T U M

I N S T I T U T F Ü R I N F O R M A T I K

Der informationszentrierte Ansatz
Ein Vorschlag für eine zeitgemäße Form des Informatikunterrichtes am Gymnasium

Peter Hubwieser, Manfred Broy

TUM–I9624
Mai 96

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-05-96-I9624-200/1.-FI
Alle Rechte vorbehalten
Nachdruck auch auszugsweise verboten

©96

Druck: Fakultät für Mathematik und
Institut für Informatik der
Technischen Universität München

Der informationszentrierte Ansatz

Ein Vorschlag für eine zeitgemäße Form des Informatikunterrichtes am Gymnasium[1]

Peter Hubwieser
Manfred Broy

Fakultät für Informatik der Technischen Universität München
80290 München, Tel. 089-2105-2-8162
e-mail: hubwiese|broy@informatik.tu-muenchen.de

Zusammenfassung

Mit dem Vorschlag eines neuen, informationszentrierten Ansatzes für den Informatikunterricht am Gymnasium wollen wir eine Verschiebung der inhaltlichen Schwerpunkte weg von Algorithmik und Programmierung hin zu fundamentalen Strukturierungstechniken für den Rohstoff "Information" bewirken. Der Unterricht soll unserer Meinung nach, über bloßes "Computerwissen" hinausgehend, Fähigkeiten vermitteln, die Schüler und Schülerinnen bei der Handhabung großer Datenmengen und dem Verstehen von geeigneten Verarbeitungsprozessen unterstützen. Wir wollen dabei nach dem Vorbild moderner Methoden der Softwareentwicklung vor allem graphische Modellierungstechniken einsetzen, die unabhängig von den speziellen Eigenschaften der elektronischen Ausstattung der jeweiligen Schule eine detaillierte, aussagekräftige Beschreibung und Erfassung komplexer Systeme ermöglichen. Die Verwendung des Rechners am Ende der Unterrichtseinheiten dient vor allem der Veranschaulichung der erarbeiteten Modelle. Dabei stellen wir uns weniger Implementierungen mittels einer Programmiersprache, sondern eher die Verwendung von Standardsoftware vor. Zum Abchluß dieses Berichtes illustrieren wir unsere Vorstellungen mittels einiger konkreter Vorschläge für Unterrichssequenzen.

1 Einleitung

Der Informatikunterricht an den Gymnasien ist nach allgemeiner Meinung in eine (nicht ganz unerwartete) Krise geraten [Peschke89]. In Bayern äußert sich diese unter anderem in nachlassendem Engagement der Lehrkräfte, sinkender Stundenzahl der Wahlunterrichte und einer festgefahrenen Diskussion über ein Pflichtfach Informatik. Ein weiteres Symptom ist die mangelnde Anziehungskraft des gegenwärtigen Informatik-Unterrichtes auf Mädchen ([Schulz-Zander+93]), die sich unter anderem in einem erschreckenden Rückgang der Frauen-Quoten bei den Informatik-Studienanfängern (von nahezu 20% auf ca. 5% Mädchen im WS 1994/95 an der TU München) äußert.

Das nachlassende Interesse der Abiturienten an der Informatik trotz deren großer wirtschaftlicher Bedeutung und der nach wie vor vergleichsweise guten Berufsaussichten hat nicht zuletzt mit dem verzerrten Bild der Informatik zu tun, das Gymnasiasten in der Schule geboten wird. Darüber hinaus werden dort Informatiktalente nicht hinreichend angesprochen. Statt be-

[1] Dieser Text enstand aus dem Skriptum zur Vorlesung: "Didaktik der Informatik 1" (P. Hubwieser , WS 1995/96, TU München).

reits in der Schule eine gezielte Förderung zu erfahren, bleiben sie weitgehend sich selbst überlassen, verbohren sich oft frühzeitig in eine techniknahe Ebene und finden oft nicht den Weg zu einem Informatik-Studium, das ihnen den nötigen theoretischen Hintergrund und angemessene Berufsaussichten vermitteln könnte. Die schlechte Position Deutschlands im Wettbewerbsfeld der internationalen Softwareproduktion ([BSK96]) könnte unter anderem in solchen Mängeln des Ausbildungssystems begründet sein.

Auf der Suche nach Verbesserungsmöglichkeiten stellt sich die Frage, ob es Sinn es macht, am Gymnasium wertvolle Unterrichtszeit für das Erlernen einer speziellen Programmiersprache mit allen ihren technischen Eigenheiten und Zufälligkeiten zu verbrauchen. Wichtiger wäre es sicherlich, den Schülern ein Grundverständnis für die Prinzipien der Informationsverarbeitung im weiteren Sinne und für Strukturierungs- und Modellierungstechniken der Informatik nahezubringen, die auch in anderen Bereichen, wie etwa der Organisation von Personalstrukturen oder Arbeitsprozessen in Industriebetrieben vermehrt Anwendung finden ([Jacobson+95]). Die genannte Krise wurde nach Meinung der Verfasser in nicht unerheblichem Maße auch von den bisher im Unterricht umgesetzten didaktischen Ansätzen ausgelöst, die zu sehr auf vordergründige technische Details ausgerichtet waren. Die Vermittlung allgemeinbildender Inhalte der Informatik wurde dabei nicht in dem Maße berücksichtigt, wie es im Rückblick angebracht gewesen wäre.

Einen weiteren Anlaß für eine Diskussion über neue Inhalte der Informatik an den Schulen bietet das Heraufziehen des Informationszeitalters mit allen seinen Konsequenzen wie die zunehmende elektronische Vernetzung der Haushalte, der rasante Zuwachs an menschlichem Wissen mit dem sich daraus ergebenden Zwang zu ständiger Neuinformation sowie die absehbare Einrichtung von Heimarbeitsplätzen in strukturschwachen Gebieten mit Online-Verbindung zum Arbeitgeber ([BSK95]).

Alle diese Entwicklungen lassen die zukünftige Bedeutung des Wirtschaftsgutes "Information" erahnen. Sie stützen sich dabei auf bestimmte Konzepte der Informationsverarbeitung, die nicht nur unserer Meinung nach in den Mittelpunkt des Informatik-Unterrichtes gestellt werden müssen ([Breier94], [Baumann90]). Anstatt zu lernen, wie man einen Computer programmiert, sollten die Schüler Antworten auf Fragen geben können, wie sie in [Glaser95] sehr treffend formuliert wurden: "Die großen Fragen einer Informationsgesellschaft lauten also: Wie schützt man sich vor Daten? Wie vor lebenszeitfressendem, unnützen Wissen? Wie lernt man Informationen nach seinen Interessen zu bewerten, auszuwählen und zu strukturieren? Und wie und mit welchen Instrumenten - also Programmen - kann man effizient, alltagstauglich und elegant an der Erkundung und Benutzung der elektronischen Welt teilhaben?". Es kann nicht genügen, die Schulen rein technisch an moderne Informationssysteme wie das Internet anzuschließen ([Sarnow96]) oder sie mit modernsten Multimedia-Systemen auszustatten. Gleichzeitig müssen wir auch dafür sorgen, daß die Lehrer und die Schüler mit dem nötigen gedanklichen Rüstzeug zur Bewältigung der damit über sie hereinbrechenden Informationsmassen versorgt werden ([Schulz-Zander96]).

Unser Vorschlag bezieht sich ausdrücklich auf den Informatik-Unterricht an Gymnasien außerhalb der *Informationstechnischen Grundbildung* (ITG), wie sie etwa in [BAL88] konzipiert wurde. Das Ziel ist die Herstellung eines breiten Konsenses über Inhalte der Informatik, die derartig allgemeinbildend sind, daß sie in den Pflichtunterricht einbezogen werden müssen ([ZS93]].

In diesem Papier sollen zunächst in Abschnitt 2 kurz die wesentlichen didaktischen Ansätze der Vergangenheit charakterisiert werden, um anschließend in Kapitel 3 die bestimmenden Merkmale unseres Ansatzes dagegen abheben zu können. Daraufhin wollen wir in Abschnitt 4 Vorschläge für Unterrichtsmethodik und Lerninhalte anbieten, welche die Grundlage für die in

Kapitel 5 folgenden Beispiele konkreter Unterrichtsprojekte bilden. Abschließend soll Kapitel 6 unser Gesamtkonzept zur Verbesserung des Informatikunterrichtes vorstellen.

2 Herkömmliche didaktische Ansätze

Um unsere Vorschläge zu verdeutlichen, sollen zunächst einige der bisher im Informatikunterricht verfolgten didaktischen Ansätze beschrieben werden. Für eine eingehendere Darstellung und Kritik der genannten Ansätze verweisen wir auf [Schulz-Zander78], [Forneck90] und [Baumann90].

2.1 Die Hardware als Ausgangspunkt

Der Ansatz stammt aus den späten 60er- und frühen 70er Jahren, in denen sich die elektronischen Rechenanlagen noch im Entwicklungsstadium befanden, eigentliche Anwendungen kaum realisiert waren und die Informatik noch nicht als eigene wissenschaftliche Disziplin anerkannt war. Vom Charakter her handelt es sich um eine der Kybernetik verpflichtete "Rechnerkunde" mit dem Ziel einer Vermittlung der mathematisch-technischen Grundlagen der DV [Meißner72,75].

Daß dieser Ansatz für das Gymnasium im allgemeinen nicht der geeignete sein kann, wurde bereits hinlänglich diskutiert ([Schulz-Zander78], [Forneck90], [Baumann90]). Dennoch bestimmt er immer noch (oft unausgesprochen) in nicht geringem Maße die Denkweise von Lehrkräften und damit zahlreiche Einzelentscheidungen über Methodik und Lerninhalte, insbesondere im Wahlunterricht der Mittelstufe.

Eine Zentrierung solcher Art zwingt zu ständigem Umlernen beim Verfolgen der neuesten Hardware- und Betriebssystementwicklungen. Genau dies kann jedoch kein bleibender Beitrag des Informatik-Unterrichtes zur Allgemeinbildung sein, dessen Lerninhalte eben nicht einem derartig schnellen Wandel unterworfen sein dürfen.

2.2 Der Algorithmus als Maß aller Dinge

Ab Mitte der 70er Jahre entstand unter dem Eindruck allgemeiner Anerkennung der Informatik als neuer wissenschaftlicher Disziplin ein Ansatz mit nun auch fachwissenschaftspropädeutischem Anspruch, der sogar die Abhaltung des Unterrichtes in einer eigenen Sprache postulierte ([Brenner+82]).

Die Schüler sollten (nach [Knauer80]) Algorithmen formulieren, programmieren, Probleme mit algorithmischem Hintergrund analysieren, gefundene Algorithmen umsetzen und durch Programmierung lösen können. Methodisch folgte der Erkennung einer Problemstellung der Entwurf eines Lösungsplans, darauf die Lösung des Problems, eine Prüfung der Korrektheit der Problemlösung und schließlich eine Diskussion möglicher Verbesserungen der Lösung ([Balzert80]). Die Lösung wurde dabei vorzugsweise im Top-Down-Verfahren ([Bauer79]) entwickelt.

Mit der Forderung nach vollständiger Algorithmisierung der Problemstellungen werden diese auf relativ einfache Beispiele mit geringer Informationskomplexität beschränkt. Gesellschaftliche Auswirkungen der Informatik oder die Beherrschung komplizierterer Informationsstrukturen sind im Unterricht nicht vorgesehen.

2.3 Die vom Algorithmus beherrschte Anwendung

Kurz nach dem letztgenannten Ansatz entwickelte sich, ausgehend von der anfangs der 70-er Jahre von [Robinsohn75] erhobenen Forderung nach einer Ausrichtung des Unterrichtes an konkreten Lebenssituationen anstatt an wissenschaftlichen Disziplinen, ein neuer Ansatz. Dieser wollte neben der Lösung praktischer Probleme auch die gesellschaftlichen, kulturellen, psychologischen Dimensionen dieser Lösungsfindung miteinbeziehen.

Der Grundgedanke war, ausgehend von einer den Schülern bekannten Anwendung der Informatik, die Lösung einer bestimmten Problemstellung zu entwickeln, wobei wiederum die Algorithmik als Werkzeug zur Lösungsfindung betont wurde, um dann möglichst alle Konsequenzen dieses Vorganges zu beleuchten.

Als Lernziele werden genannt ([GI76]): die Schüler sollen in der Lage sein, algorithmische Lösungen von Problemen systematisch zu finden, algorithmische Lösungen als Programm zu formulieren, das Gelernte durch Anwendung auf praxisorientierte Probleme zu vertiefen, Auswirkungen der DV auf die Gesellschaft zu erkennen.

Methodisch ist dabei zu beachten, daß Unterrichtsgegenstände nur dann Geltung beanspruchen können, wenn sie sich in einen Anwendungsbezug einbetten lassen ([Schulz-Zander78]). Nach [Koerber+88] ergibt sich ein typischer Unterrichtsablauf, in dem sich eine problembezogene, eine modellbezogene und eine informatikbezogene Ebene abwechseln, wobei diese Ebenen bei der Software-Erstellung von oben nach unten, bei der Programmbenutzung von unten nach oben durchschritten werden.

Da durch das Nadelöhr der Algorithmik die Fülle der Lernziele nicht erarbeitbar ist und für komplexere Probleme in der Schule oft kein Lösungsalgorithmus entwickelt werden kann, überfordert dieser Ansatz durch seinen umfassenden Anspruch das Fach Informatik sowie Lehrer und Schüler ([Forneck90]). Die Trennung von "modellbezogener" und "informatikbezogener" Ebene verrät zudem den verengten Blickwinkel auf die moderne Informatik, denn gerade in den Modellierungstechniken liegt einer der wesentlichsten Beiträge der Informatik zur Allgemeinbildung ([Goos96]).

2.4 Der Benutzer im Mittelpunkt

Unter dem Eindruck des Vordringens der Mikroelektronik in Freizeit und Familie, der Weiterentwicklung kommerzieller Software mit ihren verminderten Einarbeitungszeiten und neuartigen Vernetzung von Informations- und Kommunikationstechnologien entstand in den späten 80er Jahren ein neuer Ansatz. Dieser will unter Verzicht auf Programmierung mittels Benutzung von Anwendersystemen ausschließlich lebenspraktische Orientierung vermitteln. Er geht dabei anstatt von Prinzipien der Fachdisziplin Informatik von den Auswirkungen der technologischen Entwicklung aus ([LISW87]).

Primäre Zielsetzungen sind informationstechnische Allgemeinbildung, Qualifizierung zum rationalen Umgang mit den Kommunikations- und Informationstechnologien, Steigerung der Beurteilungsfähigkeit ihrer Anwendungen und Auswirkungen sowie die Vermittlung der Fähigkeit, durch Ausbreitung und Weiterentwicklung der Technologien entstehende Probleme zu bewältigen.

Es gibt kein ausgezeichnetes methodisches Vorgehen, man durchläuft die folgenden Tätigkeitsgebiete in unterschiedlicher Reihenfolge (manche auch mehrfach): Finden, Erkennen und Analysieren eines Problems, Strukturieren des Problems und Entwickeln modellhafter Lösungsmöglichkeiten, Nutzen von Anwendersystemen und Programmierumgebungen, Beurteilen der Ergebnisse, Reflektieren und Bewerten der Nutzung der Technologien.

Für das Gebiet der ITG ist dieser Ansatz wohl passend, für den eigentlichen Informatikunterricht des Gymnasiums fehlt ihm die formale Tiefe, die bei den bisher beschriebenen Ansätzen durch die Erstellung von Algorithmen erreicht wird. Die Schüler sehen bei der Verwendung vorgegebener Standardsoftware nur die äußeren Strukturen der Programmoberfläche, die eigentlich interessanteren inneren Konzeptionen wie Datenstrukturen und Problemlösungstil bleiben ihnen verborgen, was sie sowohl in ihrer Kritik-, wie auch in ihrer Abstrahierungsfähigkeit stark einschränkt.

3 Die Information als Mittelpunkt des Unterrichtes

Wie bereits von [Baumann90] gefordert, darf der Informatikunterricht an der Schule sich unserer Meinung nach nicht auf die Vermittlung von "Computerwissen" beschränken. Es geht vielmehr um die Heranführung der Schüler an Grundsätze der Informationswissenschaften im weiteren Sinne. Die technischen Gegebenheiten, seien es nun Hard- oder Softwaresysteme, können nur ein Medium zur Veranschaulichung von Prinzipien und Konzepten sein, niemals Ausgangspunkt didaktischen Handelns. Es spricht natürlich nichts dagegen, im Rahmen der ITG exemplarisch Handhabung und Anwendung spezieller Informationssysteme kennenzulernen, man muß sich dabei jedoch immer über den schnellen Alterungsprozeß solcher Kenntnisse im klaren sein. Bleibende Beiträge können nur übergeordnete Konzepte der Informatik sein, die oft über Jahrzehnte nichts an Aktualität verlieren.

3.1 Grundsätze

Klärung der verwendeten Begriffe

Zur Abgrenzung gegenüber der umgangssprachlichen Verwendung der Begriffe *Modell*, *Modellierung* und *Beschreibung* legen wir einführend fest, was wir im Rahmen dieser Arbeit darunter verstehen wollen.

Nach [Goos95] besteht ein *Modell* aus "Begriffen von (real existierenden oder nur gedachten) Dingen, Personen, Abläufen in gedachter Zeit, Beziehungen zwischen diesen Begriffen (logisch möglichen Sachverhalten),..", im Gegensatz zur "*Wirklichkeit*: Dinge, Personen, Abläufe in der Zeit, Beziehungen zwischen diesen Gegenständen (Tatsachen), ..". Wir setzen zusätzlich noch voraus, daß sich diese Begriffe auf exakte mathematische Definitionen abstützen, wie in [Broy95] ausgeführt: "*models*: the mathematical structures forming the semantical conceptual model associated with a description technique."

Modellierungstechniken sind dagegen "*Beschreibungsverfahren* mit einer im Vornherein festgelegten Syntax und Semantik. Sie dienen dazu, bestimmte Aspekte der Anwendung abzubilden" ([Broy94b]).

Modellierung ist die Abbildung von Elementen und Strukturen der Wirklichkeit in Form von Beschreibungen im oben genannten Sinn: "*modeling*: the mapping representing and relating real life aspects of the application to software description techniques" ([Broy95]).

Ein informationszentrierter Ansatz

Das zentrale Thema unseres Ansatzes ist nicht mehr die Erstellung von Algorithmen, deren Programmierung oder die Benutzung bestimmter Anwendungsprogramme, also die Planung und Handhabung von Hard- und Software, sondern Verständnis für Beschaffungswege, Auswahlkriterien, Strukturierungsmethoden, Verarbeitungstechniken und Darstellungsformen des Werkstoffes "Information" ([Breier94]). Es soll eher der geschickte Umgang mit großen Informationsmengen erlernt werden als die Konzeption komplexer Algorithmen. Die örtliche Rechenanlage und ihre konkrete Programmierung treten in den Hintergrund, der Computer dient vor allem als Motivations- und Veranschaulichungshilfe.

Es wird bewußt auf den Zwang zur vollständigen Algorithmisierung der im Unterricht behandelten Problemstellungen verzichtet, wodurch die Behandlung weit komplexerer Strukturen möglich wird. Die Betonung liegt auf der Verwendung von Modellierungstechniken für Informationssysteme und auf der Strukturierung der Daten. Die Ausformulierung von Algorithmen und deren Realisierung in einer Programmiersprache kann zwar für geeignete Teilaspekte durchaus sinnvoll und nutzbringend sein, sollte jedoch nicht das bestimmende Element des Unterrichts sein.

Repräsentationsformen von Informationen

In der Klarstellung der wechselseitigen Beziehung zwischen einer Informationsmenge, ihrer Repräsentation und deren Interpretation, insbesondere in der Aufdeckung von Informationsverlusten oder -verfälschungen bei Modellierungen, Umformungen oder Transportvorgängen liegt einer der wichtigsten Beiträge, die das Fach Informatik zur Allgemeinbildung leisten kann. So kann man verschiedene Formen multimedialer Informationsrepräsentation samt ihrer Vor- und Nachteile behandeln und diskutieren, etwa verschiedene Arten bildlicher graphischer Datenorganisation vergleichen (siehe Abschnitt 5.3), die Problematik einer digitalen Repräsentation von analogen Tonstrukturen besprechen oder mit Hilfe von Hypertextsystemen vorgegebene Themengebiete durchstrukturieren (siehe Abschnitt 5.1).

Daneben muß die Schule Fähigkeiten vermitteln, in der Vielfalt moderner Medien gangbare Wege zur Beschaffung nützlicher Informationen ausfindig zu machen, effektiv zu nutzen und die erhaltenen Informationen kritisch zu beurteilen (siehe auch [Breier94]).

Datenstrukturen

In den Repräsentationsformen von Informationen findet man häufig wiederkehrende Grundmuster, die im Lösungsansatz als abstrakte Datenstrukturen wie Verbunde, Listen, Bäume erscheinen. Zu jedem dieser Datentypen gibt es eine Menge von fundamentalen Verwaltungsoperationen. Komplexere Datenstrukturen können dann in einer Art Baukastenprinzip aus den Grundtypen aufgebaut werden.

3.2 Der Softwareentwicklungsprozeß als Leitbild

Korrekte Software

Auf der Suche nach einer Verbesserung der Qualität industriell erstellter Software durch Systematisierung des Entwicklungsprozesses und Verifikation der Ergebnisse (etwa mit Hilfe von interaktiven Beweisersystemen wie *Isabelle*, [Paulsen94]) entstanden eine Vielzahl von Spezifikations-, Entwicklungs- und Modellierungstechniken ([Broy+92], [Broy+93], [Broy95], [Hußmann94]) [Rumbaugh+91], [Jacobson+92], [Booch94]). Bei all diesen Techniken steht am Anfang eine mehr oder weniger formale, jedoch völlig von speziellen Hard- und Softwaregegebenheiten unabhängige Spezifikation, die dann über die Konstruktion von programmierstilabhängigen Datenstrukturen, einem sprachabhängigem Programmcode und schließlich einem plattformspezifischen Compilat schrittweise an die speziellen Erfordernisse der Implementierung angepaßt wird. Die meisten Freiheitsgrade des Entwicklers liegen dabei in der Spezifikation des Systems durch einen Satz geeigneter Modelle für unterschiedliche Sichten des Systems. Die Zwischenprodukte der weiteren Entwicklertätigkeit ergeben sich daraus nach der Entscheidung für eine Realisierungsplattform so zwangsläufig, daß man diese Schritte völlig automatisieren kann ([Broy+96]).

Realisierungsunabhängige Modellierungstechniken

Es liegt auf der Hand, daß Software-Entwicklungsmethoden für den Schulunterricht umso wertvoller sind, je allgemeiner, sprach- und plattformunabhängiger sie sind. Die formale Spezifikation eines Informationsystems kann in Jahrzehnten noch aktuell sein, während die Implementierung eine Vielzahl von Plattformwechseln vollzogen haben wird. In der Schule können wir uns daher am ehesten die in den ersten Phasen des Entwicklungsprozesses eingesetzten Modellierungstechniken zunutze machen.

Während es bei der Erstellung von Software wünschenswert scheint, die Spezifikation soweit wie möglich zu formalisieren, um eine klare Semantik zu entwickeln und eine automatische Verifikation zu erleichtern, sollten wir uns bei Auswahl und Konzeption der Beschreibungstechniken für den Unterricht eher an der Schnittstelle zwischen Entwickler und Kunden orientieren, wo es vor allem auf einprägsame, intuitiv verständliche Beschreibungen der Sachverhalte ankommt.

Objektorientierung

Der große Erfolg objektorientierter Entwicklungsmethoden vor allem in jüngster Zeit ([Rumbaugh+91], [Jacobson+92], [Booch94]) stellt ein Indiz für deren intuitive Überlegenheit über die traditionellen Varianten mit ihrer globalen Unterscheidung zwischen statischen Datenstrukturen und dynamischen Operationen dar. Letzere wurden den Entwicklern über Effizienzüberlegungen von der Struktur der ersten Rechenanlagen mit ihrer Unterscheidung zwischen Programm- und Datenspeicherung aufgeprägt. Mit der immensen Steigerung der Leistungsfähigkeit moderner Rechenanlagen können wir es uns leisten, den Maschinen das menschliche Konzept einzelner, selbständig agierender Objekte aufzuzwingen. Zusätzlich kommt die innerhalb objektorientierter Entwicklungsmethoden übliche Einteilung von Objekten in hierarchisch geordnete Klassen dem menschlichen Strukturierungsbedürfnis ([Anderson85]) sehr entgegen. Aus diesen Gründen plädieren wir auch in der Schule für die Bevorzugung objektorientierter Techniken.

Business-Engineering
Als weitere Quelle geeigneter Modellierungstechniken bietet sich das Feld der Business-Engineering und -Reengineering-Methoden an, wie sie etwa von [Jacobson+95] propagiert werden. Der besondere Reiz dieses Gebietes für die Schule liegt in der Beschreibung von Systemen, die auf den ersten Blick nichts mit Programmen oder Rechenanlagen zu tun haben.

3.3 Beschreibungstechniken zur Strukturierung von Information

Nach den vorangegangenen, allgemeineren Betrachtungen wollen wir nun einige konkrete Vorschläge machen, die als Anstoß zur Entwicklung und Ausarbeitung schulspezifischer Beschreibungs- und Strukturierungstechniken verwendet werden können.

Am Anfang steht eine Partition des zu behandelnden Systems in Subsysteme (Objekte), deren innere Struktur (Attribute), Verhalten (Operationen) und hierarchische Klassifizierung (Klassen und Vererbungsbeziehungen, Instanzen) dann zu beschreiben sind. Zusätzlich muß das Verhalten des Gesamtsystems von außen gesehen und die Kommunikation zwischen den einzelnen Objekten (Assoziationen) bestimmt werden.

Wir schlagen dafür einprägsame graphische Nonationen in der Art von E/R-Diagrammen mit klar definierter Syntax vor, deren Semantik (für den Lehrer!) etwa durch axiomatische Spezifikationssprachen ([Hettler95]) abgestützt werden könnte.

Statische Systemstruktur
Um sich einen ersten groben Überblick über den Aufbau des betrachteten Systems zu verschaffen, bietet sich zunächst eine Strukturierung der Komponenten in Baumform an, etwa gemäß der Notation in Abb.1 ([Eickel95]).

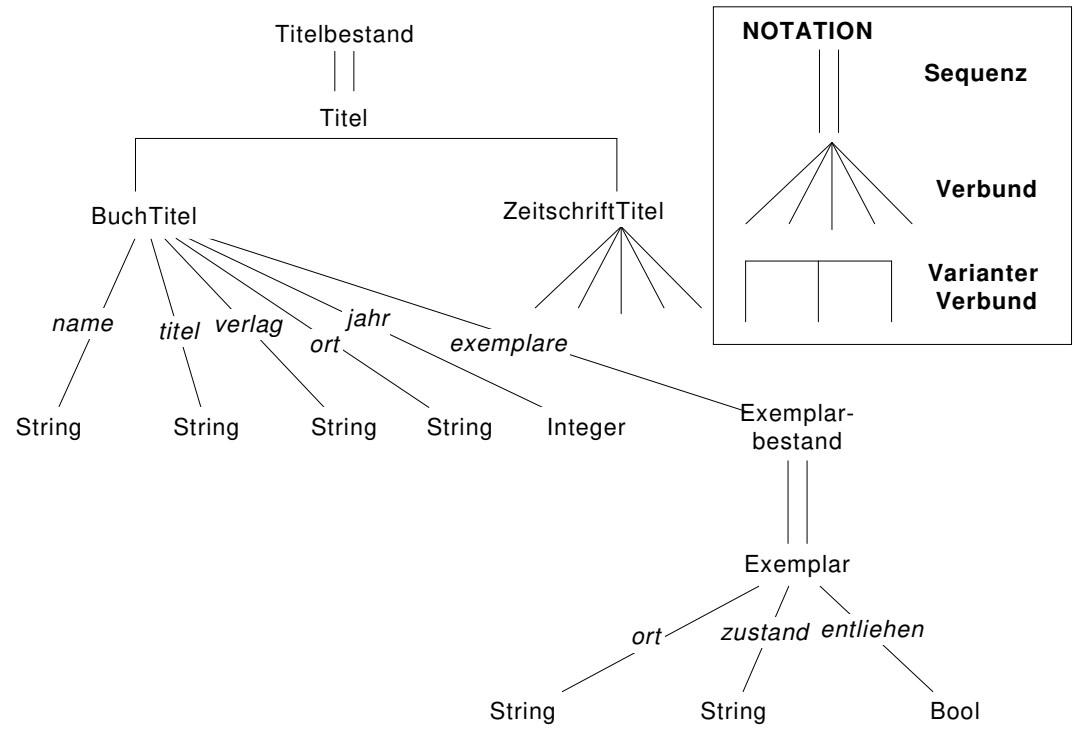

Abbildung 1: Statische Datenstrukturen eines Bibliothekssystems

Diese Beschreibungstechnik stützt sich im wesentlichen auf die abstrakten Datentypen Sequenz, Verbund und varianter Verbund. Daraus bauen wir eine spezielle Art von Baumdiagramm auf, wobei an den Knoten Sortenbezeichner stehen, während die Kanten mit den Namen der Selektoren bzw. Diskriminatoren versehen sind und durch ihre spezielle Form die Art der Komposition bezeichnen. Als vereinfachte Form kann auch ein gleichartiges Diagramm gewählt werden, bei dem zunächst die Selektor/Diskriminatornamen an den Knoten notiert werden. Die Typen (Sorten) können dann später als Klassen gleichartiger Objekte identifiziert werden.

Objektdiagramme

Um die Funktionalität der einzelnen Komponenten zu beschreiben, gehen wir nun einen Schritt weiter zum Objektmodell, bei dem das Verhalten der Komponenten zusammen mit ihrer statischen Struktur zu Objekten zusammengefaßt wird. Gleichartige Objekte bilden die Instanzen einer gemeinsamen Klasse. Wir schlagen eine graphische Beschreibung ähnlich [Rumbaugh+91] vor, die eine Vielzahl von Informationen in einem Diagramm vereinigt und in Abb.2 exemplarisch für ein Bibliothekssystem ausgeführt ist. Alternative Notationen finden sich unter anderem in [Booch94] oder [Jacobsson+92].

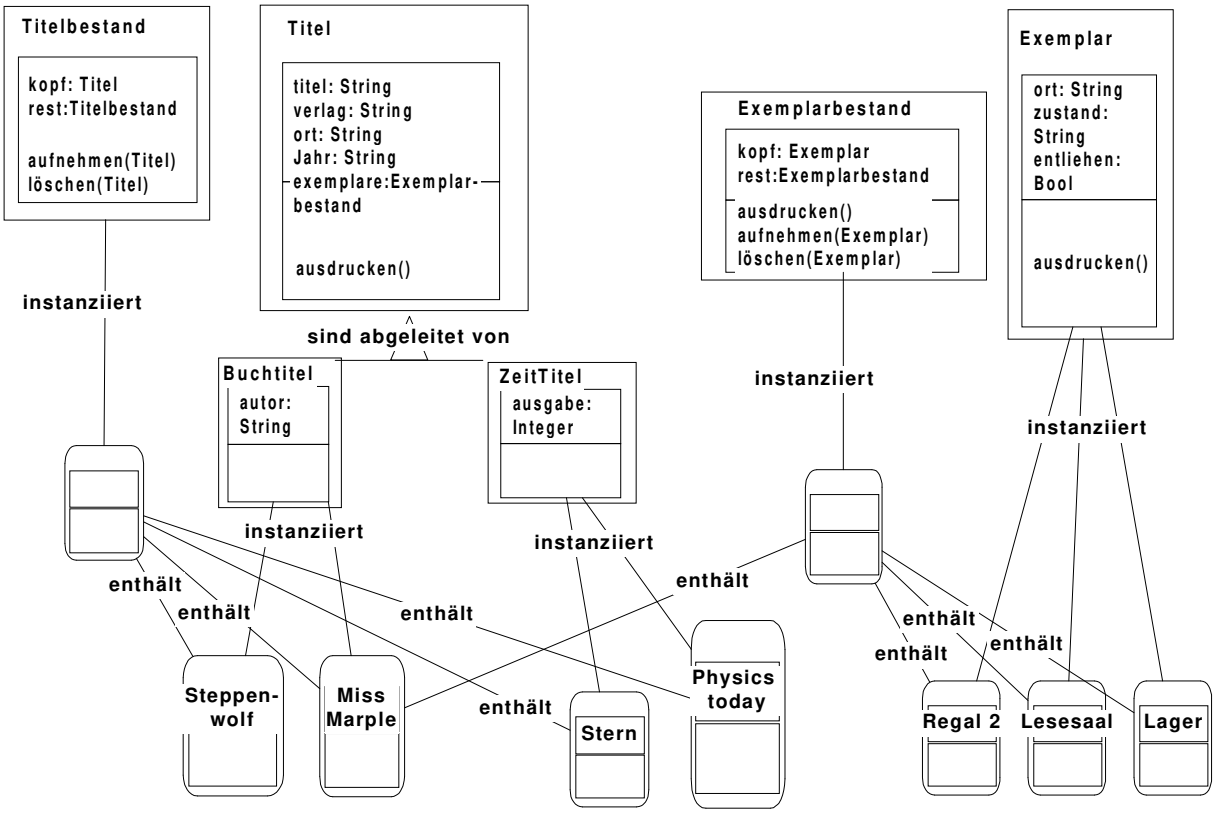

Abbildung 2: Objektmodell eines Bibliotheksystems

Variante Datentypen modellieren wir über eine gemeinsame Oberklasse durch Vererbung. Insgesamt besteht ein solches Objektdiagramm aus Knoten für Klassen und Instanzen sowie Kanten für Relationen zwischen den Objekten, die je nach Art der beteiligten Objekte in drei Klassen eingeteilt werden können: Klasse-Klasse-Relationen wie Vererbung, Klasse-Instanz-Relationen wie Instanziierung und Instanz-Instanz-Relationen wie "ist enthalten", "benutzt" oder "ruft auf". Besonderer Wert sollte auf die Klarstellung der unterschiedlichen Abstraktionsebenen von Klassen und Instanzen gelegt werden.

Zustandsübergangsdiagramme

Um dynamische Vorgänge en detail darzustellen, eignen sich besonders Zustandsübergangsdiagramme etwa wie in Abb.3. (für den Weg einer Gesetzesvorlage im Bayerischen Landtag) ausgeführt.

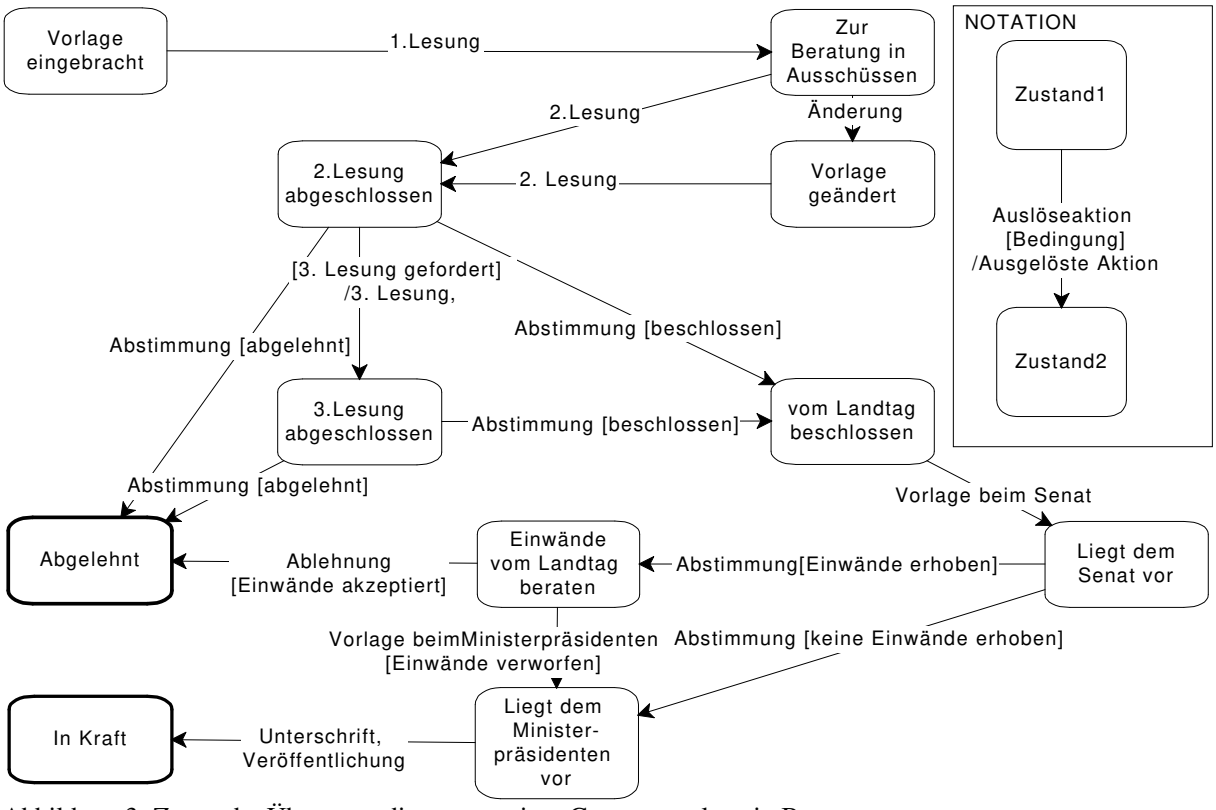

Abbildung 3: Zustands- Übergangsdiagramm einer Gesetzesvorlage in Bayern

Die Knoten des Graphen stehen für Zustände (wie Belegung von Variablen oder Stationen eine Programmablaufs), welche zusätzlich mit einer Aktion belegt werden können, die auszuführen ist, während sich das System im jeweiligen Zustand befindet. Die Semantik derartiger Diagramme ist in [Broy+96] beschrieben. Die Kanten verkörpern Zustandsübergänge, markiert mit dem Namen einer Auslöseaktion, einer Übergangsbedingung und einer ausgelösten Aktion.

Datenflußmodelle

Datenflüsse zwischen statischen Objekten können sehr anschaulich mithilfe von Datenflußdiagrammen modelliert werden. In der zeitlichen Abfolge der Lerninhalte bietet sich diese Beschreibungstechnik wohl als erste an, da sie einerseits intuitiv am leichtesten zugänglich ist, andererseits der Begriff der Funktion Voraussetzung für die Anwendung des Objektmodells ist.

Eine der ersten denkbaren Einsatzmöglichkeiten stellt die Modellierung von Termen als informationsverarbeitenden Prozessen im Rahmen des Algebraunterrichtes der Unter-und Mittelstufe des Gymnasiums dar. Bei einer Wiederholung des Bruchrechnens aus der 6.Klasse - könnte dort das folgende Datenflußdiagramm für die Addition/Subtraktion von Brüchen entstehen:

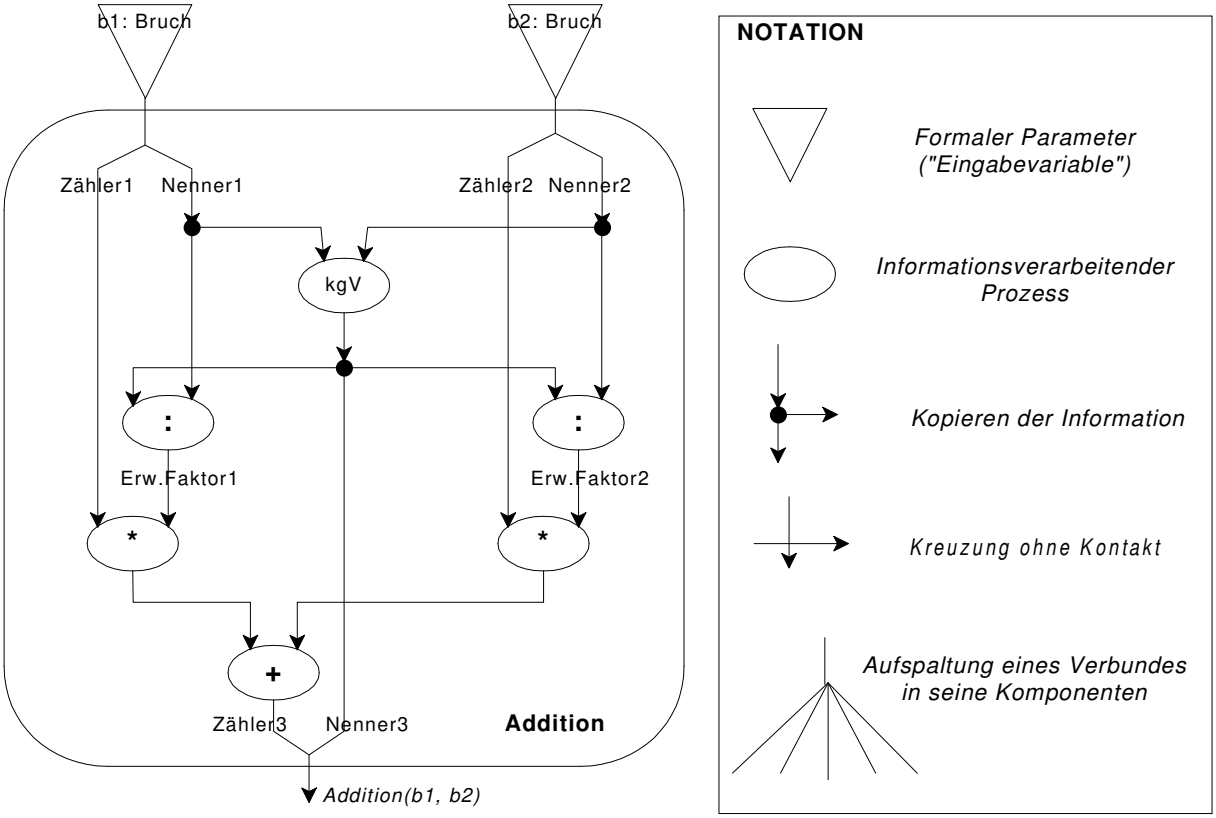

Abbildung 4: Datenflußdiagramm der Addition von Bruchzahlen

Das Diagramm aus Abb.4 wurde im Rahmen eines Versuchseinsatzes von Datenflußdiagrammen in der 7.Klasse eines Gymnasiums erfolgreich eingesetzt [Hubwieser96a]).

Datenflußmodelle können als Beschreibung der Termstruktur und der Funktionalität von Funktionen betrachtet werden. Das Modell aus Abb.4 beschreibt etwa die Funktion *Addition(b1, b2)* $=_{def}$

(Zähler(b1)(kgV(Nenner(b1),Nenner(b2)):Nenner(b1)) +*
+ Zähler(b2)(kgV(Nenner(b1),Nenner(b2)):Nenner(b2)),*
kgV(Nenner(b1),Nenner(b2)))

Nach einer derartigen Einführung können in den folgenden Klassen Funktionsdeklarationen wie *Addition(b1: Bruch, b2: Bruch)* problemlos verwendet werden. Die Schüler stellen sich dabei einfach eine entsprechend aufgebaute Maschine (Datenflußdiagramm) vor.
Auch modulare Strukturen informationsverarbeitender Prozesse können hier sehr gut ver anschaulicht werden.

Die Grenzen dieser Beschreibungstechnik werden vor allem dort deutlich, wo es um die Kommunikation zwischen dynamisch erzeugten Objekten geht [Broy94b].

Aktionsgraphen

Der Einblick in die genaue "innere" Struktur von informationsverarbeitenden Abläufen kann durch Zerlegung von Prozessen in atomare Ereignisse ermöglicht werden. Dafür eignen sich Aktionsdiagramme, das sind gerichtete Graphen, deren Knoten Ereignisse und deren Kanten die Kausalbeziehung "ist Voraussetzung für" symbolisieren. Insbesondere können die wechselseitigen Abhängigkeiten nebenläufiger Prozesse gut veranschaulicht werden [Broy94a].

In Abb. 5 ist als Beispiel das Verhalten zweier Züge, die gleichzeitig im Uhrzeigersinn auf einem Gleissystem wie in Abb. 5 verkehren, beschrieben (Zug1 befährt Gleis1, Zug 2 Gleis2):

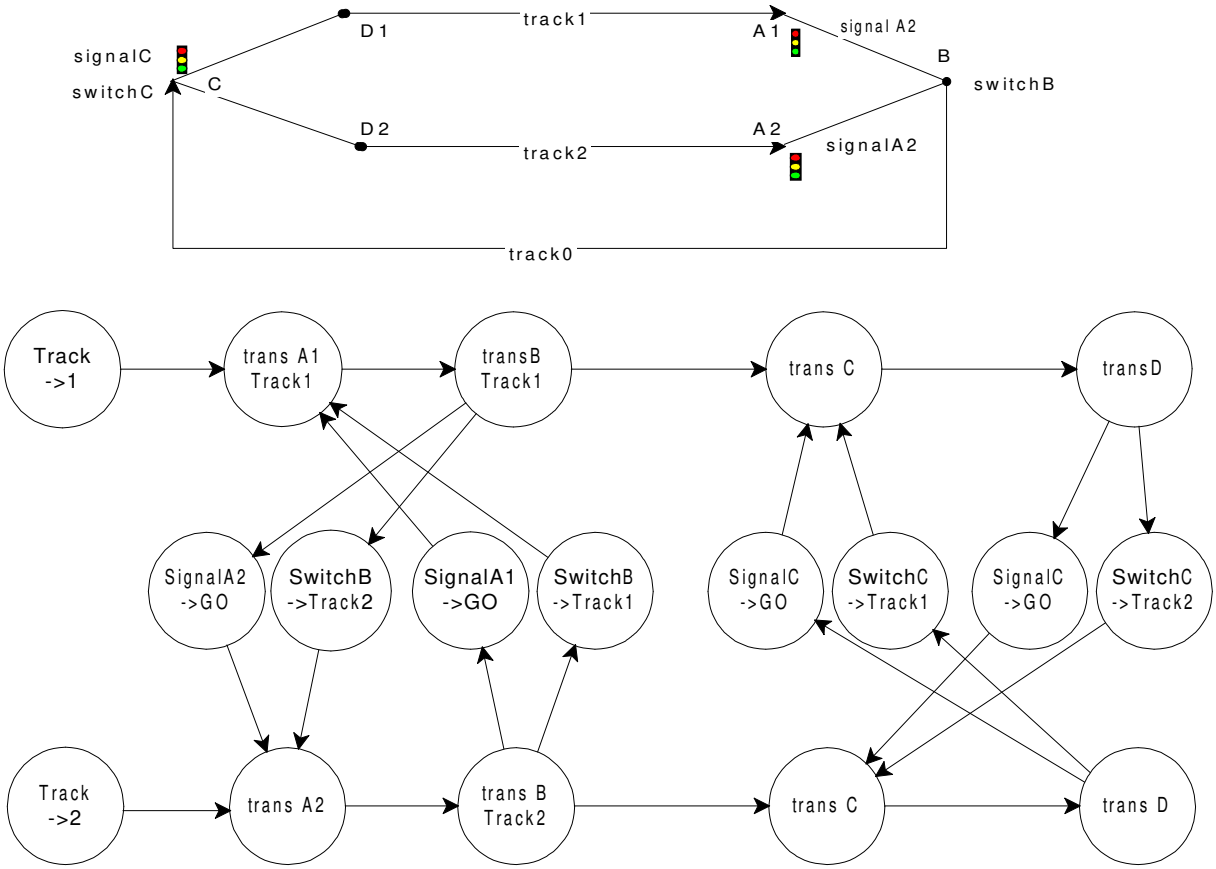

Abbildung 5: Gleisplan und Aktionsdiagramm für eine Eisenbahnanlage

Zusammenfassung der einzelnen Sichtweisen

Nach [Rumbaugh+91], [Goos96] teilen wir die Ergebnisse der einzelnen Beschreibungstechniken in ein *Objektmodell*, ein *funktionales Modell* und ein *dynamisches Modell* auf (siehe Abb.6). Das *Objektmodell* beschreibt mithilfe von Objektdiagrammen die Einteilung in Subsysteme, deren Eigenschaften und Verhalten sowie Kommunikation und Relationen zwischen den Subsystemen. Das *funktionale Modelle* besteht aus Datenflußdiagrammen und enthält Aussagen über Datenflüsse zwischen informationsverarbeitenden Prozessen, während sich das *dynamische* Modell aus Zustands-Übergangsdiagrammen, Aktionsdiagrammen und Message-Sequence-Charts zusammensetzt und die Einzelheiten dynamischer Vorgänge innerhalb der Objekte und zwischen den Objekten beschreibt. Message-Sequence-Charts beschreiben

exemplarische Abläufe von Prozessen und sind daher zur Erarbeitung allgemeiner Merkmale der Systeme im Unterricht nicht so geeignet. Ihr Einsatz wäre eher in einer Simulationsphase vorstellbar, wo die Ergebnisse der Beschreibungen mit den Anforderungen verglichen werden.

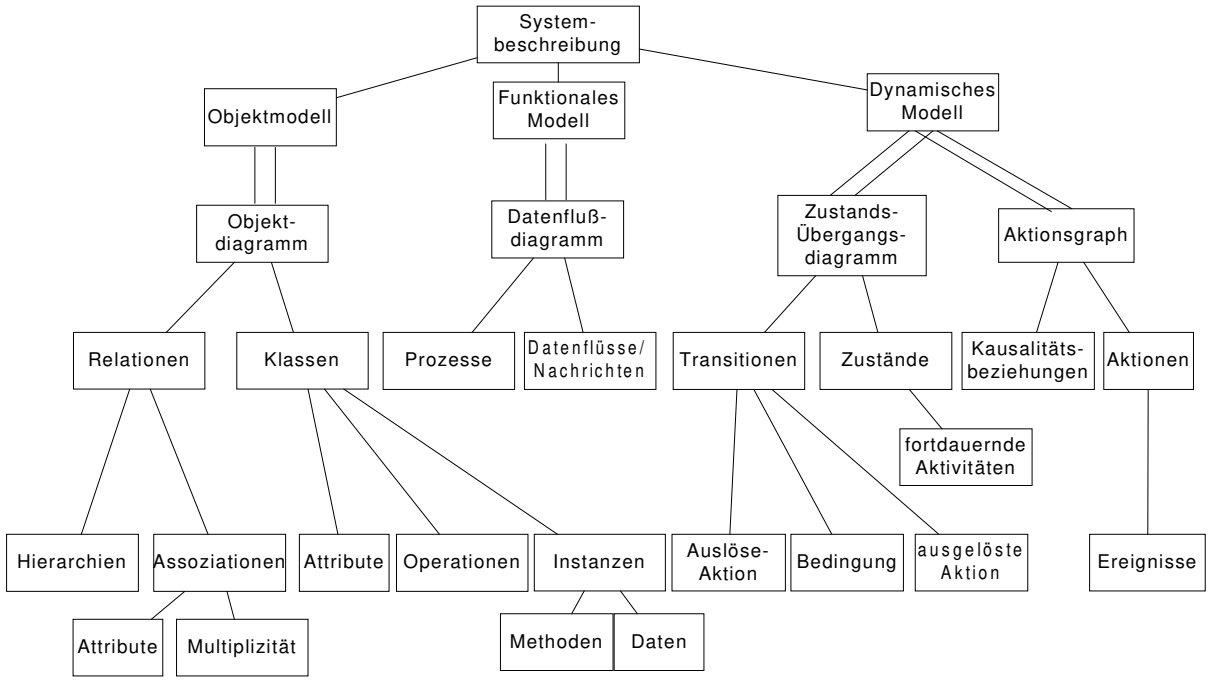

Abbildung 6: Zusammenfassung der Beschreibungstechniken

3.4 Lernziele

Zur weiteren Verdeutlichung unseres Unterrichtsentwurfs geben wir exemplarisch einige mögliche Lernziele auf verschiedenen Ebenen an. So sollen die Schüler etwa

• komplexe Sachverhalte überschauen und strukturieren können,

• Problemlösungsstrategien in unterschiedlichen Stilen beherrschen,

• sich bewußt sein, daß die schnelle Entwicklung von informationsverarbeitenden Techniken nur mit Hilfe übergeordneter Konzepte beherrscht werden kann, insbesondere verschiedene Modellierungstechniken für Informationssysteme kennen und anwenden können,

• verschiedene Repräsentationsformen von Information interpretieren, beurteilen und vergleichen können,

• in der Lage sein, einen Überblick über die Vielfalt der angebotenen Informationsquellen zu geben, diese zu nutzen und zu beurteilen,

• für gegebene Lösungsansätze jeweils das geeignete elektronische Hilfsmittel auswählen und benutzen können,

• realisierte Lösungen im Hinblick auf deren Kosten, Leistungsfähigkeit und deren innerbetriebliche oder gesellschaftliche Auswirkungen abschätzen können.

3.5 Lerninhalte

Wie die oben aufgeführten beispielhaften Lernziele dienen die folgenden Vorschläge für Lerninhalte hauptsächlich dem Zweck, den verfolgten didaktischen Ansatz näher zu charakterisieren und seine Grenzen zu verdeutlichen. Dabei denken wir nicht etwa an einen bloßen Transfer von Inhalten des Informatikstudiums in den Schulunterricht, sondern an eine alters- und schülergemäß konzipierte Darbietung von ausgewählten Themen aus den unten aufgeführten Bereichen, die vor dem Einsatz im Unterricht einer ausführlichen didaktischen Aufbereitung bedürfen. Wir schlagen als Inhalte eines Informatikunterrichtes am Gymnasium vor:

Information und ihre Repräsentation
Analoge und digitale Darstellungen von Bild und Ton, Texte, Hypertextsysteme, Informationsverluste, Mehrdeutigkeiten, Zusammenhang von Verarbeitungszeit und Speicheraufwand

Mathematische Grundlagen der Informatik
Logik, Relationen, Funktionen, Funktionale, Rekursion

Modellierungstechniken der Informatik
Objektmodelle, dynamische Modelle, funktionale Modelle, Entity-Relationship-Modell, Automatenmodelle, Ablaufdiagramme,

Prozesse
Ereignisse, Aktionen, Kausalität, Abläufe und ihre Darstellung, Zustandskonzepte

Datenstrukturen
Aufbau aus Sorten und Funktionen, Verbund, Varianter Verbund, Sequenz, spezielle Sequenzen (Schlange, Keller), Graphen, spezielle Graphen (Bäume)

Repräsentation - Interpretation - Semantik
Aussagen und Aussageformen, Boolsche Terme, Zahlen und Zeichensequenzen, Rechenstrukturen, Äquivalenztransformationen von Repräsentationen

Formale Sprachen - Syntax
Reguläre Ausdrücke, BNF-Notation, Erkennung natürlicher Sprachen

Netzwerke und Protokolle
Logische Strukturen von Netzwerken, einfache Beispiele für Kommunikationsprotokolle, Namenssysteme, Adressierungskonzepte, Transportkapazitäten, grundlegende Routing-Konzepte, Suchsysteme, Dienste in globalen Netzen

Aspekte des Datenschutzes
Personenbezogene Daten, Möglichkeiten zur Gewinnung von Informationen durch Kombination von Daten

4 Unterrichtsmethodik

Nach den Erfahrungen mit der Umsetzung abstrakter Konzepte wie Gruppentheorie oder Abbildungsgeometrie im Mathematikunterricht des Gymnasiums scheint eine Vermittlung der

genannten Inhalte und Techniken nur dann erfolgversprechend, wenn durch konkrete, anschauliche Problemstellungen eine erhöhte Aufnahmebereitschaft der Schüler geschaffen wird. Um Oberflächlichkeit zu vermeiden, sind die Modellierungen zudem möglichst vollständig auszuführen. Es bietet sich als Grobstruktur deshalb eine Einteilung in projektorientierte Unterrichtssequenzen an, die zum Beispiel aus jeweils 5-10 Unterrichtsstunden bestehen könnten.

4.1 Ablauf einer typischen Unterrichtssequenz

Problemgewinnung
Zunächst erfolgt ein erster Kontakt mit einer Problemstellung aus der Praxis, die Notwendigkeit des Einsatzes von neuen Techniken der Informationsverarbeitung wird klar. Ziele dieser Phase sind Motivierung, Veranschaulichung, Wiederholung und Festigung von bereits Gelerntem. Der Lehrer wird hierbei naturgemäß im Mittelpunkt stehen, Lehrervortrag, Schülervortrag und Unterrichtsgespräch dominieren. Als Medien sind originale Begegnungen, Film- und Bildmaterial sowie Simulationen am Rechnernetz denkbar.

Informelle Problembeschreibung
Die Schüler versuchen informell, das Problem verbal oder graphisch möglichst ausführlich mit allen seinen Randbedingungen zu beschreiben. Erziehung zu planmäßigem Handeln, Training von Sorgfalt und Umsicht stehen im Vordergrund, die Schüler arbeiten alleine oder in Gruppen unter Beratung durch die Lehrkraft. Zur Unterstützung bieten sich Textverarbeitungsprogramme, wenn möglich im Netz, und Grafikprogramme an. In der Softwareentwicklung wäre das Ergebnis der entsprechenden Phase ein Pflichtenheft.

Formale Modellierung
Unter Einsatz der oben beschriebenen Modellierungstechniken sollen die Schüler Techniken der Informationsstrukturierung erlernen, zu Sorgfalt, Genauigkeit, systematischem Denken und Handeln erzogen werden, im Hinblick auf ein späteres Studium standardisierte Modellierungsverfahren kennenlernen, Einblicke in Methoden zum Entwurf komplexer Informationssysteme gewinnen sowie eine Förderung ihrer Abstrahierungsfähigkeit erfahren. Die Gruppe ist die beherrschende Sozialstruktur, typisch wäre etwa die Entwicklung unterschiedlicher Modellklassen durch einzelne Gruppen mit abschließender gemeinsamer Diskussion der Ergebnisse. Geeignetes Hilfsmittel bei der Erarbeitung der Diagramme könnte ein geeignetes Flow-Chart-Programm sein, das eine saubere Anordnung der Elemente auf dem Arbeitsblatt, leichte Korrekturen und eine Einbindung in ein Abschlußdokument ermöglicht. Zusätzlich könnten andere Grafikprogramme und Textverarbeitungen zum Einsatz gelangen.

Realisierung von Lösungsansätzen
Das Ziel ist vor allem eine Überprüfung und Veranschaulichung der Ergebnisse der Modellierungsphase, weniger die Erlangung profunder Kenntnisse über ein spezielles Programm- oder Programmiersystem. Die früher im Rahmen der ITG erworbenen Kenntnisse kommen den Schülern hier zugute, sie sind bereits in der Lage, ein Programmiersystem oder eine Palette von Standardsoftware zu bedienen. Auch hier sollte die Gruppenarbeit dominieren, etwa in der Programmierung einzelner Module oder in der Realisierung derselben Problemstellung mittels unterschiedlicher Standardsoftware.

Bewertung

Zur Wiederholung, Festigung, Einordnung und Förderung der Kritik- und Urteilsfähigkeit der Schüler folgt abschließend eine Bewertungsphase, die sich an Review-Techniken orientiert. In allgemeiner Diskussion werden die Beschreibungen und Realisierungen bewertet und mit Alternativen verglichen. Es werden Fragen beanwortet wie:

- Was könnte man verbessern ?
- Welche Teilprobleme wurden nicht gelöst ?
- Wo haben wir das Problem vereinfacht ?
- Welche alternativen Realisierungen gäbe es ?
- Welche ähnlichen Probleme können wir mit unserer Lösung behandeln ?

4.2 Medien zur Veranschaulichung der Strukturen

Zur Veranschaulichung der Ergebnisse unserer Beschreibungen sind alle Medien geeignet, die eine Implementierung der erarbeiteten Strukturen unter möglichst geringem Syntax-Aufwand zulassen. Dabei soll den Schülern immer bewußt sein, daß das Ziel lediglich die Veranschaulichung der modellierten Zusammenhänge ist und nicht das Erlernen der Feinheiten einer bestimmten Programmiersprache oder die perfekte Beherrschung eines Programmsystems.

Standardsoftware
Tabellenkalkulationen und Datenbanksysteme, letztere insbesondere in relationaler Ausprägung stellen geeignete Implementierungsmedien dar.

Hypertextsysteme
Mit Hypertext-Sprachen wie *Hypertext Markup Language* (HTML) bieten sich herausragende Gelegenheiten, die Schüler in Techniken der Informationsstrukturierung und -darbietung einzuführen. Die unmittelbare, "visuelle" Semantik der syntaktischen Konstrukte kann die Konzepte von Informationsrepräsentation und -interpretation hervorragend veranschaulichen, ohne auf komplexe Zustands- oder Auswertungskonzepte zurückgreifen zu müssen.

Programmiersprachen
Implementierungen mit Hilfe von Programmiersprachen sollten eher die Ausnahme als wie bisher die Regel im Informatikunterricht sein. Wenn dieser Weg schon eingeschlagen werden muß, dann sollte man zumindest eine Sprache wählen, die mit einer möglichst kompakten, einprägsamen Syntax auskommt. Funktionale Programmiersprachen sind wohl am ehesten geeignet, komplexere Datenstrukturen, insbesondere in rekursiver Ausprägung, zu veranschaulichen.

Um den Syntax-Overhead so schmal wie möglich zu halten, kann die Lehrkraft vorbereitend spezielle Module anfertigen, die sprachspezielle Konstrukte in problembezogene umsetzen, um so die wesentlichen, aus der Modellierung folgenden Strukturen zu betonen anstatt sie mit sprach- oder plattformspezifischen Details zu verschütten.

Code-Generatoren
Eine interessante Perspektive für den Unterricht liefern Code-Generatorsysteme, die aus formalen Beschreibungen auf sehr abstrakter Ebene lauffähigen Programmcode generieren ([Broy+96]). Damit wird den Schülern unmittelbar die Bedeutung der Modellierungsergebnisse vor Augen geführt. An den Hochschulen werden derzeit zahlreiche derartige Systeme entwickelt, die den Schulen kostengünstig überlassen werden könnten. Die Unterrichtswirksamkeit solcher Systeme könnte durch Unterstützung von Simulationen noch gesteigert werden. Die Schüler hätten dann die Möglichkeit, unmittelbar nach der Modellierungsphase deren Ergebnisse mit der Anforderungsbeschreibung zu vergleichen.

5 Exemplarische Projekte

Im folgenden führen wir einige Beispiele für Unterrichtssequenzen auf, deren Zweck vor allem die Veranschaulichung des unterrichtslichen Niveaus unserer Vorschläge ist.

5.1 Die Darstellung der eigenen Schule im WWW

JGSt.	9-11
Vorkenntnisse:	- Zustands-Übergangsdiagramme
Lernziele:	- Strukturierung von Informationen
	- Aufbau einer Hypertext-Sprache am Beispiel *Hypertext Markup Language* (HTML)
Medien:	- Rechnernetz
	- HTML-Editor
	- HTML-Browser

Eine Anwendung in HTML ist hervorragend geeignet zur Verabschaulichung unserer Anliegen, da HTML eine sehr einfache Syntax mit direkter "visueller Semantik" besitzt, eine "algorithmenfreie" Lösung ermöglicht und zudem die Strukturierung von Information als Unterrichtsziel außerordentlich deutlich macht.

Problemgewinnung
Die Schule soll über eine Reihe von HTML-Dokumenten im *World Wide Web* (WWW) repräsentiert werden. Wir sehen uns dazu einige andere WWW-Seiten an, möglichst aus dem nichtschulischen Bereich, um eine Beeinflussung der Schüler durch die Strukturierungen anderer Schulen zu vermeiden. Einführend werden dann einige Hypertext-Prinzipien erklärt und an Beispielen demonstriert.

- Hinter Unterstreichungen verbergen sich Verweise auf Adressen von anderen Dokumenten in standardisierter Form (*Uniform Resource Locators*).
- Es können Bild-, Ton-, Videodokumente in bestimmten Dateiformaten eingebunden werden.

Lernziel ist auf dieser Stufe lediglich ein Überblick über die Möglichkeiten von HTML, der den Schülern eine Ahnung von den Strukturierungsmöglichkeiten verschaffen soll.

Problembeschreibung (Stoffsammlung)
Welche Informationen wollen wir anbieten ? Wir kommen auf folgende Themen:

- die Geschichte unserer Schule,
- unser Schulgebäude,
- kulturelle Ereignisse an der Schule,
- Aktuelles und Neuigkeiten,
- Elternbeirat : Arbeit, Mitglieder, Mitteilungen,
- SMV: Arbeit, Mitglieder, Mitteilungen,
- Vorstellung des Lehrerkollegiums.

Wir versuchen dann, die geplanten Informationen zu gliedern (Medium: Textverarbeitung). Aus einer ungeordneten Informationsmenge entsteht eine Struktur, etwa

- Gebäude
- Geschichte
- Kulturleben an der Schule: Theatergruppe, Literaturtage, Projektwochen
- Institutionen der Schule: Elternbeirat, SMV, Schulleitung, Lehrerkollegium

Modellierung
Ein Zustands-Übergangsdiagramm gibt die Vernetzungsstruktur wieder, ein Objektmodell die Inhalte der jeweiligen Dokumente

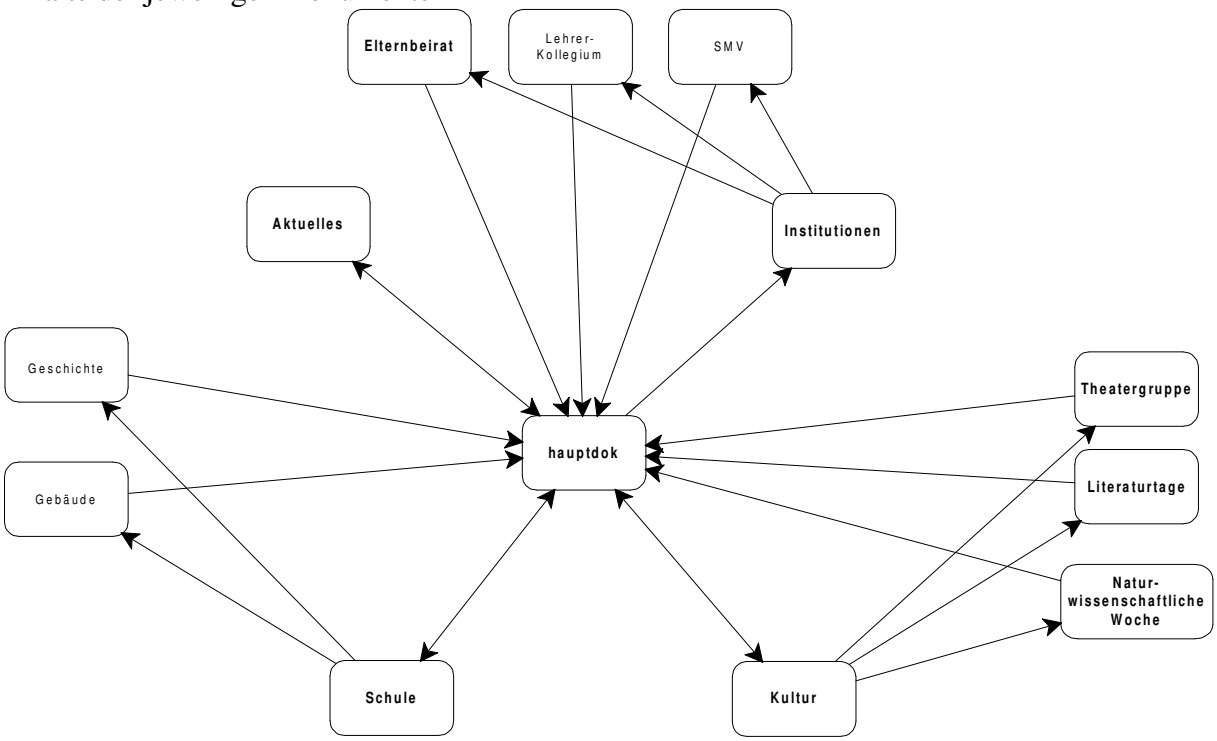

Abbildung 7: Dokumente einer HTML-Struktur als Zustände

Umsetzung
Einer genaueren Einführung in HTML anhand eines Beispiels folgt die Erarbeitung der einzelnen Dokumente in Gruppenarbeit sowie ein Test und eine Verbesserung der fertigen Dokumente. Denkbar wäre auch der Einsatz eines HTML-Systems, das eine interaktive Erstellung von Seiten ohne Syntax-Kenntnisse ermöglicht.

Wertung
Wir diskutieren Verbesserungs- und Ausbaumöglichkeiten, etwa den Einsatz von Formularen sowie die Unzulänglichkeiten der Lösung. Einer Nutzungsphase folgt die Auswertung der Kritiken, die etwa via e-mail zurückkommen. Schließlich kann man auf Unterschiede zu anderen Darstellungsformen wie Schaukästen, Zeitungsanzeigen o.ä. eingehen.

5.2 Zugsteuerung

Jahrgangsstufe:	11
Voraussetzung:	- Automatenmodell
	- Kenntnis der Grundlagen eines geeigneten Programmiersystems
Lernziele:	- Modellierung von Prozessen
	- Nebenläufigkeit, Synchronisation
	- Die Begriffe Ereignis, Aktion, Kausaliät
	- Unterschiede zwischen Kausalität und zeitlicher Abfolge

Problemgewinnung
Ideal zur Einführung wäre eine kleine Modelleisenbahnanlage an der Schule. Auch ein Video-Film der Anlage könnte genügen.

Wir betrachten die bereits oben in Abb. 5 beschriebene Situation: Ein Bahnhof besteht aus zwei parallen Gleisen (Gleis1, Gleis2), die nach je einer Signalanlage über eine Weiche A auf ein gemeinsames Gleis0 führen. Dieses erreicht über eine Schleife wiederum den Bahnhof, wo es nach Signal3 durch die Weiche C aufgeteilt wird.

Es sollen zwei Züge (Zug1, Zug2) gleichzeitig auf der Anlage verkehren, wobei wir von der folgenden Startsituation ausgehen: Signal1 steht auf FAHRT, Signal2 auf STOP, Signal3 auf FAHRT. Zug 1 steht auf Gleis1 vor A1, Zug 2 auf Gleis2 vor A2. Beide Weichen verbinden Gleis0 und Gleis1.

Problembeschreibung
Wir stellen mithilfe einer Textverarbeitung einen genauen Ablaufplan für einen Zyklus in freier Formulierung auf:

> Zug1: startet, schaltet Signal 1 und 2 auf STOP, passiert B, die Weiche W2 wechselt auf Gleis 2, Signal2 auf FAHRT, Zug erreicht C, passiert C, Signal3 schaltet auf STOP, Zug passiert D, Weiche W2 auf Gleis2, Signal3 schaltet auf FAHRT.
>
> Zug 2:startet, wenn Signal2 auf fahrt, schaltet Signal1 und 2 auf STOP, passiert B, die Weiche W2 wechselt auf Gleis 1, Signal1 auf FAHRT, Zug erreicht C, wartet bis Signal3 auf FAHRT und W2 auf Gleis2, passiert C, Signal 3 schaltet auf STOP, Zug passiert D, Signal 3 schaltet auf FAHRT.

Modellierung
Wir modellieren zunächst Signale, Weichen und Züge mittels Zustandsübergangsdiagrammen (siehe Abb.8).

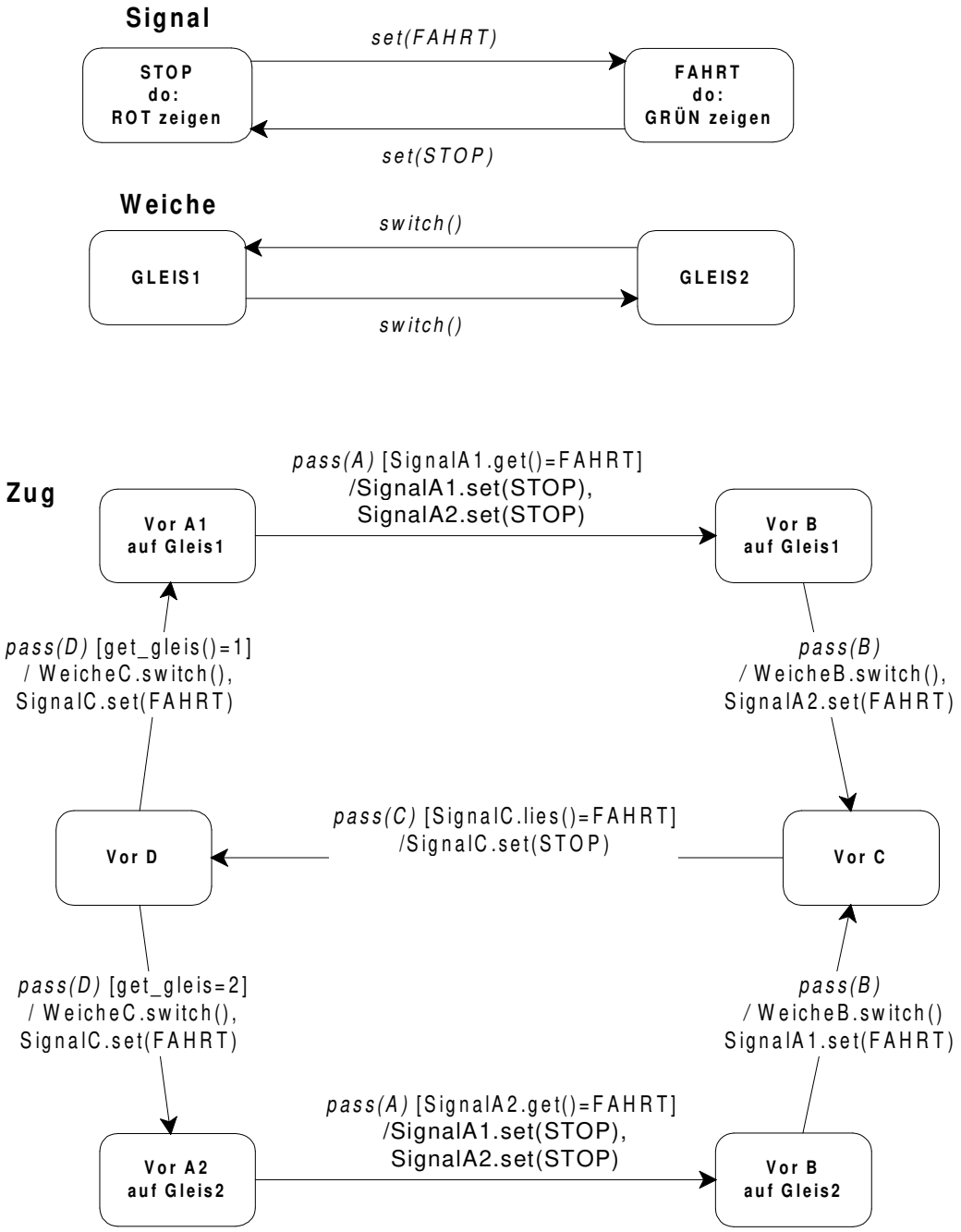

Abbildung 8 :Zustandsübergangsdiagramme

Dann definieren wir die benötigten Begriffe.

Aktion
Transitionsweg eines Automaten. Beispiel: a_1: set_S(3, FAHRT)
Eine Aktion kann mehrmals auftreten (stattfinden). Wir benötigen also ein weiteres KOnzept, nämlich

Ereignis
Ein einzelnes Auftreten (Stattfinden, Instanz) einer Aktion an einem bestimmten Ort zu einem bestimmten Zeitpunkt (vergeichbar wäre ein Punkt im Minkowski-Raum).
Beispiel: a_1 tritt im Lauf eines Umlaufzyklus zweimal auf, wir bezeichnen diese Ereignisse mit e_1, e_2, wobei gilt:

e_1: Signal 3 schaltet auf FAHRT nach Durchfahrt von Zug1
e_2: Signal 3 schaltet auf FAHRT nach Durchfahrt von Zug2

Ereignisse der gleichen Aktion haben zwar dieselben Auswirkungen (sie bewirken die gleichen Übergänge des jeweiligen Automaten), sie unterscheiden sich jedoch unter Umständen in ihren Ursachen (in ihren kausalen Vorbedingungen).

kausale Abhängigkeit
Das Ereignis e_1 ist Voraussetzung für das Ereignis e_2, kurz: $e_1 \rightarrow e_2$. Hier sollte eine deutliche Abgrenzung zum Konzept zeitlicher Abläufe erfolgen.

Prozeß
Ein Prozeß besteht aus einer Menge von Ereignissen, die untereinander in kausaler Beziehung stehen (ist eine Aktionsstruktur). Exemplarische nennen wir die einzelnen Ereignisse während eines Umlaufes eines der beiden Züge mit ihren kausalen Beziehungen.

Teilprozeß
Eine Teilmenge der Ereignisse eines Prozesses samt den zugehörigen "inneren" Kausalbeziehungen.

Mithilfe der Verbalbeschreibung des Problems stellen wir einen Aktionsgraphen auf, der bereits in Abb. 5 vorweggenommen wurde. Darin erkennen wir die beiden nebenläufigen Teilprozesse ZUG 1 und ZUG 2, die über Signal- und Weichenanlagen synchronisiert werden. Anhand des Graphen definieren wir dann weitere Begriffe.

Parallelität
Zwei Ereignisse $e_1$und e_2, die in keiner kausalen Beziehung zueinander stehen, heißen parallel. Es gilt also weder $e_1 \rightarrow e_2$ noch $e_2 \rightarrow e_1$. Beispiele wären etwa *pass(1,C)* und *pass(2,B)*.

Zwei Prozesse p_1 und p_2, deren Abläufe sich zeitlich überschneiden (können), heißen parallel. Es gibt also Ereignispaare aus p_1 und p_2 , die zueinander parallel sind.

Nebenläufigkeit
Zwei parallele Prozesse, die sich gegenseitig beeinflussen, heißen nebenläufig. Es gibt damit kausale Beziehungen zwischen Ereignissen aus p_1 und p_2. (z.B. ZUG 1 und ZUG 2 wegen *pass(1,B)* $\rightarrow$ *pass(2,A)*).

Verklemmung (von zwei Prozessen)
Ein Prozess wartet auf eine Aktion des anderen Prozesses, für die wiederum eine seiner Aktionen Voraussetzung ist. Beide Prozesse kommen damit in ihrer Ausführung nicht weiter.
Mathematische Definitionen der genannten Begriffe findet man unter anderem in [Broy94a].

Realisierung
Nun setzen wir unser Modell in die Praxis um. Ausnahmsweise haben wir eine Implementierung mittels einer Programmiersprache (*Turbo-Pascal*) gewählt, da dieses Beispiel sehr stark

ablauforientiert ist. Dabei sollte den Schülern allerdings ein vorgefertigter Modul (Unit SETPASS.TPU) mit den benötigten Routinen bereitgestellt werden, der sie von uninteressanten Syntax-Spezialitäten (wie Speichern und Laden von Dateien unter Vermeidung konkurrierender Zugriffe) abschottet. Wir achten darauf, daß je Aktion (Transition) eine Prozedur / Funktion zur Verfügung steht. Die Funktionalitäten lauten in *Pascal-Notation*:

Signal setzen:	procedure **set_Signal**(nrstr: string;state: string);
Signal abfragen:	function **read_Signal**(nrstr: string):string;
Weiche setzen:	procedure **set_Weiche**(nrstr:string;state:string);
Weiche abfragen:	function **read_Weiche**(nrstr: string):string;
Zug vorrücken:	function **pass**(nrstr: string; ort:char):char;

Voraussetzung für die Realisierung ist ein Betriebssystem, das zumindest quasiparallele Prozesse zuläßt (etwa *MS-Windows® mit* einem eigenen *MS-DOS®*-Fenster je Prozeß). Die Semaphoren realisieren wir dann mittels kleiner Dateien namens SIGNAL1, SIGNAL2, SIGNAL3, WEICHE1, WEICHE2.

Ansonsten benötigen wir noch eine Initialisierungsroutine (INIT) (und in der "Luxusversion" eine Statusinformationsanzeige SHOWSTAT). Die Zugautomaten werden durch ein gemeinsames Programm ZUG repräsentiert, dem mittels Kommandozeilenparameter die Zugnummer übergeben wird. Man beachte dabei besonders die Struktur des Programms ZUG.PAS:

- Aus dem Automatenmodell erhalten wir eine mehrfache Fallunterscheidung (case-Anweisung), wobei sich je Zustand ein Fall ergibt.
- Die kausalen Vorbedingungen aus dem Aktionsgraphen (abgesehen von der trivialen Bedingung, daß der jeweilige Vorzustand erreicht ist) liefern je eine if-Bedingung, die jedoch hier in den repeat-until-Warteschleifen versteckt sind.

Es folgen die Listings der Pascal-Programme, wobei die Existenz eines Units SETPASS.TPU vorausgesetzt wird, in dem die oben genannten Funktionen definiert werden.

```
{       Projekt ZUG:    INIT.PAS                                    }
{       (c) P.Hubwieser, München 1996                               }
{       Prozess zur Initialisierung der Signale und Weichen         }
program init;
uses crt,setpass;
begin
set_Signal('1','FAHRT');
set_Signal('2','STOP');
set_Signal('3','FAHRT');
set_Weiche('1','GLEIS1');
set_Weiche('2','GLEIS1');
end.
```

```
{       Projekt ZUG:    SHOWSTAT.PAS                                    }
{       (c) P.Hubwieser, München 1996                                   }
{       Prozess zur Anzeige des Systemzustands                          }
{       SIGNAL1-3, WEICHE1-2                                            }
program show_Signal;
uses crt,setpass;
var nrstr: string;
    ch:char;
begin
nrstr:=ParamStr(1);
clrscr;
while true do begin
      gotoxy(30,1);write('SIGNAL 1 -> ',read_Signal('1'),'   ');
      gotoxy(30,9);write('SIGNAL 2 -> ',read_Signal('2'),'   ');
      gotoxy(1, 3);write('SIGNAL 3 -> ',read_Signal('3'),'   ');
      gotoxy(50,5);write('WEICHE 1 -> ',read_Weiche('1'),'   ');
      gotoxy(1, 5);write('WEICHE 2 -> ',read_Weiche('2'),'   ');
      end;
end.
```

```
       Projekt ZUG:   ZUG .PAS                                        }
{       (c) P.Hubwieser, München 1996                                   }
{       Prozess für die Zugbewegungen                                   }
{       Aufruf mit ZUG x  (x = Zugnummer)                               }
program zug;
uses crt,setpass;
var ort: char;
    altstr,nrstr: string;

begin
nrstr:=ParamStr(1);
if nrstr='1' then altstr:='2'
             else altstr:='1';
ort:='A';
while true do
   case ort of
   'A': begin
        writeln('Zug ',nrstr,' hält vor ',ort);
        repeat
        until (read_Signal(nrstr)='FAHRT') and
             (read_Weiche('1')='GLEIS'+nrstr);
        set_Signal('1','STOP');
        set_Signal('2','STOP');
        ort:=pass(nrstr,'A');
        end;
   'B': begin
        ort:=pass(nrstr,'B');
        set_Weiche('1','GLEIS'+altstr);
        set_Signal(altstr,'FAHRT');
        end;
   'C': begin
        writeln('Zug ',nrstr,' hält vor ',ort);
        repeat
        until(read_Signal('3')='FAHRT')and
             (read_Weiche('2')='GLEIS'+nrstr);
        set_Signal('3','STOP');
```

```
        ort:=pass(nrstr,'C');
        end;
    'D': begin
        set_Signal('3','FAHRT');
        set_Weiche('2','GLEIS'+altstr);
        ort:=pass(nrstr,'D');
        end;
    end;
    delay(5000);
end.
```

Ablaufbeispiel

Nach dem Start der Programme INIT.EXE, SHOWSTAT.EXE, ZUG.EXE erhält man die in Abb. 9, 10 gezeigte Bildschirmausgabe.

```
                              SIGNAL 1 -> STOP
SIGNAL 3 -> STOP                                      WEICHE 2 -> GLEIS1
WEICHE 1 -> GLEIS1
                              SIGNAL 2 -> STOP
```

Abbildung 9: Ausgabe des SHOWSTAT-Prozesses

```
Zug 2 hält vor A
                                    Zug 1 hält vor A
Zug 2 passiert A
                                    Zug 1 passiert A
Zug 2 passiert B
                                    Zug 1 passiert B
Zug 2 hält vor C
                                    Zug 1 hält vor C
Zug 2 passiert C
                                    Zug 1 passiert C
Zug 2 passiert D
                                    Zug 1 passiert D
Zug 2 hält vor A
```

Abbildung 10: Ausgabe der ZUG-Prozesse

Wertung

Weitere Anwendungen könnten Arbeitsabläufe in Industriebetrieben, etwa die Fertigung von Kraftfahrzeugen auf parallelen Förderbändern o.ä. sein. Wir besprechen die Realisierungsmöglichkeiten unserer Lösungen. Welche Aufgabe haben die Männer im Stellwerk ? Welche Vor- und Nachteile hat die Umstellung auf elektronische Zugführung ? Wie kann man die Kombination Weiche / Signal verbessern ?

5.3 Bildverarbeitung

JGSt.: 9-10
Voraussetzungen: - Tabellenkalkulation (aus der ITG bekannt)
Lerninhalte: - Repräsentationsformen graphischer Information:
Pixelgrafiken, Vetorgrafiken
- Informationsverluste bei der Umwandlung der
Repräsentationsformen untereinander
- Manipulationsmöglichkeiten für photographische
Dokumente in Pixeldarstellung

Problemgewinnung

Wir simulieren ein Grafikprogramm mittels einer Tabellenkalkulation. Zur Vorbereitung stellen wir in einem Bereich der Tabelle Spaltenbreite und Zeilenhöhe so ein, daß die einzelnen Zellen am Bildschirm quadratisch erscheinen. Dann erstellen wir am linken und oberen Rand eine Zahlenfolge als Koordinatensystem. Nun sollen die Schüler zwei Kreise *k1(A;5)* und *k2(B;10)* mit *A(30;20)* und *B(20;20)* aus "Pixeln" zeichnen, wobei ein Pixel durch den Eintrag eines schwarzen Quadrats in eine Zelle simuliert wird. In einem ersten Versuch werden sie die Pixel manuell eintragen.

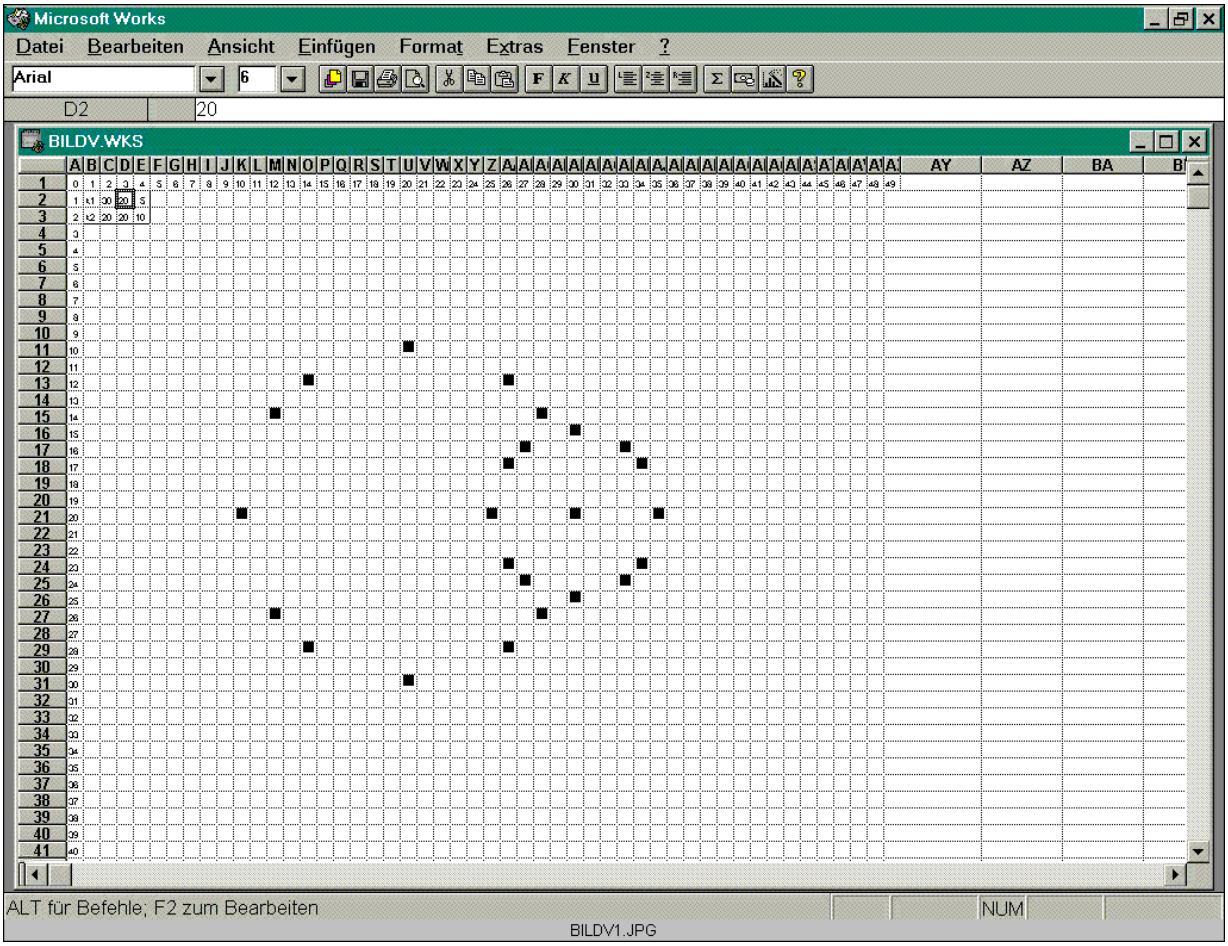

Abbildung 11: Pixelgrafik in einer Tabellenkalkulation

Die nächste Aufgabenstellung wirft dann Probleme auf: "Verschiebe den Kreis um B so, daß beide Kreise konzentrisch um den Mittelpunkt A liegen !" Die Lösung mittels Ausschneiden

und Kopieren beinhaltet die Schwierigkeit, daß die Ränder des kopierten Rechtecks mitgenommen werden. Teile des Kreises um A werden daher ebenfalls verschoben. Eine erneute Zeichung beider Kreise scheint auf den ersten Blick die einzige Lösungsmöglichkeit zu sein.

Wir versuchen einen zweiten Ansatz, indem wir die Pixel automatisch eintragen lassen, je nach Position der Zelle. Dazu benötigen wir zunächst Kreisgleichungen, bei Bedarf erfolgt eine Herleitung mithilfe des Satzes von Pythagoras. Es zeigt sich:

$$k: \qquad (x - mx)^2 + (y - my)^2 = r^2$$

Also soll ein Pixel gesetzt werden, falls gilt $P(x \mid y) \in k1$ oder $P(x \mid y) \in k2$, oder

$$(x - 30)^2 + (y - 30)^2 = 25 \ \vee \ (x - 20)^2 + (y - 20)^2 = 100$$

Dies übersetzen wir nun gemäß der Syntax der Tabellenkalkulation in einen bedingten Ausdruck (im Beipiel von MS-WORKS® in einen WENN-Ausdruck):

```
WENN(ODER((B$1-30)^2 + ($A1-20)^2 = 25;
(B$1-20)^2 + ($A1-20)^2 = 100);"n";" "),
```

das wir zunächst in eine Zelle eintragen, um es anschließend in alle noch freien Zellen zu kopieren.

Nun werden die Kreise zwar "automatisch" gezeichnet, das Verschieben eines Kreises ist allerdings immer noch mühsam: die Formel muß geändert und dann wieder in alle Zellen kopiert werden. Als Verbesserung wird vorgeschlagen, daß ein Bereich der Tabelle für den Eintrag der Radien und Mittelpunkte reserviert wird. Die Formel nimmt dann auf diese Einträge bezug:

```
WENN(ODER((F$1-$C$2)^2+($A2-$D$2)^2 = $E$2^2;
(F$1-$C$3)^2 + ($A2-$D$3)^2 = $E$3^2);"n";" ")
```

Nun kopieren wir das Ergebnis wieder in alle noch freien Zellen und können Kreise mit beliebigen Mittelpunkten und Radien auf Knopfdruck (Befehl: "Tabelle berechnen") zeichnen lassen.

Es gibt offensichtlich zwei Arten der Darstellung von digitalen Bildinformationen. Einerseits können wir für jeden Punkt der Grafikebene einen Farb/Grauwert festhalten (Pixelgrafik, Bitmap), andererseits die geometrischen Eigenschaften der darzustellenden Objekte definieren (Vektorgrafik).

Modellierung
Wir erarbeiten nun je ein Datenmodell für die beiden Möglichkeiten:

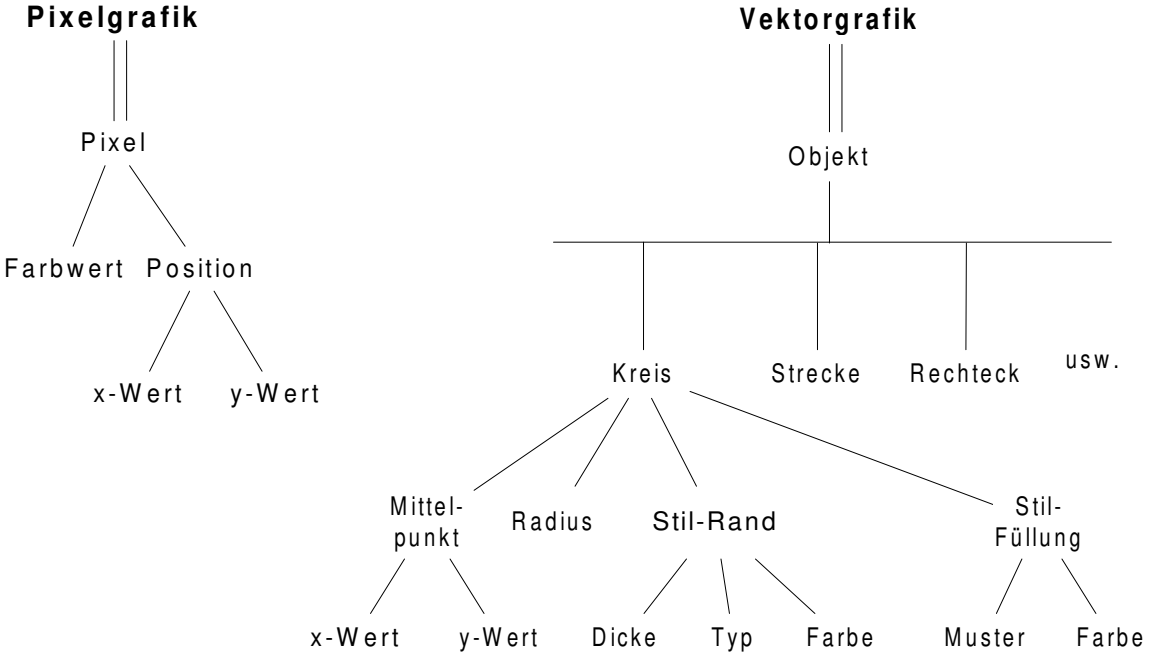

Abbildung 9: Datenmodelle für Grafiken

Anschließend besprechen wir die Eigenheiten der beiden Darstellungen.

Pixelgrafik
Man hat es immer mit einer festen Anzahl zu zeichnender Pixel zu tun (Bildschirm, Drucker-Raster,..). Deshalb kann man die Position aus der Lage des Pixel-Elementes in der Sequenz bestimmen. Falls das Bild *xmax* Spalten und *ymax* Zeilen hat, so liegt ein Pixel mit den Koordinaten *x,y* an der Stelle $s = y*(xmax) + x$. Umgekehrt ergeben sich die Koordinaten zu $y = s \text{ div } xmax$ und $x = s \bmod xmax$.

Der Farbwert kann unterschiedlich fein abgestuft sein (Farbtiefe), ein Bild mit 1014 x 768 Pixel beansprucht etwa in Abhängigkeit von der Farbtiefe folgenden Speicherplatz:

Farbtiefe	Anzahl der Farbstufen	Speicherplatz ca.
1 Byte	256	500kB
2 Byte	65536	1MB
3 Byte	17 Mio	1,5MB

Vektorgrafik
Ein einzelner Kreis in Vektorgrafik verbraucht dagegen kaum Speicherplatz. In unserem Datenmodell könnte man etwa 4 Byte je Komponente zugestehen und erhielte einen Platzbedarf von 8 x 4 = 31 Byte.

Zusätzlich zu den Objekten müssen bei den Vektorgrafiken allerdings Informationen über Bildgröße, Hintergrund, usw. mitgeführt werden, die zusätzlich Platz beanspruchen.

Jetzt sind wir in der Lage, die Umwandlungsmöglichkeiten zwischen den beiden Darstellungsarten zu besprechen:

Vektorgrafik -> Bitmap
Diese Richtung macht kaum Probleme. Der Algorithmus sieht ewa so aus:

Wiederhole für alle Spalten und Zeilen der Pixelgrafik:
Prüfe, ob Punkt zu einem Objekt gehört,
wenn ja, wende dessen Attribute auf den Punkt an
wenn nein, zeichne in Hintergrundfarbe.

Bitmap -> Vektorgrafik
Diese Richtung ist weitaus aufwendiger. Hier muß ein ausgeklügelter, zeit- und rechenintensiver Algorithmus das Pixelmuster auf das Vorhandensein bekannter Muster absuchen, diese identifizieren und ihre Attribute richtig setzen.

Realisierung
Wir stellen uns nun eine Aufgabe je Datenmodell. Wir wollen den Umkreis eines Dreiecks als Vektorgrafik produzieren und in eine Bitmap-Grafik umwandeln. Das Ergebnis bearbeiten wir mit einem Bitmap-Grafik-Programm, um die Unterschiede festzustellen und versuchen eine Rücktransformation in eine Vektorgrafik.

Als typische Bitmap-Anwendung wollen wir dagegen in einer Photo-Retusche ein Werbephoto für einen Urlaubsort "verschönern" (alternativ eine geplante Umgehungsstraße in eine vorher unberührte Landschaft einpassen oder ähnliches).

Vektorgrafik und Umwandlungen
Wir zeichnen ein Dreieck, drei Strecken und einen Kreis. Dann versuchen wir durch Verschieben, Drehen und Skalieren der Objekte, die gewünschte Situation annähernd zu realisieren. Hier werden die Möglichkeiten der Vektorgrafik verdeutlicht (Abb.13).

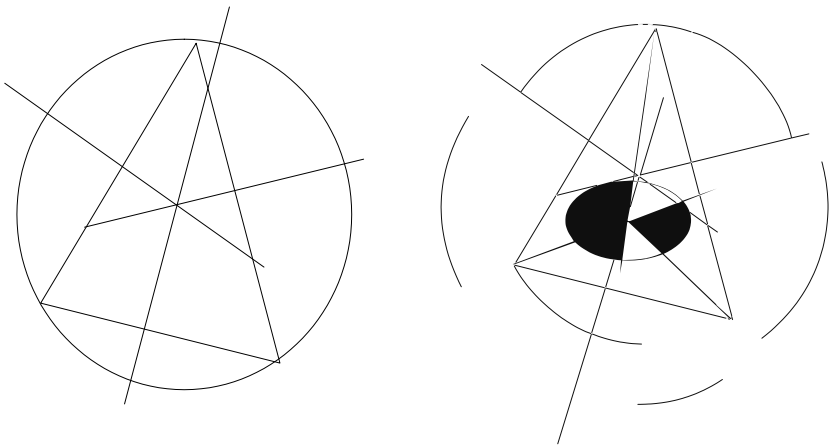

Abbildung 13: Vektorgrafik vor (links) und nach (rechts) Hin- und Rücktransformation in Bitmap-Grafik

Nach der Erstellung und Speicherung wandeln wir das Bild in eine Bitmap-Grafik mit 16-bit Farbtiefe um. Ein Vergleich des Dateiumfangs zeigt dann, daß die Pixelgrafik ungefähr den 50-fachen Platz auf der Festplatte belegt.

Nun laden wir unsere Bitmap-Grafik in ein Pixel-orientiertes Zeichenprogramm und versuchen, mittels der Winkelhalbierenden einen Inkreis einzuzeichnen. Wegen der mangelnden Korrekturmöglichkeiten wird dies auf Anhieb kaum gelingen.

Im nächsten Schritt vektorisieren wir unsere veränderte Bitmap-Grafik mittels eines geeigneten Programmes und speichern das Ergebnis. Wenn wir dieses nun wiederum in das Vektor-Zeichenprogramm laden, stellen wir fest, daß wir nun wieder Zugriff auf einzelnen

Objekte haben. Allerdings sind diese nun anders gruppiert, so sind die Kreise meist in Einzelteile zerlegt (Abb.13).

Pixelgrafik, Photoretusche
Mittels eines Scanners lesen wir das Photo einer griechischen Urlaubsortes ein und retuschieren den darauf abgebildeten monströsen Hotelkomplex weg. Eine Vektorgrafik wäre für dieses Bild ungeeignet, da es kaum identifizierbare geometrische Objekte aufweist
(Die obige Realisierung wurde mit Hilfe von *Corel-Draw®*, *Corel-Photo-Paint®* und *Corel-Trace®* erstellt).

Wertung und Ausblick
Abschließend fassen wir Vor- und Nachteile der beiden Darstellungsarten zusammen und besprechen spezielle Einsatzgebiete. Als Ausblick kann man noch mögliche Komprimierungsarten von Pixelbildern nennen und mit den entprechenden Formaten arbeiten.

6 Einbettung in ein Gesamtkonzept

Wie in [Hubwieser+96] beschrieben, versucht die Fakultät für Informatik der Technischen Universität München derzeit mit einer ganzen Reihe von flankierenden Maßnahmen, die Entwicklung des Informatikunterrichtes voranzutreiben.

Lehrerausbildung
In allen herkömmlichen Unterrichtsfächern ist es unbestritten, daß eine solide Hochschulausbildung eine Grundvoraussetzung für einen souveränen Unterrichtsstil ist. Deshalb sollte auch im Fach Informatik darauf hingewirkt werden, daß der Unterricht von einschlägig ausgebildeten Lehrkräften durchgeführt wird ([Burkert94]). Mit dieser Absicht wurde der Lehramtsstudiengang Informatik (vertieft und nichtvertieft) an der TU München konzipiert, der erstmals zum WS 1995/96 angeboten wurde und in sechs Semestern zum Staatsexamen in Informatik führen soll, was bis auf weiteres allerdings nur im Rahmen einer Erweiterungsprüfung möglich ist.

Zusätzlich führt die Technische Universität München in Zusammenarbeit mit dem Bayerischen Staatsministerium für Unterricht, Kultus, Wissenschaft und Kunst seit dem WS 1995/96 in einem Pilotversuch eine Gruppe von 11 bereits berufstätigen Lehrkräften innerhab von zwei Jahren zum Staatsexamen in Informatik. Diesem Kurs werden im WS 1996/97 zwei weitere mit je 30 Teilnehmern (an der Technischen Universität München und der Friedrich-Alexander-Universität Erlangen-Nürnberg) folgen.

Medien
Der Transport multimedialer Informationen findet in Zukunft wohl hauptsächlich über Datenautobahnen statt, deren Ausbaustufe einen der wichtigsten Faktoren für die Wahl von Wirtschaftsstandorten darstellen wird. Um den Schülern die Bedeutung dieser Medien zu vermitteln und sie zu einem kritischen, ökonomischen Gebrauch zu befähigen, muß zumindest eine Online-Demonstration der Möglichkeiten, der Vor- und Nachteile dieser Techniken im Unterricht möglich sein. Neben einer sinnvollen internen Rechnerausstattung und LAN-Vernetzung ist unserer Meinung nach deshalb ein Anschluß der Gymnasien an globale Netze nötig. In einer ersten Stufe kann dies über lokale Einwählknoten geschehen, die sich vor allem an den Fachhochschulen und Universitäten befinden könnten. Auf diese Weise könnte den meisten Gymnasien ein Internet-Zugang zum Ortstarif zur Verfügung gestellt

werden. Das Konzept "Bayern Online" [BSK95] der Bayer.Staatsregierung sieht einen entsprechenden Ausbau des Wissenschafts-Netzes für die Fachhochschulen und Universitäten - vor.

In einem Pilotversuch unter Beteiligung des Bayer.Staatsministeriums für UKWK, des Leibniz-Rechenzentrums der Bayerischen Akademie der Wissenschaften und der Fachhochschule Rosenheim initiierte die Technische Universität München einen Pilotversuch, in dessen Verlauf die technischen Erfordernisse und die Einsatzmöglichkeiten eines solchen Anschlusses für die Schulen im Landkreis Rosenheim untersucht werden sollen. Die dabei erworbenen Erkenntnisse sollen zur genauen Ermittlung des Bedarfes der Gymnasien dienen und den anderen Universitäten und Fachhochschulen bei der Erstellung einer möglichst einfachen, erprobten Lösung dienen.

7 Literatur

[Anderson85]
Anderson J.R.: Cognitive Psychology and Its Implications. Freeman & Co, New York, Oxford, 1985

[Arlt78]
W. Arlt (Hrsg.): EDV-Einsatz in Schule und Ausbildung. München, 1978

[BAL88]
Bayerische Akademie für Lehrerfortbildung: Informationstechnische Grundbildung. Dillingen/Donau, 1988

[Balzert80]
Balzert H.: Informatik. Vom Problem zum Programm. München, 1980

[Bauer79]
Bauer F.L.: Top down teaching im Informatikunterricht. In [Weinhart79]

[Baumann90]
Baumann R.: Didaktik der Informatik. Klett, Stuttgart, 1990

[Booch94]
Booch G.: Object-Oriented Analysis and Design. Benjamin / Cummings, Redwood City, CA., 1994

[Breier94]
Breier N.: Informatische Bildung als Teil der Allgemeinbildung. LOG IN 14 (1994) S.90

[Brenner+81]
Brenner A., Gunzenhäuser R.: Informatik, Didaktische Materialien für Grund- u.Leistungskurse, Stuttgart, 1981

[Broy+91]
Broy M., Dederichs F., Dendorfer C., Fuchs M., Gritzner T., Weber R.: The Design of Distrubuted Systems-An Introduction to FOCUS. TUM-I 9101, Technische Universität München, 1991

[Broy+93]
Broy M., Facchi C., Grosu R., Hettler R., Hussmann H., Nazareth D., Regensburger F., Slotosch O., Stoelen K.: The Requirement and Design Specification Language SPECTRUM. An Informal Introduction (Version 1.0). TUM-I 9311, Technische Universität München, 1993

[Broy94a]
Broy M.: Informatik. Eine grundlegende Einführung, Teil III. Springer, München, 1994

[Broy94b]
Broy M.: Modellierungstechniken. Syslab-Konzept. Fakultät für Informatik der Technischen Universität München, 1994

[Broy95]
Broy M.: Mathematics of Software Enineering. Invited talk at MPC 95. In: Möller, B. (Hrsg.): Mathematics of Program Construction, July 1995, Kloster Irsee, Lecture Notes of Computer Science 947, S. 18-47, Springer, 1995

[Broy+96]
Broy M., Geisberger E., Grosu R., Huber F., Rumpe B., Schätz B., Schmidt A., Slotosch O., Spies K.: Das AUTOFOCUS-Bilderbuch. Technische Universität München, 1996

[BSK95]
Bayerische Staatskanzlei: Bayern Online - Das Konzept. München, 1995

[BSK96]
Bayerische Staatskanzlei: Softwaretechnik. Ein Bericht des Wissenschaftlich-Technischen Beirats der Staatsregierung. München, 1996

[Burkert94]
Burkert J.: Umorientierung des Informatikunterrichtes. LOG IN 14 (1994) S.86

[Eickel95]
Eickel J.: Einführung in die Informatik I. Skriptum zur Vorlesung aus dem WS 1994/95. Technische Universität München, 1995

[Cyranek90]
Cyranek G.., Forneck H.J., Goorhuis H. (Hrsg.): Beiträge zur Didaktik der Informatik. Frankfurt a. Main, 1990

[Forneck90]
Forneck H.J.: Entwicklungstendenzen und Problemlinien der Didaktik der Informatik. In [Cyranek90]

[GI76]
Gesellschaft für Informatik: Zielsetzungen und Inhalte des Informatik-Unterrichtes. Zentralblatt für Didaktik der Mathematik, (ZDM), 8.Jg(1976) S.35

[Glaser95]
Glaser P.: Arbyter aller Länder! Süddeutsche Zeitung, Magazin 17 (1995) S.16

[Goos95], [Goos96]
Goos G.: Vorlesungen über Informatik, Band 1,1. Springer, Berlin Heidelberg, 1995,1996

[Hettler95]
Hettler R.: Entity/Relationship-Datenmodellierung in axiomatischen Spezifikationssprachen. Dissertation an der Fakultät für Informatik der Technischen Universität München, 1995

[Heuser95]
Heuser U.J.: Ungeliebte Infobahn. Die Zeit **33** (1995) S.11

[Hubwieser+96]
Hubwieser P., Broy M., Brauer W.: A New Approach in Teaching Information Technologies: Shifting Emphasis from Technology to Information. Case-study für die IFIP-Konferenz "Information Technology: Supporting Change Trough Teacher Education", Ramat Gan, Israel, 1996. (To appear)

[Hubwieser96a]
Hubwieser P.: Datenflußdiagramme im Algebraunterricht der Sekundarstufe I. (To be published)

[Hußman94]
Hußmann H.: Zur formalen Beschreibung der funktionalen Anforderungen an ein Informationssystem. Technische Universität München, 1994

[Jacobson91]
Jacobson I., Christerson M., Jonsson P., Övergard G.: Object-Oriented Software Engineering - A Use Case Driven Approach. Addison-Wesley, Reading (MA), 1991

[Jacobson+95]
Jacobson I., Ericsson M., Jacobsson A.: The Object Advantage. ACM Press, New York, 1995

[Knauer80]
Knauer W.: Informatik als Schulfach, Tübingen, 1980

[Koerber+88]
Koerber B., Peters I.R.: Grundlagen einer Didaktik der Informatik. FU Berlin, Sommersemester 1988, mss

[LISW87]
Landesinstitut für Schule und Weiterbildung: Neue Informations- und Kommunikationstechnologien, Soest, 1987

[Meißner71], [Meißner75]
Meißner H. (Ed.): Informatik I, II. Der Mathematikunterricht, **18** (1971) H.1 und **21** (1975) H.5

[Paulsen94]
Paulsen L.C.: Isabelle: A Generic Theorem Prover. Jgg. 818 of LNCS. Springer, Berlin, Heidelberg, New York, 1994

[Peschke89]
Peschke R.: Die Krise des Informatik-Unterrichtes in den neunziger Jahren. In: [Stetter+89]

[Robinsohn75]
Robinsohn S.B.: Bildungsreform als Revision des Curriculum. Neuwied, 1975

[Rumbaugh+91]
Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W. : Object-Oriented Modeling and Design. Prentice-Hall, Englewood Cliffs, NJ, 1991

[Sarnow96]
Sarnow P. : Schulen an das Netz. c't 4 (1996) S.80

[Stetter+89]
Stetter F., Brauer W.: Informatik und Schule 1989. Zukunftsperspektiven der Informatik für Schule und Ausbildung. Springer, Berlin, Heidelberg, New York, 1989

[Schulz-Zander78]
Schulz-Zander R.: Analyse curricularer Ansätze für das Schulfach Informatik. In [Arlt78]

[Schulz-Zander+93]
Schulz-Zander R. u.a.: Veränderte Sichtweisen für den Informatik-Unterricht. GI-Empfehlungen für das Fach Informatik in der Sekundarstufe II allgemein bildender Schulen. Informatik-Spektrum 16 (1993) S.349

[Schulz-Zander96]
Schulz-Zander R.: Eröffnungsrede zur 6.Fachtagung der GI zum Thema: "Informatik und Schule". Computer und Unterricht 11(1996) S.69

[Weinhart79]
Weinhart K.: Informatik im Unterricht. München, 1979

[ZS93]
Zentralstelle für Computer im Unterricht: Das Fach Informatik am Gymnasium. Augsburg, 1993