Fundamentale Ideen: Eine Studie zur Methodologie der Informatik

Andreas Schwill
Fachbereich Mathematik/Informatik - Universität Paderborn
D-33095 Paderborn - Germany
email: schwill@uni-paderborn.de

Zusammenfassung: Jede Wissenschaft muß sich von Zeit zu Zeit fragen, welche begrifflichen und methodischen Grundlagen sie nutzt, um zu Erkenntnissen zu gelangen, und was ihre unverwechselbaren Denkweisen sind. Diese Aufgabe ist von der Informatik noch kaum registriert worden. Wir werden in diesem Aufsatz in einem ersten Schritt versuchen, die Methodologie der Informatik zu klären. Die Methodologie sei dabei charakterisiert durch gewisse langlebige Konzepte, sog. *fundamentale Ideen*, auf die sich große Bereiche der Informatik abstützen, die den informatischen Forschungsbestrebungen eine Richtung geben und die sich ferner an natürlichen kognitiven Prozessen orientieren und in der historischen Entwicklung der Informatik erkennbar sind.

1 Einleitung

Trotz einer nun mehr als 40-jährigen Entwicklungsgeschichte besitzt die Informatik noch immer eine erstaunliche Dynamik. Ein Anhaltspunkt hierfür ist die Vielzahl der innerhalb eines Zeitabschnitts beobachteten, prognostizierten oder geforderten Paradigmenwechsel, unter denen in letzter Zeit mindestens folgende im Gespräch sind:

- von der sequentiellen zur parallelen Verarbeitung mit autonomen und intelligenten Verarbeitungseinheiten, die miteinander kommunizieren (beobachtet von [B91]);
- vom Programmieren als Kunst zum Programmieren als Ingenieurwissenschaft;
- von der strukturierten (top-down-orientierten) zur objektorientierten Programmierung (prognostiziert von [Cl89]);
- von der imperativen zur deklarativen Programmerung;
- von einer struktur-/ingenieurwissenschaftlichen Sicht der Informatik, deren Forschungen sich um den Computer zentrieren, zu einer hermeneutischen Disziplin, in deren Mittelpunkt der Mensch und die technische Gestaltung seiner Interaktionen steht (gefordert von [C90]);
- im Bereich des Software-Engineering von der produktorientierten zur prozeßorientierten Software-Entwicklung, bei der ganzheitliche Gesichtspunkte unter Einbeziehung der Nutzer des evolutionär zu entwickelnden Systems eine besondere Rolle spielen (gefordert von [F87]).

Der Begriff des Paradigmas wurde zuerst durch T.S. Kuhn [K62] im Rahmen seiner Forschungen zur Wissenschaftsentwicklung durch Revolutionen einer breiten Öffentlichkeit zugänglich. Kuhn bezeichnet als Paradigma

> "allgemein anerkannte wissenschaftliche Leistungen, die für eine gewisse Zeit einer Gemeinschaft von Fachleuten maßgebende Probleme und Lösungen liefern" [K62, S. 10].

Wissenschaftsentwicklung vollzieht sich nach Kuhn nicht evolutionär sondern revolutionär: Ein Paradigma bestimmt für einen gewissen Zeitabschnitt (Phase der "Normalwissenschaft") das wissenschaftliche Vorgehen und wird *schlagartig* durch ein anderes abgelöst, falls das bisherige zu Widersprüchen führt oder für die Weiterentwicklung der Wissenschaft nicht mehr tragfähig genug erscheint (*Grundlagenkrise*).

Die obigen und andere hier nicht genannte Paradigmenwechsel der Informatik haben bisher keine Grundlagenkrise ausgelöst. Sie wurden vielmehr stets mit einer erstaunlichen Gelassenheit aufgenommen und fügten sich organisch in die bisherigen Aktivitäten innerhalb der Informatik ein. Hierfür scheinen im wesentlichen zwei Gründe maßgebend zu sein:

1. *die inflationäre Verwendung des Begriffs "Paradigma" in der Informatik*.
   In der Informatik wird der Begriff des Paradigmas nur selten in der von Kuhn intendierten Bedeutung verwendet. Es erscheint sogar zweifelhaft, daß sich in der Informatik überhaupt schon *ein* anerkanntes Paradigma herausgebildet hat; die obige Beobachtung von gleichzeitig existierenden und teilweise miteinander konkurrierenden Strömungen belegt nach Kuhn eher, daß sich die Informatik noch in einer vorparadigmatischen Orientierungsphase befindet und ihr der Status einer "normalen Wissenschaft" auf der Basis *eines* Paradigmas erst bevorsteht[1]. Darauf könnten auch die vielen in der Informatik kursierenden Begriffsbildungen hindeuten, die auf "orientiert" enden [Cl89], wie anwendungsorientiert, algorithmenorientiert, objektorientiert, logikorientiert usw. Die o.g. Paradigmenwechsel sind jedenfalls keine wissenschaftlichen Revolutionen im Sinne Kuhns; vielmehr lassen sie sich meist auch als konsequente Weiterentwicklungen bestehender Konzepte auffassen (z.B. die objektorientierte Programmierung als Fortsetzung der strukturierten [M93]).
2. *die trotz ihrer stürmischen Entwicklung schon relativ stabilen Grundlagen der Informatik*.
   Die Robustheit, mit der eine sich noch orientierende Wissenschaft wie die Informatik auf die obigen, wenn auch relativ sanften Verschiebungen ihres Wissenschaftsgefüges reagiert, erstaunt dennoch. Es muß folglich schon ein gewisses stabiles Fundamentum existieren, auf dem das Gebäude der Informatik ruht und das von "Paradigmenwechseln" (zumindest der obigen schwachen Art) kaum oder gar nicht beeinträchtigt wird. Wie läßt sich dieses Fundamentum charakterisieren? Diese Fragestellung, für die in der vorliegenden Arbeit Lösungsansätze angeboten werden, ist eng verknüpft mit einer wissenschaftstheoretischen Analyse der gegenständlichen, begrifflichen und methodischen Grundlagen der Informatik. Solch eine "Theorie der Informatik" wurde zwar in der Vergangenheit bereits von verschiedenen Seiten mehrfach (allerdings zumeist mit

---

[1] P. Wegner [W83] meint allerdings umgekehrt, daß in einer stark interdisziplinär geprägten Wissenschaft wie der Informatik die "friedliche" Koexistenz mehrerer Paradigmen gerade *für* eine wissenschaftliche Reife spricht.

anderer Intention) gefordert (z.B. [L86,S90,Co89,L89]), dieser Forderung wurde jedoch zumindest mit der hier verfolgten Zielrichtung bisher nur in sehr geringem Umfang nachgekommen [D84,L86,G91,K89,F79].

In der vorliegenden Arbeit wollen wir daher aufbauend auf den obigen Vorüberlegungen zentrale Elemente der Methodologie der (Kern-)Informatik bestimmen, indem wir ausgehend vom Software-Entwicklungsprozeß typische Vorgehensweisen von Informatikern analysieren. Die Methodologie sei dabei charakterisiert durch gewisse langlebige Konzepte, sog. *fundamentale Ideen* (ein Begriff, dessen Ursprung in den Erziehungswissenschaften liegt), auf die sich große Bereiche der Informatik abstützen, die den informatischen Forschungsbestrebungen eine Richtung geben und die sich ferner an natürlichen kognitiven Prozessen orientieren und in der historischen Entwicklung der Informatik erkennbar sind.

Zunächst werden wir in Abschnitt 2 die Beiträge der Philosophie (vor allem von I. Kant) und der Erziehungswissenschaften (vor allem von J.S. Bruner) zur Klärung des Begriffs der fundamentalen Ideen und ihrer erkenntnistheoretischen und methodologischen Bedeutung für eine Wissenschaft analysieren. Aus dieser Untersuchung werden wir insgesamt fünf griffige Merkmale von fundamentalen Ideen ableiten und zu einer eigenen Definition zusammenfassen. In Abschnitt 3 entwickeln wir auf dieser Grundlage fundamentale Ideen der Informatik, indem wir ausgehend vom Software life cycle die einzelnen Aktivitäten bei der Erstellung von Software analysieren und zur Vorgehensweise in anderen Teilgebieten der Informatik in Beziehung setzen. Die ermittelten Ideen werden wir im Ansatz auf der Basis der angegebenen Kriterien als fundamental verifizieren.

2. Zur Definition fundamentaler Ideen

2.1 Der Ideenbegriff in der Philosophie

Der Begriff der Idee besitzt in der Philosophie eine lange Tradition und geht auf Plato zurück, wurde später u.a. von R. Descartes, J. Locke, D. Hume und G. Berkeley wieder aufgegriffen, jedoch einer neuen Bedeutung zugeführt. Erst Kant orientiert sich wieder am Platonischen Ideenbegriff. Wir wollen im folgenden die wichtigsten Eigenschaften von Ideen aufführen, soweit sie für die hier beabsichtigte Begriffsklärung von Bedeutung sind.

Plato.

Plato besitzt die abstrakteste Vorstellung von Ideen. Sie bilden hier neben der Realität eine eigene Klasse (einen "Kosmos") von reinen Gegenständen (eine höhere Wirklichkeit) die sich an einem himmlischen Ort befinden, also unabhängig vom Denken eines Einzelsubjekts existiert. Alle realen Dinge sind nur mehr oder weniger unvollkommene Abbilder ("Schattenbilder") der Ideen. Typische Beispiele für Ideen bei Plato sind die Ideen des Kreises, des Stuhls, der Gerechtigkeit, des Guten oder des Schönen, zu denen es in der Realität nur unvollkommene Nachbildungen gibt. Die Funktion von Ideen ist normativ: sie

dienen dem Menschen als Richtschnur, um sein Verhalten und seine Lebenswelt diesen Ideenvorstellungen anzunähern.
Alle Ideen sind angeboren und bilden die Grundlage aller menschlichen Erkenntnis. Jeder Mensch hat vor der Geburt die Gelegenheit, einen kurzen Blick in den "Ideenhimmel" zu werfen. Erkenntnis erfolgt dann nicht durch Neuerwerb einer Idee, sondern durch Wiedererinnerung.

Mit Descartes und in der Folge Locke, Leibniz, Hume und Berkeley verliert der Begriff der Idee seine kosmologische Prägung; vielmehr wird "Idee" mehr und mehr zu einer Bezeichnung für jeden Denkakt bzw. zu einem Synonym für "Begriff". Erst Kant trennt sich wieder von dieser Identifizierung.

Kant.
Die Analyse menschlicher Erkenntnisprozesse in der "Kritik der reinen Vernunft" bringt wesentliche Neuerungen und Präzisierungen des Ideenbegriffs und der Funktion von Ideen. Im folgenden wollen wir kurz auf die Hintergründe eingehen.

Überblick über die Kritik der reinen Vernunft.
Kant setzt die schon von Leibniz begonnenen rationalistischen Überlegungen fort. Gehen wir zur Erläuterung auf ein Bild J. Locke's zurück, wonach sich alle Erfahrungen in ein bei der Geburt zunächst unbeschriebenes "Wachstäfelchen" (=Bewußtsein) eingraben. Leibniz hatte zur Widerlegung Locke's bereits argumentiert, daß das Täfelchen eine ganz bestimmte Struktur besitzt, auf deren Grundlage überhaupt nur gewisse Erfahrungen möglich sind. Um im Bild zu bleiben: Erfahrungen können nur gemacht werden, wenn sie mit einem Stift in die Tafel geritzt werden. Signale, die auf anderem Wege, etwa akustisch oder optisch, das Täfelchen erreichen, werden nicht registriert und führen nicht zu Erfahrungen. Es muß also eine gewisse Übereinstimmung zwischen den erfahrunggebenden Signalen einerseits und den Sensoren sowie den Wahrnehmungs- und Verarbeitungsmechanismen des Bewußtseins andererseits vorliegen, um überhaupt Erfahrungen machen zu können. Diese Gemeinsamkeiten zwischen Sender und Empfänger analysiert Kant in der "Kritik der reinen Vernunft". Dabei kommt er zum Ergebnis, daß nicht die Erfahrung die Erkenntnis bestimmt, sondern daß umgekehrt die Erfahrungswelt weitgehend ein Produkt unseres Verstandes ist ("kopernikanische Wende" [K93, B XVI]). Alle Strukturen, die wir in der Erfahrung vorfinden, haben wir ihr selbst aufgeprägt ("Der Verstand schöpft seine Gesetze nicht aus der Natur, sondern schreibt sie dieser vor" [K57, §36]). Objektive Erkenntnis, die Wahrnehmung des "Dings an sich", ist folglich nicht möglich, da alle Erfahrungen durch das Bewußtsein ge- bzw. verformt werden, bevor sie Erkenntnismaterial werden.

Die Strukturen, die nach Kant unsere Erkenntnis ermöglichen und leiten, selbst aber nicht aus der Erfahrung gewonnen werden können, gliedern sich in drei Ebenen mit zunehmender Distanz von der objektiven Realität:

- die "reinen Anschauungsformen" (Raum und Zeit), durch die jede Wahrnehmung von Gegenständen geprägt ist, die selbst aber keine Eigenschaften von Gegenständen sind. In das Bewußtsein gelangen also nicht die wahrgenommenen Gegenstände "an sich", sondern nur ihre durch die Anschauungsformen geprägten *Erscheinungen*.
- die "reinen Verstandesbegriffe", sog. *Kategorien*, wie Quantität, Kausalität, Möglichkeit, Notwendigkeit usw., die das begriffliche Gerüst für jegliche Form menschlichen Denkens bilden; es sind reine Formen zur Konstituierung der Erfahrung.
- die "reinen Vernunftbegriffe", sog. *transzendentale Ideen*, wie Seele, Welt, Gott, die als idealisierte Zielvorstellungen das methodologische Instrumentarium zur Vermehrung der Erkenntnis bereitstellen. Diesen Aspekt werden wir im folgenden genauer vorstellen, weil er sich mit den hier verfolgten Absichten, den Ideenbegriff zu präzisieren, weitgehend deckt.

Ideen.

Kant löst sich von dem Ideenbegriff seiner Vorgänger und Zeitgenossen und schließt wieder an den Platonischen Ideenbegriff an, jedoch mit einigen bedeutenden Modifikationen. Zunächst unternimmt Kant mit Blick auf Locke und Hume eine sorgfältige Abgrenzung von Ideen zu Vorstellungen, Begriffen usw., um der "sorglosen Unordnung" zu begegnen, mit der gewöhnlich allerlei Vorstellungsarten als Ideen bezeichnet werden und "die Wissenschaft dabei einbüßen".

Kant definiert den Begriff der Idee in expliziter Form an vielen Stellen in seinen Werken, u.a. als

(1) "nichts anderes als der Begriff von einer Vollkommenheit, die sich in der Erfahrung noch nicht vorfindet" [K67],

(2) "Begriff aus Notionen, der die Möglichkeit der Erfahrung übersteigt" [K93, B377] (Notion="reiner Verstandesbegriff"),

(3) "notwendiger Vernunftbegriff, dem kein kongruierender Gegenstand in den Erfahrungen gegeben werden kann" [K93, B384].

Ideen sind also Produkte des reinen Denkens und in der Erfahrung, wie die Ideen Platos, nicht in der gedachten Form, höchstens als unvollkommene Abbilder anzutreffen.

Wie ergibt sich nun die methodologische Bedeutung der Kant'schen Ideen?

Jede menschliche Erkenntnis beginnt mit einer sinnlichen Wahrnehmung, die durch reine Anschauungsformen (Raum/Zeit) geprägt wird. Diese wird anschließend nach Maßgabe der reinen Verstandesbegriffe (Kategorien) unter dem Gesichtspunkt "was ist?" begrifflich strukturiert. Die reinen Vernunftbegriffe (Ideen) schließlich ordnen die Verstandesbegriffe unter den Gesichtspunkten "warum gilt?" oder "wie hängt zusammen?", und sie setzen dem

Verstand das Ziel, die größtmögliche Einheit (Systematik) des Erkenntnismaterials zu suchen. Diese Idee von der "absoluten Totalität der Bedingungen" [K93, B379] offenbart sich u.a. an folgenden Methoden der Wissenschaft oder allgemein des menschlichen Wissenserwerbs:

- an der Suche nach der Gesamtheit aller Bedingungen, die ein beobachtetes Ereignis ausgelöst haben (*Beispiel:* Versuch einer exakten Wettervorhersage).
- an der Suche nach der letzten, gleichsam unbedingten Bedingung einer Bedingungskette (*Beispiel:* Urknalltheorie).
- an der Suche nach der vollständigen Einheit der Teile in einem System, wobei hier im übertragenen Sinne die Teile eines Systems als Bedingungen des Systems aufgefaßt werden (*Beispiel:* Entwicklung einer vollständigen Systematik aller Lebewesen).

Die Idee von der Totalität der Bedingungen hat Kant weiter strukturiert und dabei drei zentrale Wirkungsbereiche ermittelt, an denen sich die Idee konkretisiert, den psychologischen (*Seele*), den kosmologischen (*Welt*) und den theologischen Bereich (*Gott*).

Zur regulativen Funktion von Ideen.

Sehr sorgfältig hat Kant den Zweck von Vernunftideen gegenüber Verstandesbegriffen abgegrenzt. Während Begriffe der Erkenntnis von Sachverhalten dienen, indem sie Wissen konstitutieren (durch Definitionen, Sätze, Beweise), besitzen Ideen eine *regulative Funktion*, indem sie den Verstand anleiten, seinen Erkenntnisbestand durch die Suche nach geeigneten Erfahrungen in Richtung auf die Ziele, die die Ideen beschreiben, zu erweitern. Problematisch werden Ideen jedoch, sobald man versucht, Erkenntnisse über sie zu gewinnen, sie also nicht regulativ sondern wie Begriffe konstitutiv zu verwenden: Betrachtet man die Ideen also statt als Erkenntnisziel als Gegenstände, über die man Aussagen machen, Schlüsse ziehen oder Beweise führen kann, so stößt man unweigerlich auf Widersprüche. Dies zeigt Kant auf mehrfache Weise, indem er aus einer Annahme, z.B. der, daß Bedingungstotalität nicht bloß Idee sondern tatsächlich gegeben sei, als auch aus ihrer Negation einen Widerspruch (Paralogismus, Antinomie) ableitet. Der Grund für diesen Widerspruch besteht in dem Versuch, Erkenntnisse über Gegenstände (Ideen) zu gewinnen, die sich außerhalb des Bereichs möglicher Erfahrungen befinden.

Fassen wir zusammen: Ideen sind idealisierte Vorstellungen, mit denen möglicherweise nicht erfahrbare Ziele verbunden sind; sie kanalisieren jedoch den menschlichen Forschungsdrang und leiten den Verstand an, seinen Erkenntnisbestand in Richtung auf das Ziel auszudehnen, ohne es womöglich jemals erreichen zu können.

2.2 Der Begriff der fundamentalen Ideen in der Pädagogik

Bedeutende Beiträge zur Klärung des Ideenbegriffs stammen aus der Pädagogik. Fundamentale Ideen sollen hier vor allem zur Verbesserung des Fachunterrichts an allgemeinbildenden Schulen dienen.

2.2.1 Hintergründe und Motivation

Bereits 1929 schlug A.N. Whitehead [W29] vor, sich im Fachunterricht

> "offenkundig auf unmittelbare und einfache Weise mit einigen wenigen allgemeinen Ideen von weitreichender Bedeutung"

zu befassen, denn:

> "Die Schüler stehen ratlos vor einer Unmenge von Einzelheiten, die weder zu großen Ideen noch zu alltäglichem Denken eine Beziehung erkennen lassen."

Im Jahre 1960 formulierte dann J.S. Bruner [B60] das didaktische Prinzip, wonach sich der Unterricht in erster Linie an den **Strukturen** der zugrundeliegenden Wissenschaft orientieren soll. Im deutsch-sprachigen Raum hat sich für diese Strukturen die Bezeichnung "**fundamentale Ideen**" durchgesetzt.

Bruner begründet seinen Zugang wie folgt: Lernen in der Schule hat vor allem den Sinn, die Schüler zu präparieren, das zukünftige Leben erfolgreicher zu bewältigen. Da kontrolliertes Lernen nach Anleitung - sieht man mal von der beruflichen Fort- und Weiterbildung ab - mit dem letzten Schuljahr oder Semester weitgehend abgeschlossen ist, können später eintretende Veränderungen im Privatleben, in Wirtschaft und Gesellschaft nur durch Übertragung (**Transfer**) früher erworbener Kenntnisse auf die neuen Situationen gemeistert werden.

Man kann diese Transferleistung vor allem hinsichtlich zweier Aspekte klassifizieren:

- Ähnelt die neue Situation einer bereits bekannten, so daß deren Lösungsschema nur geringfügig erweitert oder geändert werden muß, um auch auf die neue Situation anwendbar zu sein, so spricht man vom **spezifischen Transfer**. Spezifische Transferleistungen beziehen sich auf relativ lokale Effekte und werden überwiegend dann gefordert, wenn es um die kurzfristige Anwendung handwerklicher Fertigkeiten innerhalb eines begrenzten Fachgebiets geht.
- Beim **nichtspezifischen Transfer**, der sich auf langfristige (i.a. lebenslange) Effekte bezieht, lernt man anstelle von oder zusätzlich zu handwerklichen Fertigkeiten grundlegende Begriffe, Prinzipien und Denkweisen (sog. **fundamentale Ideen**). Ferner bildet man Grundhaltungen und Einstellungen aus, z.B. zum Lernen selbst, zum Forschen, zur Wissenschaft, zu Vermutungen, Heuristiken und Beobachtungen, zur eigenen Leistung, zur Lösbarkeit von Problemen usw. Später auftretende Probleme können dann gewissermaßen als Spezialfälle dieser Grundkonzepte, als Varianten eines vertrauten Themas, erkannt, eingeordnet und mit den zugehörigen Lösungsverfahren in

transferierter Form behandelt werden. Während der spezifische Transfer also etwas Neues direkt auf etwas Bekanntes derselben logischen Ebene zurückführt, bezieht der nichtspezifische Transfer eine Metaebene ein (Abb. 1).

Abb. 1: Nichtspezifischer Transfer

In Berufsschulen und in der betrieblichen Fort- und Weiterbildung dominiert der spezifische Transfer. Die vermittelten Fertigkeiten werden überwiegend nicht in einer Weise behandelt, die bei den Lernenden fundamentale Ideen ausbildet. Diese Fertigkeiten können daher i.a. nur durch spezifischen Transfer auf neue Probleme und Situationen übertragen werden. Dem gegenüber steht der nichtspezifische Transfer im Zentrum des gesamten Bildungsprozesses an allgemeinbildenden Schulen: Fortwährendes Erzeugen, Erweitern und Vertiefen von Wissen in Form fundamentaler Ideen. Daher - und jetzt kommen wir wieder auf die Ausgangsforderung Bruner's zurück - hat sich der Unterricht vor allem an den fundamentalen Ideen zu orientieren.

2.2.2 Charakterisierung des Ideenbegriffs in der Pädagogik

Über die Merkmale von Ideen herrscht in den Erziehungswissenschaften weitgehend Konsens, der zu einigen bemerkenswerten Definitionen geführt hat.

Bruner.

Bruner selbst gibt keine explizite Definition oder Charakterisierung des Begriffs, er überläßt es im wesentlichen dem Leser, sich anhand vieler Beispiele eine intuitive Vorstellung von fundamentalen Ideen zu verschaffen. Lediglich einige wenige Hinweise sind über das Buch verstreut, z.B.:

> "... daß die basalen Ideen ... ebenso einfach wie durchschlagend sind." [B60, S. 26];
> "... denn 'fundamental' bedeutet ... ja gerade, daß ein Begriff eine ebenso umfassende wie durchgreifende Anwendbarkeit besitzt." [B60, S. 31].

Fundamentale Ideen sind also in vielen Bereichen einer Wissenschaft umfassend anwendbar, und sie besitzen eine tragende Rolle, indem sie eine Vielzahl von Phänomenen ordnen und integrieren. Wir sprechen hier vom **Horizontalkriterium**. Hiermit ist

folgende Anschauung verbunden: Eine horizontale Achse, die eine Vielzahl von Wirkungsbereichen durchstößt (Abb. 2).

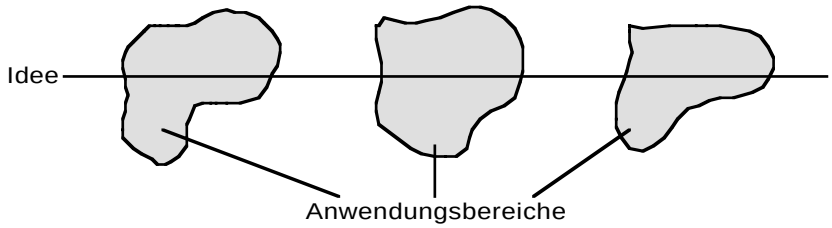


Abb. 2: Veranschaulichung des Horizontalkriteriums

Die folgende Aussage

> "Wenn früheres Lernen späteres Lernen erleichtern soll, dann muß es ein allgemeines Bild geben, das die Beziehungen zwischen den früher und den später begegnenden Dingen deutlich macht." [B60, S. 25]

zeigt in Verbindung mit der berühmten These Bruner's

> "... daß die Grundlagen eines jeden Faches jedem Menschen in jedem Alter in irgendeiner Form beigebracht werden können." [B60, S. 26],

daß fundamentale Ideen den Stoff innerhalb eines Anwendungsbereiches auch vertikal strukturieren: Eine fundamentale Idee kann auf nahezu jeder beliebigen geistigen Ebene (also einem Primarschüler ebenso wie einem Hochschüler) erfolgreich vermittelt werden (**Vertikalkriterium**). Unterschiede bestehen auf den verschiedenen Ebenen nur hinsichtlich des Niveaus sowie des Grads der Detaillierung und Formalisierung, mit dem die Idee präsentiert wird. Kontrapositiv formuliert (nach R. Fischer [F84]): Was nicht prinzipiell auch einem Grundschüler vermittelt werden kann, kann keine fundamentale Idee sein.
Das Vertikalkriterium ist von besonderer Bedeutung für die Ausbildung: Erfüllt eine Idee dieses Kriterium, so kann sie als Richtschnur verwendet werden, um den Unterricht auf jeder Ebene des gesamten Bildungsprozesses daran zu orientieren.

Fundamentale Ideen stehen also am unteren Rand intellektueller Fähigkeiten und müssen daher schon in frühen Stadien der Ausbildung des menschlichen Gehirns aufgenommen werden können. Wenn man dies akzeptiert, dann müssen solche Ideen mit der Entwicklungsstruktur unseres Gehirns in Einklang stehen; sie können also nicht in irgendeinem abstrakten Sinne "beliebig objektiv" sein oder sich als vom Menschen unabhängige "Naturgesetze" darstellen. Dies erinnert an die Kant'schen Überlegungen zum Verhältnis zwischen den Dingen "an sich", die uns verborgen bleiben, und ihren Erscheinungen. Ähnlich verhält es sich mit fundamentalen Ideen einer Wissenschaft: Sie sind nicht Elemente der Wissenschaft an sich, sondern Produkte unseres Verstandes, die wir der Wissenschaft aufprägen. Folglich können sie nur relativ zum Menschen objektiviert werden.

Visionär gesehen: Wollte man fundamentale Ideen der Informatik für *intelligente Maschinen* erarbeiten, so erhielte man vermutlich völlig andere Ansätze.

Anschaulich kann man den Anwendungsbereich einer fundamentalen Idee in verschiedene Ebenen mit wachsenden geistigen Anforderungen unterteilen. Eine fundamentale Idee ist dann durch eine vertikale Achse repräsentiert, die jede der Ebenen schneidet (Abb. 3).

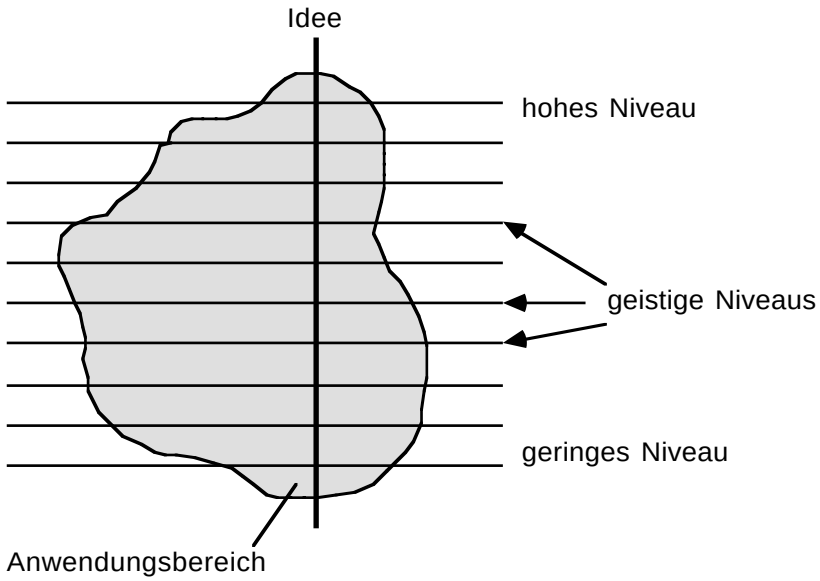


Abb. 3: Veranschaulichung des Vertikalkriteriums

Soweit Bruner's Kriterien für fundamentale Ideen. Im folgenden stellen wir noch die Ansicht zweier anderer Autoren vor, die sich zwar auf die Mathematik oder Teilgebiete davon beziehen, aber auch allgemeine Hinweise enthalten.

Schreiber.

A. Schreiber [S83] orientiert sich im wesentlichen an den Bruner'schen Überlegungen. Sein Beitrag besteht u.a. in der Angabe recht griffiger Anhaltspunkte für fundamentale Ideen der Mathematik, und zwar nennt er:

- **Weite**, d.h. logische Allgemeinheit. Gemeint ist hier wohl folgendes: Eine Idee besitzt Weite, wenn sie einen gewissen Spielraum für Interpretationen zuläßt und ein gewisses Maß an Anpassungsfähigkeit besitzt. Exakt formulierte Vorschriften, Axiome oder Regeln scheiden danach als Ideen aus.
  Beispiele: Das Kommutativgesetz der Addition a+b=b+a ist also keine Idee, da ihm die Allgemeinheit fehlt. Stattdessen könnte man die *Invarianz* als Idee mit Weite betrachten, der sich das Kommutativgesetz (Invarianz gegen Vertauschung von Operanden) wie auch viele andere Phänomene unterordnen. Ebenso sind hiernach die berühmte

Gleichung $E=mc^2$ oder das Quicksort-Verfahren keine fundamentalen Ideen. Die zugrundeliegenden Ideen sind vielmehr der Materie-Energie-Dualismus bzw. das Divide-and-Conquer-Verfahren

- **Fülle**, d.h. vielfältige Anwendbarkeit und Relevanz. Dieses Kriterium deckt sich im wesentlichen mit unserem Horizontalkriterium.
- **Sinn**, d.h. Verankerung im Alltagsdenken, lebensweltliche Bedeutung. Mit diesem Kriterium - wir sprechen im folgenden vom **Sinnkriterium** - geht Schreiber über die Bruner'schen Kriterien hinaus. Für ihn zählen fundamentale Ideen noch zur Sphäre des Alltagsdenkens. Ihr Kontext ist vortheoretisch, noch unwissenschaftlich. Die Präzisierung zu einem exakten Begriff steht einer Idee noch bevor. Man vergleiche hierzu auch das Zitat von Whitehead zu Beginn von Abschnitt 2.2.1.

  Beispiel: Das skizzierte Verhältnis zwischen einer Idee "mit Sinn" und einem Begriff besteht z.B. zwischen "Reversibilität" (d.h. Umkehrbarkeit von Operationen mit Wiederherstellung des Ausgangszustandes) als Idee und "Umkehrfunktion" als Begriff. Während "Umkehrfunktion" ein rein mathematischer Terminus ohne Bezug zur Alltagswelt ist, läßt sich Reversibilität an vielen Stellen im täglichen Leben aufzeigen.

Schweiger.

F. Schweiger [S82] erarbeitet in seinem Aufsatz fundamentale Ideen der Analysis. Seiner Ansicht nach sind fundamentale Ideen ein

> "Bündel von Handlungen, Strategien oder Techniken, sei es durch lose Analogie oder Transfer verbunden, die
> (1) in der historischen Entwicklung der Mathematik aufzeigbar sind,
> (2) tragfähig erscheinen, curriculare Entwürfe vertikal zu gliedern,
> (3) als Ideen zur Frage, was ist Mathematik überhaupt, zum Sprechen über Mathematik, geeignet erscheinen,
> (4) den mathematischen Unterricht beweglicher und zugleich durchsichtiger machen könnten. Weiters erscheint mir
> (5) eine Verankerung in Sprache und Denken des Alltags, gewissermaßen ein korrespondierender denkerisch sprachlicher oder handlungsmäßiger Archetyp, notwendig zu sein."

Schweiger's Ansichten weichen hier in mehrfacher Hinsicht von denen Bruner's und Schreiber's ab. Zwar führt er das Vertikalkriterium (Punkt (2)) an, erwähnt jedoch nicht das ebenso wichtige Horizontalkriterium. Andererseits hält er (wie Schreiber auch) einen Bezug fundamentaler Ideen zur Lebenswelt für erforderlich (Punkt (5)).

Zwei weitere Aspekte sind neu: der historische (Punkt (1)) und der philosophische (Punkt (3)). Der historische Aspekt - wir wollen fortan vom **Zeitkriterium** fundamentaler Ideen sprechen - ist aus zwei Gründen bedeutsam. Einerseits liefert er einen Anhaltspunkt, wie fundamentale Ideen zu gewinnen sind: Durch Beobachtung der geschichtlichen Entwicklung fachwissenschaftlicher Begriffe, Konzepte und Strukturen. Andererseits wird hiermit

angedeutet, daß fundamentale Ideen einer Wissenschaft längerfristig gültig bleiben. Diese Eigenschaft hat auch J. Nievergelt [N90] erkannt, wenn er sein "quest for classics" formuliert:

> "How do we recognize ideas of long lasting-value among the crowd of fads? The 'test of time' is the most obvious selector. Other things being equal, ideas that have impressed our predecessors are more likely to continue to impress our successors than our latest discoveries will." (S. 5)

Den philosophischen Aspekt verstehen wir weniger als Kriterium für, denn als Konsequenz einer ideenorientierten Wissenschaftsbetrachtung: Hat man eine Wissenschaft erst einmal durch das Aufstellen fundamentaler Ideen strukturiert, so ist man gleichzeitig im Besitz ihrer philosophischen Fundierung, weiß um ihr "Wesen" und kann sie gegen andere Wissenschaften abgrenzen.

Wir wollen nun als Ergebnis der Vorüberlegungen eine eigene Definition des Begriffs "fundamentale Idee" anschließen: Die fünf herausgearbeiteten Kriterien (Horizontal-, Vertikal-, Ziel-, Zeit- und Sinnkriterium) sind zu erfüllen, wobei die ersten beiden die Bedingungen der Fundamentalität sind und die drei anderen notwendig sind, um überhaupt von einer Idee sprechen zu können.

Definition:

Eine **fundamentale Idee** bezgl. eines Gegenstandsbereichs (Wissenschaft, Teilgebiet) ist ein Denk-, Handlungs-, Beschreibungs- oder Erklärungsschema, das

(1) in verschiedenen Gebieten des Bereichs vielfältig anwendbar oder erkennbar ist (**Horizontalkriterium**),
(2) auf jedem intellektuellen Niveau aufgezeigt und vermittelt werden kann (**Vertikalkriterium**),
(3) zur Annäherung an eine gewisse idealisierte Zielvorstellung dient, die jedoch faktisch möglicherweise unerreichbar ist (**Zielkriterium**),
(4) in der historischen Entwicklung des Bereichs deutlich wahrnehmbar ist und längerfristig relevant bleibt (**Zeitkriterium**),
(5) einen Bezug zu Sprache und Denken des Alltags und der Lebenswelt besitzt (**Sinnkriterium**).

Fundamentale Ideen im vergleichbaren Sinne wurden bisher vor allem für die Mathematik und eine Reihe ihrer Teilgebiete vorgestellt, u.a. für Wahrscheinlichkeitsrechnung, Analysis, lineare Algebra und analytische Geometrie, Numerik, Gruppentheorie und Geometrie [F76,H81,H75,K81,M80,S82,T79]. Weitere Ansätze, zumeist mit einem anderen Ideenverständnis, gibt es in Physik, Chemie, Biologie, Sozial- und Wirtschaftswissenschaften [S70,S81,G77,P65,M68].

3 Fundamentale Ideen der Informatik

Bei der Entwicklung einer Kollektion von fundamentalen Ideen der Informatik werden wir im wesentlichen nach folgendem Programm vorgehen, das von den Inhalten der Informatik zu ihren Ideen abstrahiert und ggf. iterativ mehrfach durchzuführen ist:

1. Schritt: Analyse konkreter Inhalte der Informatik und Ermittlung von Beziehungen und Analogien zwischen ihren Teilgebieten (Horizontalkriterium) sowie zwischen unterschiedlichen intellektuellen Niveaus (Vertikalkriterium). Dies liefert eine erste Kollektion von fundamentalen Ideen.

2. Schritt: Verbesserung und Modifikation dieser Liste durch Überprüfung, ob jede der Ideen auch eine lebensweltliche Bedeutung besitzt und im Alltag nachweisbar ist (Sinnkriterium).

3. Schritt: Nachzeichnung der historischen Entwicklung jeder Idee. So gewinnt man evtl. weitere Gesichtspunkte und stabilisiert die Ideenkollektion. Hierzu beachte man den Vorschlag von Nievergelt (Abschnitt 2.2.2).

4. Schritt: Abstimmung der Ideen untereinander und Analyse von Beziehungen zwischen ihnen: Besitzen die Ideen ein vergleichbares Abstraktionsniveau? Lassen sich die Ideen irgendwie strukturieren oder gruppieren? Bestehen hierarchische oder netzwerkartige Abhängigkeiten zwischen den Ideen?

Natürlich können wir nicht jeden der vier Schritte des Verfahrens im Detail nachvollziehen, uns genügt hier das Prinzip. Ebenso wollen wir nicht für jede der gewonnenen Ideen nachprüfen, ob sie alle Kriterien der Definition erfüllt. Hierzu sollen einige Beispiele genügen. Stets ist aber zu berücksichtigen, daß jede Liste ein gewisses subjektives Element enthält, weil die beteiligten Begriffe und Bedingungen nicht formal definiert sind und auch nicht formal definiert werden können (regulative versus konstitutive Funktion von Ideen), also immer Interpretationsfreiheiten lassen. Auch gibt es kein Kriterium, um nachzuweisen, ob eine Ideenkollektion alle Elemente der Wissenschaft vollständig und in angemessenem Umfang beschreibt. Diese Eigenschaft gewinnt sie allenfalls durch Diskussion und regelmäßigen Gebrauch innerhalb der Wissenschaft.

Zum 1. Schritt: Welche konkreten Inhalte der Informatik soll man auf Ideen untersuchen?
Eine zentrale Aufgabe der Wissenschaft Informatik ist es, den Softwareentwicklungsprozeß im weitesten Sinne zu erforschen und Methoden hierfür bereitzustellen. Folglich erscheint es sinnvoll, gerade die Tätigkeiten innerhalb dieses Entwicklungsprozesses auf Vorkommen von fundamentalen Ideen zu analysieren.

3.1 Softwareentwicklung

Die Grundlage für die Untersuchung von Fragestellungen bei der Softwareentwicklung bildet der Software life cycle (Standardform s. Abb. 4). Legt man andere Modelle, wie das

Wasserfallmodell, das Spiralmodell o.ä. zugrunde, so ändern sich die Aktivitäten dabei kaum, nur ihre zeitliche Abfolge [S89].

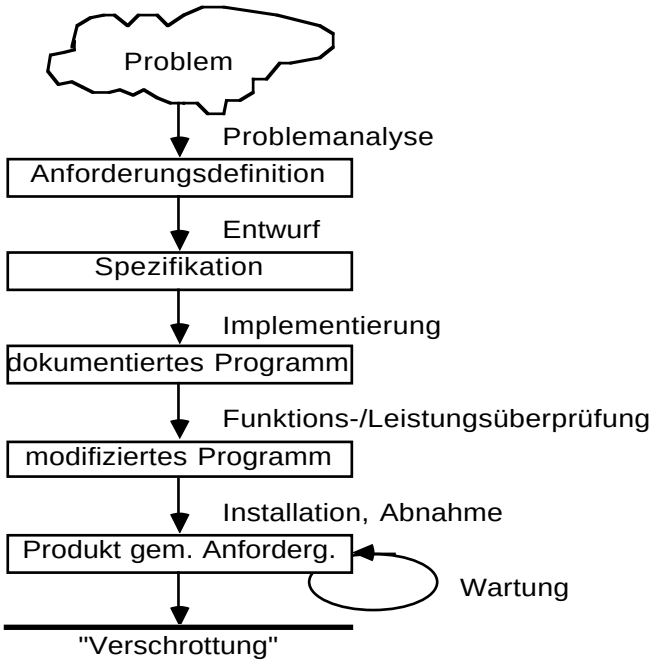


Abb. 4: Software life cycle

Gehen wir die einzelnen Phasen im Überblick durch.

Problemanalyse. In dieser Phase werden das zu lösende Problem und alle wichtigen Umgebungsbedingungen vollständig und eindeutig erfaßt. Ferner wird die Durchführbarkeit des Projekts untersucht. Die Phase gliedert sich in vier Teilphasen:

- Istanalyse: Untersuchung und Beschreibung des vorliegenden Systems durch Betrachtung seiner Komponenten, ihrer Funktionen und ihres Zusammenwirkens.
- Sollkonzeptentwicklung: Festlegung der Anforderungen an die zu erstellende Software durch Angabe u.a. des Benutzermodells, der Basismaschine, der Benutzermaschine usw.
- Durchführbarkeitsstudie: Sie liefert Erkenntnisse darüber, ob die Vorstellungen über das Softwareprodukt überhaupt realisiert werden können, ob sie prinzipiell durchführbar (Gibt es etwa Teile, die nicht oder nicht effizient berechenbar sind?) und sozial/ökonomisch vertretbar sind.
- Projektplanung: Erstellung von Zeitplänen, Zusammenstellung von Teams, Verteilung des Personals und der übrigen Hilfsmittel.

Zentrale Idee dieser Phase ist die *strukturierte Zerlegung:* Die verschwommenen Vorstellungen des Auftraggebers werden präzisiert und strukturiert zu Papier gebracht sowie auf Durchführbarkeit untersucht. Die Struktur des gegebenen Systems, in das das Softwareprodukt eingebettet werden soll, wird durch *Zerlegung* in Komponenten und

Ermittlung ihrer Beziehungen aufgedeckt. Üblicherweise erfolgt solch eine Analyse schrittweise, indem man zunächst eine grobe Zerlegung vornimmt und einzelne ihrer Komponenten weiter verfeinert bis ein hinreichend großer Detaillierungsgrad erreicht ist. So erhält man eine *Hierarchie* von Abstraktionsstufen. (Nicht nur) in dieser Phase ist die Aufprägung hierarchischer Beziehungen ein zentrales Strukturierungsmittel.
Die Durchführbarkeitsanalyse bezieht sich von seiten der Kerninformatik auf Berechenbarkeitsfragen und auf Überlegungen zur *Komplexität* im theoretischen Sinne. Damit zusammenhängende Ideen sind u.a. *Reduktion*, *Diagonalisierung*.

Entwurf. Während in der Problemanalysephase die Eigenschaften des Softwareprodukts ohne Hinweise auf ihre Realisierung beschrieben werden, entwickelt man in der Entwurfsphase ein Modell des Systems, das, umgesetzt in ein Programm, die Anforderungen erfüllt. Hierzu wird das komplexe Gesamtsystem fortlaufend in unabhängig voneinander realisierbare und in ihren Zusammenwirken überschaubare Einzelbausteine (Module) zerlegt und die Funktionen dieser Bausteine und ihre Schnittstellen spezifiziert. Üblich ist die hierarchische Modularisierung, für die es zwei verschiedene Reinformen gibt, die Top-down-Methode und die Bottom-up-Methode, die jedoch in der Praxis häufig gemischt werden. Um das Zusammenwirken der Module möglichst übersichtlich zu gestalten, sollte

- jedes Modul die Funktionen möglichst weniger anderer Module verwenden,
- die Schnittstelle jedes Moduls möglichst klein sein,
- jedes Modul möglichst viele Informationen über seine Struktur vor anderen Modulen verbergen (Geheimnisprinzip).

Das Ergebnis der Entwurfsaktivität ist die Spezifikation, in der Funktion, Schnittstellen und Hinweise zur Anwendbarkeit eines jeden Moduls, sowie ein Gesamtüberblick über die Abhängigkeit der einzelnen Module untereinander enthalten sind. Die Spezifikation kann graphisch oder sprachlich mit Spezifikationssprachen oder abstrakten Datentypen formuliert werden.

Im Zentrum der Entwurfsphase steht die Idee der *Modularisierung*, ebenfalls eine Form der *strukturierten Zerlegung*, mit ihren beiden Ausprägungen (*Top-down* und *Bottom-up*). Damit überhaupt modularisiert werden kann, müssen gewisse Bedingungen erfüllt sein: Die Wirkung jedes Moduls muß formal mit einer *Spezifikationssprache* definiert werden, die auch ein Konzept zur Kapselung (*Parameter-/Blockkonzept*) enthält, um die innere Struktur eines Moduls vor anderen Modulen *geheim* zu halten.

Implementierung. Erstellung eines lauffähigen Programms, das in seinem Ein-/Ausgabeverhalten der Anforderungsdefinition entspricht. Von besonderer Bedeutung in dieser Phase ist die Wahl einer Programmiersprache. Bei der Implementierung stehen die

einzelnen Module im Vordergrund. Um sie später einfacher testen und verändern zu können, sind u.a. folgende Prinzipien zu beachten:

- strukturierte Programmierung,
- klar definierte Schnittstellen, z.B. durch Parametrisierung,
- Verwendung semantisch einfacher Sprachelemente der Programmiersprache.

Wesentlich ist, daß sich die Module gemäß Entwurf als überschaubare Einheiten im Programm wiederfinden.

Kern der Implementierungsphase ist die Idee der *Algorithmisierung* (*Zerlegung* in Einzelschritte) und die anschließende Umsetzung des Algorithmus in ein *ablauf*fähiges Programm einer *Programmiersprache* bestehend aus *Kontrollstrukturen* (*Sequenz*, *Schleife*, *Alternative* usw.) und *Datenstrukturen* (*Aggregation*, *Generalisation* usw.).
Je nach zu lösendem Problem stehen für die Wahl der Algorithmen gewisse Grundmuster, häufig Paradigmen genannt, zur Verfügung, z.B. *Divide-and-Conquer*, *Branch-and-Bound* usw. Bei der Programmierung ist *strukturiert* vorzugehen. Damit die Module und ihre Beziehungen im Programm erkennbar werden, muß die Programmiersprache mindestens über ein *Block*- und ein *Parameterkonzept* sowie über Mittel zur graphischen oder sprachlichen Darstellung von *Hierarchien* (Klammern, begin...end, Einrückung) verfügen.

Funktionsüberprüfung. Anhand der Anforderungsdefinition wird das Ein-/Ausgabeverhalten des Programms durch eine Kombination von Verifikation und Testen überprüft. Man beginnt mit dem Modultest und prüft anhand der Spezifikation, ob jedes Modul das vorgeschriebene Funktionsverhalten besitzt. Nach dem Zusammenbau der getesteten und verifizierten Module beginnt der Integrationstest, dann der Installationstest und schließlich der Abnahmetest.

Zentrale Idee in dieser Phase ist die Qualitätskontrolle, d.h. die Untersuchung und *Bewertung* des fertigen Programms oder seiner Teile auf *Korrektheit*. Formal oder durch Testeingaben werden die Programmteile *partiell* und *total* verifiziert. Korrekte Bausteine werden durch Umkehr des vorangegangenen Zerlegungsprozesses zu größeren zusammengesetzt, die dann ihrerseits überprüft werden. Bei nebenläufigen Programmen spielen hierbei auch Ideen wie *Konsistenz* und *Fairneß* eine Rolle.

Leistungsüberprüfung. Nach der Korrektheitsüberprüfung müssen Leistungsmessungen durchgeführt werden.

Im Zentrum dieser Phase steht, ebenfalls unter dem Aspekt, die Qualität des Produkts zu *bewerten*, die Idee der *Komplexität*. Damit zusammenhängende Ideen sind u.a. *Ordnung*, *worst-case Laufzeit*.

Die letzten Phasen des Software life cycle bilden Installation, Abnahme und Wartung des Softwareprodukts. Neue bisher nicht genannte Ideen tauchen hierbei nicht auf.

3.2 Die Ideenkollektion

Unter den in Abschnitt 3.1 genannten Ideen fallen drei besonders auf, weil sie, gewissermaßen als phasenübergreifende Ideen, in allen Stadien der Softwareentwicklung eine herausragende Rolle spielen: Die Ideen der *Algorithmisierung*, der *Sprache* und der *strukturierten Zerlegung*. Diese Ideen - wir nennen sie *Masterideen* - wollen wir im folgenden genauer analysieren, da sie auch Anhaltspunkte für eine Reihe weiterer Ideen liefern.

Algorithmisierung.

Über die Bedeutung der Algorithmisierung als fundamentale Idee der Informatik besteht weitgehend Konsens. Mit dieser Idee verbindet sich die Zielvorstellung, alle Probleme ließen sich durch maschinell nachvollziehbare Verfahren, deren Korrektheit jederzeit gesichert ist, effizient lösen. Diese Zielvorstellung wird z.B. sichtbar

- an dem Versuch, die Algorithmenentwicklung selbst zu automatisieren,
- an der harten KI-These, die - unabhängig davon, ob man sie vertritt oder nicht - entsprechende Forschungen motiviert und ihnen eine Richtung gibt,
- an dem Versuch, auch nicht berechenbare Probleme durch Betrachtung immer größerer berechenbarer Teilklassen noch zu attackieren,
- an dem Versuch, nicht effizient lösbare Probleme durch andere Rechenmodelle (besser: Gedankenmodelle) doch noch effizient lösen zu können (z.B. nichtdeterministische Turingmaschinen, randomisierte Algorithmen, Orakel).

Eine genauere Analyse der Aktivitäten beim Algorithmisieren liefert noch eine Fülle weiterer Ideen. Wir unterscheiden vier große Bereiche: Beim Entwurf eines Algorithmus bedient man sich häufig der oben schon erwähnten Grundmuster, sog. Entwurfsparadigmen (auch hier nicht im Kuhn'schen Sinne), wie *Divide-and-Conquer*, *Backtracking* (für eine detaillierte Aufstellung s. [M83]). Diesen Entwurf setzt man anschließend in ein Programm um. Hierfür stehen gewisse gemeinsame Konzepte zur Verfügung, um den Daten- und Kontrollbereich zu beschreiben, wie z.B. *Konkatenation*, *Alternative*, *Rekursion* usw. Diese auf den ersten Blick imperativ geprägten Ideen finden sich auch bei funktionalen und prädikativen Programmiersprachen, etwa die Konkatenation als Komposition. Das fertige Programm wird auf einem (oder mehreren) Prozessoren ausgeführt. Damit verbunden ist die Idee des *Prozesses*, d.h. die Trennung von Beschreibung und Ausführung eines Algorithmus. Die letzte Ideengruppe im Rahmen der Algorithmisierung befaßt sich mit der Bewertung von Algorithmen unter Qualitätsgesichtspunkten. Die beiden zentralen Bewertungskriterien sind *Korrektheit* und *Komplexität*, jeweils mit einer Reihe weiterer zugehöriger Ideen.

Sprache.

Nicht nur bei der Programmierung (Programmiersprachen), bei der Spezifikation (Spezifikationssprachen), bei der Verifikation (Logikkalküle), in Datenbanken (Anfragesprachen), bei Betriebssystemen (Kommandosprachen) spielt die Idee der Sprache eine herausragende Rolle, vielmehr besteht in der Informatik eine generelle Tendenz zur Versprachlichung von Sachverhalten. Dies gilt auch für Bereiche, bei denen zunächst kein unmittelbarer Bezug zu einer sprachlichen Darstellung erkennbar ist, z.B. beim VLSI-Entwurf oder bei der Bildverarbeitung. Dieses Vorgehen bietet u.a. folgenden Vorteil: Es vereinheitlicht die Sichtweise, denn jedes Problem reduziert sich dann auf ein Problem über Wörtern; die algorithmische Manipulation von Sprachen und Wörtern ist andererseits gut erforscht.

Mit der Idee der Sprache verbindet sich die Zielvorstellung, daß sich alle informatikrelevanten Sachverhalte durch gewisse Zeichenfolgen beschreiben lassen, die nach einfachen Bildungsgesetzen aufgebaut sind und die untereinander auf effiziente algorithmische Weise semantikerhaltend transformierbar sind.

Beispiel: Eine Rechnerarchitektur veranschaulicht man sich häufig durch ein Ebenenmodell (Abb. 5) [T84]. Jeder Ebene wird ein anderes Sprachniveau zugeordnet, mit dem die Objekte und Aktionen der Ebene adäquat beschrieben werden können. Die Arbeitsweise des Rechners besteht in diesem Modell aus einer Folge von Übersetzungsprozessen, die die Benutzersprache auf der obersten Ebene sukzessive auf die unterste Ebene semantikerhaltend umsetzt.

In engem Zusammenhang mit Sprachen stehen offenbar die Ideen von *Syntax* und *Semantik*, ferner die verschiedenen Ansätze zur semantikerhaltenden Transformation von Wörtern einer Sprache in eine andere, z.B. die Ideen der *Übersetzung*, *Interpretation* oder *operationalen Erweiterung*.

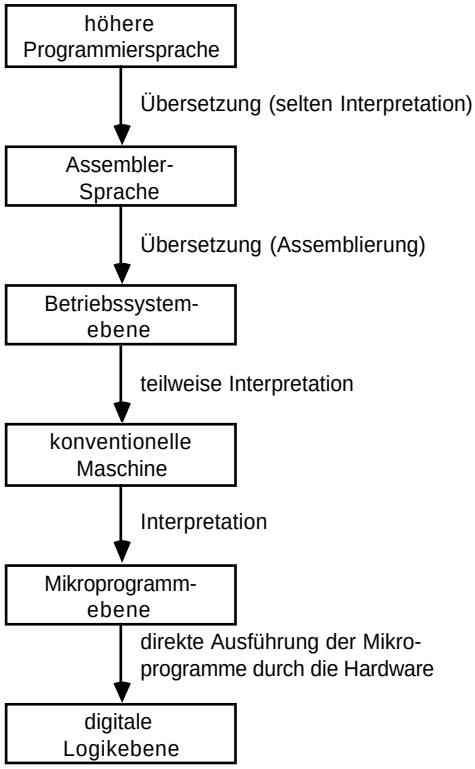


Abb. 5: Ebenenmodell der Rechnerarchitektur

Strukturierte Zerlegung.

Die Zerlegung konkretisiert sich im Software life cycle z.B.

- bei der Istanalyse an der wiederholten Zerlegung des bestehenden Systems in Komponenten und an der Bildung von Teams,
- beim Entwurf an der hierarchischen Modularisierung,
- bei der Implementierung an der wiederholten Zerlegung von Abläufen in Einzelschritte (auch Algorithmisierung) oder an der Zerlegung eines Problems in einfachere (Divide-and-Conquer),
- beim Software life cycle selbst an der Zerlegung des Entwicklungsprozesses in sechs Phasen, die jeweils wiederum aus mehreren Unterphasen bestehen.

Bei der strukturierten Zerlegung können wir offenbar zwei zueinander orthogonale Aspekte unterscheiden:

- Der vertikale Aspekt, die *Hierarchisierung* (Abb. 6), beschreibt die Zerlegung eines Gegenstands in aufeinander aufbauende Ebenen unterschiedlichen Abstraktionsniveaus. Geleitet wird diese Vorgehensweise von der Zielvorstellung einer schrittweisen totalen Zerlegbarkeit jedes Systems in eine endliche Folge von Hierarchieebenen, die einen unterschiedlichen Abstraktionsgrad besitzen, aber semantisch äquivalent sind und algorithmisch ineinander überführt werden können.

Die Idee der Hierarchisierung finden wir u.a. in folgenden Zusammenhängen (vgl. [P74]): Ebenenmodelle der Rechnerarchitektur (s.o.), Sprachhierarchien (z.B. die Chomsky-Hierarchie), Maschinenmodelle, Komplexitäts- und Berechenbarkeitsklassen, virtuelle Maschinen, ISO-OSI-Ebenenmodell.

- Der horizontale Aspekt, die *Modularisierung* (Abb. 7), beschreibt die Zerlegung eines Gegenstands in einzelne Teile gleichen Abstraktionsniveaus. Hinter der Modularisierung verbirgt sich die Zielvorstellung, jedes System ließe sich durch die Eigenschaften seiner Teile vollständig erklären und als Summe total voneinander unabhängiger Teile (bottom-up) auffassen bzw. in total voneinander unabhängige Teile zerlegen (top-down). Diese Zielvorstellung ist zwar faktisch unerreichbar, denn die Eigenschaften eines Systems lassen sich nicht nur aus den Eigenschaften der Subsysteme allein ableiten, sondern sie werden auch durch ihre Beziehungen untereinander beeinflußt; die normative (Plato) bzw. regulative (Kant) Funktion dieser Idee äußert sich jedoch in dem Bestreben, die Abhängigkeiten zwischen den Teilsystemen (Schnittstellen) dann wenigstens so gering wie möglich zu halten.

Die hierarchische Modularisierung entsteht als Mischform aus diesen beiden Grundformen (Abb. 8).

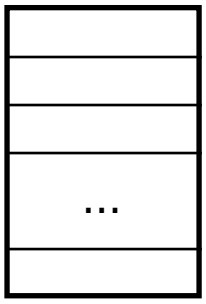


Abb. 6: Hierarchisierung

Abb. 7: Modularisierung

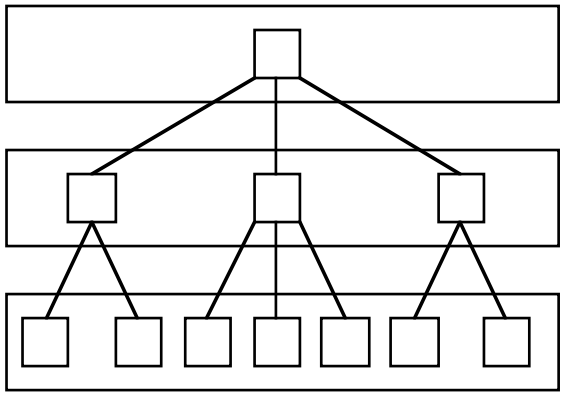


Abb. 8: Hierarchische Modularisierung

Einen weiteren wichtigen Aspekt der Zerlegung haben wir bisher noch nicht erwähnt: Offenbar kommt jeder Zerlegungsprozeß irgendwann zum Ende, spätestens dann, wenn ein atomares Niveau erreicht wird.

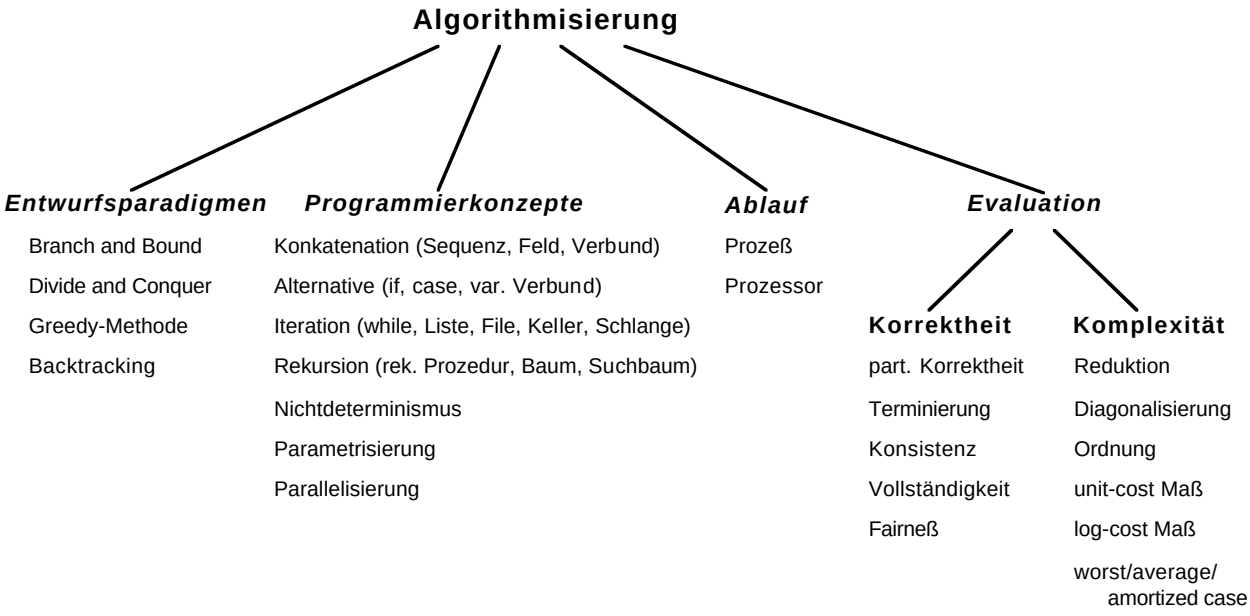
Beispiele: Beim Divide-and-Conquer-Verfahren bricht die Zerlegung ab, sobald ein Problem vorliegt, für das die Lösung unmittelbar angegeben werden kann. Bei der hierarchischen Modularisierung stoppt sie spätestens dann, wenn eine Modulspezifikation erreicht wird, die sich direkt in eine einzelne Anweisung der Programmiersprache umsetzen läßt.

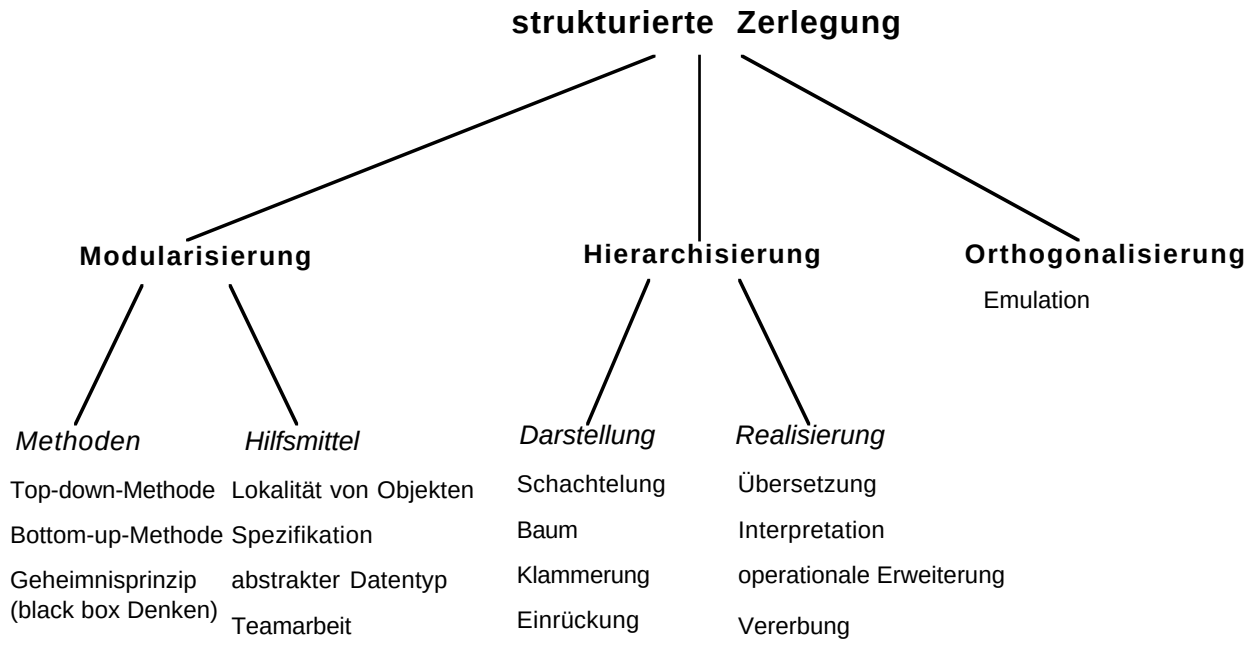
Diese Beobachtung führt auf die Idee der Erzeugendensysteme, wir wollen in Anlehnung an eine ähnliche Operation in der linearen Algebra von der Idee der *Orthogonalisierung* sprechen. Unter Orthogonalisierung eines Objektbereiches verstehen wir die Angabe einer möglichst kleinen Zahl von Basiselementen des Bereiches sowie von Operationen auf dieser Basis, so daß jedes beliebige andere Objekt des Bereichs durch endliche Anwendung der Operationen aus den Basiselementen erzeugt werden kann. Hinter dieser Idee verbirgt sich der Wunsch nach "Einfachheit" und die reduktionistische Zielvorstellung, jeder Objektbereich ließe sich in der beschriebenen Weise orthogonalisieren, was für viele in der Informatik vorkommenden Bereiche auch tatsächlich sehr häufig zutrifft, wie die folgenden Beispiele belegen. Sie zeigen zugleich, daß es sich um eine fundamentale Idee innerhalb und außerhalb der Informatik (vgl. Horizontal- und Sinnkriterium) handelt:

- Programmiersprachen. Ein zentrales Prinzip der Sprache ALGOL68 ist ihr Orthogonalentwurf: Es gibt einige wenige elementare Daten- und Anweisungstypen, die jedoch in beliebiger Weise kombiniert werden dürfen. PASCAL ist hier wesentlich restriktiver; von Orthogonalität ihrer Strukturen kann nicht gesprochen werden.
- imperativen Programmiersprachen. Die Strukturen Zuweisung, Konkatenation und while-Schleife bilden eine Basis der Kontrollstrukturen. Jede andere Anweisung läßt sich durch diese drei Typen darstellen.
- funktionalen Programmiersprachen. Die Grundoperationen bei Sprachen, die sich am λ-Kalkül orientieren, sind Abstraktion (=Definition einer Funktion mit Parametern), Applikation (=Aufruf einer Funktion mit aktuellen Parametern) und Substitution (=Parameterersetzung).
- Maschinen. Die universelle Turingmaschine bildet die (einelementige) Basis der Klasse aller Turingmaschinen.
- primitiv-rekursiven und μ-rekursiven Funktionen. Es gibt eine Menge von Grundfunktionen (z.B. konstante Funktion 1, Nachfolgerfunktion) und eine Reihe von Operationen (z.B. Komposition, Einsetzung von Funktionen, μ-Operator), mit der man jede primitiv-rekursive bzw. μ-rekursive Funktion erzeugen kann.
- formalen Sprachen. Dyck-Sprache, reguläre Sprachen und Homomorphismus orthogonalisieren die kontextfreien Sprachen (Satz von Chomsky-Schützenberger).
- booleschen Funktionen. UND, ODER und NICHT bilden eine Basis (einen sog. Bausteinsatz) für alle Booleschen Funktionen. NAND ist ebenfalls ein Bausteinsatz.
- der Fertigung von Automobilen im Baukastensystem,
- bei Schrankwänden oder Regalsystemen,
- bei Häusern aus Fertigteilen,
- bei der DNS, die aus vier Grundsubstanzen Adenin, Guanin, Cytosin und Thymin besteht.

Zum Nachweis der Nicht-Orthogonalität verwendet man üblicherweise die Idee der *Emulation:* Gegeben sei ein Erzeugendensystem. Kann man eines der Elemente des Systems durch die übrigen darstellen, so ist das System nicht orthogonal.

Nach diesen Vorüberlegungen können wir jetzt den vollständigen Katalog fundamentaler Ideen der Informatik aufstellen. Er enthält alle bisher genannten Ideen, thematisch gruppiert und hierarchisch (da ist die Idee wieder!) strukturiert. Einige neue Ideen runden die einzelnen Gruppen ab. Masterideen sind wie erwähnt Algorithmisierung, strukturierte Zerlegung und Sprache (Abb. 9). Man beachte: Bei den kursiv dargestellten Bezeichnungen handelt es sich um Oberbegriffe für Ideengruppen, die zur Systematisierung hinzugenommen wurden, und nicht um Ideen.





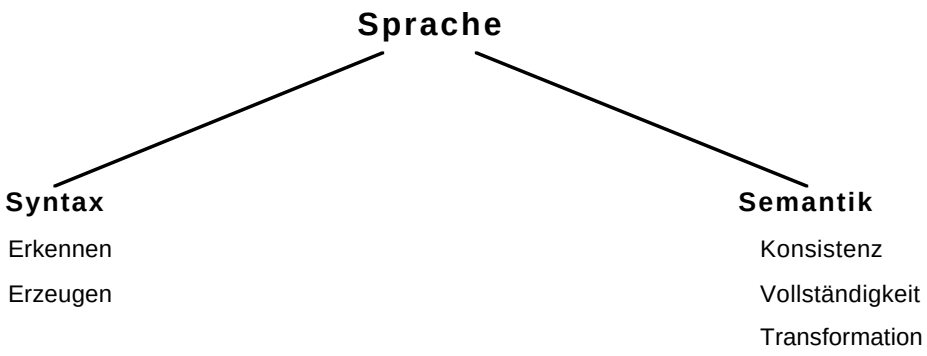


Abb. 9: Fundamentale Ideen der Informatik

Offenbar ist die Zuordnung einzelner Ideen zu Masterideen nicht immer eindeutig. So enthält z.B. das Divide-and-Conquer-Verfahren eine algorithmische und eine Zerlegungskomponente. Wir haben in solchen Fällen die Ideen *dann* der Algorithmisierung zugeordnet, wenn der dynamische Anteil (Prozeßaspekt, Ablaufaspekt) überwiegt. Bei der Zerlegung dominiert im Gegensatz dazu der statische Aspekt, also das Ergebnis der Zerlegung und nicht der Weg dorthin. Ferner tauchen einzelne Ideen in verschiedenen Ausprägungen an mehreren Stellen der Aufstellung auf. Zum Beispiel bezeichnen Reduktion und Transformation Übersetzungsprozesse; Übersetzung selbst erscheint dann noch einmal als Idee zur Realisierung von Hierarchien. Man erkennt also, daß die Ideen teilweise in vielfältiger Weise miteinander verflochten sind. Eine exakte Zuordnung und scharfe Abgrenzung voneinander ist kaum möglich. Auch hier offenbart sich in dem Versuch, dies doch zu tun, die regulative Funktion der Idee der strukturierten Zerlegung.
Auf einen genauen Nachweis der Fundamentalität der Ideen im Sinne der fünf Kriterien wollen wir hier verzichten. Ansätze findet man in [S93].

Wir haben die Sammlung von Ideen willkürlich auf einer bestimmten Hierarchiestufe abgebrochen, drei Beispiele mögen jedoch zeigen, daß sich die Hierarchie nach unten weiter fortsetzen läßt, wobei die Ideen natürlich immer spezieller und ihr Wirkungsbereich im Sinne des Horizontalkriteriums immer enger wird.

Beispiele:

1. Plane-sweeping ist eine fundamentale Idee im Bereich der *Greedy-Methode*, gewissermaßen eine bereichsspezifische Ausprägung der Greedy-Methode für die Belange der Algorithmischen Geometrie.
2. Eine fundamentale Idee der *Iteration* ist der Eingabezyklus prompt-read-check-echo (vgl. [F79]).

3. Eine fundamentale Idee der *Konkatenation* ist die bei funktionalen Sprachen im Zusammenhang mit Listen regelmäßig vorkommende Funktionskomposition generate-transform-filter-accumulate.

Das Ideen-Tripel Algorithmisierung, strukturierte Zerlegung und Sprache läßt sich noch auf eine weitere interessante Weise abstützen: Bei der Suche nach Antworten auf die Frage "Was ist Informatik?" wird häufig der Modellierungsaspekt der Informatik betont [Co89,L86, L89,D89]. Gelegentlich wird die Informatik auch als *Wissenschaft von der Modellbildung* bezeichnet. Akzeptiert man diese Charakterisierung, so lassen sich die drei Masterideen als tragende Säulen der Modellbildung auffassen:

- Mit der *strukturierten Zerlegung* sind die Ideen verbunden, mit deren Hilfe man ein reales System analysiert und die modellrelevanten Eigenschaften ableitet.
- Das Modell wird anschließend auf der Basis einer Beschreibungs*sprache* präzisiert und öffnet sich so weiteren syntaktischen und vor allem semantischen Analysen und Transformationen.
- Der dynamische Aspekt von Modellen, die Möglichkeit, sie zu simulieren, wird durch die *Algorithmisierung* erfaßt. Die zugehörigen Ideen dienen dem Entwurf und dem Ablauf von Simulationsprogrammen.

4 Schlußbemerkungen

Jede Wissenschaft muß sich von Zeit zu Zeit fragen, welches ihr Forschungsgegenstand ist, welche begrifflichen und methodischen Grundlagen sie nutzt, um zu Erkenntnissen zu gelangen, und was ihre unverwechselbaren Denkweisen sind. Dieser Aufgabe, der sich in etablierten Wissenschaften philosophisch oder didaktisch ausgerichtete Forschungsgruppen widmen, ist von der Informatik noch kaum registriert worden. Lediglich zum Forschungsgegenstand sind in letzter Zeit einige Überlegungen angestellt worden [C92]. Wir haben in diesem Aufsatz einen ersten Schritt zur Klärung der methodischen Grundlagen unternommen. Als Leitfaden diente uns hierbei das Konzept der fundamentalen Ideen, ein vielseitiger Begriff mit einem weitreichenden Spektrum, das durch die vorliegende Arbeit bei weitem nicht ausgeschöpft werden kann.

Die Ergebnisse können innerhalb der Informatik vor allem für die Weiterentwicklung *der* beiden Bereiche herangezogen werden, die auch Beiträge zur Klärung des Ideenbegriffs geliefert haben:

- *Erforschung der Grundlagen der Informatik* mit dem Ziel, eine informatische Erkenntnistheorie zu entwickeln.
- *Didaktik der Informatik*.
  Jeder Student wird im Laufe seines Berufslebens vermutlich mehreren Paradigmenwechseln der Informatik gegenüberstehen, wobei jeweils ein größerer Teil seines Wissens überflüssig oder fehlerhaft wird. Daher sollten die Fähigkeiten, die er während

des Studiums erwirbt, möglichst robust gegenüber neuen wissenschaftlichen Entwicklungen sein und ihn befähigen, Paradigmenwechsel zu bewältigen. Folglich müssen Studenten ein Bild von den dauerhaften Grundlagen, den fundamentalen Ideen, Prinzipien, Methoden und Denkweisen der Informatik erlangen. Dazu sind in Vorlesungen und Lehrbüchern stets die fundamentalen Ideen, die sich hinter den jeweils behandelten Sachgebieten verbergen, herauszuarbeiten, zu betonen, zu anderen Teilgebieten in Beziehung zu setzen und so in einen übergeordneten Zusammenhang einzuordnen[2]. [S93,S94] enthält Einzelheiten hierzu.

Literatur

[B91] Brauer, W.: "The new paradigm of informatics", in: New Results and New Trends in Computer Science, (H. Maurer, ed.) (1991) 15-24

[B60] Bruner, J.S.: "The process of education", Cambridge Mass. 1960 (dt. Übers.: "Der Prozeß der Erziehung", Berlin 1970)

[C89] Claus, V.: "Entwicklung von Informatik-Methoden", HMD-Theorie und Praxis der Wirtschaftsinformatik 150 (1989) 26-44

[C90] Capurro, R.: "Ethik und Informatik", Informatik-Spektrum 13 (1990) 311-320

[C89] Coy, W.: "Brauchen wir eine Theorie der Informatik?", Informatik-Spektrum 12 (1989) 256-266

[C92] Coy, W. (ed.): "Sichtweisen der Informatik", Vieweg 1992

[D89] Denning, P.J. et al.: "Computing as a discipline", Comm. of the ACM 31,1 (1989) 9-23

[D84] Dörfler, W.: "Fundamentale Ideen der Informatik und Mathematikunterricht", Symposium über Schulmathematik, Österr. Mathem. Gesellschaft, Didaktik Reihe Nr. 10 (1984) 19-40

[F76] Fischer, R.: "Fundamentale Ideen bei den reellen Funktionen", Zentralblatt für Didaktik der Mathematik 3 (1976) 185-192

[F84] Fischer, R.: "Unterricht als Prozeß von der Befreiung vom Gegenstand - Visionen eines neuen Mathematikunterrichts", J. für Mathematik-Didaktik 1 (1984) 51-85

[F87] Floyd, C.: "Outline of a paradigm change in software engineering", in: Computers and Democracy - A Scandinavian Challenge (G. Bjerknes, P. Ehn, M. Kyng, eds.) (1987) 191-210

[F79] Floyd, R.W.: "The paradigms of programming", Comm. of the ACM 22,8 (1979) 455-460

[G77] Gärtner, H.: "Lehrplan Biologie - Analyse und Konstruktion", Sample Verlag 1977

[G91] Gotlieb, C.C.: "Fashions and fundamentals in computer science education", J. of Education and Computing 7 (1991) 97-103

[H81] Halmos, P.R.: "Does mathematics have elements?", The Mathematical Intelligencer 3 (1981) 147-153

[H75] Heitele, D.: "An epistemological view on fundamental stochastic ideas", Educational Studies in Mathematics 6 (1975) 187-205

[K93] Kant, I.: "Kritik der reinen Vernunft", Ausgabe B, Nachdr. Meiner 1993

[K57] Kant, I.: "Prolegomena", Nachdr. Meiner 1957

[K67] Kant, I.: "Über Pädagogik", Nachdr. Kamp 1967

[K81] Klika, M.: "Fundamentale Ideen der Analysis", mathematica didactica 4 (1981) 1-31, Sonderheft

[K89] Knöß, P.: "Fundamentale Ideen der Informatik im Mathematikunterricht", Deutscher Universitäts-Verlag 1989

[K62] Kuhn, T.S.: "The structure of scientific revolutions", Chicago 1962

[L86] Lockemann, P.C.: "Konsistenz, Konkurrenz, Persistenz - Grundbegriffe der Informatik?", Informatik-Spektrum 9 (1986) 300-305

[L89] Luft, A.L.: "Informatik als Technikwissenschaft - Thesen zur Informatik-Entwicklung", Informatik-Spektrum 12 (1989) 267-273

[M84] Mehlhorn, K.: "Data structures and algorithms", Springer 1984

[M68] Michaelis, J.U.: "Social studies for children in a democracy", 1968

[M93] Müller, B.: "Is object-oriented programming structured programming?", ACM SIGPLAN Notices 28,9 (1993) 57-66

---

[2] K. Mehlhorn [M84, Chapt. IX] hatte die Wahl zwischen einem ideenorientierten ("paradigm oriented") und einem themenorientierten ("problem oriented") Zugang, entschied sich dann aber der kürzeren Darstellung wegen für den themenorientierten Ansatz. In einem besonderen Kapitel ordnet er die Lösungsansätze jedoch, wie von uns angeregt, in eine übergeordnete Ideenstruktur ein.

[M80] Müller, M.W.: "Fundamentale Ideen der Numerischen Mathematik", Beitr. zum Mathematikunterricht (1980) 238-245
[N90] Nievergelt, J.: "Computer science for teachers: A quest for classics, and how to present them", Proc. of the 3rd Intern. Conference on Computer Assisted Learning (1990) 2-15
[P74] Parnas, D.L.: "On a 'buzzword': Hierarchical structure", Information Processing (1974) 336-339
[P65] Price, R.A., Smith, G.R., Hickman, W.L.: "Major concepts for the social studies", 1965
[S81] Schmidt, H.J.: "Fachdidaktische Grundlagen des Chemieunterrichts", Vieweg Verlag 1981
[S83] Schreiber, A.: "Bemerkungen zur Rolle universeller Ideen im mathematischen Denken", mathematica didactica 6 (1983) 65-76
[S89] Schulz, A.: "Software-Lifecycle- und Vorgehensmodelle", Angewandte Informatik (1989) 137-142
[S82] Schweiger, F.: "Fundamentale Ideen der Analysis und handlungsorientierter Unterricht", Beitr. zum Mathematikunterricht (1982) 103-111
[S93] Schwill, A.: "Fundamentale Ideen der Informatik", Zentralbl. für Didaktik d. Mathematik (1993) 20-31
[S94] Schwill, A.: "Fundamental ideas of computer science", Bulletin of the EATCS No. 53 (1994), im Druck
[S90] Siefkes, D.: "Wende zur Phantasie - Zur Theoriebildung in der Informatik", 20. GI-Jahrestagung, Informatik-Fachberichte 257 (1990) 242-255
[S70] Spreckelsen, K.: "Strukturelemente der Physik als Grundlage ihrer Didaktik", Naturwiss. im Unterr. 18 (1970) 418-424
[T84] Tanenbaum, A.S.: "Structured computer organization", Prentice-Hall 1984
[T79] Tietze, U.-P.: "Fundamentale Ideen der linearen Algebra und analytischen Geometrie - Aspekte der Curriculumsentwicklung im MU der SII", mathematica didactica 2 (1979) 137-163
[W83] Wegner, P.: "Paradigms of information processing", in: The Study of Information (F. Machlup, U. Mansfield, eds), Wiley 1983, 163-175
[W29] Whitehead, A.N.: "Die Gegenstände des mathematischen Unterrichts", Neue Sammlung 2 (1962) 257-266 (Originalarbeit von 1929)