lösung und Übertragung in die zugehörige Sprache,
- Variantenvergleich zwischen Problemlösungen in verschiedenen Programmierparadigmen.

Der Einblick in verschiedene Modellierungsmethoden bekommt einen deutlich höheren Stellenwert als die einseitige Vertiefung in einem Paradigma.

b) Die Suche nach neuen Lehrmethoden

Interessant wird eine Anregung von Walter Dosch [Dosch93] zu Ebenen der Programmentwicklung (siehe Abbildung 54) von 1992 heute, da sie die Verbindung von unterschiedlichen Programmierparadigmen ohne Vertiefung in mehreren Sprachkonzepten ermöglicht. Daraus lässt sich eine Lehrmethode ableiten, die z.B. zum Entwickeln einer Spezifikation in das prädikative Modellieren mit Fakten und Regeln einführt, ohne auf eine Programmiersprache angewiesen zu sein. Analog kann zum Verständnis für einen Algorithmenentwurf das funktionale Modellieren eingesetzt werden. Virtuelle Maschinen sind eine anspruchsvolle Metapher für Informatiksysteme, deren abstrakte Implementierungen mit einem prozeduralen Modell erklärt werden kann. Selten wird eine maschinennahe Repräsentation im Informatikunterricht gewählt. Gerade sie eignet sich aber ausgezeichnet, um konkrete Implementierungen zu verstehen. Auf die Notwendigkeit technischer Basiskonzepte der Informatik wird hier mit Nachdruck hingewiesen. Die Wirkprinzipien von Informatiksystemen lassen sich nicht mit der weitverbreiteten Black–Box–Sichtweise erfassen.

Der neue Lehrplanentwurf für die Informatik in der Gymnasialen Oberstufe in Nordrhein–Westfalen eröffnet mit den Abschnitten „2.1 Bereiche: Herleitung und didaktische Funktion“ und „2.2 Zuordnung der Themen und Gegenstände zu den Bereichen“ eine didaktische Perspektive, die Modellieren und Konstruieren dem Analysieren und Bewerten gegenüberstellt. Diese selbstgewählten didaktische Ansprüche werden im Abschnitt „2.3 Obligatorik und Freiraum“ nicht überzeugend umgesetzt. Hier soll eine abweichende Sicht auf diese Bereiche, die mit Sicherheit eine zentrale Rolle in der informatischen Bildung spielen, vorgestellt werden.

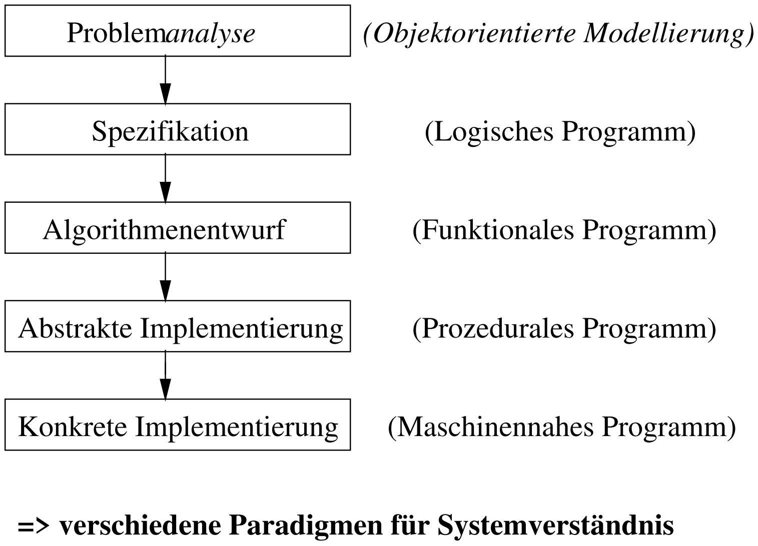


Abbildung 54: Ebenen der Programmentwicklung [Dosch93], S. 13

Modellieren und Konstruieren

Ziel: Informatikmodell / Informatiksystem

- Probleme eingrenzen und strukturieren
- Anforderungen an Informatikmodell
- reduzierte Systeme beschreiben
- Strategien modifizieren

- Daten abstrahieren
- Standardalgorithmen anwenden
- Implementation des Modells
- Evaluation des Systems

Methode:

vom Bauplan zum System

Informatikprinzipien müssen auf Vorrat erlernt werden, um eine Gestaltungsaufgabe sinnvoll zu bewältigen.

Analysieren und Bewerten

Ziel: Informatikprinzipien und –methoden

- Aufbau und Funktionsweise von Systemen
- Technikfolgenabschätzung und Gestaltungskriterien
- historische Rahmenbedingungen
- Begriff der Berechenbarkeit:
  berechenbar, entscheidbar, akzeptiert
- Begriff der Unentscheidbarkeit
- Modellierungsparadigmen
  (prozedural, funktional, prädikativ)
- Objektorientierte Denkweise / Problemlösung
- formale Sprachen und Grammatiken

Methode:

vom System zu Kriterien

Das Gestaltungsergebnis ermöglicht das entdeckende Lernen auf der Suche nach den Kriterien, die dem Entwicklungsprozess zugrunde lagen. Es wird nachvollziehbar, welche Informatikprinzipien die Entscheidungen beeinflussten.

Die Bedeutung des Modellierens und Konstruierens für den Informatikunterricht ist nicht neu und darf auch nicht verdrängt werden. Wichtig ist der ergänzende Zugang zu den Informatikprinzipien und –methoden mittels Systemanalyse, der den Informatikunterricht von dem Dauerstress des Problemlösens befreit, mit dem die Schüler in der Sekundarstufe II häufig überfordert wurden. In den Abituranforderungen [KMK91] wird das Entwickeln von Algorithmen dem Anforderungsbereich III zugeordnet, der nicht im Mittelpunkt stehen soll.

Den deutlichsten Bruch mit den existierenden Lehrmethoden stellten Peter Hubwieser und Manfred Broy 1996 vor [HB96]. Ausgehend von einer Kritik an der zu engen Sicht auf die Programmierung im Informatikunterricht schlagen sie vor, auf die Programmiersprachen zu verzichten und mit den Schüler Diagramme als Beschreibungsform für Modellierungen zu wählen. Sie bezeichneten diese Lehrmethode als „Informationszentrierten Ansatz“.

Informationszentrierter Ansatz
von Hubwieser, P. u. Broy, M.

Ziel:

- weg von Algorithmik
- hin zu fundamentalen Strukturierungstechniken für den Rohstoff „Information“

Inhalt:

- realisierungsunabhängige Modellierungstechniken
  z.B. formale Spezifikation
- Objektorientierung
  z.B. hierarchisch geordnete Klasse
- Beschreibungstechniken

  z.B. E/R–Diagramme (Entity–Relationship–Diagramme)
  → Semantik in Baumdiagrammen für statische Modelle (Systemaufbau)

  oder Zustandsübergangsdiagramme für dynamische Modelle (Verarbeitung)

! Begriff der Information in der Informatik
Begriff Daten

Tatsächlich existieren in der Informatik eine Reihe von graphischen Darstellungsformen mit strenger Syntax und Semantik, die sich eignen Lösungshypothesen zu präzisieren und zu formalisieren. Es fehlt die empirische Überprüfung der Eignung für den Informatikunterricht. Mit Sicherheit sind diese Repräsentationen nicht einfach zu erlernen und anzuwenden. Der Vorteil liegt in der Betonung universeller Modellierungstechniken gegenüber wechselnden Sprachen. Der Nachteil besteht in der mangelnden Überprüfung der modellierten Lösung mit Informatiksystemen, da die „experimentelle“ Basis zur Veranschaulichung von Grundkonzepten ohne Programm fehlt bzw. unzureichend ist.

Unterrichtsmethodik

- Problemgewinnung
  (aus der Praxis)
- informelle Problembeschreibung
  (verbal oder graphisch)
- formale Modellierung
  zur Informationsstrukturierung (Gruppenarbeit typisch)
- Realisierung von Lösungsansätzen
  zur Veranschaulichung der Modellierung (Programmierung einzelner Module ohne Programmiersprache)
- Bewertung

→ Veranschaulichung der Strukturen erforderlich
  – Datenbanksysteme
  – Code–Generatoren

? Verifikation der Modellierung

1997 veröffentlichten die beiden Autoren [HB97] eine Fortsetzung ihres Ansatzes, der

1. Informationsbeschaffung,
2. Informationsübertragung,
3. Informationsgewinnung,
4. Informationsspeicherung,
5. Informationsdarstellung,
6. Informationsbewertung,
7. Informationssicherung

als Bereiche des potentiellen Lehrstoffes vorstellt und untersetzt. Daraus werden Anwendungskategorien abgeleitet. Der Versuch, die Grundkonzepte der Informatik mit Informations– und Kommunikationssytemen (IuK–Systemen) zu lehren, wirkt sehr aktuell und attraktiv. Damit wird eine spezielle Art von Informatiksystemen betont und die informatische Bildung deutlich reduziert. Zu den Konsequenzen fehlt eine Diskussion.

Viele Vorschläge zur Gestaltung der Schulinformatik gehen davon aus, dass es ein Fach für wenige Interessenten sein soll, die es nur wählen, wenn es Spaß macht und dem aktuellen Trend der Fachwissenschaft dicht auf den Fersen bleibt. So entsteht der Eindruck der Beliebigkeit bei der Stoffauswahl und –darbietung. Die Lehrbarkeit wird in Frage gestellt. Wer kann und möchte schon im Kurstakt die Konzepte und Beispiele wechseln. Das Gegenstück dazu sind Forderungen nach der konsequenteren Abkopplung des Schulfaches von der Bezugswissenschaft Informatik. Sie findet zur Zeit dort statt, wo sie kaum wahrgenommen wird, in der Komplexität der Informatiksysteme, deren Transparenz nicht mehr gewährleistet ist. Die Nutzung professioneller

Sprachen in der Schule bringt das verlorene Systemverständnis nicht zurück, wirkt aber scheinbar sehr modern. Aber altmodische Lehrmethoden können nicht mit modernen Fachmethoden kompensiert werden.

Die Idee eines Informatikpraktikums für die handlungsorientierte Einführung der Grundkonzepte liegt nahe, erfordert aber vorerst konzeptionelle Studien und Versuchsbeispiele, bevor eine Erprobung in der Unterrichtspraxis beginnen kann. Die Dortmunder Didaktikgruppe verfolgt diesen Ansatz mit folgender Intention (Schubert, Steinkamp: Vortrag auf der MNU Jahrestagung in Saarbrücken 1999):

- Informatikunterricht fördert konstruktive Schülertätigkeiten, um unstrukturierte Anforderungen (Ausschnitte der realen Welt) in Informatiksysteme zu überführen. Dieser Prozess des informatischen Modellierens kostet sehr viel Lernzeit, da umfangreiche Vorkenntnisse beginnend vom Verstehen kleiner Bausteine bis zu deren Integration zu einem komplexen System erforderlich sind.
- In Analogie zum naturwissenschaftlichen und technischen Unterricht wird vermutet, dass eine Folge von Schülerexperimenten den Lernprozess unterstützen kann. Ziel ist das handlungsorientierte Aneignen von Grundkonzepten aus verschiedenen Teilgebieten der Informatik. An die experimentelle Komponente der Informatik mit dem „Denkzeug“ Computer werden hohe Erwartungen geknüpft. Logische Strukturen, die im Kopf des Entwicklers existieren, können in einer künstlichen Systemwelt auf ihre Wirksamkeit überprüft werden.
- Bisher wird das Programmieren im Kleinen erlernt, da die Tätigkeiten innerhalb einer Programmentwicklungsumgebung das einzige sind, was Schülern handlungsorientiert zugänglich gemacht werden konnte, nachdem der hardwareorientierte Ansatz des Informatikunterrichts, in dem Computer aus Hardware–Bausteinen zusammengebaut wurden, nicht mehr umgesetzt wird. Die Folge davon ist die Beschränkung auf einfache Algorithmen und Datenstrukturen in einem oder zwei Programmierparadigmen.
- Eine Erweiterung der Palette der zu erlernenden Konzepte erfordert deshalb schülergerechte Tätigkeitsfelder, die es ermöglichen:
  - Modellierungen auszuführen (auch ohne Programmiersprache),
  - Architekturen zusammenzusetzen (und zu zerlegen),
  - Datenübertragungen zu steuern (und zu beobachten).
- Wie in jedem Schülerpraktikum sind der Versuchsaufbau, seine Erkenntnisstruktur und die möglichen Fehler didaktisch aufzubereiten.
- In einer ersten Forschungsarbeit wird das Fundamentum der Informatik auf notwendige und geeignete Experimentierfelder untersucht und eine Aufwand–Lernerfolgs–Studie vorgenommen. Die Ergebnisse bilden das Konzept für ein Didaktik–Laborpraktikum für Lehramtsstudierende der Informatik, mit dem sich diese auf ihren künftigen Unterricht vorbereiten können. Die Einzelexperimente können für unterschiedliche Zielgruppen und Bildungsaufgaben zu alternativen Sequenzen verbunden werden.
- Die empirische Erprobung kann beginnen, wenn das erste Experiment als Informatiksystem vorliegt. Dazu wird eine Testvariante bereitgestellt und der potentielle Einsatzrahmen in Lehrkonzepten skizziert. Exemplarisch wird festgestellt, mit welchen Materialien die Versuchsanleitung, –durchführung und –auswertung unterstützt werden muss und wie hoch der zeitliche Übungsbedarf ist.
- Besonders kompliziert wird die Modellierung der künstlichen Experimentierumgebung eingeschätzt. Es besteht die Gefahr, die Komplexität von Prozessen mit störanfälligen Handlungsumgebungen nicht transparenter zu gestalten. Deshalb sind Erkenntnisse zum prinzipiellen Vorgehen bei der Entwicklung solcher Experimente besonders wichtig.

Ein interessanter Nebeneffekt dieser Überlegungen ist die damit verbundene Abkopplung der Informatiklehrmittel von den kommerziellen Systemen, die für Zielgruppen und Anwender außerhalb der Schule entwickelt wurden. Auch das Werkzeug für die Hand des Lehrers weicht dann deutlich vom Lern– und Arbeitsmittel des Schülers ab. Die Arbeit mit reduzierten Systemen als Unterrichtsmittel ist so gewollt, fördert die Transparenz und Modellbildung, folgt lehrmethodischen Zielen und wird nicht länger als Ausdruck eines finanziellen Mangels interpretiert. Schulspezifische Lösungen können sich durchaus als kostbarer erweisen.

9.4.4. Probleme

Die Diskussion um die Einführung in ein alternatives Programmierkonzept führt zu Meinungsverschiedenheiten darüber, ob dies besser vor oder nach der prozeduralen Programmierung erfolgen soll und welche Konsequenzen daraus erwachsen.
Unsicherheit besteht im Umgang mit der Theorie der Informatik. Das Spektrum reicht in den Lehrplänen vom 14–Stunden–Abschnitt bis zu einem Halbjahreskurs im letzten Schuljahr. Über die Breite und Tiefe der Behandlung einzelner Themen wird gestritten.
Die Informatik ändert ihre Forschungsziele und Denkgewohnheiten. Brauer formulierte [BB95]: „Computersystem als Gruppe gleichrangiger, selbständiger Akteure, die bestimmte Aufgaben erledigen und dazu miteinander und mit der Umgebung interagieren.“ Die Lern– und Anpassungsfähigkeit der Akteure spielt dann eine wesentliche Rolle. Eine aktuelle Zusammenfassung der Probleme sieht so aus:

1. überhöhte Erwartungen an die notwendigen Informatikkenntnisse vieler Berufsgruppen
2. Unklarheit über die „neue Kulturtechnik“ (Umgang mit dem Computer)
3. fehlende Motivation für die Unterrichtsinhalte bei attraktiver Software
4. Zweifel an zeitbeständigen Grundlagen und der Existenz von Bildungswert und Bildungskern (Algorithmen und prozedurale Programmiersprachen veraltert?)
5. keine Vorkenntnisse aus Sekundarstufe I
6. altersuntypische Aufgaben in Sekundarstufe II
7. Inhalte erwiesen sich als schwierig
   → Frage nach geeigneten Lehrmethoden
8. Sprachenstreit auf neuer Ebene um Paradigmen
9. Missverständnis zum Informatikprojekt
10. Theorie der Informatik:
    über Auswahl, Umfang und Anordnung der Themen wird gestritten
11. Forschungsziele der Informatik und Denkgewohnheiten verändern sich stark,

    Brauer:
    „Computersystem als Gruppe gleichrangiger, selbständiger Akteure, die bestimmte Aufgaben erledigen und dazu miteinander und mit der Umgebung interagieren.“
12. Zerrbild der Informatik

Das hat Konsequenzen für die Lehrpläne des Schulfaches Informatik. Ein Arbeitskreis der GI entwickelt deshalb zur Zeit ein Gesamtkonzept der informatischen Bildung, um solche Ziele des vernetzenden Denkens und Handeln (verteilt, interaktiv, nichtsequentiell) mit Inhalten und Lehrmethoden zu untersetzen.

10. Projektarbeit an Informatiksystemen

10.1. Mehrdeutigkeit des Projektbegriffs

Die Begründung der Projektarbeit im Informatikunterricht wurde ursprünglich aus der Fachwissenschaft abgeleitet, deren Prinzipien und Methoden, wie z. B. strukturierte Programmierung, Modularität, Phasen der Software–Entwicklung und Gestaltung einer Dokumentation für den Lernenden erst ab einer entsprechenden Komplexitätsstufe notwendig und in ihrer Bedeutung verständlich wurden. Mit dem Ablösen der strukturierten Programmierung durch das objektorientierte Modellieren (siehe Kapitel 6) und der Einführung von neuen didaktischen Konzepten für die informatische Bildung, die auf die Software–Entwicklung verzichten wollen, wird diese frühere Begründung neu zu überdenken sein. Der Projektbegriff wird in der Informatik anders interpretiert als in der Erziehungswissenschaft. Daraus resultieren viele Missverständnisse unter den Lehrern [Ambros92] zur Projektmethode als Unterrichtsform:

- längere, fächerübergreifende Unterrichtsform,
- Projektwochen erwecken den Eindruck: „Wir hatten keinen Unterricht“,
- schulische Erfahrungsmöglichkeiten erweitern,
- ohne Leistungs– und Notendruck,
- ganzheitliche und praktisch durchführbare Arbeitsvorhaben oder unterrichtliche Gesamtthemen.

Deshalb sollen hier beide Sichtweisen vorgestellt werden und auf ihre Wirksamkeit im Informatikunterricht untersucht werden. Die einseitige Betonung einer Sichtweise brachte nicht den gewünschten Aneignungsprozess bei den Lernenden zustande.

10.2. Projektlernen

1942 stellten C. Nelson und L. Bossing die Projekt–Methode als Unterrichtsmethode vor, die aus C. R. Richards Forderung von 1900 nach der selbständigen Schülerlösung im Werkunterricht hervorging. Der Begriff der Projekt–Methode wurde 1927 von W. S. Monroe umgewandelt zu einem neuen Lernweg, bei dem der Schüler das tun kann, was er möchte. Später setzte sich die ursprüngliche Auffassung durch. Zur Projekt–Methode gehören folgende Phasen (Stufen):

1. Zielsetzung,
2. Planung,
3. Ausführung,
4. Beurteilung.

Einzel– und Gruppenprojekte werden unterschieden. Der Lehrer soll die Schüler selbsttätig planen und arbeiten lassen, aber „doch die Fäden in der Hand behalten“[NB42], S. 137. Für die Projektwahl ist die Zustimmung der Schüler erforderlich, und das Projekt soll:

- einen Bildungswert besitzen,
- eine gute Zeitaufwand-Ertrag-Relation aufweisen,
- zugängliches Material erfordern,

- mit den Mitteln der Schule finanzierbar sein.

Es gilt als sehr kompliziert bei dieser Lehr–Lern–Methode den Unterrichtszusammenhang zu erhalten. H. Pütt stellte 1982 fest:

> „Der nach wie vor anhaltende Drang nach der Durchführung von Projektunterricht und sogenannten Projektwochen ist wohl damit zu erklären, dass Schüler und Lehrer gleichermaßen in dieser dem schülerorientierten Unterricht adäquatesten Form einen Weg aus der Routine suchen und damit Möglichkeiten zur Öffnung der Schule ergreifen.“ [Pütt82], S. 63.

Er gliedert seine Projektvorschläge in einem Verlaufdiagramm nach Projektphasen, Themenaspekt und Arbeits– und Handlungsformen. Dem Projektunterricht ordnet er sehr ausführlich Merkmale zu, die aber nicht alle erfüllt werden können [Pütt82], S. 101, wobei 4. und 8. nicht fehlen dürfen:

1. problemhaltige Aufgabe,
2. Motivation,
3. Zielorientierung,
4. planvolles, selbständiges und selbstbestimmtes Lernen,
5. Verbindung von schulischem und außerschulischem Tun,
6. Hingabe und ernstes Engagement,
7. individuelles und kooperatives Handeln,
8. Verbindung von Theorie und Praxis,
9. Ausdauer,
10. Abbau von Lehrerdominanz,
11. typische Verlaufsstruktur,
12. fächerübergreifenden Charakter,
13. Methodenvielfalt,
14. Abschluss und Aufgabenbeurteilung,
15. gesellschaftliche Relevanz.

Von den Sozialformen des Unterrichts aus betrachtet wird das Projektlernen zu Gruppenunterricht führen [Meyer88], S. 237–277.

> Es ist aber auch von „Projektwochen – Etikettenschwindel?“ auf S. 334–340 zu lesen. Diese sehr komplizierte Form des Projektlernens wird nicht selten missverstanden und verstärkt so die schulinternen Probleme. „Oft genug passiert es, dass innerlich ablehnend eingestellte Lehrer während der Projektwoche so gut wie nichts tun und sich dann auch noch hinterher über den dürftigen Erfolg der Projektwoche beschweren.“ , S. 338. Es wird unzureichend beachtet, dass Themenvorschläge von den Schülern und Lehrern gemeinsam so zu entwickeln sind, dass die Fragestellungen für beide Seiten neu sind. „Lehrer bieten ihre Hobbies als Arbeitsgruppenthemen an ... Die Projektwoche ist dann nichts anderes als Epochalunterricht im Zwangsverband.“, S. 339.

10.3. Informatikprojekt

> „Das Ziel der Software–Technologie ist, ökonomisch Software zu erstellen, die zuverlässig und effizient in der realen Umgebung arbeitet." [Kroha97], S. 19.

Mit Ingenieurmethoden (Prozessmodelle, Teamarbeit, CASE–Werkzeuge (CASE steht für computer aided software engineering)) werden Softwareprojekte entwickelt. Probleme bilden:

- die unterschiedliche Interessenlage und Sprachbarrieren zwischen Auftraggeber, Entwickler, Anwender,
- die Komplexitätsreduzierung realer Problemstellungen,
- die Qualitätssicherung und Gewinnoptimierung,
- Abweichungen vom Plankonzept, die verhindert werden müssen.

Die ursprüngliche, naive Vorstellung, dieser Prozess eigne sich für das Projektlernen im Gruppenunterricht allgemeinbildender Schulen musste aufgegeben werden. Software als Produkt hat aber spezielle Eigenschaften, die durchaus unterrichtsrelevante Auswirkungen in der Gesellschaft zeigen. Der Systementwicklungsprozess besteht aus Denkweisen und Arbeitsmethoden der Informatik, die zum Verständnis des Faches gehören, wie:

- Funktionsmodellierung durch Prozesse,
- Datenmodellierung durch Diagramme,
- objektorientierte Analyse, Spezifikation, Modularisierung,
- Systemarchitektur und verteilte Verarbeitung,
- Echtzeitsysteme,
- Inspektion, Testen, Verifikation,
- Modelle für den Entwicklungsprozess,
- Qualitätsmerkmale, Zertifikation und Rechtsschutz von Informatiksystemen.

Zum lehrmethodischen Zugang über Projektlernen liegen Erfahrungen und Hypothesen vor, die empirisch zu überprüfen sind.

10.4. Informatikdienstleistung für andere Fächer

Informatiklehrer leisten mit Anwendungssystemen viele Dienstleistungen in der Schule. Ein enttäuschter Lehrer formulierte: „Ich möchte nicht immer nur die Schülerzeitung in der Projektwoche gestalten." Die ITG wurde in Nordrhein–Westfalen an Projektlernen gebunden, um exemplarische Anwendungen der Informatiksysteme in verschiedenen Fächern vorzustellen (siehe Kapitel 9). Das ist sowohl im themengleichen als auch im themendifferenzierten Gruppenunterricht möglich. Hier soll auf Erfahrungen mit dem Projektlernen in der ITG näher eingegangen werden. In [AKHK+90], S. 58 werden Probleme analysiert, die mit der Einführung der ITG verbunden waren:

- Die Lehrer beherrschten die Informatiksysteme unzureichend.
- Die Lehrer beherrschten das projektorientierte Unterrichten unzureichend.

- Die Unterrichtseinheit „Zeitung“ verlor Zeit durch die langwierige Einführung in die Textverarbeitung.
- Bei der Unterrichtseinheit „Industrieroboter“ fehlten den Schülern die Lernvoraussetzungen.
- Der Unterrichtseinheit „Wärmeschutz“ fehlte die motivierende Wirkung auf die Schüler.
- Bei der Unterrichtseinheit „Warenhaus“ kam bei den Schülern Langeweile auf, als Waren– und Kundenkarten mühsam herzustellen waren.
- Die wichtigen Reflexionsphasen entfielen nicht selten, da den Lehrern deren Integration nicht gelang.

Die Schüler beurteilten den Gruppenunterricht sehr positiv, da ihnen die Handlungsorientierung gefiel. Sie hätten gerne noch mehr solche Projekte bearbeitet. Die Lehrer bewerteten die Lernvoraussetzungen der Schüler als unzureichend.

10.5. Systementwicklung im Informatikunterricht

10.5.1. Ziel

Ziel ist das Lösen komplexer Anwendungsprobleme mit folgenden Vorteilen:

- Die Integration von Wissen und Können aus verschiedenen Teilbereichen des Informatikunterrichts wird notwendig und möglich.
- Daraus erwächst die Motivation für die Vertiefung und Erweiterung des Gelernten.
- Traditionelle Unterrichtsabschnitte können mit den Projektphasen verbunden werden. Der voraussichtliche Erkenntnisgewinn wird im Projektrahmen kalkulierbar.
- Die Realitätsnähe der Schülertätigkeit fördert das Verständnis für die Auswirkungen von Informatikanwendungen. Dieser Vorteil schwindet mit dem Abstand zwischen den existierenden Anwendungssystemen und den Möglichkeiten der Systementwicklung mit einer Schülergruppe.
- Die kooperativen Arbeitsformen sind so informatiktypisch, dass sie im Rahmen einer wissenschaftspropädeutischen Ausbildung unverzichtbar erscheinen. Die Notwendigkeit und Realisierung von Schnittstellenbeschreibungen und der Integration von Teillösungen zu einem Gesamtsystem werden so am besten verstanden und erlebbar.
- Die komplexen Aufgaben sind meist anderen Fächern entnommen, so dass interdisziplinäres Problemlösen erforderlich und möglich wird. Man sprach von pädagogischer Doppelfunktion des Informatikunterrichts, in dem immer noch ein zweites Fach mit vertieft wurde.

10.5.2. Gestaltung und Kooperation

Themendifferenzierter Gruppenunterricht ist typisch. Die ungewohnten Spielregeln müssen von den Schülern geübt werden können (nach [Meyer88]):

- Eigenverantwortung und Verantwortung für die Gruppe,
- Diskussionskultur mit Zuhören und Argumentieren,

- Gruppenleitung,
- Konfliktbewältigung,
- Zeitplanung und –kontrolle,
- Dokumentation der Zwischen- und Endergebnisse,
- Präsentation und Verteidigung der Ergebnisse,
- höhere Beteiligung an Reflexion und Steuerung des Arbeits– und Lernprozesses.

Der Lehrer muss sich stark zurücknehmen und möglichst lange stiller Gruppengast sein können. Deshalb kann die Ergebniserwartung nicht ein hochkomplexes Informatiksystem sein. Diesen Irrtum gab es und gibt es noch immer. 1982 gaben H. Haas und D. Wildenberg ein Didaktikbuch der Schulinformatik heraus [HW82] mit dem Kapitel B „Projektarbeit im Informatikunterricht" (S. 81–169) mit sehr ausführlichen Hinweisen zu:

- Projekte der Software–Entwicklung,
- Teamarbeit,
- Planung, Durchführung und Kontrolle von Projekten,
- Projektarbeit mit Schülern,
- Lernerfolgsmessung bei Projektarbeiten.

wobei die Unterrichtsphasen unmittelbar aus der Software–Technik abgeleitet wurden (S. 113):

1. Projektauftrag (Wahl des Projektthemas) und Projektplanung,
2. Systemanalyse und funktionale Spezifikation,
3. Entwurfsspezifikation,
4. Modulprogrammierung und Modultest,
5. Integration des Gesamtsystems und Test,
6. Dokumentation.

Dazu empfahl E. Lehmann eine Zeitaufteilung [Lehmann85]:

- ca. 30 % der Zeit für:
  - Auswahl des Themas aus den Vorschlägen der Lernenden,
  - Analyse des Problems bis zur Präzisierung der Aufgabe,
  - Materialsammlung,
- ca. 45 % der Zeit für:
  - Bestimmung der Teilaufgaben, Zuordnung der Bearbeitungsgruppen, Schnittstellendefinition,
  - Entwerfen und Erproben der Lösungsteile,

- ca. 25 % der Zeit für:
  - Integration des Gesamtsystems,
  - Nutzung des Gesamtsystems, Bestimmung der Auswirkungen,

Dieser strenge Versuch fachliche Arbeitsmethoden auf Lernprozesse zu übertragen führte zu einem sehr einseitigem Unterrichtsmodell, das schwer realisierbar war und Frust bei den Lehrern auslöste, die damit scheiterten, da:

- das Thema so komplex sein muss, dass der Alleingang eines Schülers ausgeschlossen wird,
- die Motivation der Schüler schwer über den Zeitraum mehrerer Monate mit einer Systementwicklung erhalten werden kann,
- die Anforderungen an die Selbständigkeit der Schüler sehr hoch sind,
- Konzentrationsschwierigkeiten bei der Gruppenarbeit auftreten und intensive Hausarbeit erforderlich ist,
- die Lehrertätigkeit zur parallelen Betreuung komplexer Aufgaben ungewohnt und mit sehr hohem Vorbereitungs– und Betreuungsaufwand verbunden ist,
- die Integration der Teillösungen zu Funktionsstörungen führt,
- der Lehrer nicht mit einer Musterlösung auskommt, aber auch nicht alle möglichen Schülerlösungsvarianten vorhersehen kann,
- das Scheitern eines Projektes meist nicht als Lernfortschritt bewusstgemacht werden kann,
- die Freude am Unvorhergesehenem schnell an Grenzen stößt,
- keine Leistungsnivellierung stattfinden darf.

Der Lehrer soll die Projektaufgabe nicht vorher lösen, muss aber zu jedem Zeitpunkt den Weg der Fortsetzung erkennen. Das ist für komplexe Informatiksysteme von einem einzelnen nicht zu leisten. Der Lehrer kann aber den Ausgang eines Unterrichtsprojektes nicht beliebig offen halten, denn gerade das Ergebnis bringt Motivation und Stolz auf das Erreichte. Systemfragmente wirken aber eher enttäuschend. Einen Ausweg zeigte 1988 H. Löthe [Löthe88], indem er zwei typische Arbeitsweisen der Informatik mit folgender Charakteristik vorstellte:

- Arbeitsstil des ingenieurhaften Programmentwickelns:
  - Es findet ein Konkretisierungsprozess statt, bei dem von einer allgemeinen Beschreibung der Anforderungen über die Systemanalyse in einer Folge von Schritten bis zum auf dem Computer lauffähigen Produkt fortgeschritten wird.
  - Charakteristisch ist für diesen Vorgang, dass nur die letzte Version lauffähig ist.
  - Die vorhergehenden Versionen sind Pläne, bei denen sich der Programmautor auf einzelne Aspekte konzentriert.
  - Das Problem der Anfänger besteht darin, dass sie die Folge der Entscheidungen für den späteren Lauf auf dem Computer nicht einschätzen können.
  - Die Idealvorstellung besteht darin, dass die Korrektheit der Lösung nicht aus dem Programmtest, sondern aus dem Ablauf des Entwicklungsprozesses resultiert. Das Austesten soll nur „kleinere“ Irrtümer verbessern helfen.

- Arbeitsstil des forschenden Programmfindens:
  - Sachzusammenhang oder Lösungsverfahren mit dem Computer sind nicht von vornherein klar.
  - Begonnen wird mit der Realisierung einiger Ideen, und das entstehende Programmpaket wird durchgespielt und ausgetestet, um Erfahrungen zu sammeln.
  - Die Lösung wächst schrittweise. Als Methode des „strukturierten Wachstums“ wird das bezeichnet.
  - Jede Version des entstehenden Systems ist lauffähig. Der Einfluss der Zwischenergebnisse auf den Entwicklungsprozess wird gewünscht.
  - Charakteristisch ist das Experimentieren mit dem Ziel der theoretischen Systematisierung.
  - Jede Abstraktionsebene (Datenstrukturen und Operationen) baut auf der vorhergehenden auf.
  - Leitbild ist der experimentell forschende Entwickler, der die Klärung eines Sachverhaltes herbeiführt. Die Ergebnisse stehen nicht von vornherein unter dem Zwang einer festen Zielvorgabe.

Beide Arbeitsweisen wurden in der Projektphase des Informatikunterrichts mit Schülern angewendet. Die zweite Methode mildert den Produkt– und Erfolgsdruck, aber erst die Verknüpfung mit dem Projektlernen führte zu einem erfolgreichen Lernprozess. Die Planung der Projektarbeit und die Durchführung der Beratungen übernehmen die Lernenden selbst. Der Lehrer wird zum Berater der Projektgruppe. Er beobachtet und vermittelt im Konfliktfall. Die Projektleitung kann bei einem Lernenden oder beim Lehrer liegen. Sinnvoll ist folgende Staffelung der Anforderungen bei 13 Jahrgangsstufen:

- Analyse eines Informatik–Systems,
- Kurzprojekt ohne Themenwahl, um die Vorgehensweise kennenzulernen (1 – 2 Monate),
- umfangreicheres Projekt (mit Themenwahl) über ein viertel oder halbes Schuljahr.

Kriterien für die Gruppenbildung führen zu einem Kompromiss zwischen informellen Gruppen (Freundschaftsgruppen) und leistungsdifferenzierten Gruppen. Nicht vergessen werden darf die Projektreflexion, in der sich die Projektteilnehmer bewusst machen, was überhaupt abläuft:

- Methodenkritik,
- Diskussion von Schwierigkeiten,
- Aufarbeitung von Beziehungsproblemen.

Dadurch wird das Projekt erst zur Bildung und bleibt nicht Beschäftigung.

10.5.3. Lernerfolgskontrolle

Alle bekannten Formen der Lernerfolgskontrolle des Informatikunterrichts, wie Klausuren, Referate, Diskussionsbeiträge, Protokolle, Arbeit am Gerät, sollen einbezogen werden. Die Formen und Kriterien sind vor Projektbeginn zu erklären. Für Schüler kann es durchaus neu sein, dass komplexes soziales Verhalten gefördert wird, aber individuelle Leistungsbewertung stattfinden muss. Die Arbeitsdokumente der Schüler sind ebenfalls in die Lernerfolgskontrolle einzubeziehen. Die praktische Schülertätigkeit am Computer zeigt ein ebenso breites Spektrum, wie die geistig–planende. Einige entwickeln sehr kreativ eigene Arbeitsstrategien. Andere agieren hilflos, verstehen die Beispielstrategien nicht und scheitern an deren Anwendung. Die Klausuren gestalten sich besonders kompliziert, da der gemeinsame Lernbereich schwer zu ermitteln ist. Das Einfügen von traditionellen Unterrichtsabschnitten kann den Lernprozess auf die dafür erforderlichen Schwerpunkte fokussieren. Die Schüler teilen sich in Aufgabenbereiche, es gibt aber auch rotierende Tätigkeiten, die bewertet werden können, z. B.:

- Diskussion im Plenum leiten,
- Arbeitsprozess protokollieren,
- Fachthema im Referat vorstellen,
- Arbeitsergebnisse vorstellen,
- Arbeitsergebnisse bewerten.

10.5.4. Beispiele und Erfahrungsberichte

Die Sammlung der Beispiele müsste eigentlich sehr lang sein, aber viele gute Beispiele wurden in der Vergangenheit nicht über die Schulgrenzen hinaus bekannt. Das änderte sich mit der Telekommunikation und Telekooperation der Schulen. Die frühen Beispiele versuchten meist schulinterne Prozesse zu verbessern:

- Schülerbibliothek verwalten,
- Lehrmaterialbestellung,
- Kurswahl und Abiturbewertung,
- Berufswahl und Studienberatung,
- Buchungssystem für ein Reisebürounternehmen,
- U–Bahn–Auskunftssystem,
- Wahlauswertung.

Die Erfahrungsberichte sind warnend und ermutigend zugleich.

E. Modrow 1991 und 1992
Er löste sich in seinem Didaktikbuch vollständig von dem Software–Technik–Modell und kommt zu sehr positiver Beurteilung der „Unterrichtsprojekte“ [Modrow92]. Zitate:

- S. 143 „Im Anfangsunterricht besteht die Gefahr, dass überwiegend so eng gefasste Themen behandelt werden, dass der Überblick über den Sinn des Ganzen verloren geht. ... Neben dem Einsatz von Leitthemen sollte es deshalb schon im Anfangsunterricht Phasen projektartigen Arbeitens geben, in denen die Kursteilnehmer/innen die Auswahl des Themas und die Art und den Umfang der bearbeiteten Fragestellungen zumindest mitbestimmen.“
- S. 143–150 fünf Beispiele, davon drei unterteilt in Teilaufgaben
  - Komplexe Zahlen und Juliamengen,
  - Geld und Banken,
  - Überprüfung der Kurswahlen,

– Effizienz von Algorithmen.

Wolfgang Ambros 1992
Er wählte nicht zufällig den Titel „Das Informatikprojekt — oder: Die Angst vor dem Chaos“ [Ambros92] für seinen Erfahrungsbericht und schildert den Konflikt, in dem sich der Lehrer befindet, der die Richtlinien erfüllen möchte, aber es nicht kann, da er mit überzogenen Forderungen konfrontiert wird. Zitate:

- S. 189 „In 13 mußt du ein Projekt machen. ... Keiner erfährt davon, ob du ein Projekt machst oder nicht.“
- S. 193 „Die Diskrepanz zwischen realen Projekten und schulischen Möglichkeiten kann nur zu verzerrenden Überzeugungen führen.“
- S. 195 „Softwaretechnologie ... verbreitet zunehmend das Gefühl des Ausgeliefertseins an fremde, undurchschaubare und unkontrollierbare Mächte ...“
- S. 198 „Eine der Stärken imperativer Sprachen liegt gerade darin, dass man überprüfen kann, ob der Computer den Befehlen gehorchen kann.“

Rüdiger Baumann 1996
Er geht in seinem Didaktiklehrbuch auf „Software–Projekte im Informatikunterricht“ [Baumann96] ein. Zitate:

- Widerspruch: S. 191 „In der Schulrealität kommt diese Unterrichtsform praktisch nicht vor, und sie lässt sich durch Vorschriften allein auch nicht durchsetzen. ... Kein Schulfach eignet sich so gut für diese Unterrichtsform wie die Informatik; sie ist für den Informatikunterricht kennzeichnend und typisch.“
- S. 192 „Jedes Team-Mitglied stellt dem gesamten Kurs über seine teamspezifischen Aufgaben hinaus eine oder mehrere Dienstleistungen zur Verfügung. ... Neben diesen Aufgaben fungiert jeder Kursteilnehmer bei den Plenumsitzungen reihum als Sitzungsleiter oder als Protokollführer.“

Anhang

A. Lernziele

Kognitive Lernziele – Was man weiß
Der kognitive Bereich umfasst das Erinnern, die Erkenntnis von Wissen und die Entwicklung intellektueller Fähigkeiten und Fertigkeiten.
Die Bloom'sche Taxonomie [Bloom76] nennt:

1. **Wissen** (im Sinne von Kennen):
   - Reproduzieren des Gelernten,
   - Kennen von Verallgemeinerungen, Einzelheiten, Methoden, Prozessen, Mustern, Strukturen, Definitionen.
2. **Verstehen**:
   - einfachste Form des Begreifens,
   - Erfassen vorgegebener Informationen und Benutzen in geringem Umfang,
   - Reorganisieren des Gelernten (Inhalt mit eigenen Worten wiedergeben können).
3. **Anwenden**:
   - Abstraktion verstandener Begriffe (allgemeine Erkenntnisse),
   - Übertragung auf neue Probleme (konkrete Situationen).
4. **Analysieren**:
   - Zerlegen von komplexen Zusammenhängen in ihre Bausteine,
   - Identifizieren und Analysieren von Beziehungen zwischen ihnen,
   - logisches Schließen und Verknüpfen von Fakten und Hypothesen,
   - Bestandteil des Problemlösens.
5. **Synthetisieren**:
   - kreatives Zusammenfügen bekannter Elemente zu Neuem,
   - Entdecken bisher nicht bekannter Gesetzmäßigkeiten,
   - Bestandteil des Problemlösens.
6. **Bewerten**:
   - Voraussetzung für eigene Entscheidungen,
   - Auswerten und Bewerten von Lösungen, Methoden, Ideen zu einem bestimmten Zweck, wobei der Bewertungsmaßstab vorgegeben oder von Schülern selbst festgelegt wird.

Affektive Lernziele – Was man will
Der affektive Bereich umfasst die Interessen, Einstellungen und Wertungen. Die Bloom'sche Taxonomie [KBM78] nennt:

1. **Aufmerksamwerden, Aufnehmen, Beachten**:
   Sensibilisieren für bestimmte Phänomene oder Reize, wobei man zwischen der bloßen „Zur-Kenntnisnahme", der Aufnahmebereitschaft und der gerichteten, gegen Störungen unempfindliche Aufmerksamkeit unterscheiden kann.

2. **Reagieren**:
   Bereitschaft des Lernenden, seine Aufmerksamkeit aktiv (durch „Mittun“) auf etwas zu lenken.

3. **Werten**:

   - Zuordnen von Werten an und Einschätzung von Reizen, Phänomenen oder Objekten durch den Lernenden,
   - Gewinnen von Haltungen und Einstellungen.

4. **Entwickeln von Werte–Strukturen**:

   - Erkennen des Konfliktes zwischen mehreren Werten,
   - Herstellen von Beziehungen zwischen Werten,
   - Überprüfen auf Konsistenz (Beständigkeit),
   - Strukturieren der Werte (z.B. in einer Hierarchie).

5. **Werte–Verinnerlichung**:

   - Aufnehmen von Überzeugungen und Ideen in die eigene Weltanschauung (Philosophie),
   - festes Verankern von Werte–Strukturen durch den Lernenden, dessen Verhalten dadurch weitgehend bestimmt wird,
   - Weiterentwickeln der eigenen Weltanschauung.

Koerber und Peters beziehen sich auch auf Bloom [KP95], gehen aber auf ethische Fragen ein:

- Frage nach dem höchsten „Gut“,
- Frage nach dem „richtigen Handeln“:
  Handle so, dass die Maxime deines Willens jederzeit zugleich als Prinzip einer allgemeinen Gesetzgebung gelten kann (kategorischer Imperatriv).
- Frage nach der Freiheit des Willens.

Psychomotorische Lernziele – Was man kann

1. **Imitation**:
   Nachahmen einer beobachteten Handlung.

2. **Manipulation**:
   Ausführen bestimmter Handlungen nach Anweisung.

3. **Präzision**:
   Verbessern der Genauigkeit von Handlungen, die nun losgelöst vom Vorbild selbst kontrolliert werden.

4. **Strukturierung**:
   Gliedern einer Handlung in Einzelhandlungen und koordiniertes Ausführen.

5. **Naturalisierung**:
   Weitgehendes Verlagern des Bewegungsablaufs in das Unterbewusstsein (die Handlung ist in Fleisch und Blut übergegangen).

B. Unterrichtsvorbereitung

B.1. Modellvorstellungen

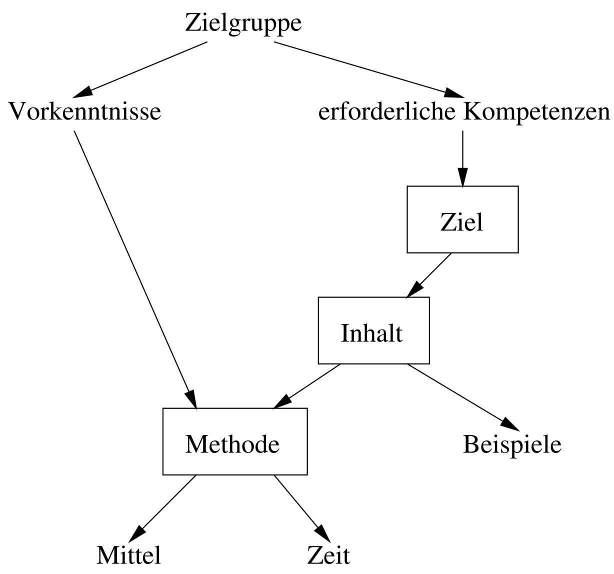
B.1.1. Schnittstelle zur allgemeinen Didaktik

Klafki, W.: Neue Studien zur Bildungstheorie und Didaktik, S. 266 ff

zur Unterrichtsplanung

- Lehrergruppen (nach Möglichkeit unter Beteiligung von Schülern)
- Lehrgangssequenz als Grundeinheit
- Ausgangssituation: landesspezifische Lehrpläne
- offener Entwurf
  - für Lehrer didaktisch begründbares, flexibles Handeln
  - für Schüler produktive Lernprozesse
- Dimensionen des Unterrichts und deren Beziehungen
  - Begründungszusammenhang
  - thematische Struktur
  - Zugangs– und Darstellungsmöglichkeiten
  - methodische Strukturierung (Lehr-Lern-Prozessstruktur)

B.1.2. Ausschnitt aus dem Bedingungsgefüge



Empfehlungen

- Ziel–Inhalt–Methode–Relation
- sorgfältige Analyse der Zielgruppe
- Abbildung von Kompetenzen (sehr allgemein) auf Ziele und Teilziele (sehr konkret)
- tiefes Verständnis für die Sachlogik des Stoffes
- Suche nach interessanten Beispielen und Geschick bei deren Aufbereitung
- empirische Erfahrungen zur Lehrmethode (besonders zur Zeit)
- genaue Kenntnis der Unterrichtsmittel
  - Komplexität von Informatiksystemen (Einarbeitungsaufwand)
  - eingeschränkte Auswahl (Finanzbedarf)

B.1.3. Beschreibung von Kursbeispielen

- Unterrichtseinheit
  - Didaktische Einordnung
    - ▷ Vorbemerkungen
    - ▷ Lernziele der Reihe
    - ▷ Unterrichtliche Voraussetzungen
    - ▷ Mögliche Fortsetzungen
  - Unterrichtsverlauf (Tabellenform möglich)

Zeit	**Ziel**	**Inhalt**	**Methode**	Mittel
			Lehrervortrag (LV)	Lehrbuch (LB)
			Schülervortrag (SV)	Computer (CO)
			Unterrichtsgespräch (UG)	Arbeitsblatt (AB)
			Einzelarbeit (EA)	Schema (Modellierung)
			Gruppenarbeit (GA)	Entwurf (Dynamik)
			Partnerarbeit (PA)	Programmtext (Statik)
			Hausaufgabe (HA)	Tafelbild (TB)
			Demonstrationsexperiment (DE)	Folie (FO)
			Schülerexperiment (SE)	Computerprojektion (CP)

- Lernbereich
- Beschreibung des Kursverlaufs

zentrale Frage

Fachsystematik	vs.	Schulfachsystematik
Ziel: Berufsbild		**Ziel:** Beitrag zur Allgemeinbildung (z.B. Studierfähigkeit)

beide **verbindet** eine gewisse Sachlogik:

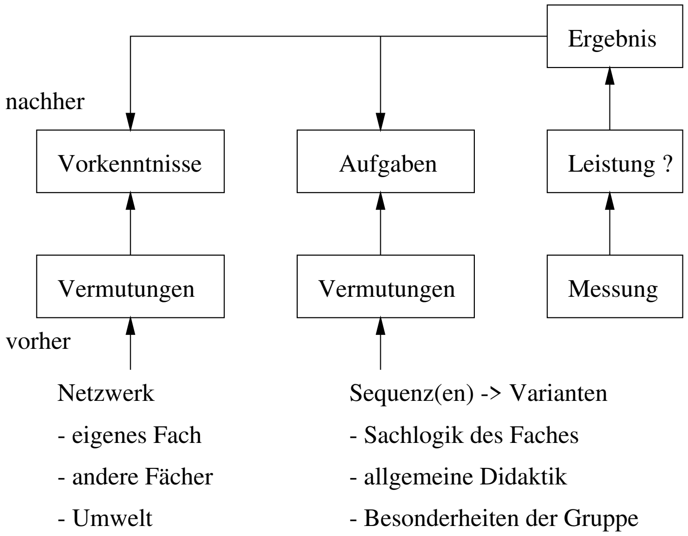
- vom Einfachen zum Komplizierten

- Vorkenntnisse / Kenntnisse (analog Fähigkeiten und Fertigkeiten) bauen aufeinander auf
- Prinzip der Wissenschaftlichkeit

beide **trennt**:

- das kognitive Niveau (z.B. Abstraktionsgrad)
- die Lehrmethodik (z.B. Spiralcurriculum)
- die Kompetenzanforderungen (z.B. Breite statt Tiefe in einem Teilgebiet)

Wir wissen nach dem Unterricht recht genau, was den Lernenden fehlte.



Es gibt immer interessante Varianten der Unterrichtssequenz, zwischen denen man aber nicht ungestraft wechseln kann. Wir müssen uns entscheiden! Die gewählte Sequenz wird nie allen Lernenden gleich gut gerecht.

Unterrichtsentwurf

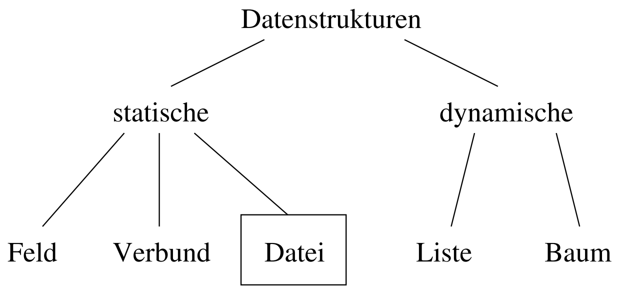
1. Begründung von Vorkenntnissen, Aufgaben, Leistungen für die konkrete Zielgruppe
2. Sequenz in Tabellenform
   - Ansatzpunkte für Alternativen
   - Problembereiche markieren
   - Differenzierungsmöglichkeiten
3. Lehrbücher und Informatiksysteme
   Beispiele, Hausaufgaben, Experimente

B.2. Einarbeiten in ein traditionelles Thema

z.B. Datenstrukturen

1. Begründungszusammenhang
   - Informatik als Strukturwissenschaft
   - Algorithmen und Datenstrukturen als Basiskonzepte
   - Richtlinien des Lehrplans für das imperative (prozedurale) Konzept
2. thematische Struktur (fachlicher Zusammenhang)
   - konkrete Datentypen
     - Teilmenge der ganzen Zahlen
     - Teilmenge der reellen Zahlen
     - Zeichenketten
     - Wahrheitswerte
   - abstrakte Datentypen
   - Datenstrukturen

B.2.1. Zugangs– und Darstellungsmöglichkeiten



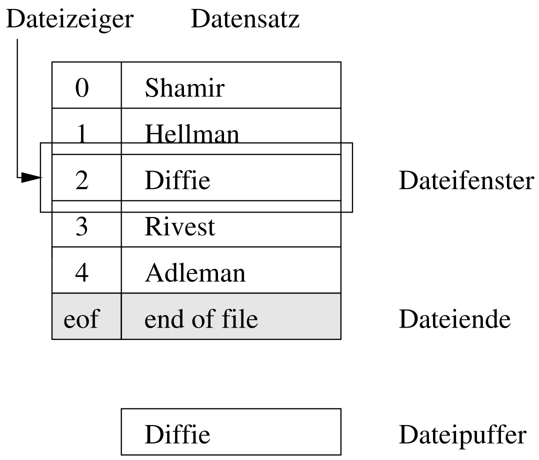
Vorteile:

- Feld:
  viele Daten unter einem Namen (V_1)
- Verbund:
  V_1+ Daten unterschiedlichen Typs (V_2)
- Datei:
  $V_1 + V_2$+ dauerhafte Speicherung (V_3)

Nachteile:

- Feld:
  nur Daten gleichen Typs
- Verbund:
  flüchtige Speicherung

Modellvorstellungen zur Datei



Blickfenster

- gehört zur Datei
- kann positioniert werden
- rückt bei jedem Lesen oder Schreiben automatisch eine Position (einen Datensatz) weiter
- gelesen werden kann nur, was im Blickfenster steht, bzw. geschrieben werden kann nur ins Blickfenster
- alle Ein– und Ausgaben werden über einen Puffer geleitet, der genau die Größe eines Datensatzes haben muß
- keine Warnung beim Überschreiben von Daten
- Fehler beim Lesen über die "eof"–Marke hinaus

Grundoperationen:

- Schlüsselwörter:
  Öffnen und Schließen vor / nach Schreiben und Lesen
- Unterprogramme:
  Erzeugen, Aktualisieren, Sortieren, Suchen, Ausgeben

  können in Anwendersystemen auch Schlüsselwörter sein

Eine Datei besteht aus einer Folge von gleichartig aufgebauten Datensätzen.

algorithmische Konstruktur: bedingter Zyklus

wiederhole, solange <Bedingung>
    <Anweisungen>

wiederhole
    <Anweisungen>
bis <Bedingung>

typische Bedingung für den Abbruch ist "eof"

Ziel: Erweitern des Dateimodells (Datei sortieren)

Zeit	Teilziel	Inhalt	Methode / Mittel
7:45	Wiederholung	Datei anlegen (Schreiben)	DE + SV / CP1 + B1
		Datei anzeigen (Lesen)	
7:55	Motivation	Datei sortieren (Lesen + Schreiben)	UG / CP2+B1
8:00	Arbeit am	Erweitern des Dateimodells	LV / TB1
	neuen Stoff	Begründung der beiden Puffer	UG/TB2
		Entwurf des Unterprogramms	UG / AB1 + B2
8:30	Übung	Implementation und Erprobung	SE/ CO+B2
9:00	Zusammenfassung	Lösungsvarianten vorstellen	DE+SV/ CP3+B2

B.3. Beispiel für prädikative Modellierung

Vorbereitung einer Informatik–Doppelstunde

Praktikant:	N.N.
Universitätsbetreuer:	N.N.
Mentor:	N.N.
Ort:	Gesamtschule
Datum:	
Zeit:	8:35–9:20
Thema der ersten Stunde:	Einführung in die Arbeit mit dem Prolog–System
Zeit:	9:30–10:15
Thema der zweiten Stunde:	Einführung in das praktische Arbeiten mit dem Prolog–System

B.3.1. 1. Stunde

1. Getroffene Entscheidungen

1.1 Thematischer Zusammenhang

a.) Reihenthema

Einführung der Programmiersprache Prolog, als Vertreter für das deklarative Sprachparadigma

b.) Stundenthema

Einführung in die Arbeit mit dem Prolog–System

c.) Einordnung der Stunde

- **Reihe:** Organisation und Durchführung gemeinsamer Arbeit — Gemeinsam arbeiten im Internet
- **Einzelstunde: Einführung in die Arbeit mit dem Prolog–System**

1.2 Ziele der ersten Stunde

a.) Stundenlernziel

Die Schüler[1] sollen die Prolog–Wissensbasis anhand einer Ahnentafel kennenlernen und in der Lage sein, eine eigene Wissensbasis aufzustellen.

b.) Teillernziele

Die Schüler

- erkennen die logische Struktur der Ahnentafel und bringen dies so in Verbindung zum deklarativen Sprachparadigma,
- nennen mögliche Problemstellungen,
- benennen Eigenschaften aus der Ahnentafel,
- lernen die Prolognotation dieser Eigenschaften kennen,

[1] im Folgenden mit S. abgekürzt

- kennen die Begriffe Prädikat, Objekt, Funktor, Argument, Fakt,
- nennen (umgangssprachlich) mögliche Anfragen,
- lernen die Prolog–Notation einer Existenz–Anfrage kennen,
- wissen, dass nur Variablen mit einem großen Buchstaben beginnen,
- sind in der Lage, das Prolog–System zu bedienen (elementare Funktionalität), d.h. sie können
  - eine neue Wissensbasis erstellen,
  - diese editieren,
  - speichern,
  - aktivieren und
  - einfache Anfragen an das Prolog–System stellen,
- editieren die Ahnentafel von Folie 1,
- stellen erste Anfragen an das System.

1.3 Geplanter Unterrichtsverlauf

Zeit	Phasen	Unterichtsinhalte	Unterrichts-formen	Medien
8:35	Eingangsimpuls / Erarbeiten einer Problemstellung	Die Folie mit Ahnentafel in Diagrammform wird an die Wand projiziert. Der Zusammenhang der Ahnentafel mit dem in der letzten Stunde angesprochenen Prädikativen Paradigma wird hergestellt. Die S. nennen mögliche Problemstellungen, die an der Tafel festgehalten werden. Wenn die S. keine Problemstellung nennen, die sowohl eine Beziehung als auch eine Eigenschaft erfordern, so werde ich versuchen, das Gespräch auf eine „ist Schwester von“–Problemstellung zu lenken, oder diese als LV hinzufügen.	UG / SB LV	OHP, Folie 1 Leere Folie
8:40	Erarbeitung I / Sicherung I	L.: „Was benötigen wir dazu?“ Die Eigenschaften (und Beziehungen), die aus der Ahnentafel zu entnehmen sind, werden umgangssprachlich an der Tafel festgehalten. Die Notation der Eigenschaften in Prolog wird durch den L. ergänzt. Die Begrifflichkeit wird erläutert (Prädikat, Objekt, Funktor, Argument, Fakt). Besonderer Hinweis auf den Punkt am Ende jeder Zeile und auf die Kleinschreibung des ersten Buchstabens der Prädikate und Objekte.	UG / SB LV LV	Tafel OHP, Folie 2

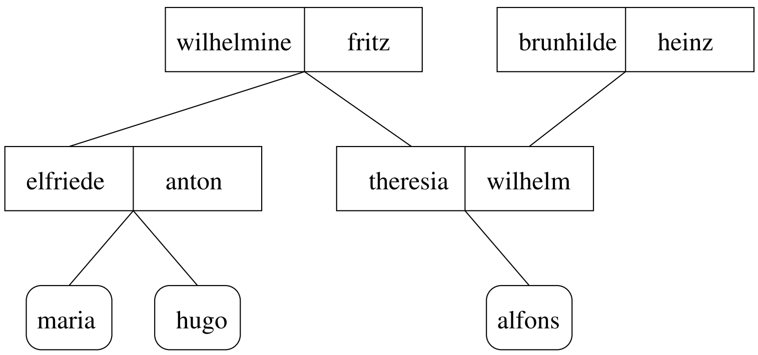
8:55	Erarbeitung II	L.: „Welche Fragen könnte man jetzt an das System stellen, so dass es in der Lage ist, sie zu beantworten?“ Die S. nennen Fragen, die an der Tafel festgehalten werden. Dann werden sie durch die Prolog–Notation ergänzt. Dabei eird der Begriff Variable erklärt und auf die Schreibweise hingewiesen.	SB	Tafel OHP, Folie 2
9:00	Einschub	Bevor sich die S. an die Rechner setzen, gebe ich die Hausaufgaben bekannt.	LV	Tafel
	Demonstration	Zur Nutzung des Systems zeige ich am Lehrerrechner, wie zu verfahren ist (editieren, speichern, aktivieren, Anfragen)	LV	Folie 3, CO, CP
9:05	Sicherung II	Die S. sollen die Ahnentafel in die Prolog–Notation übertragen (und so die Datenbasis in das Prolog–System eingeben) und speichern. Weiterhin sollen sie erste Anfragen an das System stellen.	EA, PA	CO

Legende zum Unterrichtsverlauf:

AB:	Arbeitsblatt	CO:	Computer	CP:	Computerprojetkion
EA:	Einzelarbeit	FO:	Folie	GA:	Gruppenarbeit
LV:	Lehrervortrag	OHP:	Overheadprojektor	SE:	Schülerexperiment
SV:	Schülervortrag	TB:	Tafelbild	UG:	Unterrichtsgespräch

1.4 Folien

Folie 1:



Folie 2:

weiblich(wilhelmine).
weiblich(brunhilde).
weiblich(elfriede).
weiblich(theresia).
weiblich(maria).

maennlich(fritz).
maennlich(heinz).
maennlich(anton).
maennlich(wilhelm).

maennlich(hugo).
maennlich(alfons).

Folie 3:

elternVon(wilhelmine,fritz,elfriede).
elternVon(wilhelmine,fritz,theresia).
elternVon(brunhilde,heinz,wilhelm).
elternVon(elfriede,anton,maria).
elternVon(elfriede,anton,hugo).
elternVon(theresia,wilhelm,alfons).

1.5 Geplantes Tafelbild

(die linke Tafelhälfte ist dabei zugeklappt)

	3. außen links	1. innen halb rechts	2. innen ganz rechts
	Abfrage: Umgangssprachlich: Wer ist weiblich ? Prolog-Notation: Weiblich(Wer). Variable	Datenbasis Umgangssprachlich Eigenschaften: Elfriede ist weiblich. ... Anton ist männlich. ...	Datenbasis Prolog-Notation weiblich(elfriede). ... maennlich(anton). ... Prädikat (Funktor) Objekt(e) (Argument(e)) Faktum

2 Begründung zentraler didaktischer Entscheidungen

2.1 Lerngruppe

Die Unterichtsstunde findet statt in einem Informatik–Grundkurs der Jahrgangstufen 11 und 12 mit insgesamt 27 S. Von diesen sind 10 aus der Jahrgangsstufe 12 (davon 4 Mädchen) und dementsprechen 17 aus der Jahrgangstufe 11 (davon 10 Mädchen).

2.2 Darstellung der Entscheidungen

Sowohl Grundwissen in Logik, als auch die Kenntnis von Verwandschaftsbeziehungen haben durchaus Alltagsrelevanz.
Die Ahnenafel bietet deshalb eine so gute Einstiegsmöglichkeit, weil Beziehungen, wie z.B. Vater, Mutter, Eltern, sehr leicht von den Schülerinnen und Schülern zu erblicken sind (Ergebnisse sind einfach nachzuvollziehen) und dennoch erreicht man mit Verwandtschaftsbeziehungen wie Großeltern, Schwager, Schwippschwager, Nichte, usw. eine Vielzahl von Komplexitätsgraden, bei denen die S. den Vorteil des Prolog–Systems einsehen. Die gleiche Ahnentafel ist den S. aus einer vorangegangenen Stunde zur Verknüpfung von html–Seiten bekannt. Es sollten html–Seiten erstellt und so verknüpft werden, dass die Strutur der Ahnentafel der html–Seite entspricht.

B.3.2. 2. Stunde

1. Getroffene Entscheidungen

1.1 Thematischer Zusammenhang

a.) Reihenthema

Einführung der Programmiersprache Prolog, als Vertreter für das deklarative Sprachparadigma

b.) Stundenthema

Einführung in das praktische Arbeiten mit dem Prolog–System

c.) Einordnung der Stunde

- **Reihe:** Organisation und Durchführung gemeinsamer Arbeit — Gemeinsam Arbeiten im Internet
- **Einzelstunde:** Theoretische Einführung in die Arbeit mit dem Prolog–System
- **Einzelstunde: Einführung in das praktische Arbeiten mit dem Prolog–System**

1.2 Ziele der zweiten Stunde

a.) Stundenlernziel

Die Schüler *erwerben Sicherheit im Umgang* mit dem Prolog–System.

b.) Teillernziele

Die Schüler

- kontrollieren die Ahnentafel des Mitschülers,
- nennen behandelte Anfrageformen,
- erkennen, dass die bisherige Modellierung der Ahnentafel nicht vollständig ist,
- erkennen, dass die Relationen zwischen den Personen noch nicht realisiert sind,
- wiederholen die Begriffe Prädikat, Objekt, Funktor, Argument, Fakt,
- *neue Anfrageformen*,
- ergänzen ihre Wissensbasis durch Relationen und editieren diese,
- notieren bereits in der letzten Stunde behandelte Anfragen und geben sie als Anfrage an das Prolog–System ein,
- notieren die Anfrage nach den weiblichen und männlichen Personen und nach den eigenen Eltern und geben diese als Anfrage ein,
- notieren ihre Vermutung der Ausgabe bzgl. einer vorgegebenen Anfrage und überprüfen diese durch Eingabe und Ausführung der Anfrage ins Prolog–System,
- deuten die Ausgabe des Prolog–Systems bzgl. einer vorgegebenen Anfrage mit Nutzung der anonymen Variablen,

- erkennen dabei, dass die anonyme Variable (Zeichen '_') eine Nichtbeachtung des Arguments, dessen Platz es einnimmt, bedeutet,
- notieren zwei auf dem Arbeitsblatt genannte Anfragen in der Prolog–Notation unter Verwendung der anonymen Variablen,
- notieren, unter Verwendung der anonymen Variablen, die Anfragen nach dem Namen aller Mütter,
- stellen diese dem System,
- dokumentieren die Ausgabe.

1.3 Hausaufgaben

1. Vervollständigen der am Anfang der letzen Stunde eingeführten Ahnentafel.
2. Angabe der Eigenschaften einer Vaterbeziehung.
3. Angabe der Eigenschaften einer Schwesterbeziehung.

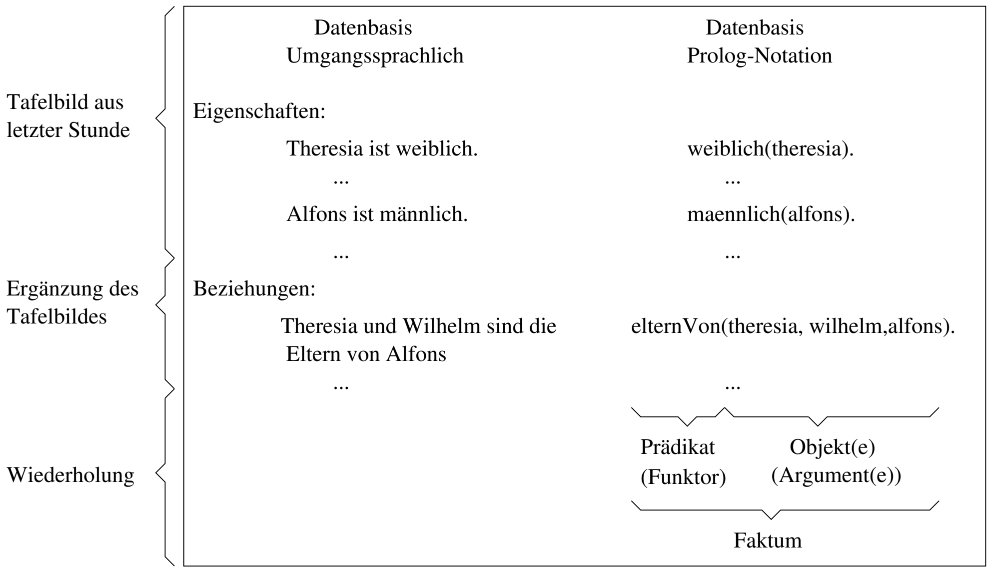
1.4 Geplanter Unterrichtsverlauf

Zeit	Phasen	Unterichtsinhalte	Unterrichts-formen	Medien
9:30	Vorbereitung	Die S. editieren ihre Ahnentafel am Computer.	EA	CO
9:35	Besprechung der Wissensbasis, Wiederholung	Ein S., der seine Wissensbasis komplett hat, erläutert diese. Anhand dieses Beispiels werden die in der letzten Stunde angesprochenen Anfrageformen wiederholt.	UG	CP
9:40	Modellierungserweiterung	Die Ahnentafel aus der vorangegangen Stunde und die dazu erstellte Wissensbasis werden an die Wand projiziert. Die Diskussion der letzen Stunde wird aufgegriffen und die Modellierung der Wissensbasis wird duch Beziehungen ergänzt. Die Notation dieser wird hinzugefügt und die Begriffe Prädiakt, Funktor, Objekt, Argument werden wiederholt. Eine neue Form der Anfrage (Funktor mit **drei** Argumenten) wird kurz demonstriert.	UG LV, UG LV	FO, OHP CP
9:50		Die Hausaufgabe zur nächsten Stunde wird gestellt		FO, OHP
9:55	Sicherung	Implementierung der begonnenen Modellierung mit praktischen Experimenten.	SE (EA / GA)	AB, CO

Legende zum Unterrichtsverlauf:

AB:	Arbeitsblatt	CO:	Computer	CP:	Computerprojektion
EA:	Einzelarbeit	FO:	Folie	GA:	Gruppenarbeit
LV:	Lehrervortrag	OHP:	Overheadprojektor	SE:	Schülerexperiment
SV:	Schülervortrag	TB:	Tafelbild	UG:	Unterrichtsgespräch

1.5 Möglicher Folienanschrieb



1.6 Arbeitsblatt

Arbeitsblatt
Implementierung der Modellierung am Beispiel der Ahnentafel

1. Ergänzen Sie die Wissensbasis zu ihren Ahnen um ihre Elternbeziehung und die Elternbeziehung Ihrer Eltern.

2. Stellen Sie dem Prolog–System drei Fragen, notieren Sie diese und vergleichen Sie alle Antworten mit Ihrem Diagramm:

   a) Frage nach weiblichen Personen: --------------------------------
   b) Frage nach männlichen Personen: --------------------------------
   c) Frage nach den Eltern Ihres Vaters: --------------------------------

3.

   a) Notieren Sie alle Antworten auf folgende Fragen:

      i. Halten Sie zunächst Ihre Vermutung fest und überprüfen Sie sie anschließend:

```
elternVon(<die eigene Mutter>, Vater, Kind).
```

      Vermutung: --

      Tatsächliche Antwort: --

      ii.

```
elternVon(Oma,_,<der eigene Vater>)
```

      Ausgabe: --

   b) Beschreiben Sie die Funktionalität des Zeichen „_ “

   ----

4. Notieren und überprüfen Sie zwei Anfragen, so dass einmal nur der Name Ihrer Mutter, und bei der zweiten Abfage nur der Name Ihres Vaters ausgegeben wird. (Dabei sollen Sie davon ausgehen, dass bei jeder Anfrage <u>nur Ihr Name bekannt</u> ist.)

   Anfrage nach Ihrer Mutter: ____

   Anfrage nach Ihrem Vater: ____

5. Untersuchen Sie die Anfrage nach den Namen aller Mütter und dokumentieren Sie Ihr Ergebnis.

   Anfrage: ____

   Dokumentation: ____
   ____
   ____

2 Begründungen zentraler didaktischer Entscheidungen

Die Schülerinnen und Schüler haben in der vorangegangenen Stunde Grundlegendes über eine aufzustellende Wissensbasis gelernt, wobei sich die Fakten auf solche mit nur einem Argument beschränkten. Anhand einer Ahnentafel wurden bereits Eigenschaften von Elementen der zu modellierenden Welt erarbeitet.

2.1 Lerngruppe

Die Unterrichtsstunde findet statt in einem Informatik–Grundkurs der Jahrgangsstufen 11 und 12 mit insgesamt 27 S. Von diesen sind 10 aus der Jahrgangsstufe 12 (davon 4 Mädchen) und dementsprechend 17 aus der Jahrgangstufe 11 (davon 10 Mädchen).

2.2 Darstellung der Entscheidungen

Ein erster großer Teil der Unterrichtstunde wird auf die Besprechung der Hausaufgabe verwendet, um zum einen das in der letzen Stunde behandelte zu wiederholen und zum anderen um die Ernsthaftigkeit der Hausaufgabe deutlich zu machen. Ein eher kleiner Teil der Zeit soll darauf verwendet werden, die Modellierung aus der letzen Stunde aufzugreifen und so die Wissensbasis durch die noch fehlenden Beziehungen zu vervollständigen und genannte Begriffe zu wiederholen. Den größten Teil der Unterrichtsstunde soll die praktische Arbeit am System in Anspruch nehmen. Es ist notwendig, die S. ausreichend zu motivieren, da in der letzten Stunde der theoretische Teil überwog und leider nicht genug Zeit blieb, um am Prolog–System zu arbeiten. Dies wird in dieser Stunde nachgeholt. Anhand eines Arbeitsblattes werden die S. durch praktische Arbeit geleitet. Es werden Grundarbeitstechniken geübt und die anonyme Variable wird von den S. selbst erarbeitet. Um die Ernsthaftigkeit der Hausaufgabe zu untersteichen, nimmt das Arbeitsblatt Bezug auf diese. Die S. sollen auf der Grundlage ihrer Ahnentafel den Zettel bearbeiten und Anfragen an das System stellen. Damit dies für jeden S. individuell möglich ist, kann sich das Arbeitsblatt nur auf die Elternbeziehung und auf die Elternbeziehung der Eltern stützen, da nur diese in allen Ahnentafeln der S. vorhanden sind.

3 Literatur

[Bratko87], [CM87], [GH91], [Schubert92], [SS86] und siehe Anhang G

4 Mögliche Probleme

- S. kennt seine Eltern oder seine Großeltern nicht
  => das Arbeitsblatt kann nicht vollständig bearbeitet werden.
- Unkonventionelle Verwandtschaftsverhältnisse der S., die beim Bearbeiten der Hausaufgabe zu Problemen führen (Stiefvater, ...)

C. Hospitation (Beispiel)

Fragenkatalog für die Hospitation

C.1. Lehrstoff

C.1.1. Übereinstimmung mit dem Lehrplan

Welche Unterrichtseinheit ist nach dem Lehrplan vorgeschrieben?
Hat der Lehrer einen Stoffverteilungsplan?
Welches Thema wurde in der vorhergehenden Stunde behandelt?
Welche Wochenstundenzahl ist für die Unterrichtseinheit vorgesehen?

C.1.2. Umfang und Auswahl des Stoffes

Sind Umfang und Auswahl des Stoffes auf die Klasse abgestimmmt?
Ist die Auswahl begründet?
Wird der Stoff medienwirksam ausgewählt und veranschaulicht (in der Vorbereitung)?
Werden bei der Stoffauswahl Schwerpunkte gebildet?

C.1.3. Fachliche Richtigkeit

Beherrscht der Lehrer den Lehrstoff?
Wird der Stoff fachlich richtig erarbeitet?
Werden Stoffstrukturen nach ihrer Bedeutsamkeit gebildet?
Welche Ursachen haben fachliche Fehler des Lehrers?

C.1.4. Didaktische Analyse

Welche Lernzeile werden formuliert?
Sind die Lernziele verständlich operationalisiert und taxonomiert?
Welche Lernzielebenen und Lernzielbereiche werden angesprochen?

C.1.5. Methodische Aufbereitung des Stoffes

Wie sollen die Lernziele im Verlauf der Unterrichtsstunde erreicht werden?
Welche Unterrichtsverfahren und Unterrichtsformen sollen angewendet werden?
Wie soll die Motivation erfolgen?
Welche Unterrichtsphasen sind vorgesehen?
Welcher zeitliche Ablauf ist geplant?
Wie sollen Lernkontrollen erfolgen?
Wird die Schülerselbsttätigkeit berücksichtigt?
Welche Medien sollen zum Einsatz kommen?
Welche Sozialformen werden geplant?

C.2. Unterrichtsablauf

C.2.1. Motivation

Wie weckt der Lehrer die Lernbereitschaft der Schüler?
Welche Problemstellung erfolgt durch den Lehrer?
Wie interessant gestaltet der Lehrer den Unterricht?
Wie beteiligen sich die Schüler am Unterricht?

C.2.2. Methodischer Aufbau

Sind die Unterrichtsmethoden schüler– und lernzielgerecht?
Erfolgen funktionsgerechte Wechsel der Unterrichtsmethoden?
Ist die Informationsfolge logisch aufgebaut?
Wird die Mitarbeit der Schüler aktiviert?

C.2.3. Wechsel der Unterrichtsform

Erfolgen Wechsel zwischen Frontalunterricht und anderen Unterrichtsformen?
Welche Unterrichtsformen werden praktiziert: Fragend-entwickelnder Unterricht, Gruppenarbeit, darbietender Unterricht, Fallbeispiele, Rollenspiel, exemplarischer Unterricht, deduktive oder induktive Methode u.a.?
Besteht zwischen der Unterrichtsform und dem Lernziel ein Bezug?

C.2.4. Schülermitarbeit und Schülerselbständigkeit

Haben die Schüler sich aktiv am Unterricht beteiligt?
Fand mit den Schülern ein Gespräch statt?
Wurde die Eigentätigkeit der Schüler angeregt?
Gab der Lehrer Impulse zur Mitarbeit?
Hat der Lehrer Schülerbeiträge aufgegriffen?

C.2.5. Medieneinsatz, Tafelanschrieb

Ist der Tafelanschrieb leserlich, sauber und fehlerfrei?
Werden die Versuche für alle sichtbar durchgeführt?
Ist der Medieneinsatz lernzielorientiert?
Werden die Medien sparsam eingesetzt?
Ist ihr Einsatz methodisch gut vorbereitet?
Wie ist das Verhältnis von Aufwand zu Erfolg?
Werden Anstöße zu eingenständigem Denken gegeben?

C.2.6. Zielstrebigkeit des Unterrichtsablaufs

Gestaltet der Lehrer den Unterricht ökonomisch?
Werden die Lernziele zügig angestrebt?
Werden die vorgesehenen Lernziele erreicht und gefestigt?
Redet der Lehrer nicht zuviel?

C.2.7. Erfolgssicherung, Lernkontrollen

Was tut der Lehrer zur Erfolgssicherung: Fragen stellen, Merksätze formulieren, Teil– bzw. Gesamtergebnisse festhalten, Lernabschnitte wiederholen, Übungsaufgaben stellen, das Gelernte anwenden lassen u.a.?
Gibt der Lehrer Aufgaben zur Nachbearbeitung des Gelernten?
Wie vergewissert sich der Lehrer, ob die Schüler die Lerninhalte verstanden haben und das Neue anwenden und übertragen können?
Führt der Lehrer eine Gesamtwiederholung des Lehrstoffes der Unterrichtsstunde durch?

C.2.8. Unterrichterfolg

Welcher Lernerfolg wurde erzielt?
Welche Lernstufe wurde erreicht: Reproduktion, Reorganisation, Transfer, Problemlösung?
Wie sicher ist der erreichte Unterrichtserfolg?
Welche Kenntnisse, Fähigkeiten und Fertigkeiten wurden beobachtet?
Welche effektiven Lernziele wurden angestrebt?

C.3. Lehrerpersönlichkeit

C.3.1. Auftreten, Unterrichtsstil

Welche Haltung zeigt der Lehrer?
Wirkt er freundlich, kontaktfreudig oder distanziert?
Ist der Unterricht lebendig und humorvoll?
Strahlt der Lehrer Sicherheit und Vertrauen aus?

C.3.2. Gesprächsführung, Mimik, Gestik

Ist die Sprache des Lehrers deutlich und klar?
Spricht der Lehrer zu viel, zu schnell, für die Schüler verständich?
Werden Mimik und Gestik sparsam eingesetzt?
Wechselt er die Lautstärke und die Betonung?
Spricht er Dialekt?

C.3.3. Kreativität, Flexiblität

Geht der Lehrer auf die Schüler ein?
Zeigt der Lehrer Engagement und Einfallsreichtum?
Wie reagiert er bei auftretenden Schwierigkeiten?
Kann er neue Ziele setzen, und erreicht er diese Ziele?

C.3.4. Erzieherische Einflussnahme

Fördert der Lehrer die Interaktion zwischen den Schülern?
Wirkt der Lehrer autoritär?
Wie verhält er sich bei Disziplinschwierigkeiten?
Welchen Einfluss nimmt er auf das Verhalten der Schüler?
Wie hilft er den schwachen und unsicheren Schülern?
Wie stärkt er das Selbstvertrauen der Schüler?
Wie reagiert er auf Kritik?

C.3.5. Kontakte Lehrer — Schüler

Wirkt der Lehrer kontaktfreudig?
Berücksichtigt der Lehrer die individuellen Anlagen der Schüler?
Wie fördert der Lehrer Interesse und Konzentration der Schüler?
Wie verhalten sich die Schüler dem Lehrer gegenüber?
Dominiert der Lehrer, oder fördert er die Eigeninitiative der Schüler?

C.3.6. Soziales Verhalten

Ist das Unterrichtsklima freundlich und entspannt?
Wie verhalten sich die Schüler zueinander?
Welche Umgangssprache pflegen die Schüler untereinander?
Gibt es Cliquen innerhalb der Klasse?
Beherrschen einzelne Schüler die Klassen?
Wie verstärkt der Lehrer das Gemeinschaftsgefühl?

C.4. Situative Faktoren

Ist der Klassenraum zu klein oder zu gross?
Wie wirkt die Akustik im Klassenzimmer?
Wirkt der Raum freundlich oder kalt?
Ist die Tafel zu klein, schlecht beschreibbar, nicht verstellbar?
Welche Unterrichtsstunde wird besucht (am Anfang oder am Ende des Unterrichts)?
Wurde zuvor eine Arbeit geschrieben?
Welchen Unterricht hatte die Klasse zuvor?
Ist das Begabungs– oder Vorbildungsniveau unterschiedlich?

C.5. Beispiel für prädikative Modellierung

Hospitation einer Informatik–Doppelstunde

Praktikant:	N.N.
Universitätsbetreuer:	N.N.
Mentor:	N.N.
Ort:	Gesamtschule
Datum:	
Zeit:	8:35–9:20
Thema der ersten Stunde:	Einführung in die Arbeit mit dem Prolog–System
Zeit:	9:30–10:15
Thema der zweiten Stunde:	Einführung in das praktische Arbeiten mit dem Prolog–System

C.5.1. 1.Stunde

<table>
<tr><th>Zeit</th><th>Unterrichtsverlauf</th></tr>
<tr><td>8:35</td><td>UG:
Problemsituation Struktur des Familienstammbaums, Schüler nehmen Gespräch an;
Analogie zum Hypertextprinzip, Wiederholung knüpft an die vorhergehende Stunde an:
Einrichten von OpenProlog
Programm zum Lösen von Problemen, Programmiersprache
logische Probleme, prädikatives Paradigma</td></tr>
<tr><td>8:40</td><td>Arbeit am neuen Stoff (LV): Beziehungen (Relationen), Lehrer schreibt auf Folie; Schreibweise spezieller Relationen
Frage nach dem Wissen zur Beantwortung (UG): „Wer ist mit Alfons verwandt? Wer sind die Eltern von ...? Wer ist der Vater von Alfons? ...“
Schüler: „Der Strich der zu den Eltern führt!“</td></tr>
<tr><td>8:45</td><td>Verallgemeinerung: Tafelbild eröffnet, Wissensbasis umgangssprachlich vs. Wissensbasis in Prolog–Notation
„Verbindung haben“ (meint die Eltern–Beziehung)
Begriff Objekt durch den Lehrer eingeführt</td></tr>
</table>

Zeit	Unterrichtsverlauf
8:49	• Darstellung von Eigenschaften: Lehrer „alle Namen dem Prolog–System mit Geschlecht bekannt machen“ • Prologschreibweise • Einführung neuer Begriffe: Prädikat (Funktor), Objekt (Argument), Fakt(um); LV, Tafelbild, Schüler schreiben mit
8:54	• Folie mit Hinweisen zur Syntax von Prolog mit dem Ziel der Editierhilfe für das Beispiel, Schüler: Übertrag ins Heft
8:56	• neuer Stoff Anfragen: UG, TB, Ja–Nein–Antworten • Hinweis auf eine besondere Taste am Mac–System • Fehler 1: nicht auf Funktion des „?“ eingegangen (Systembesonderheit „?“ nicht als Promptzeichen)
9:00	• neuer Stoff logische Variable (ist flüchtig): Lehrerhinweis auf Großschreibweise, Schüler schreiben mit • Fehler 2: muss gründlicher erarbeitet werden
9:02	• Hinweise zu Anfragen/Variablen (Folie); Schüler schreiben mit
9:06	• Nutzung des Prolog–Systems (Folie); CP wird vorbereitet
9:11	• Nutzung des Systems (consult): Lehrerdemonstration, CP; Beispiele für Existenz und Allabfragen
9:15	• Wissensbasis implementieren, SE

C.5.2. 2.Stunde

Zeit	Unterrichtsverlauf
9:30	• Wissensbasis implementieren, SE
9:36	• Wissensbasis vergleichen: DE Schüler, CP • Schülerfehlerquelle: Geschlechtszuordnung von Vornamen
9:40	• Wiederholung (CP): Abfragen, Notation (Unterschied weiblich(theresia) vs. weiblich(Theresia)), Variable; gute Konzentration • typischer Fehler: Wissensbasis wurde nicht konsultiert • Lehrer: „Was habe ich noch für Möglichkeiten?“ Existenzabfrage, Allabfrage
9:45	• Relationen üben: Wissensbasis erweitern, Startbeispiel der letzten Stunde wird wieder aufgegriffen • Lehrer: „Ist Wissensbasis vollständig? Wird da alles beschrieben?“ UG zu Beziehungen folgt: „Was ist da eine Beziehung? Wie kann man das feststellen? Kann man das markieren? Was möge ich dann umgangssprachlich sagen?“ • Beispielbeziehung umgangssprachlich und in Prolognotation (OH), Reihenfolge der Objekte in Beziehung egal, aber einheitlich • Bespielbeziehung: elternVon; Schüler: „Das ‘Von’ in ‘elternVon’ muss immer groß geschrieben werden?“; Lehrer: Verbesserung der Lesbarkeit, im Prinzip egal aber innerhalb einer Wissensbasis einheitlich
9:50	• Zusammenfassung der Grundbegriffe (Prädikat, Funktor, Objekt,...): UG, OH; Lehrer: Abfragen machen können
9:53	• Üben von Relationen und Eigenschaften am Beispiel Ahnentafel als HA: Murren der Schüler, Erläuterung Lehrer dazu, Schüler notieren

9:55	• Üben zum Implementieren und Erkunden der Modellierung • Beginn der Arbeit mit dem Arbeitsblatt • Zielorientierung durch Lehrer: „an Eurer Ahnentafel weiterbasteln“ • produktive Unruhe • fünf Mädchen an zwei Rechnern, heftige Wortwechsel • Problem: wo liegt die Ahnentafel? z.T. auf anderen Rechnern, daher Platzwechsel erforderlich, Unruhe • aufgrund der Kursstärke entsteht mehr Hilfebedarf, als Lehrer leisten kann; Unruhe; Diskussion mit dem Stammlehrer, Gruppenberatung, Mentor berät mit • Mädchen jauchzt auf (Erfolgserlebnis) • Problem: Prolog-System fehlt auf einigen Rechnern!
10:05	• erste Schülerergebnisse: PA, Lehrer betreut individuell • zwei Mädchen in der 5er–Gruppe in der zweiten Reihe plaudern: worüber? keine Gruppenarbeit! • Syntaxfehler; großer Hilfebedarf

D. Unterrichtsauswertung (Beispiel)

D.1. Beispiel für prädikative Modellierung

Hospitation einer Informatik–Doppelstunde

Praktikant:	N.N.
Universitätsbetreuer:	N.N.
Mentor:	N.N.
Ort:	Gesamtschule
Datum:	
Zeit:	8:35–9:20
Thema der ersten Stunde:	Einführung in die Arbeit mit dem Prolog–System
Zeit:	9:30–10:15
Thema der zweiten Stunde:	Einführung in das praktische Arbeiten mit dem Prolog–System

Fragen

1. Welche Ziele hat sich der Lehrer gestellt? Welche davon sind erreicht worden? Wie gelang es dem Lehrer, den Lehr–Lern–Prozess umzusetzen?
2. Gelang die Umsetzung der Struktur? Schlussfolgerungen?
3. Wie waren die Schüleraktivitäten?

Spontaneinschätzung durch Lehrer (Lehramtsstudierender)

- 1.Stunde mit Theorie überladen,
- SE kamen zu kurz (5 Minuten vor der Pause),
- Abschreiben durch die Schüler dauert länger als erwartet,
- Prolog–System auf einigen Rechnern nicht vorhanden, ungünstig für den Gast an der Schule, der das nicht vorbereiten kann; Lehrerwechsel im Labor
- nicht ganz mit 2.Stunde zufrieden,
- zuviel Lehrerbewegung zwischen Beamer und OH–Projektor?
- Lehrer hatte Schreibfehler in der Prolog–Syntax an der Tafel; Schüler machen Lehrer darauf aufmerksam, problematisch?
- Struktur wie gedacht umgesetzt, aber zu häufiger Wechsel zwischen Medien?
- wenige Meldungen in der Theoriephase, fast nur Jungen

Einschätzung durch Universitätsbetreuer

- gute Stunde: Konzept, Unterrichtsgespräch in Ordnung, Platz gut genutzt (Tafel, Folie, ...),
- Lehrer kommt gut bei Schülern an,

- Schüler können dem Unterrichtsverlauf gut folgen,
- Schülerkommentar „... der Strich, der zu den Eltern führt.“ hätte vom Lehrer gut zur Einführung des Begriffs Relation verwendet werden können,
- zu Fehler 1: auf „?“ als Eröffnung einer Anfrage muss eingegangen werden
- zu Fehler 2: Variablenbegriff war Schülern nicht präsent, muss gründlicher erarbeitet werden,
- gut, dass ein Schüler zum Anschrieb an die Tafel geholt wurde,
- manche Schüler wollten im Prolog–System im Kommandofenster editieren,
- Lernziel überhöht: „Schüler erlangen Sicherheit ...“
$\rightarrow$ kein geeignetes Lernziel für diese zwei Stunden,
- Lebhaftigkeit durch Medienwechsel im Unterricht kein Problem!
- Fragetechnik; mehr Fachvokabular verwenden,
- Fragen so stellen, dass Ein–Wort–Antworten vermieden werden,
- sprachliche Selbstbeobachtung stärken (z.B. Füllworte vermeiden).

Einschätzung durch den Mentor

- grundlegender Einsatz im Umgang mit der großen Gruppe positiv,
- technischer Medieneinsatz positiv,
- Wiederholung von Schülerantworten (Lehrerecho) nicht so gut, besser: Schüler wiederholen Antworten,
- nicht nur W-Fragen (z.B. Wer kann mir das sagen?), sondern verstärkt zielgerichtete Aufträge an Schüler formulieren (Aufforderungscharakter!),
- Anschrieb während der Stunde an verdeckte Tafel kann von Schülern als mangelndes Vertrauen interpretiert werden, besser: vor der Stunde anschreiben oder Folie vorbereiten,
- Lernziele verfeinern (statt Ziel Z_1 besser $Z_{1,1}, Z_{1,2}, ...$); diese müssen nicht alle gleichgewichtig sein (Detail, Fertigkeit, Fähigkeit, ...),
- Strukturierung der Lernziele: es muss zu erkennen sein, welche Lernziele nicht von allen Schülern erreicht werden müssen,
- segr schlecht, wenn Lehrer während eines Schülervortrags etwas anderes tut,
- intellektuelles Anforderungsniveau recht niedrig am Anfang,
- Schüler haben freiwillig z.T. bis in die Pause hinein gearbeitet.

E. Leistungskontrolle und –bewertung

Funktionen der Lernerfolgsüberprüfung

- Planung und Steuerung konkreter Unterrichtsverläufe
- Grundlage für Schülerberatung
- Grundlage für Beurteilungen, die rechtliche Konsequenzen haben
- Evaluation von Lehr–Lern–Prozessen mit Konsequenzen für Konzepte

Probleme

- Pionierzeit: Mittelmaß weder definierbar noch herstellbar,
- Werkzeuge konkurrieren mit dem Lehrerurteil,
- Gruppenarbeit,
- Aufgaben aus anderen Fächern,
- Auswertung: Persönliche Verletzbarkeit und Datenschutz beachten.

Maßstab

- Sehr gut: konstruktive und effektive Lösungsvorschläge,
- Gut: konstruktive und funktionierende Lösungsvorschläge,
- Befriedigend: überwiegend konstruktive und funktionierende Lösungsvorschläge,
- Ausreichend: übernehmen Lösungen von anderen und können sie reproduzieren,
- Mangelhaft: zu kurzschlüssige und falsche Vorschläge (erkennen nicht den Punkt, auf den es ankommt).

E.1. Einheitliche Prüfungsanforderungen in der Abiturprüfung Informatik (EPA)

Quelle: [KMK91]

- Prüfungsgegenstände
  - Fachliche Inhalte
  - Vertiefungsmöglichkeiten im Leistungsfach
  - fachliche Qualifikationen
- fachspezifische Anforderungsbereiche
- schriftliche Prüfung: Erstellen von Aufgaben und Bewertung
- mündliche Prüfung: Ziele und Bewertungskriterien
- Aufgabenbeispiele für Grund– und Leistungsfach

Prüfungsgegenstände, fachliche Inhalte

- Algorithmische Grundlagen, Gegenstände und Methoden der Informatik
  - Algorithmen und Datenstrukturen
  - Programme
  - Methoden der Software–Entwicklung
- Funktionsprinzipien von Hard– und Software–Systemen
  - Anwendersoftware
  - Programmiersprachen und –umgebungen
  - Betriebssoftware
  - Rechnermodelle und reale Rechnerkonfigurationen
  - Theoretische Grundlagen
- Anwendungen von Hard– und Softwaresystemen und deren gesellschaftliche Auswirkungen
  - Anwendungsgebiete
  - Mensch–Maschine–Schnittstelle
  - Grenzen und Möglichkeiten / Chancen und Risiken des Einsatzes der Informations– und Kommunikationstechniken

I	**Reproduktion**	ca. 40 %
	• die Wiedergabe ... • die Beschreibung ...	
II	**Transfer**	ca. 50 %
	• selbständiges Auswählen ... • selbständiges Übertragen ...	
III	**Problemlösendes Denken** planmäßiges Verarbeiten komplexer Gegebenheiten mit dem Ziel, zu selbständigem Gestalten ...	ca. 10 %

Bewertung von Prüfungsleistungen [KMK91], S. 21

„Die Note „ausreichend“ (5 Punkte) soll nur verteilt werden, wenn annähernd die Hälfte (mindestens vier Zehntel) der erwarteten Gesamtleistung erbracht worden ist. Oberhalb und unterhalb dieser Schwelle sollen die Anteile der erwarteten Gesamtleistung den einzelnen Notenstufen jeweils ungefähr linear zugeordnet werden, um zu sichern, dass mit der Bewertung die gesamte Breite der Skala ausgeschöpft werden kann.“

E.2. Diskussion von Kontrollformen

Klausuren [KMNW84], [MSNW96], S. 100 ff.

- schriftliche Begründung der Note
- Sonderstellung relativiert durch „Sonstige Mitarbeit“
- Anzahl (meist 2) und Dauer (meist 2–3 Unterrichtsstunden)
- wachsender Schwierigkeitsgrad der Teile
- Gefahr der Überforderung:
  für die Suche nach algorithmischen Lösungsverfahren in der Informatik sind erhebliche kreative Fähigkeiten erforderlich

- Einzelarbeit mit klarem Ziel und Zeitlimit,
- gleiche Anforderungen an alle Schüler(innen),
- gute Vergleichsmöglichkeit der erzielten Leistungen,
- kaum Täuschungsmöglichkeit,
- Möglichkeiten und Probleme der Gerätenutzung (mit und ohne Computereinsatz),
- Abhängigkeit von Tagesform und Aufgabengestaltung (mehrere kleine Aufgaben besser als eine komplexe Aufgabe, Zusammenhang oder Unabhängigkeit der Aufgaben(teile)).
- Aufgabentypen: Analysieren, Vergleichen, Modifizieren, Entwickeln (Gefahr der Überforderung der kreativen Fähigkeiten),
- Korrektur:
  Auf Grund des Variantenreichtums ist der Zeitaufwand für die Korrektur sehr hoch. Die Art und Schwere der Fehler kann unterschiedlich bewertet werden. Wurden ein oder mehrere Lernziele in Frage gestellt? Bewährt hat es sich, den Lösungsschritten Punktzahlen zuzuordnen.
  Der **Bewältigungsaspekt**, bei dem schwierigere Teile größeres Gewicht bekommen, führt zu einer Häufung unterdurchschnittlicher Noten.
  Der **Ausfallaspekt** geht davon aus, wer leichtere Aufgaben nicht lösen kann, hat gravierendere Lücken. Dafür werden mehr Punkte abgezogen.

Mündliche Leistungskontrolle (auch Referate)

- unterschiedliche Leistungsanforderungen, Frage der Vergleichbarkeit,
- Abhängigkeit von Tagesform und Aufgabengestaltung,
- Mitarbeit im Unterricht, Einbringen eigener Ideen und Stellungnahmen.

Praktikum

- Beobachtung der Arbeit am Computer – Kriterien:
  - Vertrautheit,
  - zweckmäßige und effektive Nutzung,

  - angemessene Reaktion auf Rückmeldungen des Systems,
  - in welchem Zustand wird der Arbeitsplatz hinterlassen.
- untypische Einzelbeobachtungen nicht überbewerten,
- Arbeit mit fertiger Software, Reduktion und Abstraktion, vom Problem zur Aufgabe, von der Aufgabe zur Lösung,
- Bearbeiten von „Test“–Aufträgen:
  - Texte,
  - Dateien,
  - Tabellenkalkulation,
  - Syntaxfehler–Korrektur,
  - Logikfehler–Korrektur,
  - Programm–Analyse,
  - Programm–Modifikation,
- Projektarbeit mit komplexerer Aufgabe ermöglicht die Anwendung der einzelnen Informatikerkenntnisse und –methoden. Im Mittelpunkt steht nicht das Informatikprodukt, sondern die schrittweise Erweiterung des Wissens und Könnens der Lerngruppe. Die Motivation kann mit freier Themenwahl erhöht werden. Die soziale Kompetenz in der Teamarbeit wird geübt. Eine Beratung in der Gruppe kann die Transparenz der Einzelleistung erleichtern.
- Leistung entsteht über längeren Zeitraum und erfaßt den gesamten Tätigkeitszyklus.
- differenziertes Arbeiten, Vergleichbarkeit der Leistungen sichern, Note 1 muß erreichbar sein,
- Fremdeinflüsse,
- Verteidigung der Ergebnisse vor der Gruppe oder im Einzelgespräch,
- hoher Betreuungsaufwand.

Hausaufgaben

- Analogie zu anderen Fächern, aber Gefahr der Vorteile für Computerbesitzer,
- Entlastung der Unterrichtsstunde durch Festigung des Grundwissens,
- Unterstützung des Praktikums durch Entwurfsarbeiten,
- fachliche Korrektheit und Systematik der Schülernotizen.

E.3. Erfahrungsberichte und Probleme

- Schulcomputerjahrbuch [Bosler92], S. 203–221
  „Zur Leistungsbewertung im Informatikunterricht“
- LOGIN
  - „Mündliche Abiturprüfung im Fach Informatik“ [Baumann91a]
  - „Klausuren und mündliches Abitur im Informatikunterricht“ [BV91]
  - „Leistungsmessung im Informatikunterricht“ [Baumann86]
  - „Leistungsmessung bei der Projektarbeit im Informatikunterricht“ [Koerber86]

E.4. Beispiele

- Grundbegriffe definieren und in Zusammenhänge einordnen,
- Basismechanismen erläutern und vergleichen,
- Informatikprinzipien und –methoden anwenden,
- Analysieren:
  - Eingabe und Ausgabe bestimmen,
  - Datentypen bestimmen,
  - Hierarchie der Teilprobleme,
  - Was leistet das folgende Programm oder die Anwendersoftware?
  - Eigenschaften einer Lösung erkennen und beschreiben (Struktururierung, Benutzung, Verbesserung).
- Modifizieren: Rahmenprogramm oder Rahmenlösung zur Verfügung stellen, Änderungen und Ergänzungen fordern,
- Entwickeln: Pläne entwerfen, Daten strukturieren.

F. Themen zur Vertiefung

F.1. Computermodell

- Zeichenverarbeitung betonen,
- **zentrale Steuerschleife – von–Neumann–Prinzip**,
- Problemklassen,
- Es existieren Alternativen (z.B. Transputer, neuronale Netze).
- Programm muss nicht statisch sein. (z.B. maschinelles Lernen).
- Computer kann programmiert werden, um auf unterschiedliche Situationen zu reagieren, Entscheidungen zu treffen, sich etwas zu merken.
- System von Schaltern,
- Zustand eines Systems als Gedächtnis,
- Analogie zum Problemlösen von Mensch und Computer,
- Programm ist eine Menge von Befehlen.
- Das wichtigste Programm ist das Betriebssystem.

F.2. Textverarbeitung

- Anforderungen an Anwendersoftware – Benutzungsfreundlichkeit:
  - Aufgabenangemessenheit,
  - Selbstbeschreibungsfähigkeit,
  - Steuerbarkeit - Dialogeigenschaften:
    - ▷ Aufgaben: Objekt–Repräsentation, Interpretation der Nutzerinteraktion,
    - ▷ Initiative beim Anwender,
    - ▷ hohe Flexibilität, Nutzermodellierung,
    - ▷ geringe Komplexität, Minimieren von Befehlsoptionen,
    - ▷ Dialogstile: Menüs, Eingabemasken, Kommandosprachen, natürliche Sprache, direkte Manipulation.
  - Erwartungskonformität, kognitives Modell des Werkzeuges,
  - Fehlerrobustheit.
- Objektorientiertes Modellieren:
  - Klassen (z.B. Glückwunschkarte),
  - Objekte (z.B. Glückwunschkarte zum Schulabschluss) zerlegen in Teilobjekte (z.B. Bild, Standardtext, persönlicher Text),
  - Instanzen (z.B. Glückwunschkarte zum Schulabschluss für Anne),
  - Aktionen (Methoden): Malen, Abschreiben, Formulieren,
  - Eigenschaften (Attribute): Seitenformat, Schriftart, Hervorhebungen, Farben.

F.3. Problemlösen durch Programmieren

- Analyse, Spezifikation (Anforderungsbeschreibung), Entwurf, Codierung, Implementation, Test, Anwendung, Wartung,
- Modellierung:
  Ein Weltausschnitt von unstrukturierten Ausgangsdaten wird unterlegt mit einem Strukturkonzept, das das zu entwickelnde System zerlegt und gliedert. Dabei wird häufig mit Analogieschluss auf einen Vorrat bekannter Strukturkonzepte zurückgegriffen.
- Anfangs–, Ziel– und Zwischenzustände, Objekte, Beziehungen zwischen den Objekten, Operatoren mit Anwendungsbedingungen,
- Ziel in Teilziele zerlegen, Abhängigkeiten zwischen den Teilzielen erkennen und bei der Planung berücksichtigen,
- Schlüssel zum Erfolg: Repräsentation des Problems (operatorgerecht),
- Verantwortung des Menschen: Interpretation, Wertung der Ergebnisse.

F.4. Prinzip der strukturierten Programmierung (mit einer prozeduralen [imperativen] Sprache)

- Jeder Strukturbaustein besitzt genau einen Eingang und einen Ausgang. Es werden keine Sprungbefehle verwendet.
- Drei Grundstrukturen (Sequenz [Folge, Reihung], Zyklus [Wiederholungen, Schleifen], Alternativen [Fallunterscheidungen]) werden zu immer umfangreicheren (komplexeren) Lösungen verknüpft.
- Unterprogrammtechnik sichert Übersichtlichkeit und Wiederverwendbarkeit der Lösungsbausteine (Fehlerquelle Parameterübergabe).
- Die Wahl der Datenstruktur beeinflusst die Lösungskomplexität.
- Methode: Top–down–Entwurf mit schrittweiser Verfeinerung,
- logische Fehler bei der Formulierung der Bedingungen, fehlende oder falsche Anfangswerte, Überschreiben von Speicherinhalten.

F.5. Programmieren mit Turtle–Grafik

- Vorteil: Anschaulichkeit des programmierten Ergebnisses,
- Bedingung: anfängergerechte Programmierumgebung,
- LOGO (funktionale Programmiersprache) von Seymour Papert hatte ursprünglich eine mechanische Schildkröte zum Malen für Kinder.
- Nachteil: eingeschränkte Anwendbarkeit dieser Grafiken,
- ausgezeichnete Lehrbücher für jüngere Schulkinder.

F.6. Problemlösen durch Anwendersoftware

- neue Quelle: Baues, J.; Hillebrand, H.-P.; Hüster, E.; Mersch, B.: Informatik erleben. Dümmler Verlag, Bonn 1995 (mit 3 Disketten).
- Möglichkeiten und Grenzen der Informationsgesellschaft:
  - Veränderung der Arbeitswelt (Rationalisierung, Humanisierung),
  - Veränderung des Lernens (lebenslang, selbstorganisiert),
  - Kommunikationsgewohnheiten.
- Aufgabenanalyse führt zur Werzeugauswahl,
- Modellierung:
  - Kreativität im Anwendungsbereich erforderlich,
  - Datenbankmodelle (Relationen, Hierachie, Netzwerk):
    Jedes Modell erfordert eine besondere konzeptionelle Beschreibung der Informationen und der Beziehungen zwischen ihnen.
  - elektronische Dokumente als Zielvorstellung (Netzpräsentation) oder als Brückenfunktion (Hausarbeit, Schülerzeitung, Lebenslauf),
- Objekte und Teilobjekte entwerfen, bearbeiten, speichern, nutzen:
  - Arbeitsanleitung (Schrittfolgen, Dialogpläne),
  - Musterlösungen für wiederkehrende Aufgaben,
  - Lösungsrahmen mit individuellen Informationen füllen.
- Funktionen des Systems in der richtigen Reihenfolge aktivieren, Entlastung von traditionellen Tätigkeiten:
  - Reduzierung der Komplexität,
  - Wiederverwendbarkeit von Bausteinen,
  - Makrosprachen zur Programmierung von anwenderspezifischen Befehlsfolgen.
- Informatiksysteme als Medium (Multimedia–Anwendungen und Telekommunikation):
  - Präsentationen (Texte, Bilder, Töne, Filme),
  - Publizieren (Verlust der Qualität),
  - Suchen (Datenbanken, Server),
  - Gruppenarbeit (auch über größere Entfernungen).
- Vorstellung vom Informatiksystem (kognitives, mentales Modell):
  - Funktionsumfang, Zugriffsmöglichkeiten,
  - Dialoggestaltung, Bildschirmaufbau,
  - Manipulationsmöglichkeiten und Glaubwürdigkeit,
  - Datenschutz und Datensicherheit.

F.7. Tabellenkalkulation und Diagramme

- Das Automatisieren von Berechnungen und deren Verknpfung zu komplexen Auswertungen ist möglich. Das Formelgeflecht entspricht einem Netzwerk von Bausteinen.
- Nutzer plant die Zusammenhänge (Struktur) und die Berechnungsreihenfolge (Programm).
- Der Diagrammtyp muss zur Zielstellung passen. Das Problem entscheidet über die Werkzeugauswahl (Funktionsauswahl).
- Angestrebt wird die Wiederverwendbarkeit von Lösungen (Rahmenlösungen) für Aufgabenklassen.
- Die Dynamik des Arbeitblattes (Hintergrundprogramm) aktualisiert bei Änderung eines Wertes sofort alle Ergebnisse.
- Grundbegriffe: Arbeitsblatt, Tabelle, Zelle, Spalte, Zeile, Adresse, Zeichen, Datentyp, Wert, Formel, Diagramm.
- Objektorientiertes Modellieren:
  - Klassen (z.B. Tabelle, Diagramm),
  - Objekte (z.B. Reisefinanzierung) zerlegen in Teilobjekte (z.B. Ausgaben (Fahrt, Unterkunft, Verpflegung, Eintritt), Einnahmen (Arbeitsleistung, Auszeichnung, Zuwendung),
  - Instanzen (z.B. Schulabschlussfahrt nach Prag),
  - Aktionen (Methoden): Struktur festlegen (anlegen), Werte eintragen (ausfüllen), Formeln eintragen (ausrechnen).
  - Eigenschaften (Attribute): Spaltenanzahl und Spaltenbreite, Währung wählen, Ergebnisse hervorheben.
- Mit jeder Zelle können vier wichtige Informationen verbunden werden:
  - Ihre Bezeichnung (Adresse) liegt immer fest.
  - Eine Formel **im Hintergrund** kann der Nutzer zuordnen.
  - Den Wert **im Vordergrund** ermittelt entweder das System selbst auf Grund der Formel oder der Nutzer trägt ihn ein.
  - Den Datentyp bestimmt der Nutzer mit der Formatierung (z.B. Text, ganze Zahl, Dezimalzahl).
- Werzeugbestandteile:
  - Bildschirmaufbau (Dialoggestaltung),
  - Funktionen (Unterprogramme) liegen in Ikonen und Menüs,
  - Dateiverwaltung.
- Manipulationsmöglichkeiten bei Erhebung, Reduktion und Darstellung der Daten sind zu beachten.

F.8. Multimedia–Anwendung (Medienverbund)

- Vorläufer sind die Hypertext–Systeme, mit denen Text und Grafik (zeitunabhängige oder diskrete Medien) zu einem **interaktiven Programm (Netzwerk von Seiten)** verbunden werden können.
- Multimedia–Systeme verbinden die zeitunabhängigen Medien mit den zeitabhängigen oder kontinuierlichen Medien (Audio und Video) und ermöglichen deren **Steuerung durch den Nutzer**. DTP–Systeme sind nicht multimedial.
- Multimedia–Anwendungen sind sehr gut geeignet zum Speichern, Präsentieren und Erkunden (Kennenlernen) komplexer Zusammenhänge und großer Informationsbestände. Fakten werden leichter zugänglich, wenn die Relation bekannt ist. Der Mensch muss wissen, wonach er sucht.
- Die Mensch–Computer–Interaktion erfolgt über Kommunikationskanäle. Dabei werden Sprechen und Hören unterstützt.
- Die Informationsfülle kann nach individuellen Wünschen reduziert werden. Diese Filterung von Informationen erfordert Aufklärung und Erfahrung. Eine neue Informatikgrundbildung wird notwendig.
- Anwendungsklassen (Überschneidungen sind typisch):
  1. Telekommunikation (z.B. Videokonferenz),
  2. Verteilte Systeme (z.B. Deutscher Bildungsserver),
  3. Kooperations–Systeme (z.B. Ressourcen simultan nutzbar in computerunterstützter Gruppenarbeit)
- Multimedia–Anwendungen eignen sich für Übungen, die viele Sinne des Lernenden aktivieren wollen, z.B. Sehen, Hören und Sprechen beim Erlernen von Fremdsprachen.
- Multimedia–Anwendungen können Menschen mit Behinderungen (z.B. Blinde) besonders unterstützen.
- Es wird eine erweiterte Simulation möglich, z.B. virtuelles Bauen und begehbare Kunstwerke (Modell der Frauenkirche in Dresden).
- Multimedia–Anwendungen sind die Grundlage für Systeme zur Nutzung der Virtuellen Realität (VR), die hochkomplexe Datenmengen veranschaulichen. VR erfordert 3D–Sichtsysteme (Datenhelm, –maske, –brille) und Greif– und Tastsysteme (Datenhandschuh, –anzug).
- Hypermedia führt zu nicht–linearen, elektronischen Dokumenten, in denen Multimedia–Anwendungen zu einem interaktiven Programm (Netzwerk von Seiten) verbunden sind. Navigationswerkzeuge sind erforderlich.
- Hohe Kosten bremsen den Einsatz solcher Systeme. Die Urheberrechte sind zu klären. Leistungsfähige Kompressionstechniken für die sehr großen Datenmengen werden entwickelt.

F.9. Der Variablenbegriff in der Informatik

- Daten (Variable und Konstante),
- Erarbeitung des Variablenbegriffs der Informatik (Vergleich zur Mathematik),
- Variable mit Name, Typ, Wert, Adresse,

- Typen:
  - Teilmenge der ganzen Zahlen,
  - Teilmenge der reellen Zahlen,
  - Zeichen,
  - Zeichenkette,
  - Wahrheitswert,
- Zugriffsmöglichkeit:
  - schreibender Zugriff zerstört den alten Wert,
  - lesender Zugriff verändert den Wert nicht,
  - Variable ermöglicht schreibenden und lesenden Zugriff,
  - Konstante gestattet nach einmaligem Schreibzugriff (Festlegen des Wertes) im Vereinbarungsteil nur noch das Lesen des Wertes im Anweisungsteil.
- Wertzuweisung:
  - $X \Leftarrow X + 1$ ist eine Fehlerquelle,
  - der alte Wert wurde überschrieben,
  - Unterschied zwischen Transportbefehl und Gleichung.
- lokale und globale Variablen:
  - Hierarchie von Hauptprogramm mit Unterprogrammen,
  - Relativität der Eigenschaften „lokal" und „global",
  - Struktur und Beziehungen erkennen (Fehlerquelle).
- Beispiel Speicherplatzwechsel von a und b:
  - Analogie zum Umfüllen zweier Flüssigkeiten führt zu der Erkenntnis, dass ein Hilfsbehälter (Speicherplatz $hilf$) benötigt wird.
  - Anweisungen: $hilf \Leftarrow a$, $a \Leftarrow b$, $b \Leftarrow hilf$.

F.10. Grundstrukturen und deren Erweiterung

Grundstrukturen sind Sequenz, Zyklus, Alternative und Unterprogramm. Zur Darstellung eignen sich Struktogramme und Entwurf in der Muttersprache (Pseudocode). Algorithmus und Daten können gleichzeitig zusammengestellt werden. Die Methode der schrittweisen Verfeinerung hat sich bewährt.

- Grundalgorithmus Summe:
  - Anfangswert $Summe = 0$ setzen,
  - Anfordern und Eingabe des zu summierenden Wertes w,
  - Wiederholen, solange Wert $w \neq 0$:
    - ▷ $Summe \Leftarrow Summe + w$,
    - ▷ Anfordern und Eingabe des zu summierenden Wertes w,
  - Ausgabe von $Summe$.
  - Der Abbruch erfolgt, wenn die Bedingung falsch wird.

- Grundalgorithmus Tabelle:
  - Anfordern und Eingabe von Anfangswert t_a, Endwert t_e und Schrittweite dt für die Temperatur,
  - Anfangswert t auf t_a setzen,
  - Ausgabe des Tabellenkopfes,
  - Wiederholen, solange Wert $t \leq t_e$:
    - ▹ Berechnung der Länge l,
    - ▹ Ausgabe von Temperatur t und Länge l,
    - ▹ Berechnen von $t \Leftarrow t + dt$.
  - Jeder abweisende Zyklus kann in einen nichtabweisenden Zyklus umgewandelt werden.

Weitere Beispiele für Grundalgorithmen sind Maximum oder Minimum, Speicherplatzwechsel, Sortieren und Suchen. Mit dem Logiktest (Trace) kann man die Spur der Berechnung verfolgen.

Schwerpunkte zur Rekursion:

- Problemlösung durch Rückgriff auf Bekanntes,
- Aufruf und Abbruch,
- rekursive Datenstrukturen,
- Unterschied zwischen Zyklus und Rekursion,
- Stapel aufbauen und abräumen.

Beim Datenkonzept können Datentypen und Datenstrukturen (statische und dynamische) unterschieden werden.

Zur Datei gehören die Grundbegriffe Dateifenster, Dateiende, Dateizeiger, Puffer, Öffnen, Schließen, Erzeugen, Aktualisieren, Sortieren, Suchen, Ausgeben.

F.11. Unterprogrammtechnik – Prozeduren und Funktionen

F.11.1. Prinzipien und Methoden

- Prozedurenkonzept ist eines der mächtigsten Konzepte imperativer Programmiersprachen.
- Ziel: Bewältigung der Komplexität durch Zerlegung einer Aufgabe in Teilaufgaben und Zusammensetzung der Lösung aus Bausteinen (Unterprogrammen),
- Vorteile: Mehrfachverwendung der Bausteine und Übersichtlichkeit,
- Prozeduren und Funktionen als spezielle Unterprogramme, Analogie zu den Funktionen in der Anwendersoftware,
- Vereinbarung: Bereitstellen des Bausteines durch Beschreiben des Lösungsweges (Algorithmus) und der allgemeinen Daten (formale Parameter),
- Aufruf: Anwenden des Bausteines durch Namensangabe und Zuordnen von speziellen Daten (aktuelle Parameter),
- Veranschaulichung der Unterprogrammtechnik, Datenverarbeitungsmodell,

- Geltungsbereich der Variablen: Datenbereich des Hauptprogramms und Datenfenster des Unterprogramms werden als Stapel realisiert,
- Schnittstellendefinition:
  - Prinzip der Geheimhaltung: Es ermöglicht die klar definierten Schnittstellen. Die Lokalität von Objekten und die Spezifikation von abstrakten Datentypen sind die Methoden zur Realisierung.
  - Parameterkonzept: Der Parameterübergabe liegt die Idee der Vergabe eines Synonyms für ein Objekt zugrunde. Der Bezeichner des formalen Parameters wird innerhalb des Unterprogramms als Synonym für den aktuellen Parameter verwendet. Es existieren mindestens zwei verschiedene Modelle, Wertparameter und Variablenparameter. Das Prinzip der Adressenübergabe (call by referenz) ermöglicht den Export von Ergebnissen.
  - abstrakter Datentyp (ADT): Der Schwerpunkt liegt auf den Eigenschaften von Operationen und Wertebereichen. Nur der Spezifikationsteil (Gebrauchsanweisung) ist nach außen sichtbar (z.B. längenbeschränkter Keller, Analogie zum Münzbehälter).
- Typische Fehler sind Unklarheit über die Parameterart und folglich Verwechslungen (Modellierungsfehler) und Unverständnis für die Adressenübergabe (Werteverlust).

F.11.2. Lokale und globale Variablen

Beispiel ohne Parameter:

```
program Sterne1;
  var Zeilen : integer; (* globale Variable *)
  procedure Druck; (* UP - Vereinbarung *)
     var Spalten : integer; (* lokale Variable *)
  begin for  Spalten := 1  to 50 do write('*')   end;
begin for  Zeilen := 1  to 25 do
  begin Druck; (* UP - Aufruf *)   writeln   end
end.
```

Hauptprogramm und Unterprogramm sind Blöcke, die die Reichweite der Variablen begrenzen.

Beispiel mit Parametern:

```
program Sterne2;
  var Zeilen, Anzahl : integer; (* globale Variable und aktueller Wertparameter*)
  procedure Druck(Laenge : integer); (* UP-Vereinbarung mit formalem Wertparameter*)
     var Spalten : integer; (* lokale Variable *)
  begin for  Spalten := 1  to Laenge do write('*')
  end;
begin for  Zeilen := 1  to 25 do
  begin Druck(Anzahl); (* UP - Aufruf mit aktuellem Wertparameter*)
  writeln
  end
end.
```

Der Wertparameter bringt mehr Flexibilität in das Programm.

Die Relativität der Eigenschaften „lokal“ und „global“ kann an einer Hierarchie von Hauptprogramm mit geschachtelten Unterprogrammen erläutert werden.

- Beispiel: Ein Hauptprogramm $H1$ vereinbart die Variable x und das Unterprogramm $V1$. Das Unterprogramm $V1$ vereinbart die Variable y und das Unterprogramm $V2$. Das Unterprogramm $V2$ vereinbart die Variable z.
- Frage: Welche Variablen sind lokal und welche global?
- Antwort mit Begründung – Struktur und Beziehungen erkennen:
  - x besitzt als globale Variable des Hauptprogramms die größte Reichweite, gilt also auch in den Unterprogrammen $V1$ und $V2$.
  - y besitzt als lokale Variable des Unterprogramms $V1$ keine Gültigkeit im Hauptprogramm, ist aber zugleich global bezüglich des Unterprogramm $V2$.
  - z besitzt als lokale Variable des Unterprogramm $V2$ die kleinste Reichweite, gilt also nur hier.

F.11.3. Parametertest

- Beispiel [HJS82]:
  program Parametertest;
  var a, b, c, d : integer;
  procedure Test(var a1 : integer; b1 : integer); (∗ UP − Vereinbarung ∗)
  var c : integer;
  begin a1 := 1; b1 := 1; c := 1; d := 1
  end;
  begin a := 0; b := 0; c := 0; d := 0;
  Test(a, b); (∗ UP − Aufruf ∗)
  writeln(a, b, c, d)
  end.
- Frage: Welche Zahlen werden ausgegeben?
- Antwort – Analyse der Daten
  - Hauptprogramm: globale Variablen $a = 0|1,\ b = 0,\ c = 0,\ d = 0|1$,
  - Unterprogramm:
    - ▷ lokale Variable $c = 0|1$,
    - ▷ Wertparameter $b1 = 0|1$,
    - ▷ Variablenparameter $a1$ gestattet den Zugriff auf den Speicherplatz a des Hauptprogramms.
  - Ausgegeben werden: $a = 1,\ b = 0,\ c = 0,\ d = 1$.
  - Begründung:
    - ▷ $a1$ ist ein formaler Parameter vom Typ Variablenparameter, der den aktuellen Parameter a im Hauptprogramm verändern darf. $\rightarrow\ a = 1$
    - ▷ $b1$ ist ein formaler Parameter vom Typ Wertparameter, der den aktuellen Parameter b benutzen, aber im Hauptprogramm nicht verändern darf. $\rightarrow\ b = 0$
    - ▷ c ist kein Parameter, aber diese Bezeichnung wird lokal und global verwendet. Diese doppelte Bezeichnung bildet eine Fehlerquelle. Im Unterprogramm gilt nur die lokale Variable. Im Hauptprogramm gilt nur die globale Variable. $\rightarrow\ c = 0$
    - ▷ d ist eine globale Variable, d.h. sie kann im Unterprogramm benutzt und auch verändert werden. $\rightarrow\ d = 1$

F.12. Programmieren einer Datenbank

- Dem Datenbanknutzer werden verschiedene Benutzungsschnittstellen (Betriebsarten) zur Verfügung gestellt, z.B. Dialogmodus (Menühierarchie), Kommandomodus, Programmiermodus. Das ermöglicht Vergleiche der Nutzerzugänge. Die Wirkung von Interpreter und Compiler kann gegenübergestellt werden.
- Die Nutzung eines Werkzeuges zum Umsehen (Browser) ist typisch für große Datenmengen. Bekannt ist das heute vom WWW (World Wide Web).
- Analogie zur strukturierten Programmierung (Zyklen, Alternativen, Unterprogramme),
- Am Anfang steht häufig die Neuerstellung einer Datei. Dieser Vorgang setzt sich aus zwei Teilprozessen zusammen. Zuerst wird eine Dateistruktur entworfen. Das erfordert Kreativität und Erfahrung. Von der Qualität dieser Modellierung wird der Anwendungserfolg ganz wesentlich bestimmt. Der zweite Teil ist die Eingabe der Datensätze. Diese Aufgabe kann sehr langwierig sein, gehört aber eher zu den Routinearbeiten.
- Typischerweise sind die Dateien einer Datenbank nach der Neuerstellung regelmäßig zu aktualisieren. Bausteine (Programme) für das Einfügen, Löschen und Korrigieren von Datensätzen können nützlich sein.
- Das gleiche gilt für wiederkehrende Ausgaben.
- Komplexe Befehle können eingesetzt werden, z.B. für Sortieren und Suchen.
- Die Grundbegriffe Datenbank, Datei, Datensatz, Datenfeld und Datentyp werden vorausgesetzt.
- Dateien werden im Gegensatz zu früheren Erfahrungen **logisch geöffnet und geschlossen**.
- Das System verwaltet den Dateizeiger, den der Nutzer auch selbst steuern kann. Logische Fehler können dann zum Systemabsturz führen.
- Grundeinstellungen für das Datenbanksystem wirken wie Schalter.
- Neue Prinzipien und Methoden werden angewendet, z.B. Sortierkriterien, Indexdateien.
- Lernschwierigkeiten: Die Liste der möglichen Befehle ist lang. Zu den Grundbefehlen gehören viele Parametervarianten mit spezieller Auswirkung.

F.13. Rekursion

- Anschaulichkeit: Optische Rückkopplung – Spiegelbild im Spiegel,
- Problem durch Rückgriff auf Bekanntes (ein bereits gelöstes Problem) lösen → Begriff Rekursion (lat. recursare – zurückkehren) → Merkmale eines rekursiven Algorithmus:
  - Selbstaufruf (Wiederholung) nach einer
  - Abbruchbedingung (Bedingung zur Steuerung des Selbstaufrufes) und
  - Veränderung eines Wertes, der den Abbruch herbeiführt.
- Andreas Schwill: Formulierung fundamentaler Ideen der Informatik und Konsequenzen für den Informatikunterricht, z.B. Algorithmisierung → Programmierkonzepte → Rekursion (rekursive Prozedur, Baum, Suchbaum),

- Turtle–Grafik–Beispiele zum Zeichnen von ineinandergeschachtelten Quadraten, von Spiralen oder von Stufen,
- Es existieren Programmierstile, in denen können Zyklen ausschließlich mit Rekursion modelliert werden (z.B logische Programmierung, auch prädikative oder deklarative Programmierung genannt). Die rekursive Definition des Terms als Baustein der Datenstrukturen Liste und Baum ist der Grund dafür. Das Prolog–Beispiel zeigt eine solche rekursive Regel.

  Fakten:

  - $teil\_von(computer, zentraleinheit).$
  - $teil\_von(computer, peripherie).$
  - $teil\_von(zentraleinheit, steuerwerk).$
  - $teil\_von(zentraleinheit, rechenwerk).$
  - $teil\_von(zentraleinheit, hauptspeicher).$

  Regeln:

  - $besteht\_aus(X, Y) : -teil\_von(X, Y).$
  - $besteht\_aus(X, Y) : -teil\_von(Z, Y), besteht\_aus(X, Z).$

  Aufruf des Beispiels zur Rekursion mit $? - besteht\_aus(computer, Teile)$

  - 1. Regel: $besteht\_aus(computer, Teile) : -teil\_von(computer, Teile).$
    $Teile = zentraleinheit$ und $Teile = peripherie$
  - 2. Regel:
    $besteht\_aus(computer, Teile) : -teil\_von(Z, Teile), besteht\_aus(computer, Z).$

    $Z = zentraleinheit$
    - ▷ $Teile = steuerwerk$
    - ▷ $Teile = rechenwerk$
    - ▷ $Teile = hauptspeicher$

  Die Abbruchbedingung steht in der 2. Regel am Anfang: $teil\_von(Z, Y)$. Sie bewirkt den Abbruch, wenn keine Fakten der geforderten Form mehr gefunden werden.
- Es existieren Programmierstile, in denen die Rekursion das tragende Prinzip ist (z.B funktionale Programmierung, auch applikative Programmierung genannt). Zu dieser Sprachfamilie gehören Lisp und Logo.
- Modellvorstellungen: Speichermodell Keller (Stack), Kellerautomat.

Datenschutz und Datensicherheit

- Die Begriffe Datenschutz und Datensicherheit werde häufig verwechselt. Definition, Abgrenzung und Zusammenhang sind ausführlich zu erläutern.
- Es besteht die Möglichkeit, auf die historische Entwicklung einzugehen:
  - Die Informatik bringt mit erstaunlicher Geschwindigkeit neue Fachgebiete hervor. Am Beispiel der Informationssicherheit wird deutlich, wie diese schnelle Entwicklung möglich wird.

  - Die Kryptologie (Lehre von den Geheimschriften) und die Mathematik brachten in jahrzehntelanger Forschung die Definitionen, Sätze und Basismechanismen hervor, die in die Informationssicherheit einflossen.
  - Solange die Rechenzentren als Informatikanwendung dominierten wurden die Forderungen nach Schutzvorrichtungen für Hardware und Software durch Abschließen und strenge Zugangskontrolle realisiert. Mit der Dezentralisierung und Verbreitung der Personalcomputer reichten diese Verfahren nicht mehr aus.

- Datensicherheit – Grundbedrohungen und Schutzmechanismen
  - ***Verlust der Vertraulichkeit*** führt zu unbefugtem Informationsgewinn (z.B. Passwortdiebstahl). Verschlüsselung, Authentifizieren und Zugriffsschutz stellen geeignete Schutzmechanismen dar.
  - ***Verlust der Integrität*** bedeutet unbefugte Modifikation von Daten. Solche Fälschungen erkennt man durch Prüfsummen (zusätzliche Informationen über den Sender) oder Verschlüsselung.
  - ***Verlust der Verfügbarkeit*** äußert sich in unbefugter Beeinträchtigung der Funktionalität. Relevante Attacken können etwa durch Authentifizieren (Nachweis der Identität und digitale Unterschrift) eingeschränkt werden.
  - ***Verlust der Anonymität*** kann je nach Schwerpunkt mit unterschiedlichen Methoden verhindert werden, z.B. Nachricht an viele Teilnehmer senden, Nutzung von Pseudonymen, erhöhte Aktivität verschleiern. Wenn sicherheitsrelevante Daten aufgezeichnet werden, geht die Anonymität verloren. Es kann stets nachvollzogen werden, wer hat wann von wo mit welchen Mitteln was veranlasst und worauf zugegriffen.
  - ***Verlust der Originalität*** tritt auf, wenn Original und Kopie nicht zu unterscheiden sind (z.B. Mehrfacheinlösen elektronischer Schecks). Die digitale Unterschrift (z.B. RSA–Verfahren) kann mit einem Zeitstempel verknüpft werden.
  - ***Verlust der Verbindlichkeit*** tritt ein, wenn einer der Partner nachträglich die Teilnahme oder die Korrektheit eines Informationsaustausches bestreitet. Auch hier bieten digitale Unterschriften Schutz.

- Datenschutz – Grundbegriffe und Gesetze
  - Die Risiken, die beim täglichen Umgang mit ***personenbezogenen Daten*** auftreten können (gefälschte oder veraltete Daten, unberechtigte Einsicht, Verknüpfung scheinbar unwichtiger Informationen), sollen erkannt werden.
    Unter den Bedingungen der modernen Informatik gibt es kein belangloses Datum mehr. Daten, die einzeln nicht viel aussagen, können zu problematischen Verknüpfungen führen.
  - Ziel: Die Persönlichkeitsphäre des Menschen muss geschützt werden.
  - Grundbegriffe: personenbezogene Daten, automatisierte und nicht–automatisierte Datei, Akte, Erheben, Verarbeiten (Speichern, Verändern, Übermitteln, Sperren, Löschen) und Nutzen solcher Daten, Dritter, Datengeheimnis, Erlaubnisvorbehalt.
  - ***Grundgesetz*** (Artikel 1, Abs. 1 und 2) garantiert den Schutz der informationellen Selbstbestimmung.
  - ***Bundesdatenschutzgesetz*** enthält:
    - ▷ detaillierte Vorgaben
    - ▷ mit rechtlichen Sanktionen,
    - ▷ Kontrolle (Datenschutzbeauftragte) und
    - ▷ Rechte der Betroffenen.

    Erforderlich sind Maßnahmen nur, wenn ihr Aufwand in einem angemessenen Verhältnis zu dem angestrebten Schutzzweck steht.
  - ***Landesdatenschutzgesetz*** konkretisiert für den öffentlichen Bereich der Länder den Umgang mit personenbezogene Daten.
  - ***Datenschutz an Schulen*** liefert für den Einstieg in das Thema die Motivation, indem Betroffenheit hergestellt wird.
  - Die ***Umsetzung*** der gesetzlichen Bestimmungen erfolgt durch die Anwendung der Datensicherheit (Verfahren und Werkzeuge) und durch organisatorische Maßnahmen.

G. Logische (deklarative) Programmierung

G.1. Charakteristik

Eine typisch Sprache ist Prolog. Sie besitzt deklarative und prozedurale Bestandteile. Dieser Programmierung liegt die Prädikatenlogik 1. Stufe zugrunde. Die Resolution wird als Beweisverfahren bei der Lösungsbestätigung angewendet, d.h. das Prolog–System geht wie ein Theorembeweiser vor. Die Wissensbasis entspricht dem Programm (Fakten, Regeln). Fragen wirken wie Eingabewerte, mit denen das Programm gestartet wird. Die Antworten stellen die Ausgabewerte dar. Der Nutzer schreibt auf, was er über das Problem weiß und nicht, wie man die Lösung erhält. Das Prolog–System besitzt einen vorgefertigten Verarbeitungsmechanismus, der die Fakten und Regeln auswertet und zur Lösung verknüpft. Man spricht von Wissensverarbeitung, die gegenüber der traditionellen Datenverarbeitung eine neue Qualität darstellt. Der Arbeitsstiel des experimentierend forschenden Arbeitens [Löthe88] eignet sich gut für die Logische Programmierung. Die fundamentale Kontrollstruktur ist die Rekursion. Damit werden Zyklen ausgedrückt.
Die typischen Anfängerfehler lassen sich in Modellierungs– und Dialogprobleme einteilen.

Modellierungsprobleme:

1. Abgrenzung des Weltausschnittes (Boxmodell, abgeschlossene Welt),
2. Bestimmung der Objekte,
3. Zuordnung der Eigenschaften,
4. Aufstellen von Regeln (Regeln als virtuelle Aussagen).

Dialogprobleme:

1. Notwendigkeit von Anfragen,
2. keine Beurteilung der Aussagen durch das System,
3. mangelnde Transparenz der Lösungssuche (virtuelle Prolog–Maschine),
4. Flüchtigkeit von logischen Variablen (beziehen sich auf Objekte, nicht auf Speicherzellen).

Die Fachdidaktische Gestaltung konzentriert sich auf:

1. Wissen strukturieren:
   a) Objekte, Eigenschaften, Beziehungen,
   b) Modell abgeschlossene Welt,
   c) Formalisierung zu Fakten und Regeln,
   d) Varianten für Datenstrukturen.
2. Erkunden der Prolog–Maschine:
   a) Faktenbasen erweitern,
   b) Ableitungsbaum als Darstellungemittel,
   c) Ableitungsregeln,
   d) Regelverkettung,

   e) Beobachtung der Variablenbindung,
   f) Backtracking steuern.

3. Programmierheuristik:
   a) Reihenfolge der Fakten und Regeln,
   b) Gefahr von Endlosschleifen,
   c) Strukturierung von Regelwerken.

4. Heuristik zur Beschränkung von Suchräumen.

G.2. Skizze einer Unterrichtsreihe mit Prolog

Begründung
Die Wahl der logischen Programmierumgebung mit Prolog für diese einführenden Informatik–Lernbereiche in der Klasse 7 des Gymnasiums hat drei Gründe:

1. Dem Anfänger wird ein erheblicher Teil des Programmieraufwandes abgenommen. Er beschreibt sein zu lösendes Problem mit Fakten und Regeln, ohne sich um die Lösungssuche selbst zu kümmern. Diese wird ihm in vorgefertigter Software zur Verfügung gestellt (Tiefensuche mit Rückkehrverfahren). Das hat für den Anfänger den Vorteil, dass er mit einfachen Anfragen an sein Programm, komplizierte Lösungsmechanismen auslösen kann, die er schrittweise erkunden und für sich nutzen lernt.

2. Es werden rekursive Denk– und Arbeitsweisen, die zu den fundamentalen Ideen der Informatik gehören, fast spielerisch erlernt, weil einfache Daten– und Algorithmenstrukturen in Prolog darauf basieren und nur damit möglich sind. Sie stehen damit nicht am Ende einer langen Einführung, sondern gehören bereits zur Gestaltung erster, einfacher Programme. So wird Rekursion zu einer rasch vertrauten Entwurfsmethode bei den eigenen Problemlösungen.

3. Der Anfänger von dem sonst üblichen Ballast der Syntaxbesonderheiten einer Programmiersprache entlastet. Er braucht keine Schlüsselworte erlernen, er bildet sie selbst. Die wenigen Schreibregeln von Prolog lassen sich an einer Hand aufzählen. Dagegen liegt der Schwerpunkt auf den Anforderungen an das Modellieren der Problembeschreibung. Logisches Denken wird gefördert. Die eigene Verantwortung für die Korrektheit der Wissensbasis (des Prolog–Programms) ist leichter einzusehen als in anderen Programmierumgebungen.

Damit steht den Lehrern eine Programmierumgebung zur Verfüng, die die Bezeichnung Lernumgebung verdient. Die folgende Unterrichtsskizze soll mehr Lehrer ermutigen, diesen neuen Weg zur Einführung in das Problemlösen mit Mitteln und Methoden der Informatik zu wählen, und ihnen zugleich helfen, mit vertretbarem Einarbeitungsaufwand zu Unterrichtserfolgen zu gelangen. Am Ende der Informatikallgemeinbildung könnte dann für immer mehr Schüler die Erkenntnis stehen, dass zu jeder Problemklasse die richtigen Programmierumgebung auszuwählen ist.

Plan für zehn Doppelstunden

- Einführung in die gewählte Programmierumgebung

  1. Starten des Systems – Kennenlernen der Benutzungsoberfläche, Laden und Nutzen des 1. Beispiels – Anfragen an die Faktenbasis, Analyse des 1. Beispiels – Lösungssuche als fertige Software,
  2. Erweiterung des 1. Beispiels um Regeln – Programm als Sequenz, Laden, Editieren, Nutzen und Speichern des veränderten Programms, Kennenlernen der Ableitung durch Mustergleichheit,
  3. neue Faktenbasis des 2. Beispiels entwerfen, Wiederholung des Editierens und Speicherns, Anfragen mit Variablen (Name, Wert, Typ) – Variablenbindung als Selektion,

- Daten– und Algorithmenstrukturen mit Rekursiver Arbeitsweise

  4. Erweiterung des 2. Beispiels um eine rekursive Regel, Zyklus durch Rekursion (Rekursionsaufruf und Rekursionsabbruch), Praktische Realisierung der rekursiven Arbeitsweise,
  5. Entwerfen einer rekursiven Lösung für das 3. Beispiel, Wiederholung Rekursionsaufruf und –abbruch, Spur der Ableitung verfolgen – Suchprozess mit Rückkehr,
  6. Prozedurmodell entwickeln – Aufruf, Erfolg, Mißerfolg, Rückkehr, Einführen der Datenstruktur Liste – Kopf und Restliste, Anwendung der Datenstruktur Liste im 4. Beispiel
  7. Analyse des 5. Beispiels – rekursive Datenstruktur, Beobachten des rekursiven Abbaus der Liste bis zur leeren Liste, Wiederholung des Suchprozesses mit Rückkehr am 5. Beispiel,

- Komplexe Übungen

  8. Ziel: Färben einer Landkarte – Analyse des Programms, Kennenlernen der Beschreibung eines Suchraumes durch Aufzählung, Programm als Wissensbasis aus Fakten und Regeln, Wiederholung des Prozedurmodells,
  9. Ziel: Vorfahren–Relation in einen Familienstammbaum einfügen, Wiederholung der rekursiven Arbeitsweise, Programmierheuristik zur Reihenfolge der Fakten und Regeln,
  10. Ziel: Mahnung zu einer Bibliotheksdatei entwerfen, Wiederholung des Variablen– und Datenkonzepts, Zusammenfassung zum Problemlösen mit einer logischen Programmiersprache.

Beispiele

1. Freund(innen)auswahl: Als Freund(in) soll jedes Mädchen (oder jeder Junge), das (der) klug oder sportlich ist, in Frage kommen.
   Fakten:

   a) $klug(anne)$.
   b) $klug(isabel)$.
   c) $klug(piet)$.
   d) $klug(thomas)$.

e) $sportlich(anne).$

f) $sportlich(denise).$

g) $sportlich(katrin).$

h) $sportlich(piet).$

i) $sportlich(nico).$

j) $sportlich(sven).$

k) $maedchen(anne).$

l) $maedchen(isabel).$

m) $maedchen(denise).$

n) $junge(piet).$

o) $junge(sven).$

p) $junge(nico).$

Regeln:

a) $freundin(X) : -maedchen(X), klug(X).$

b) $freundin(X) : -maedchen(X), sportlich(X).$

oder

a) $freund(X) : -junge(X), klug(X).$

b) $freund(X) : -junge(X), sportlich(X).$

Der Beweisverlauf kann die Mängel der Wissensbasis nicht korrigieren.

2. Vierecksarten
Fakten:

a) $ist\_ein(trapez, konvexes\_Viereck).$

b) $ist\_ein(drachenviereck, konvexes\_Viereck).$

c) $ist\_ein(parallelogramm, trapez).$

d) $ist\_ein(rechteck, parallelogramm).$

e) $ist\_ein(rhombus, parallelogramm).$

f) $ist\_ein(rhombus, drachenviereck).$

g) $ist\_ein(quadrat, rechteck).$

h) $ist\_ein(quadrat, rhombus).$

Regeln:

a) $gehoert\_zu(X, Y) : -ist\_ein(X, Y).$

b) $gehoert\_zu(X, Y) : -ist\_ein(Z, Y), gehoert\_zu(X, Z).$

3. Computeraufbau
Fakten:

a) $teil\_von(computer, zentraleinheit).$

b) $teil\_von(computer, peripherie).$

c) $teil\_von(zentraleinheit, steuerwerk).$

d) $teil\_von(zentraleinheit, rechenwerk).$

e) $teil\_von(zentraleinheit, hauptspeicher).$

Regeln:

a) $besteht\_aus(X, Y) :- teil\_von(X, Y).$

b) $besteht\_aus(X, Y) :- teil\_von(Z, Y), besteht\_aus(X, Z).$

4. Nachbarländer
Fakten:

a) $nachbarn(andorra, [frankreich, spanien]).$

b) $nachbarn(belgien, [frankreich, luxemburg, niederlande]).$

c) $nachbarn(deutschland, [belgien, daenemark, frankreich, oesterreich, polen, schweiz, tschechien]).$

d) ...

$?- nachbarn(belgien, Nachbarlaender).$
$Nachbarlaender = [frankreich, luxemburg, niederlande]$

5. Listenelement
Fakt: $listenelement(X, [X|Restliste]).$
Regel: $listenelement(X, [Listenkopf|Restliste]) :- listenelement(X, Restliste).$

6. Färben einer Landkarte
Fakten:

a) $farbe(blau).$

b) $farbe(gelb).$

c) $farbe(gruen).$

d) $farbe(rot).$

Regeln:

a) $nachbar(L1, L2) :- farbe(L1), farbe(L2), L1 \backslash = L2.$

b) $karte(L1, L2, L3, L4, L5, L6) :- nachbar(L1, L2),$
$nachbar(L1, L3), nachbar(L1, L5), nachbar(L1, L6), nachbar(L2, L3),$
$nachbar(L2, L5), nachbar(L3, L4), nachbar(L3, L5), nachbar(L3, L6),$
$nachbar(L4, L5), nachbar(L4, L6), nachbar(L5, L6).$

c) $start :- write(\text{‘}Farbenvorschlag\ fuer\ diese\ sechs\ Laender: \text{‘}),$
$nl, write(\text{‘}Land1\quad Land2\quad Land3\quad Land4\quad Land5\quad Land6\text{‘}), nl,$
$farbe(L1), farbe(L2), farbe(L3), farbe(L4), farbe(L5), farbe(L6),$
$karte(L1, L2, L3, L4, L5, L6),$
$write(L1), write(\text{‘}\quad\text{‘}), write(L2), write(\text{‘}\quad\text{‘}), write(L3), write(\text{‘}\quad\text{‘}),$
$write(L4), write(\text{‘}\quad\text{‘}), write(L5), write(\text{‘}\quad\text{‘}), write(L6), nl, nl.$

$?- start.$
$Farbenvorschlag\ fuer\ diese\ sechs\ Laender:$

Land1	*Land2*	*Land3*	*Land4*	*Land5*	*Land6*
blau	*gelb*	*gruen*	*blau*	*rot*	*gelb*

Diese Methode der Beschreibung des Suchraumes durch Aufzählung ist nur vertretbar, wenn die Anzahl der Objekte und der Beziehungen zwischen ihnen klein bleibt. Die Anzahl der Lösungen kann dabei durchaus beachtlich groß sein.

7. Familienstammbaum
   Fakten:

   a) $elternteil(georg, margitta).$
   b) ...
   c) $maennlich(georg).$
   d) ...
   e) $weiblich(margitta).$
   f) ...

   Regeln:

   a) $vater(X, Y) : -elternteil(X, Y), maennlich(X).$
   b) $mutter(X, Y) : -elternteil(X, Y), weiblich(X).$
   c) $brueder(X, Y) : -elternteil(Z, X), elternteil(Z, Y), maennlich(X), maennlich(Y), X \setminus = Y.$
   d) $vorfahr(X, Y) : -elternteil(X, Y).$
   e) $vorfahr(X, Y) : -elternteil(X, Z), vorfahr(Z, Y).$

   $? - vater(Vater, Kind).$
   $Vater = georg$
   $Kind = margitta$

8. Bibliotheksmahnung
   Fakten:

   a) $buch(1, \text{`}Abenteuer\text{`}, autor(\text{`}Tolkien\text{`}, \text{`}J.R.R.\text{`}), \text{`}Der\ Herr\ der\ Ringe\text{`}).$
   b) ...
   c) $leser(1, \text{`}IsabelKroll\text{`}).$
   d) ...
   e) $ausleihe(1, 1, \text{`}23.02.1994\text{`}).$
   f) ...

   Regel: $mahnung(Termin, Leser, Titel) : -ausleihe(Lesernr, Buchnr, Termin), leser(Lesernr, Leser), buch(Buchnr, \_, \_, Titel).$

   $? - mahnung(\text{`}23.02.1994\text{`}, Leser, Titel).$
   $Leser = IsabelKroll$
   $Titel = Der\ Herr\ der\ Ringe$

Erläuterungen zu den zehn Doppelstunden

Einführung in die gewählte Programmierumgebung – 1. Doppelstunde

Zielorientierung mit dem E(ingabe)V(erarbeitung)A(usgabe)–Prinzip der Informatik

Der Lehrer führt in den neuen Lernbereich ein, indem er den Schülern erklärt, dass die Problemlösemethode der Informatik mit Programmen (Folge von Anweisungen) realisiert wird, die Eingaben zur Verarbeitung nutzen und daraus Ausgaben erzeugen. Bisher wurde ein fertiges Programm (Anwendungssoftware) zur Textverarbeitung genutzt. Jetzt geht es darum, eigene kleine Programme zu entwickeln. Dazu benötigt man eine spezielle Problemlösungs–Entwicklungs–Software, die Programmierumgebung. Eine solche Programmierumgebung können die Schüler in den neuen Lernbereichen kennenlernen, um mit ihrer Hilfe die eigenen Ideen in Programme zu fassen.

Starten des Systems – Kennenlernen der Benutzungsoberfläche
Die Schüler sollen das EVA–Prinzip experimentell erkunden, indem sie ein erstes kleines Programm kennenlernen und nutzen, das der Lehrer für sie vorbereitet hat. Mit dem Schlüsselwort „profix“ kann das Prologsystem gestartet werden. Die Schüler finden eine menüorientierte Benutzungsoberfläche vor, die an ihre Erfahrungen aus den Lernbereichen der Textverarbeitung anknüpfen. Im Menü „File“ wählen sie die Option „open“ und öffnen das Programm „Freundauswahl“. Neu ist die Nutzung getrennter Fenster für die Schüler. Im Menü „Window“ wählen sie die Option „tile“ und teilen damit den Bildschirm in drei Bereiche auf, die dem EVA–Prinzip entsprechen. Ganz oben sehen sie das Ausgabefenster (Output), in der Mitte das Eingabefenster (Input) und unten das Fenster mit dem aktuellen Programm (B1.PRO), das die Verarbeitung ermöglicht.

Laden und Nutzen des 1. Beispiels – Anfragen an die Faktenbasis
Während bei der Textverarbeitung das Öffnen einer Datei (File) völlig ausreichte, um diese Datei zu nutzen, lernen die Schüler jetzt eine Besonderheit der Prolog–Programmierumgebung kennen. Das Programm „B1.PRO“ muss mit dem Menü „Edit“ und der Option „consult“ für die Verarbeitung aktiviert werden. Nun können Anfragen an die Faktenbasis gestellt werden. Die Schüler lernen, das System mit Mausklick auf „Exit“ zu verlassen.

Analyse des 1. Beispiels – Lösungssuche als fertige Software
Die Analyse führt die Schüler zu der Erkenntnis, dass ein einfaches Prolog–Programm aus Fakten besteht, die das zu lösende Problem näher beschreiben, aber keinen Lösungsweg vorschreiben. Die Lösungssuche steht als fertige Software mit dem Interpreter der Prolog–Programme zur Verfügung. Mit einer Anfrage (Eingabe) kann eine Anzahl verschiedener Lösungen (Ausgabe) erzeugt werden. Fakten, die bei der Problembeschreibung vom Menschen vergessen wurden, kann die automatische Lösungssuche nicht korrigieren.

2. Doppelstunde – Erweiterung des 1. Beispiels um Regeln – Programm als Sequenz
Der Lehrer führt den Begriff Regel als „Wenn ... dann ... “–Konstruktion an Beispielen aus dem täglichen Erfahrungsbereich der Schüler ein und bildet mit den Schülern gemeinsam solche Regeln für die Freun(innen)auswahl. Die Umkehrung der Regel in der Prologschreibweise wird besprochen.

Laden, Editieren, Nutzen und Speichern des veränderten Programms
Die Schüler wiederholen ihre Kenntnisse aus der vorhergehenden Stunde, indem sie die Programmierumgebung starten, das Programm laden und die Regeln hinzufügen. Das Speichern eines Programms wird neu eingeführt (F2–Taste oder Menü „File“ mit der Option „save as“). Nach dem Aktivieren des Programms mit „consult“ stellen die Schüler erstmals Anfragen, die Regeln zur Ausführung bringen. Sie erkunden den Unterschied zwischen einer Anfrage, die vom System mit ja oder nein beantwortet wird, und einer Existenzanfrage mit einer Variablen, bei der sie verschiedene Lösungen durch die Bindung von konkreten Objekten an diese Variable erhalten.

Kennenlernen der Ableitung durch Mustergleichheit
Der Lehrer erklärt am Beispiel, wie es zur Ableitung von Lösungen durch Mustergleichheit kommt.
Anfrage: ? – $freundin(Wer)$.
Versuch mit der 1. Regel: $maedchen(Wer), klug(Wer)$.
Versuch mit dem 11. Fakt: $maedchen(anne), klug(anne)$.

Erfolgsmeldung: $Wer = anne$

Die Suche nach identischen (mustergleichen) Fakten wird von den Schülern als Schlusspunkt der erfolgreichen Lösungssuche erkannt. Die Ableitungsfolge kann grafisch als Ableitungsbaum dargestellt werden [Schubert92]. Die Schüler verstehen den einfachsten Programmaufbau als Sequenz. Die Summe der Fakten und Regeln zu einem Problem kann auch als Wissensbasis (Prolog–Programm) bezeichnet werden. Die Schüler erfahren, dass ein Prolog–Programm immer sequentiell vom Interpreter verarbeitet wird. Damit verstehen sie, dass die Fakten am Programmanfang stehen müssen, um unnötige Suchwege zu verhindern. Schrittweise wird so eine Programmierheuristik im Unterricht entwickelt, die es den Schülern ermöglicht, eigene Lösungen zu erstellen.

3. Doppelstunde – neue Faktenbasis des 2. Beispiels entwerfen

Die Einteilung der Vierecke eignet sich gut, um erstmals mit den Schülern gemeinsam eine kleine Faktenbasis zu entwerfen. Der Lehrer weist darauf hin, dass jedes Programm eine kleinen, abgeschlossenen Weltausschnitt modelliert (Modell einer abgeschlossenen Welt = Boxmodell). Der Mensch trägt die Verantwortung für die Grenzen dieses Modells. Fehlende und fehlerhafte Fakten führen mit Sicherheit zu falschen Lösungen.

Wiederholung des Editierens und Speicherns

Die Schüler erstellen dieses Programm (Faktenbasis) mit der Programmierumgebung selbständig. Mit verschiedenen Anfragen erkunden sie die Programmausführung (Schlussfolgerungsprozess) und die Variablenbindung. Sie nutzen die Möglichkeit, alle Lösungen eines Problems vom Prologsystem zu erhalten.

Anfragen mit Variablen (Name, Wert, Typ) – Variablenbindung als Selektion

Der Begriff der Variablen in der Informatik knüpft an die aus der Mathematik bekannten Kennzeichen Name und Wert an und wird um die neue Eigenschaft des Typs ergänzt. Vorerst bleibt es beim Zeichenketten–Typ. Später im Lernbereich 6 kann exemplarisch auf die Arithmetik mit den Ziffern verwiesen werden [Baumann91b]. Die Prolog–Programmierumgebung erspart den Schülern die aufwendigen Syntaxbesonderheiten anderer Programmiersprachen. Typdeklarationen sind nicht in das Programm zu schreiben, müssen aber logisch durchdacht und in den Konsequenzen verstanden werden. Nach dem sequentiellen Programmablauf lernen die Schüler nun die Selektion kennen, die durch Fakten und Regeln ermöglicht wird, die mit dem gleichen Schlüsselwort beginnen. Sie werden zum Begriff Prozedur zusammengefasst. Neu ist für die Schüler, dass die Variablenbindungen nur vorübergehend erfolgt. Gerade die Auflösung bestehender Bindungen und die Neubindung ermöglicht die alternativen Lösungswege bei der Ausführung des Prologprogramms.

Daten– und Algorithmenstrukturen mit Rekursiver Arbeitsweise

Diese beiden Lernbereiche sind bei der Wahl der logischen Programmierumgebung nicht voneinander zu trennen, da die Algorithmenstruktur Zyklus und die einfache Datenstruktur Liste auf der rekursiven Arbeitsweise beruhen. Es hat sich gezeigt, dass damit ein erstaunlich einfacher Zugang zu diesen anspruchsvollen Methoden der Informatik gefunden wird.

4. Doppelstunde – Erweiterung des 2. Beispiels um eine rekursive Regel

Die Schüler sind unzufrieden damit, dass das 2. Programm nur feststellen kann, dass ein Parallelogramm ein Trapez ist und nicht auch, dass ein Parallelogramm ein konvexes Viereck ist. Um diesen Mangel zu beheben, entsteht das Ziel eine Regel *gehoert_zu* zu formulieren, die sich selbst wieder aktivieren (aufrufen) kann, um eine ganze Zugehörigkeitskette abzuleiten.

Zyklus durch Rekursion (Rekursionsaufruf und Rekursionsabbruch)
Dazu führt der Lehrer den neuen Begriff Rekursion anschaulich ein (z. B. Spiegelbild im Spiegel) und erklärt das Kennzeichen der rekursiven Arbeitsweise, eine Regel aktiviert sich selbst wieder. Nachdem der rekursive Aufruf in der rekursiven Regel formuliert wurde, steht die Frage, wie dieser sich ständig wiederholende Prozess (Zyklus) gestoppt werden kann. Deshalb wird vom Lehrer der Begriff der Abbruchbedingung eingeführt, die zu jeder Rekursion und zu jedem Zyklus gehört.

Praktische Realisierung der rekursiven Arbeitsweise
Die Schüler gehen zur praktischen Arbeit über. Sie laden, modifizieren und speichern das 2. Programm selbständig. Das ist für sie eine gute Wiederholung im Umgang mit der Programmierumgebung. Neu ist das Nutzen einer rekursiven Problemlösung. Deshalb sollte ausreichend Zeit zur Verfügung stehen, um verschiedene Anfragen zu stellen und mit der Lösungssuche zu experimentieren.

5. Doppelstunde – Entwerfen einer rekursiven Lösung für das 3. Beispiel
Die rekursive Arbeitsweise soll an einem einfachen Beispiel geübt werden. Der grobe Computeraufbau liefert dafür die Beziehungen zwischen dem Ganzen und seinen Teilen. Die Formulierung dieser Fakten *teil_von* stellt jetzt für die SchülerInnen schon keine Schwierigkeit mehr dar. Anders verhält es sich mit dem Anwenden der Rekursion.

Wiederholung Rekursionaufruf und –abbruch
Hier ist die Hilfe des Lehrers erforderlich, um die Analogie zum 2. Programm zu erkennen. Wieder soll eine Regel mehrfach aktiv werden, um Teile, die wiederum Teile enthalten, richtig zu verbinden. Der Zyklus wird mit dem Rekursionsaufruf realisiert. Dazu gehört ebenfalls ein passender Rekursionsabbruch. Die Schüler erfahren, dass die einfachste Lösungsidee zuerst realisiert wird. So verstehen sie, warum im Prolog–Programm der Rekursionsabbruch immer vor der Regel mit dem Rekursionsaufruf stehen muss. Diese Programmierheuristik fördert die Korrektheit und Effizienz ihrer Lösungen. Nachdem die Lösung gemeinsam entworfen wurde, erfolgt die praktische Erprobung individuell von den Schülern.

Spur der Ableitung verfolgen – Suchprozess mit Rückkehr
Der Lehrer demonstriert den Aufruf der Trace–Komponente des Prolog–Systems, um damit die Spur der Ableitung experimentell zu erkunden. Der Suchprozess mit Rückkehr wird im abschließenden Unterrichterrichtsgespräch wiederholt, indem Beispiellösungen begründet werden. Die Schüler werden erstmals mit den Systemmitteilungen Aufruf (Call), Erfolg (Exit), Misserfolg (Redo) und Rückkehr (Fail) konfrontiert. Vorerst geht es aber nur um das Beobachten ausgewählter Lösungswege. Das dazugehörende Prozedurmodell (siehe Abbildung 55) mit den

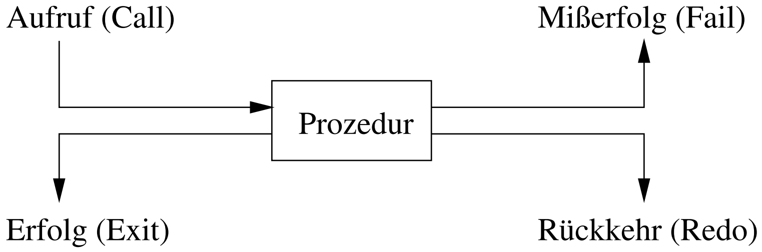


Abbildung 55: Prozedurenmodell

vier Türen wird in den folgenden Unterrichtsstunden schrittweise entwickelt. Für den weiteren Unterricht stehen jetzt die drei grundlegenden Algorithmenstrukturen Sequenz, Selektion (Alternative) und Zyklus zur Verfügung. Die Schüler werden zunehmend besser verstehen, dass alle Informatiklösungen aus diesen Bausteinen zusammengesetzt werden können.

6. Doppelstunde – Prozedurmodell entwickeln (Aufruf, Erfolg, Misserfolg, Rückkehr)
Die Tiefensuche des Prolog–Interpreters erfordert die Rückkehr von erfolglosen Wegen und die Suche nach alternativen Möglichkeiten. Die Schüler erkennen das an den vier Türen, die es für jede Prozedur (Teillösung) gibt. Der Lehrer führt nochmals die Beobachtung der letzten Stunde vor als Einstieg in die Erklärung des Modells am Beispiel.

Am Beispiel der Prozedur *teil_von* wird das Prozedurmodell vorgestellt:

Anfrage:	$teil\_von(computer, Was)$.
Aufruf der Prozedur:	$teil\_von(computer, Was)$.
Erfolg:	$teil\_von(computer, zentraleinheit)$.
	$Was = zentraleinheit$
Rückkehr:	$teil\_von(computer, Was)$.
Erfolg:	$teil\_von(computer, peripherie)$.
	$Was = peripherie$
Rückkehr:	$teil\_von(computer, Was)$.
Misserfolg:	$teil\_von(computer, Was)$.

Der Misserfolg tritt erst dann auf, wenn kein unbenutzter Fakt der Prozedur *teil_von* mit der Anfrage in Übereinstimmung gebracht werden kann. Im 3. Beispiel sind zwar noch drei unbenutzte Fakten vorhanden, aber dort steht jeweils das Objekt *zentraleinheit* an der ersten Position. Das kann nicht mit dem Anfrageobjekt *computer* zur Übereinstimmung der Muster gebracht werden. Von dieser ersten Konfrontation mit dem Prozedurmodell darf noch kein tiefes Verständnis erwartet werden. Die kontinuierliche Anwendung bei allen folgenden Beispielen bringt erst schrittweise diese Einsicht.

Einführen der Datenstruktur Liste – Kopf und Restliste
Nachdem die Algorithmenstrukturen bekannt sind, soll bei den Schülern Verständnis für die Bedeutung der Datenstruktur in der Informatik geweckt werden. In der Prolog–Programmierumgebung steht mit der Datenstruktur Liste ein anspruchsvolles Informatikkonzept mit einfachen Beschreibungsmöglichkeiten zur Verfügung. Der Lehrer kann am Beispiel einer Namensliste die neuen Begriffe Listenkopf und Restliste einführen und auf deren Schreibweise in Prolog mit dem Listenoperator eingehen.

Anwendung der Datenstruktur Liste im 4. Beispiel
Die Datenstruktur Liste kann sehr einfach zur Beschreibung von Nachbarländern eingesetzt werden. Da das Programm (4. Beispiel) nur aus Fakten besteht können sich die Schüler gut auf die Datenstruktur konzentrieren. Das Editieren, Speichern und Nutzen des Programms bereitet keine Schwierigkeiten.

7. Doppelstunde – Analyse des 5. Beispiels – rekursive Datenstruktur
Die Kenntnisse über die Datenstruktur Liste werden vertieft, indem das kleine, aber gehaltvolle Programm *listenelement* gemeinsam mit den Schülern analysiert wird. Es besteht aus einem Fakt, dem Rekursionsabbruch, und einer einzigen Regel mit dem rekursiven Aufruf. Das Wissen über die Rekursion wird wiederholt. Der Lehrer führt mit der Trace–Komponente die Wirkung des Programms vor und vereinfacht diese Informationsfülle über die Ableitungsspur zu folgendem Ableitungsbaum:

Beobachten des rekursiven Abbaus der Liste bis zur leeren Liste
Nachdem die Schüler den rekursiven Abbau der Liste bis zur leeren Liste durch die Demonstration des Lehrers und das analysierende Unterrichtsgespräch kennenlernten, wird die praktische Nutzung des Programms für beliebige Listen zur Festigung dieses neuen Stoffes eingesetzt.

Wiederholung des Prozedurmodells In der praktischen Arbeit mit der Programmierumgebung wird die Trace–Komponente von den Schülern selbständig aktiviert und damit das Prozedurmodell für die Kontrolle des Lösungsweges bewusst eingesetzt. Erste Versuche der Schüler im mündlichen argumentieren mit den neuen Begriffen Aufruf, Erfolg, Misserfolg und Rückkehr schließen den Lernbereich 5 ab.

Komplexe Übungen
Es mag etwas vermessen klingen, wenn Anfänger nach sieben einführenden Doppelstunden zum Problemlösen mit Informatikmethoden bereits komplexe Übungen lösen sollen. Im Vergleich zum Ausgangsniveau wurde jedoch so viel neuer Stoff geboten, dass Üben und Wiederholen der Einzelheiten im Zusammenhang geboten ist.

8. Doppelstunde – Ziel: Färben einer Landkarte – Analyse des Programms
Dieses Beispiel ähnelt einer Knobelaufgabe und hat den Vorteil, dass die Schüler die Lösung nicht sofort kennen. Noch anspruchsvoller wird die Frage nach allen möglichen Lösungen. Der Vorteil der Informatikanwendung wird hier erstmals so richtig deutlich. Das gelingt in anderen Programmierumgebungen mit Anfängerlösungen selten. Der Schwerpunkt wird jetzt auf die Analyse der Aufgabenstellung (Eingabe– und Ausgabewerte) und den Grobentwurf (Problembeschreibung) gelegt. Die fertige Lösung stellt der Lehrer danach zum Probieren und Experimentieren bereit.

Kennenlernen der Beschreibung eines Suchraumes durch Aufzählung
Diese grundlegende Problemlösemethode ist für die Schüler neu. Sie sollte deshalb in Ruhe genutzt und auf ihre Vor– und Nachteile hin untersucht werden. Die Schüler sollen erkennen, dass die Landkarte in vereinfachter Form in der Programmiersprache darzustellen ist.
Dazu werden die Nachbarschaftsbeziehungen aufgezählt.

Programm als Wissensbasis aus Fakten und Regeln
Neben der Wiederholung des Grundaufbaus einer solchen Wissensbasis, wird die Ausgabeprozedur *write* neu eingeführt. Die Schüler unterscheiden zwischen der Textausgabe in Hochkommas und der Wertausgabe für die Variablen. Hier bieten sich Ansatzmöglichkeiten zur Modifikation des fertigen Programms, indem neue Ausgabemöglichkeiten erprobt werden. In leistungsstarken Gruppen kann auf die Art der Lösungserzeugung durch Generieren einer Farbbelegung und deren Testen auf Verträglichkeit eingegangen werden. An diesem Beispiel lassen sich die zeitweiligen Variablenbindungen mit der Trace–Komponente besonders gut kontrollieren. Die Schüler beobachten auch erstmals, dass sie auf die Ergebnisse warten müssen, weil das System längere Suchzeiten benötigt.

9. Doppelstunde – Ziel: Vorfahrenrelation in einen Familienstammbaum einfügen
Zur Wiederholung und Zusammenfassung aller gelernten Elemente eignet sich besonderes das Beispiel Familienstammbaum. Es fehlt in keinem Anfängerlehrbuch. Der Vorteil besteht darin, dass der Lehrer die Objekte und Beziehungen sofort bei den Schülern als bekannt voraussetzten kann und trotzdem genügend anspruchsvolle Informatikelemnte üben kann. Der Lehrer stellt ein Rahmenprogramm zur Verfügung, in das die Schüler nur noch die von ihnen gewünschten Namen einzutragen brauchen. Als Übung werden die Vater–, Mutter– und Geschwisterrelation im Unterrichtsgespräch zusammengetragen.

Wiederholung der rekursiven Arbeitsweise
Mit der neuen Zielstellung, nicht nur die Eltern, sondern auch frühere Vorfahren mit einer Regel zu ermitteln, soll die Analogie zu den bereits gelösten Beispielen hergestellt werden. Wieder wird der Rekursionsabbruch und der Rekursionsaufruf formuliert. Jetzt müssten diese rekursiven Programmelemente bereits von den Schülern gefunden werden.

Programmierheuristik zur Reihenfolge der Fakten und Regeln
Die Schüler ergänzen den Programmrahmen nach eigenen Vorstellungen und nutzen das vollständige Programm. Im Umgang mit der Programmierumgebung sollte sich eine gewisse Sicherheit eingestellt haben. Dieses Beispiel eignet sich auch für eine Leistungskontrolle, die aus einem praktischen und einem theoretischen Teil bestehen könnte. Der praktische Teil ergibt sich aus der Vervollständigung des lückenhaften Programms. Für den theoretischen Teil bieten sich Fragen nach dem Programmaufbau und dem heuristischen Wissen nach der Reihenfolge der Fakten und Regeln an.

10. Doppelstunde – Ziel: Mahnung zu einer Bibliotheksdatei entwerfen
Diese Aufgabenstellung kann der Auftakt zu einem interessanten fakultativem Informatikunterricht in den folgenden Schuljahren sein. Hier wird ein praktisches Anwendungsfeld der Informatik berührt, das natürlich in dieser kurzen Einführungsphase nicht zufriedenstellend bearbeitet werden kann. Die Bibliotheksverwaltung, zum Beispiel der Schülerbibliothek, wird von den Schülern als echtes Problem verstanden. Also greift der Lehrer exemplarisch eine Teilaufgabe heraus, die mit den Möglichkeiten der Anfänger lösbar ist.

Wiederholung des Variablen– und Datenkonzepts
Die Zusammenstellung geeigneter Fakten wie „buch"„leser" und „ausleihe" bereitet den Schülern keine Schwierigkeiten. Der Lehrer kann in leistungstsarken Gruppen die zusammengesetzten Datenstrukturen neu einführen, da sie in Prolog erstaunlich leicht zu beschreiben sind. Für jedes Buch bietet sich eine Registriernummer, der Autor mit Name und Vorname und der Titel an. Gemeinsam wird die Regel für eine Mahnung entworfen. Hier kann der Lehrer die Schüler mit der anonymen Variable als Platzhalter für uninteressante Informationen bekannt machen. Die Nutzung dieses Programms kann nur der zaghafte Beginn für eine umfassendere Bibliothekslösung sein. Aber die Schüler werden so zum selbständigen Weiterlernen auf dem Gebiet der Informatik angeregt Beim Laden, Ergänzen, Speichern und Nutzen des Teilprogramms sollten keine Schwierigkeiten mehr auftreten.

Zusammenfassung zum Problemlösen mit einer logischen Programmiersprache
Den Abschluss dieser Informatikeinführung kann eine kurze Zusammenfassung der Besonderheiten des Problemlösens in Prolog bilden:

- Programm ist Wissensbasis aus Fakten und Regeln.
- Eingaben werden als Anfragen formuliert.
- Ausgaben liefert das System in Form von Antworten.
- Zur Lösungssuche wird ein Prologinterpreter verwendet.
- Sequenzen entstehen durch die Reihenfolge der Fakten und Regeln.
- Selektionen entstehen durch die verschiedenen Fakten und Regeln mit dem gleichen Namen.
- Zyklen entstehen durch Rekursionsaufruf und Rekursionsabbruch.

H. Bewertung von Informatiksystemen

H.1. Benutzungsfreundlichkeit

Das Informatiksystem soll seine Reaktionen nach den Denk– und Arbeitsweisen der Benutzer richten. Dazu müssen bereits bei der Entwicklung des Systems bekannt sein:

- der Benutzerkreis (Fähigkeiten, Fertigkeiten, Nutzungsart),
- die Arbeitsaufgaben (Inhalte, Komplexität, Struktur).

So können Funktionalität und Benutzungsoberfläche genau spezifiziert werden. Der Nutzer soll sich auf seine Aufgabe konzentrieren können. Das Werzeug „Informatiksystem" soll dabei in den Hintergrund treten und keine zusätzliche Belastung darstellen.

H.1.1. Problemangemessenheit

Dazu gehört, dass die Funktionalität geprüft wird.

H.1.2. Dialogflexibilität

Die Steuerung und Aktivität sollte prinzipiell beim Nutzer liegen. Das System hat angemessene Rückmeldungen bereitzustellen.

- Menüs können auch sehr lästig sein. Deshalb sollte der Nutzer Voreinstellungen und Sprünge vornehmen können.
- Der Zeitbedarf ist zu überprüfen und
- die Einbettung in den Gesamtprozess der Anwendung.

H.1.3. Selbsterklärungsfähigkeit (Transparenz)

Die Verständlichkeit ist zu überprüfen. Mit Farben können aktive von inaktiven Fenstern unterschieden werden. Jeder Zustandwechsel sollte von einem Farbwechsel begleitet sein.

H.1.4. Zuverlässigkeit

Es können erwartet werden:

- Fehlertoleranz: Das System soll auch dann noch korrekt arbeiten, wenn Komponenten fehlerhaft sind.
- Der Nutzer kann Fehler rückgängig machen.
- Warnungen.

H.1.5. Erlernbarkeit

Es soll eine einfache Nutzung ohne tiefere Informatikkenntnisse möglich sein.

1. Hilfen
   Hier erwartet man:

- Hilfe–Funktionen: Informationen über den Modus (Zustand) des Systems,
- ausagekräftige Fehlermeldungen,
- Korrekturvorschläge.

2. Erwartungskonformität
   Sie wird verbessert, wenn einheitliche Syntax für anwendungsunabhängige Funktionen eingesetzt wird.

H.2. Vorteile der Anwendung

Ein Vergleich mit der traditionellen Arbeitweise lässt Vor– und Nachteile erkennen. Untersuchungen zur Effizienz der Lösungsalgorithmen gehören in dazu.

H.3. Komplexität

H.3.1. Komplexitätsmaße

1. Zeit
   Die Zeitkomplexität gibt die Anzahl der Rechenschritte eines Maschinenmodells an.
2. Platz (Raum, Speicher)
   Die Platzkomplexität gibt die Anzahl der benötigten Speicherzellen eines Maschinenmodells an.
3. Schranke
   Von besonderem Interesse sind Komplexitätsschranken, z.B.:
   - konstante Komplexität,
   - logarithmische Komplexität,
   - lineare Komplexität,
   - polynomiale Komplexität,
   - exponentielle Komplexität.
4. Zeit–Platz–Relation
   Jedes Informatiksystem stellt einen Kompromiss zwischen Zeit– und Platzbedarf dar. Was man an Zeit einspart, setzt man an Platz wieder zu.

H.3.2. Komplexitätsklassen

Alle existierenden Probleme lassen sich in Komplexitätsklassen einordnen. Mit der Methode der Diagonalisierung kann man zeigen, dass es keine maximalen Komplexitätsklassen gibt. Auch das Halteproblem wird mit dieser Methode bewiesen.

1. Die Klasse P
   In diese Klasse gehören alle Probleme, die deterministisch lösbar sind mit polynomial–beschränktem Zeitaufwand. Einfacher gesagt sind das alle praktisch lösbaren Aufgaben.
2. Die Klasse NP
   In diese Klasse gehören alle Probleme, die nichtdeterministisch lösbar sind (mit Raten) mit polynomial–beschränktem Zeitaufwand. Einfacher gesagt sind das alle praktisch unlösbaren Aufgaben, denn das Raten kann nur durch vollständiges Durchmustern realisiert werden. Es existieren aber Informatiksysteme dafür, die Heuristiken enthalten, d.h. die Lösung wird nicht für jeden Fall gefunden.

Reduktion und Kompression
Für das Komplexitätsmaß „Platz“ existieren Überlegungen zur Reduktion der Speicher und deren Kompression in Maschinenmodellen. **Beschleunigung**
Für das Komplexitätsmaß „Zeit“ existieren Überlegungen zur Reduktion durch Beschleunigung der Maschinenmodelle.

H.4. Korrektheit – Programmverifikation

Gesucht sind Verfahren, mit denen man die Korrektheit eines Informatiksystems oder bescheidener ausgedrückt eines Programms beweisen kann. Die Analogie dazu ist der Beweis eines mathematischen Satzes.
Korrektheit liegt vor, wenn aus der Eingabespezifikation tatsächlich die und nur die gewünschte Ausgabespezifikation folgt.
Daraus lassen sich zwei Teilaufgaben bilden:

- Nachweis der korrekten Berechnung aller Ausgabewerte,
- Nachweis der Terminierung für alle zulässigen Eingabewerte.

Die Programmverifikation ist schwieriger und fehleranfälliger als das Programmieren selbst.

H.4.1. Konsistenz

Das System selbst und die von ihm verarbeiteten Informationen sollen keine Widersprüche enthalten. Weltausschnitt (Modell) und Realität sollten bezüglich der zu lösenden Aufgaben gut harmonieren.

H.4.2. Vollständigkeit

Die Vollständigkeit einer Spezifikation ist nachzuweisen. Es tritt auch Überspezifikation auf, wenn an die Software unnötig einschränkende Anforderungen gestellt werden.

H.4.3. Gerechtigkeit (fairness)

Die Zuteilung von Betriebsmitteln erfolgt so, dass mehrere Informatiksysteme parallel oder verzahnt (zeitlich ineinander gefügt) aktiv werden können.

H.4.4. Softwaretechnik

Die Korrektheit großer Informatiksysteme soll mit besonderen Prinzipien, Methoden und Werkzeugen für ihre Entwicklung gesichert werden. Dahinter verbirgt sich die Hypothese, dass die richtige Arbeitsweise, in diesem Fall eine ingenieurmäßige, Fehler von vornherein verhindern kann. Zu diesem Zweck wird der Entwicklungsprozess in Phasen unterteilt:

- Problemanalyse und Anforderungsdefinition,
- Entwurf und Programmentwicklung,
- Programmierung und Implementierung,
- Test,

- Wartung.

Ziel ist die Qualitätssicherung.

H.4.5. Test – Suche nach Fehlern

Testen beweist nicht die Korrektheit, sondern nur die Anwesenheit oder Abwesenheit bestimmter Fehler. Das Testen wird nicht überflüssig durch eine Verifikation, da zwischen Quelltext und Maschinenprogramm durchaus semantische Unterschiede bestehen können.

H.4.6. Terminierung

Terminierung bedingter Schleifen (Iterationen) untersuchen durch formale Methode:

- Wählen einer Funktion (Heuristik erforderlich),
- Reduktion des Funktionswertes je Schleifendurchlauf zeigen (streng monotones Fallen der Funktion).

H.4.7. Partielle Korrektheit

Methode:

- Zerlegung des Programms in überschaubare Teile,
- Verwendung von Regeln der axiomatischen Semantik (Prädikatensematik): Prädikate als Vorbedingung und Nachbedingung (Zusicherungen) ,
- Konstruktion einer Schleifeninvarianten.

H.5. Sicherheit

H.5.1. Vertraulichkeit

Verlust der Vertraulichkeit führt zu unbefugtem Informationsgewinn. Das kann etwa durch Verschlüsselung verhindert werden.
Es existieren mathematische Funktionen, die leicht zu berechnen sind (polynomialer Aufwand), deren Umkehrungen aber nur mit unvorstellbar großem Aufwand zu berechnen sind. Diese Funktionen nennt man Einwegfunktionen. Einwegfunktionen wurden bereits in den sechziger Jahren benutzt, um Passwortdateien zu verschlüsseln. Es existiert kein mathematischer Beweis dafür, dass Funktionen diese Einwegeigenschaft besitzen.
Beispiel:
Verlust der Vertraulichkeit durch Passwortdiebstahl und Zugriff auf unverschlüsselte Informationen. **Authentizität**
Authentifikation setzt sich aus zwei Teilprozessen zusammen, der Identifikation und der Verifikation der Identität.

- Authentifikation auf der Basis von Wissen (z.B. Passwort),
- Authentifikation auf der Basis von Besitz (z.B. Chipkarte),

- Authentifikation auf der Basis biometrischer Eigenschaften: Es werden persönliche Eigenschaften oder Merkmale eines Menschen zu seiner Erkennung genutzt, z.B. Fingerabdruck oder Dynamik der Unterschrift.

Elektronische Unterschrift
Unter der Gefahr der Maskerade versteht man das Vortäuschen einer falschen Identität. Mit elektronischer Unterschrift kann man das verhindern, wenn sichergestellt wird, dass diese Unterschrift nicht gefälscht werden kann. Die Idee geht auf Diffie und Hellman zurück. Wie bei jeder Unterschrift so sind auch hier zwei Anforderungen zu erfüllen:

- Nur genau eine Person kann die Unterschrift erzeugen.
- Alle anderen Personen sind in der Lage, die Korrektheit der Unterschrift zu überprüfen.

Das RSA–Verfahren, ein asymmetrisches Kryptosystem, erfüllt diese Anforderungen.
Verlust der Verbindlichkeit:
Einer der Partner könnte nachträglich die Teilnahme oder die Korrektheit eines Informationsaustausches bestreiten. Wie können Geschäftsvorgänge elektronisch unter Beibehaltung der Verbindlichkeit beweisfähig abgewickelt werden?
Zugriffsschutz
Eine Sicherheitsstrategie legt fest, welche Subjekte bestimmte Operationen mit bestimmten Objekten ausführen dürfen. Diese Rechte und Verbote müssen kontrolliert werden. Der Referenz–Monitor ist der Teil des Betriebssystems, der die Zugriffe von Subjekten auf Objekte überwacht und anhand einer Datenbank, in der die Zugriffsrechte gespeichert sind, entscheidet, ob sie zulässig sind oder nicht.

- Rechteverwaltung und Rechteprüfung, Isolation,
- Subjekte: handelnde Einheiten (z.B. Benutzer, Programme, Prozesse),
- Objekte: behandelte Einheiten (z.B. Datenfelder, Dateien, Programme),
- Operationen: Lesen, Schreiben, Erzeugen, Verändern, Löschen, Ausführen,
- Angriffe: alle unbefugten Operationen, Raubkopieren, Zeitdiebstahl,

Der Referenz–Monitor muss folgende drei Eigenschaften besitzen:

- Vollständigkeit: Es darf nicht möglich sein, den Referenz–Monitor zu umgehen.
- Unveränderbarkeit: Damit werden alle Manipulationen von außen ausgeschlossen.
- Korrektheit: Es wird auf formale Verifikation Wert gelegt. Deshalb steht beim Entwurf die Einfachheit im Vordergrund.

Modifikationen der Entscheidungs–Datenbank werden vom Referenz–Monitor kontrolliert. Außerdem protokolliert er alle sicherheitsrelevanten Ereignisse in einer speziellen Datei zu Zwecken der Beweissicherung.
Ein spezielles „Computerstrafrecht“ soll die genannten Maßnahmen durch seine Präventivwirkung unterstützen.

H.5.2. Integrität

Verlust der Integrität bedeutet unbefugte Modifikation von Daten. Solche Fälschungen erkennt man durch Prüfsummen (zusätzliche Informationen über den Sender) oder Verschlüsselung.
Beispiel:
Verlust der Integrität durch Fälschung einer Information.

H.5.3. Verfügbarkeit

Verlust der Verfügbarkeit äußert sich in unbefugter Beeinträchtigung der Funktionalität. Relevante Attacken können etwa durch Authentifikation (Nachweis der Identität und digitale Unterschrift) eingeschränkt werden.
Beispiel:
Verlust der Verfügbarkeit durch Viren.

H.5.4. Anonymität

Verlust der Anonymität:

- des Senders – Broadcasting (Nachricht an viele Teilnehmer),
- des Empfängers – Pseudonyme,
- der Kommunikationsbeziehung – Rauschen.

Sicherheitsrelevante Daten werden aufgezeichnet. Dadurch geht die Anonymität verloren. Es kann stets nachvollzogen werden: Wer hat wann von wo mit welchen Mitteln was veranlasst und worauf zugegriffen?

H.5.5. Originalität

Verlust der Originalität – Kopieren:
Hier ist weitgehend ungelöst, wie man Kopien von Originalen unterscheiden kann. Die Raubkopien von Software sind ein Beispiel dafür.

H.6. Möglichkeiten für Wartung und Anpassung

Hier können alle Vorteile der strukturierten Zerlegung bzw. des strukturierten Aufbaus eines Informatiksystems einbezogen werden.

H.6.1. Darstellungen der Algorithmen

Um die Entwurfs– und Programmierkonzepte nachvollziehen und verstehen zu können, ist man auf eine Beschreibung der Algorithmen angewiesen:

- Struktogramme,
- Pseydocode,
- Programmiersprache.

Die Kompliziertheit der Lösung hängt davon ab, ob das gewählte Programmierparadigma gut zur Promblemklasse passt. Die Sprachen unterstützen logische, funktionale und objektorientierte Denkweisen unterschiedlich gut.

H.6.2. Beschreibung der Datenstrukturen

Ziel ist das einfache Aktualisieren und Modifizieren von Datenbeständen. Unterstützt kann das werden durch:

- Verständlichkeit der Auswahlkriterien,
- Selbsterklärungsfähigkeit der Bezeichner,
- problemadäquate Datenmodellierung für Dateien und Datenbanken.

H.6.3. Modularisierung

Ziel ist die Übersichtlichkeit der Lösung und Wiederverwendbarkeit der Lösungsbausteine. Methoden:

- Top–down–Methode,
- Bottom–up–Methode,
- Geheimnisprinzip: Das Geheimnisprinzip ermöglicht die klar definierten Schnittstellen.

Hilfsmittel:

- Prozeduren,
- Lokalität von Objekten,
- Spezifikation,
- abstrakter Datentyp,
- Teamarbeit.

H.6.4. Strukturarten

Das Verständnis für ein Informatiksystem hängt entscheidend von dessen Strukturierung ab. Man spricht auch von der Art der Wissensrepräsentation. Zunehmend treten die einzelnen Strukturdarstellungen nicht mehr in Reinform, sondern gemischt auf (hybride Systeme), um die spezifischen Vorteile zu kombinieren:

- Hierarchie:
  - Schachtelung,
  - Baum,
  - Klammerung,
  - Einrückung.
- Netzwerk,
- Regelwerk.

Realisierung:

- Speicherung,
- Übersetzung,
- Interpretation,
- operationale Erweiterung.

H.6.5. Softwaretechnik

Die Entwicklung großer Informatiksysteme erfordert eine besondere Vorgehensweise. Vorteile:

- Einschränken der logischen Komplexität,
- Einsatz von Software–Werkzeugen und Programmierumgebungen (Editoren, Compiler, ...),
- Entwurfsmethode des Prototyping: Entwicklung eines Prototypen mit geringerem Aufwand zum Sammeln von Erfahrungen,
- Existenz und Qualität der Dokumentation (aktuell und konsistent).

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[AKHK+90] ALTERMANN-KÖSTER, M. ; HOLTAPPELS, H. G. ; KANDERS, M. ; PFEIFFER, H. ; WITT, C. de: *Bildung über Computer?*. Juventa Verlag, Weinheim, 1990

[Ambros92] AMBROS, Wolfgang: Das Informatikprojekt - oder: Die Angst vor dem Chaos. In: *Schulcomputerjahrbuch '93/94* (siehe [Bosler92]), S. 189–203

[Anderson96] ANDERSON, J.R.: *Kognitive Psychologie*. Spektrum der Wissenschaften, Heidelberg, 1996

[Arlt87] ARLT, W. u.: Empfehlungen zur Lehrerbildung im Bereich der Informatik. In: *LOGIN* 7 (1987), Nr. 5/6. – Beilage

[Baumann86] BAUMANN, Rüdeger: Leistungsmessung im Informatikunterricht. In: *LOGIN* 6 (1986), Nr. 4, S. 11–19

[Baumann91a] BAUMANN, R.: Mündliche Abiturprüfung im Fach Informatik. In: *LOGIN* 11 (1991), Nr. 1/2, S. 10–18

[Baumann91b] BAUMANN, Rüdeger: *Prolog. Einführungskurs*. Klett-Schulbuchverlag, Stuttgart, 1991

[Baumann93] BAUMANN, Rüdeger: Ziele und Inhalte des Informatikunterrichts. In: *Zentralblatt für Didaktik der Mathematik* 25 (1993), Nr. 1, S. 9–19

[Baumann96] BAUMANN, Rüdeger: *Didaktik der Informatik*. 2. völlig neu bearbeitete Aufl. Klett-Schulbuchverlag, Stuttgart, 1996

[BB95] BRAUER, Wilfried ; BRAUER, U.: *Informatik - das neue Paradigma (Änderung von Forschungszielen und Denkgewohnheiten der Informatik)*. In: *LOGIN* 15 (1995), Nr. 4, S. 25–29

[BH87] BUSSMANN, H. ; HEYMANN, H. W.: *Computer und Allgemeinbildung*. In: *Neue Sammlung* 27 (1987), Nr. 1, S. 2–39

[BJR96] BOOCH, G. ; JACOBSON, I. ; RUMBAUGH, J.: *The Unified Modeling Language for Object-Oriented Development, Documentation Set*. Rational Software Corporation, 1996

[BLK84] BUND-LÄNDER-KOMMISSION FÜR BILDUNGSPLANUNG UND FORSCHUNGSFÖRDERUNG (BLK) (Hrsg.): *Rahmenkonzept für die Informationstechnische Bildung in Schule und Ausbildung*. BLK, Bonn, 1984

[BLK87] BUND-LÄNDER-KOMMISSION FÜR BILDUNGSPLANUNG UND FORSCHUNGSFÖRDERUNG (BLK) (Hrsg.): *Gesamtkonzept für die Informationstechnische Bildung in Schule und Ausbildung*. BLK, Bonn, 1987. – Heft 16

[Bloom76] BLOOM, B. S.: *Taxonomie von Lernzielen im kognitiven Bereich*. Beltz Verlag, Weinheim, 1976

[Bosler92] BOSLER, Ulrich u. (Hrsg.): *Schulcomputerjahrbuch '93/94*. Metzler Schulbuch und B.G. Teubner, Hannover, Stuttgart, 1992

[BP91] BURKERT, J. ; PESCHKE, R. ; HESSISCHES INSTITUT FÜR BILDUNGSPLANUNG UND SCHULENTWICKLUNG (HIBS) (Hrsg.): *Weiterentwicklung des Informatikunterrichts - Folgerungen aus der Sicht von Lehrerbildung und Wissenschaft*. Bd. 16. HIBS, Wiesbaden, 1991

[BR95] Booch, G. ; Rumbaugh, J.: *Unified Method, Documentation Set*. Rational Software Corporation, 1995

[Bratko87] Bratko, I.: *Prolog. Programmierung für die Künstliche Intelligenz*. Addison-Wesley, Reading (Massachustts), 1987

[Brauer76] Brauer, Wilfried u.: *Zielsetzungen und Inhalte des Informatikunterrichts*. In: *Zentralblatt für Didaktik der Mathematik, Stuttgart* 8 (1976), Nr. 1, S. 35–43

[Bundestag97] Bundestag: *Informations- und Kommunikationsdienste-Gesetz (IuKDG)*. In: *Bundesgesetzblatt I* (1997), S. 1870–1880

[Buse80] Buse, D.: Informatik. In: Roth, L. (Hrsg.): *Handlexikon der Didaktik der Schulfächer*. Ehrenwirth, München, 1980, S. 256–264

[BV91] Bombei, K. D. ; Völzke, R.: *Klausuren und mündliches Abitur im Informatikunterricht*. In: *LOGIN* 11 (1991), Nr. 1/2, S. 19–30

[CH95] Crutzen, C. K. M. ; Hein, H.-W.: *Objektorientiertes Denken als didaktische Basis der Informatik*. In: *Innovative Konzepte für die Ausbildung* (siehe [Schubert95a]), S. 149–158. – Tagung „Informatik und Schule“ des Fachbereiches 7 „Ausbildung und Beruf“ - Chemnitz

[Charwat94] Charwat, H. J.: *Lexikon der Mensch-Maschine-Kommunikation*. Oldenbourg Verlag, München, 1994

[Claus79] Claus, Volker u.a: *Empfehlungen zur Ausbildung, Fortbildung und Weiterbildung von Lehrkräften für das Lehramt Informatik für die Sekundarstufe I und II*. In: *Informatik-Spektrum* 2 (1979), Nr. 2, S. 53–60

[Claus90] Claus, Volker: Perspektiven der Informatik. In: *LOGIN* 10 (1990), Nr. 6, S. 43–47

[Claus91] Claus, Volker: Die Rolle der Sprache - Anforderungen an den Informatikunterricht. In: *Weiterentwicklung des Informatikunterrichts - Folgerungen aus der Sicht von Lehrerbildung und Wissenschaft* (siehe [BP91]), S. 148–158

[Claussen93a] Claussen, U. (Hrsg.): *Objektorientiertes Programmieren*. Springer–Verlag Berlin, 1993

[Claussen93b] Claussen, Ute: *Objektorientiertes Programmieren*. Springer-Verlag, Berlin, 1993

[CM87] Clocksin, W.F. (Hrsg.) ; Mellish, C.S. (Hrsg.): *Programming in Prolog*. Springer–Verlag Berlin - Heidelberg, 1987

[CS93] Claus, Volker ; Schwill, Andreas: *Duden Informatik*. Dudenverlag, Mannheim, 1993

[CS97] Claus, Volker ; Schwill, Andreas: *Schülerduden Informatik*. Dudenverlag, Mannheim, 1997

[Czischke97] Czischke, Jürgen: *OOP - von Anfang an*. In: *Informatik betrifft uns* (1997), Nr. 1, S. 16–42

[Diener98] Diener, U. u.: *Neue Medien im Unterricht - Vorbild USA?*. Verlag Bertelsmann Stiftung, Gütersloh, 1998

[Dosch93] Dosch, W.: *Programmiersprachen - Grundlagen und Entwicklungen.* In: *Tagungsband zum Colloquium „Programmiersprachen im Unterricht“* (siehe [Keidel93]), S. 11–38

[Drosdowski89] Drosdowski, G. (Hrsg.): *Deutsches Universalwörterbuch*. Dudenverlag, Mannheim, 1989

[Fischer84] Fischer, R.: *Unterricht als Prozeßvon der Befreiung vom Gegenstand - Vision eines neuen Mathematikunterrichts.* In: *Jahrbuch für Mathematik-Didaktik*, 1984, S. 51–85

[FS97] Fothe, M. ; Schubert, Sigrid: *Informatikcurriculum und Technologieentwicklung.* In: *Informatik und Lernen in der Informationsgesellschaft* (siehe [HL97]), S. 216–219. – Tagung „Informatik und Schule“ des Fachbereiches 7 „Ausbildung und Beruf“ - Duisburg

[GIDDR88] Gesellschaft für Informatik der DDR (Hrsg.): *Tagungsband „Computer im Bildungswesen COMBI 88“* Universität Leipzig, 1988

[GH91] Göhner, H. ; Hafenbrak, B.: *Arbeitsbuch Prolog*. Dümmler, Bonn, 1991

[GLSS92] Gasper, Friedrich ; Lei, Ina ; Spengler, Mario ; Stimm, Hermann: *Technische und theoretische Informatik*. 1. Aufl. Bayerischer Schulbuch-Verlag, München, 1992

[Gorny88] Gorny, P.: *Didaktische Ansätze für die informatische Grundbildung im internationalen Vergleich.* In: *Tagungsband „Computer im Bildungswesen COMBI 88“* (siehe [GIDDR88]), S. 44–53

[Gorny91] Gorny, P.: *Anforderungen an die Lehrerbildung Informatik.* In: *Weiterentwicklung des Informatikunterrichts - Folgerungen aus der Sicht von Lehrerbildung und Wissenschaft* (siehe [BP91]), S. 139–147a

[Halsall95] Halsall, F.: *Data Communications, Computer Networks and Open Systems*. Addison-Wesley, Bonn, 1995

[HB96] Hubwieser, Peter ; Broy, Manfred: *Der informationszentrierte Ansatz - Ein Vorschlag für eine zeitgemäe Form des Informatikunterrichtes am Gymnasium.* Technische Universität München - Fakultät für Informatik. 1996 (TUM-I9624). Forschungsbericht

[HB97] Hubwieser, Peter ; Broy, Manfred: *Grundlegende Konzepte von Informations– und Kommunikationssystemen für den Informatikunterricht.* In: *Informatik und Lernen in der Informationsgesellschaft* (siehe [HL97]), S. 40–50. – Tagung „Informatik und Schule“ des Fachbereiches 7 „Ausbildung und Beruf“ - Duisburg

[HJS82] Hui, E. ; Jung, C. ; Schmid, M.: *Pascal, Informatik in 24 Stunden*. Diesterweg Verlag, Frankfurt a.M., 1982

[HL97] Hoppe, H. U. (Hrsg.) ; Luther, W. J. (Hrsg.): *Informatik und Lernen in der Informationsgesellschaft*. Springer-Verlag, Berlin Heidelberg, September 1997 (Informatik aktuell). – Tagung „Informatik und Schule“ des Fachbereiches 7 „Ausbildung und Beruf“ - Duisburg

[HT94] Hauf-Tulodziecki, Annemarie: *Informatik in der Sekundarstufe II als Teil der informationstechnischen Bildung.* In: *Computer und Unterricht* 4 (1994), Nr. 13, S. 58–62

[HW82] Haas, Hans W. (Hrsg.) ; Wildenberg, Detlef (Hrsg.): *Informatik für Lehrer*. Bd. 1, 2. Oldenbourg Verlag, München, 1982. – Bd. 1 Einführung in die Schulinformatik; Bd2. Komplexere Probleme und Didaktik der Schulinformatik

[KBM78] Krathwohl, D. R. (Hrsg.) ; Bloom, B. S. (Hrsg.) ; Masia, B. B. (Hrsg.): *Taxonomie von Lernzielen im affektiven Bereich*. Beltz Verlag, Weinheim, 1978

[Keidel93] Keidel, K. (Hrsg.): *Tagungsband zum Colloquium „Programmiersprachen im Unterricht“*. Zentralstelle für Computer im Unterricht, Augsburg, 1993

[Klafki91] Klafki, Wolfgang: *Neue Studien zur Bildungstheorie und Didaktik — Zeitgemäše Allgemeinbildung und kritisch–konstruktive Didaktik*. Beltz Verlag, Weinheim, 1991

[KMK91] Ständige Konferenz der Kultusminister der Bundesrepublik Deutschland (KMK) (Hrsg.),: *Einheitliche Prüfungsanforderungen in der Abiturprüfung „Informatik“*. Luchterhand, Neuwied, 1991. – lt. Burkert1994 Veröffentlichung: 1992

[KMK94] Ständige Konferenz der Kultusminister der Bundesrepublik Deutschland (KMK): *Zu Fragen der Gleichwertigkeit von allgemeiner und beruflicher Bildung, Beschlußvom 02.12.1994*. In: *Forschung & Lehre* 2 (1995), Nr. 2, S. 68. – gekürzt

[KMNW84] Kultusministerium des Landes Nordrhein-Westfalen (Hrsg.): *Materialien zur Leistungsbewertung in den Fächern der gymnasialen Oberstufe (Bewertung von Klausuren): Informatik*. Verlagsgesellschaft Ritterbach, Frechen, 1984

[KMNW91] Kultusministerium des Landes Nordrhein-Westfalen (Hrsg.): *Die Schule in Nordrhein-Westfalen; Vorläufige Richtlinien Leistungskurse Informatik*. Verlagsgesellschaft Ritterbach, Frechen, August 1991 (Die Schule in Nordrhein-Westfalen)

[KMNW93] Kultusministerium des Landes Nordrhein-Westfalen (Hrsg.): *Richtlinien und Lehrpläne Informatik Gymnasium Sekundarstufe I*. Verlagsgesellschaft Ritterbach, Frechen, April 1993 (Die Schule in Nordrhein-Westfalen)

[Knuth89] Knuth, D. E.: *Wissenschaft dialog*. In: *Magazin der Nixdorf Computer AG 1/89* (1989), Nr. 1, S. 10

[Koerber86] Koerber, Bernhard: *Leistungsmessung bei der Projektarbeit im Informatikunterricht*. In: *LOGIN* 6 (1986), Nr. 4, S. 20–23

[KP93] Koerber, Bernhard ; Peters, Ingo-Rüdiger: *Informatikunterricht und informationstechnische Grundbildung — ausgrenzen, abgrenzen oder integrieren?* In: *Informatik als Schlüssel zur Qualifikation* (siehe [Troitzsch93]), S. 108–115. – Tagung „Informatik und Schule“ des Fachbereiches 7 „Ausbildung und Beruf“ - Koblenz

[KP95] Koerber, Bernhard ; Peters, Ingo-Rüdiger: *Wertefreiheit und Ideologie in der informatischen Bildung*. In: *Innovative Konzepte für die Ausbildung* (siehe [Schubert95a]), S. 88–96. – Tagung „Informatik und Schule“ des Fachbereiches 7 „Ausbildung und Beruf“ - Chemnitz

[Kroha97] Kroha, Petr: *Softwaretechnologie*. 1. Aufl. Prentice Hall, München, 1997

[Krüger95] KRÜGER, G.: *Telekommunikation.* In: *Innovative Konzepte für die Ausbildung* (siehe [Schubert95a]), S. 2–15. – Tagung „Informatik und Schule" des Fachbereiches 7 „Ausbildung und Beruf" - Chemnitz

[Lehmann85] LEHMANN, Eberhard: *Projektarbeit im Informatikunterricht.* B.G. Teubner, Stuttgart, 1985

[Lehmann93] LEHMANN, Eberhard: *Software-Wartung. Ein neuartiger Einstieg in den Informatik-Anfangsunterricht.* In: *Informatik als Schlüssel zur Qualifikation* (siehe [Troitzsch93]), S. 134–140. – Tagung „Informatik und Schule" des Fachbereiches 7 „Ausbildung und Beruf" - Koblenz

[Loos86] LOOS, R. u.: *Rahmenempfehlungen für die Informatik im Unterricht der Sekundarstufe I.* In: *Informatik-Spektrum* 9 (1986), Nr. 2, S. 141–143

[Löthe88] LÖTHE, A.: *Didaktische Charakteristika von Programmierstilen.* In: *Didaktische Ansätze für die informatische Grundbildung im internationalen Vergleich* (siehe [GIDDR88]), S. 16–27

[LP81] LEWIS, H. R. ; PAPADIMITRIOU, Ch. H.: *Elements of the Theory of Computation.* Prentice-Hall, Englewood Cliffs, 1981

[Meyer95] MEYER, F.: *School-Wide Web: Eine Informations-Infrastruktur für Schulen.* In: *Innovative Konzepte für die Ausbildung* (siehe [Schubert95a]), S. 308–316. – Tagung „Informatik und Schule" des Fachbereiches 7 „Ausbildung und Beruf" - Chemnitz

[Meyer88] MEYER, H.: *Unterrichtsmethoden.* Scriptor Verlag, Frankfurt a. M., 1987/88. – 2 Bd.

[Modrow92] MODROW, E.: *Zur Didaktik des Informatik-Unterrichts.* Bd. Band 1 und 2. Dümmler, Bonn, 1991 and 1992

[MSNW96] MINISTERIUM FÜR SCHULE UND WEITERBILDUNG DES LANDES NORDRHEIN-WESTFALEN (Hrsg.): *Richtlinien Informatik, Gymnasiale Oberstufe.* Concept Verlag, Düsseldorf, 1996

[Müller92] MÜLLER, Klaus: *Objektorientierte Programmierung.* In: *Schulcomputerjahrbuch '93/94* (siehe [Bosler92]), S. 180–189

[MW95] MAIER, G. ; WILDBERGER, A.: *In 8 Sekunden um die Welt, Kommunikation über das Internet.* Addison-Wesley, Bonn, 1995

[NB42] NELSON, C. ; BOSSING, L.: *Die Projekt–Methode.* In: BLOCHMANN, E. (Hrsg.) ; GEISSLER, G. (Hrsg.) ; NOHL, H. (Hrsg.) ; WENIGER, E. (Hrsg.): *Das Problem der Unterrichtsmethode in der pädagogischen Bewegung*, Beltz, Weinheim - Berlin - Basel, 1942, S. 123–140

[Nievergelt91] NIEVERGELT, J.: *Was ist Informatik-Didaktik?* In: *Sonderdruck 4. Fachtagung „Informatik und Schule"* (1991). – Sonderdruck 4. Fachtagung „Informatik und Schule", Universität Oldenburg

[NPS98] NEUPERT, Heiko ; POHLMANN, Dietrich ; SCHUBERT, Sigrid: *Positionen zur informatischen Bildung an deutschen Schulen.* In: *LOGIN* 18 (1998), Nr. 1, S. 9

[Peschke89] PESCHKE, R.: *Die Krise des Informatikunterrichts in den neunziger Jahren.* In: STETTER, Franz (Hrsg.) ; BRAUER, Wilfried (Hrsg.): *Informatik und Schule 1989: Zukunftsperspektiven der Informatik für Schule und Ausbildung*, Springer-Verlag, Berlin Heidelberg, 1989 (Informatik-Fachberichte 220). – Tagung „Informatik und Schule" des Fachbereiches 7 „Ausbildung und Beruf" - München, S. 89–98

[Peschke90] PESCHKE, Rudi: *Grundideen des Informatikunterrichts.* In: *LOGIN* 10 (1990), Nr. 6, S. 25–33

[Peschke91] PESCHKE, R.: *Stand der informationstechnischen Bildung und Folgen für den Informatikunterricht.* In: *Weiterentwicklung des Informatikunterrichts - Folgerungen aus der Sicht von Lehrerbildung und Wissenschaft* (siehe [BP91]), S. 121–138

[Pütt82] PÜTT, Heinz: *neue pädagogische bemühungen.* Bd. 90 : Projektunterricht und Vorhabengestaltung. 1. Aufl. Neue Deutsche Schule Verlagsgesellschaft, Essen, 1982

[Rauch95] RAUCH, H: DECIDE - Entscheidungsfindung im Netz. In: *Innovative Konzepte für die Ausbildung* (siehe [Schubert95a]), S. 317–326. – Tagung „Informatik und Schule" des Fachbereiches 7 „Ausbildung und Beruf" - Chemnitz

[Rechenberg94] RECHENBERG, Peter: *Was ist Informatik?*. Carl Hanser Verlag, München, 1994

[Reischuk90] REISCHUK, K.R.: *Einführung in die Komplexitätstheorie.* B.G. Teubner, Stuttgart, 1990

[SBGK94] SCHELLER, M. ; BODEN, K. ; GREENEN, A. ; KAMPERMANN, J.: *Internet: Werkzeuge und Dienste.* Springer-Verlag, Berlin, 1994

[Schubert88] SCHUBERT, Sigrid: *Untersuchungen zur Lehrdisziplin Informatik unter dem Aspekt der Könnensentwicklung.* Technische Universität Chemnitz, 1988. – Diss. A

[Schubert91] SCHUBERT, Sigrid: *Fachdidaktische Fragen der Schulinformatik und (un)mögliche Antworten.* In: GORNY, P. (Hrsg.): *Informatik: Wege zur Vielfalt beim Lehren und Lernen*, Springer-Verlag, Berlin Heidelberg, 1991 (Informatik-Fachberichte 292). – Tagung „Informatik und Schule" des Fachbereiches 7 „Ausbildung und Beruf" - Oldenburg, S. 27–33

[Schubert92] SCHUBERT, Sigrid: *Logische Programmierung.* In: *Schulcomputerjahrbuch '93/94* (siehe [Bosler92]), S. 171–179

[Schubert95a] SCHUBERT, S. (Hrsg.): *Innovative Konzepte für die Ausbildung.* Springer-Verlag, Berlin Heidelberg, 1995 (Informatik aktuell). – Tagung „Informatik und Schule" des Fachbereiches 7 „Ausbildung und Beruf" - Chemnitz

[Schubert95b] SCHUBERT, Sigrid: *Welche Theorie-Inhalte müssen in der Lehrerausbildung vermittelt werden?* In: HUBER-WÄSCHLE, F. (Hrsg.) ; SCHAUER, H. (Hrsg.) ; WIDMAYER, P. (Hrsg.): *GISI 95, Herausforderung eines globalen Informationsverbundes für die Informatik*, Springer-Verlag, Berlin Heidelberg, 1995 (Informatik aktuell), S. 374–381

[Schubert97] SCHUBERT, Sirid (Hrsg.): *Themenheft: Computerunterstütztes Lernen.* Informationstechnik und Technische Informatik, Juni 1997. – Themenheft der Zeitschrift: it + ti - Informationstechnik und Technische Informatik

[Schwill93] SCHWILL, Andreas: Fundamentale Ideen der Informatik. In: *Zentralblatt für Didaktik der Mathematik* 25 (1993), Nr. 1, S. 20–31

[Schwill95] Schwill, A.: *Programmierstile im Anfangsunterricht.* In: *Innovative Konzepte für die Ausbildung* (siehe [Schubert95a]), S. 178–187. – Tagung „Informatik und Schule“ des Fachbereiches 7 „Ausbildung und Beruf“ - Chemnitz

[SMK92a] Sächsisches Staatsministerium für Kultus (Hrsg.): *Lehrplan Gymnasium Informatik*. SMK, Dresden, 1992

[SMK92b] Sächsisches Staatsministerium für Kultus (Hrsg.): *Lehrplan Mittelschule Angewandte Informatik*. SMK, Dresden, 1992

[SMK97] Sächsisches Staatsministerium für Kultus (Hrsg.): *Orientierungsrahmen für das Angebot Angewandte Informatik an Mittelschulen*. SMK, Dresden, 1997

[SS86] Sterling, L. ; Shapiro, E.: *PROLOG*. Addison-Wesley, Reading (Massachusetts), 1986

[Stetter88] Stetter, Franz: *Grundbegriffe der Theoretischen Informatik*. Springer-Verlag, Berlin, 1988

[SZ93] Schulz-Zander, Renate u.: *GI-Empfehlungen für das Fach Informatik in der Sekundarstufe II allgemeinbildender Schulen.* In: *Informatik als Schlüssel zur Qualifikation* (siehe [Troitzsch93]), S. 205–218. – Tagung „Informatik und Schule“ des Fachbereiches 7 „Ausbildung und Beruf“ - Koblenz

[Thüringen98] Thüringen, Kultusministerium (Hrsg.): *Vorläufiger Lehrplan für das Gymnasium für das Leistungsfach Informatik*. Kultusministerium Thüringen, Erfurt, 1998

[Troitzsch93] Troitzsch, Klaus G. (Hrsg.): *Informatik als Schlüssel zur Qualifikation*. Springer-Verlag, Berlin Heidelberg, 1993 (Informatik aktuell). – Tagung „Informatik und Schule“ des Fachbereiches 7 „Ausbildung und Beruf“ - Koblenz

[Wagner94] Wagner, K. W.: *Einführung in die Theoretische Informatik*. Springer-Verlag, Berlin, 1994

[Wirth92] Wirth, Niklaus: *Geleitwort.* In: Mössenböck, H. (Hrsg.): *Objektorientierte Programmierung in Oberon-2*. Springer-Verlag, Berlin, 1992