Fundamentale Ideen der Informatik

Andreas Schwill
Fachbereich Informatik - Universität Oldenburg
Postfach 2503 - W-2900 Oldenburg - Germany

Zusammenfassung

Es herrscht allgemein Konsens, daß die Fortschritte der Wissenschaft Informatik nicht mit gleicher Geschwindigkeit für den Schulunterricht zugänglich gemacht werden können. Daher müssen sich die Inhalte im Informatikunterricht bis auf weiteres nach den langlebigen Grundprinzipien und Denkweisen der Informatik richten. Dieses erstmals von J.S. Bruner 1960 fachneutral formulierte Prinzip der Orientierung des Unterrichts an fundamentalen Ideen greifen wir in dieser Arbeit auf. Nach einer Einführung in die Hintergründe der Brunerschen Überlegungen werden fundamentale Ideen der Informatik entwickelt und Vorschläge präsentiert, wie diese Ideen im Unterricht vermittelt werden können.

1 Einleitung

Trotz ihrer nun mehr als 40-jährigen Entwicklungsgeschichte besitzt die Informatik noch immer eine erstaunliche Dynamik. Regelmäßig kündigen sich Paradigmenwechsel an, etwa in der Programmierung von der "straight forward"-Methode zum strukturierten Programmieren nach der Softwarekrise in den 60er Jahren, später dann zum logischen und applikativen Programmieren und zur Zeit zur objektorientierten Programmierung.

Unter den wenigen Personen, die sich forschend mit dem Themenbereich "Informatik in der Schule" befassen, herrscht allgemein Konsens, daß die Fortschritte der Wissenschaft Informatik nicht mit gleicher Geschwindigkeit für den Schulunterricht zugänglich gemacht werden können. Daher müssen sich die Inhalte im Informatikunterricht bis auf weiteres an den langlebigen Grundlagen der Wissenschaft orientieren. Es ist unverzichtbar, daß den Schülern ein Bild von den *grundlegenden Prinzipien*, *Denkweisen* und *Methoden* (den **fundamentalen Ideen**) der Informatik vermittelt wird. Nur von diesen Ideen ist eine längerfristige Gültigkeit zu erwarten. Neuere Erkenntnisse erscheinen dann häufig nur als Variation eines bereits vertrauten Sachverhalts und können über die einmal gelernten Ideen leichter erschlossen werden. Diese Leitlinien wurden fachneutral erstmals von J.S. Bruner [B60] in seiner grundlegenden Arbeit propagiert.

In diesem Aufsatz übertragen wir Bruner's didaktische Vorstellungen auf den Informatikunterricht. Zunächst skizzieren wir in Abschnitt 2 die Hintergründe der Bruner'schen Arbeit. Wir entwickeln Kriterien für fundamentale Ideen und schlagen ein allgemeines Verfahren zur Ermittlung fundamentaler Ideen vor. Ein weiterer Abschnitt ist der Vermittlung fundamentaler Ideen im Unterricht gewidmet.

In Abschnitt 3 werden dann fundamentale Ideen der Informatik erarbeitet und anhand der entwickelten Kriterien exemplarisch auf ihre Fundamentalität überprüft.

2 Fundamentale Ideen als Unterrichtsprinzip

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

> "sich 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** (den sog. **fundamentalen Ideen**) der zugrundeliegenden Wissenschaft orientieren soll.

2.1 Hintergründe und Motivation

Bruner begründet seinen Zugang wie folgt: Lernen in der Schule hat vor allem den Sinn, uns 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 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. Die vermittelten Kenntnisse haben nicht in erster Linie den Zweck, unmittelbar angewendet werden zu können.
Daher - und jetzt kommen wir wieder auf die Ausgangsforderung Bruner's zurück - hat sich der Unterricht vor allem an den fundamentalen Ideen zu orientieren. Jeder Unterrichtsgegenstand ist daraufhin zu überprüfen, welche Ideen ihm zugrundeliegen. Alle Lehrpläne und Unterrichtsmethoden sind darauf abzustellen, bei jedem Unterrichtsgegenstand die fundamentalen Ideen hervorzuheben.
Natürlich werfen die letzten Forderungen eine Fülle von Fragen auf: Was sind überhaupt fundamentale Ideen? Wie sind Curricula umzustellen, damit fundamentale Ideen eine zentrale Rolle erhalten? Welche Themen eignen sich auf den verschiedenen Schulstufen am besten zur Vermittlung von fundamentalen Ideen? Einigen Fragen werden wir im folgenden auf den Grund gehen, zuerst allgemein und in Abschnitt 3 dann informatikbezogen.

Zunächst zwei Beispiele aus der Informatik und der Mathematik, die zeigen, wie bedeutsam die Forderung Bruner's ist, den Unterricht nach Ideen auszurichten und wie häufig gegen dieses Prinzip in der Praxis verstoßen wird.

Beispiel 1: In einem Informatiklehrbuch für die Schule wird die Parameterübergabeart call by reference in Prozeduren eingeführt. Der Mechanismus, der beim Aufruf einer Prozedur mit Referenzparametern abläuft, wird dort auf etwa einer Seite u.a. am Behältermodell wie folgt erläutert:

> "Der Kasten der aufrufenden Variablen wird mit in den Prozedurraum genommen und erhält während der Prozedurbearbeitung einen anderen Namen, den Namen des Parameters. Nach Abschluß der Bearbeitung wird der aufgeklebte Name wieder entfernt, der Kasten wird unter seinem ursprünglichen Namen in den Hauptraum zurückgenommen. Damit sind jetzt im Kasten die Werte enthalten, die während der Prozedurbearbeitung hineingelegt wurden."

Diese recht verwirrenden und zum Teil falschen Erklärungen überdecken vollständig die Idee, die hinter dem Konzept call by reference steckt:

Vergabe eines Synonyms für ein Objekt.

Der Bezeichner des formalen Parameters wird innerhalb der Prozedur als Synonym für den aktuellen Parameter verwendet. Diese sehr einfache wie treffende Idee beschreibt das Prinzip vollständig. Die ursprüngliche Flut von Informationen wird zu einer griffigen "Formel" *verdichtet*, die im Gedächtnis haften bleibt, auch weil sie im Gegensatz zur obigen Erläuterung noch einen sprachlichen *Bezug zur Lebenswelt* besitzt.

Beispiel 2: Das Horner-Schema ist ein Verfahren zur Berechnung von Funktionswerten eines Polynoms. Zum Beispiel stellt man für das Polynom

$$p(x)=3x^3+7x^2+5x+3,$$

für das man p(4) berechnen möchte, folgendes Schema auf:

	3	7	5	3
x=4		0	12	76	324
	3	19	81	**327**

Es gilt also p(4)=327.

Welche Idee steckt hinter dem Schema? Geht man einige Standardwerke der Schulmathematik durch, wird man enttäuscht. Eine prägnante Erklärung dieses Verfahrens findet man nicht, jedenfalls nicht offensichtlich. Sie ist zwar in der Erläuterung des Verfahrens verklausuliert, wird aber nicht herausgestellt. Dabei ist die Idee hinter diesem Schema einfach das

sukzessive Ausklammern von x

im Polynom p, also p(x)=x(x(x(3)+7)+5)+3. Behält ein Schüler diese Idee im Kopf, so kann er das Schema auch nach Jahren noch wieder entwickeln.

2.2 Fundamentale Idee: Begriffspräzisierung

Die Gretchenfrage, die hinter den bisherigen Überlegungen stand, lautet: Was sind fundamentale Ideen? Bruner selbst weicht aus, wenn es um eine explizite Definition oder Charakterisierung des Begriffs geht. Spätere Arbeiten anderer Autoren besitzen hier mehr Substanz. Bruner überläßt es im wesentlichen dem Leser, sich anhand vieler Beispiele

eine intuitive Vorstellung von fundamentalen Ideen zu verschaffen. Lediglich einige wenige, aber wichtige Hinweise sind über das Buch verstreut, z.B.:

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

Fundamentale Ideen besitzen also eine umfassende Anwendbarkeit in vielen Bereichen, und sie ordnen und integrieren eine Vielzahl von Phänomenen. Wir sprechen hier vom **Horizontalkriterium**. Hiermit ist folgende Anschauung verbunden: Eine horizontale Achse, die eine Vielzahl von Wirkungsbereichen durchstößt (Abb. 2).

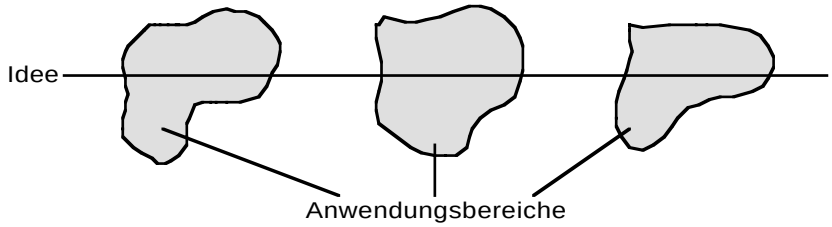


Abb. 2: Veranschaulichung des Horizontalkriteriums

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." (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." (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 dem Grad der Detaillierung und Formalisierung, mit dem die Idee präsentiert wird. Kontrapositiv formuliert (nach R. Fischer [F84]): Was nicht prinzipiell auch einem Volksschüler vermittelt werden kann, kann keine fundamentale Idee sein.

Fundamentale Ideen müssen also 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. Dennoch werden wir in Abschnitt 3 einen solchen Versuch unternehmen, dessen "Objektivität" aber nur darin besteht, daß die fundamentalen Ideen der Informatik von einer Vielzahl von Menschen akzeptiert werden (können). Wollte man fundamentale Ideen der Informatik *für intelligente Computer* erarbeiten, so erhält 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).

Für den Schulunterricht ist das Vertikalkriterium von besonderer Bedeutung: Erfüllt eine Idee dieses Kriterium, so kann sie als Richtschnur verwendet werden, um den Unterricht auf jeder Ebene des gesamten Bildungsprozesses daran zu orientieren. Wir kommen hierauf in Zusammenhang mit dem Spiralprinzip später noch einmal zurück.

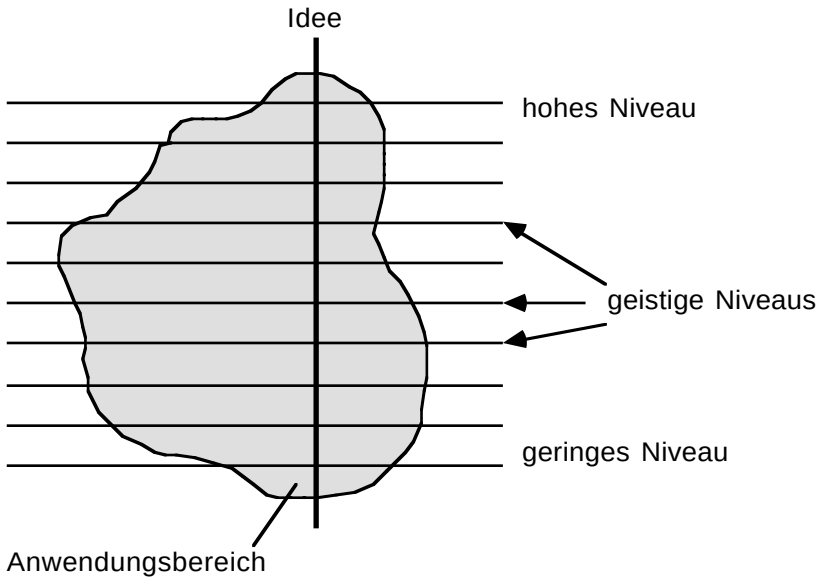


Abb. 3: Veranschaulichung des Vertikalkriteriums

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.

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.
  Beispiel: 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 auch die berühmte Gleichung $E=mc^2$ oder das Quicksort-Verfahren keine fundamentalen Ideen.

- **Fülle**, d.h. vielfältige Anwendbarkeit und Relevanz. Dieses Kriterium deckt sich im wesentlichen mit unserem Horizontalkriterium. Schreiber stellt in einer anderen Arbeit noch einmal ausdrücklich heraus, daß die *häufige* Anwendung einer Idee nicht schon zu ihrer Fülle führt. Vielmehr muß die Idee auch in *vielfältiger* Variation anwendbar sein.
- **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.

  Beispiel: Das skizzierte Verhältnis zwischen einer Idee "mit Sinn" und einem Begriff besteht z.B. zwischen "Reversibilität" 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.

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. Im Unterschied zu den beiden anderen Präzisierungsversuchen bezieht sich Schweiger durch seine Aufzählung "Handlungen, Strategien oder Techniken" offenbar eher auf relativ präzise definierte Gebrauchsregeln. Er steht damit im Gegensatz zu Schreiber's Definition, der diese scharfen Regeln durch das Kriterium "Weite" gerade ausschließt. Andererseits hält Schweiger (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 (ebenso übrigens auch Punkt (4) aus Schweiger's Liste) verstehen wir weniger als Kriterium für denn als Vorteil von fundamentalen Ideen. Hat man eine Wissenschaft erst einmal durch das Aufstellen fundamentaler Ideen strukturiert, so ist man gleichzeitig im Besitz einer philosophischen Fundierung der Wissenschaft, weiß um ihr "Wesen" und kann sie gegen andere Wissenschaften abgrenzen.

Wir wollen nun, praktisch als Synthese, eine eigene Definition anschließen. Wie diese aussieht, dürfte nach den Vorüberlegungen relativ klar sein: Die vier herausgearbeiteten Kriterien (Horizontal-, Vertikal-, Zeit- und Sinnkriterium) sind zu erfüllen, wobei die ersten beiden Bedingungen für Fundamentalität sind und die beiden anderen notwendig sind, um überhaupt von einer Idee sprechen zu können.

"Definition":
Eine **fundamentale Idee** (bezgl. einer Wissenschaft) ist ein Denk-, Handlungs-, Beschreibungs- oder Erklärungsschema, das

(1) in verschiedenen Bereichen (der Wissenschaft) vielfältig anwendbar oder erkennbar ist (**Horizontalkriterium**),
(2) auf jedem intellektuellen Niveau aufgezeigt und vermittelt werden kann (**Vertikalkriterium**),
(3) in der historischen Entwicklung (der Wissenschaft) deutlich wahrnehmbar ist und längerfristig relevant bleibt (**Zeitkriterium**),
(4) einen Bezug zu Sprache und Denken des Alltags und der Lebenswelt besitzt (**Sinnkriterium**).

Fundamentale Ideen 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 gibt es in Physik, Chemie und Biologie [S70,S81,G77].

Beispiel 1: A. Schreiber [S79] nennt als fundamentale Idee der Mathematik u.a. die **Exhaustion**. Unter Exhaustion versteht er allgemein das Annähern der gedanklichen

oder realen Welt, das Modellieren. Er unterscheidet die *ideelle* Exhaustion, bei der das Annähernde gedanklicher Natur ist, z.B. die Approximation einer Fläche durch ein Polygon, einer Funktion durch ein Polynom, einer reellen Zahl durch einen Bruch, und die *reelle* Exhaustion, bei der das Annähernde der physischen Wirklichkeit entstammt, z.B. die Annäherung eines Gebäudes durch Pappmodelle, von Längen durch einen Zollstock, von mathematischen (dimensionslosen) Punkten durch einen Bleistiftpunkt.
Offenbar erfüllt die Exhaustion das Horizontal- und das Vertikalkriterium. Sie besitzt aber auch eine lebensweltliche Bedeutung, z.B. beim Annähern von Wohnungsmobiliar durch Pappschablonen und Durchspielen möglicher Einrichtungsvorschläge, beim Annähern einer Geraden durch eine straff gespannte Schnur. Auf die historische Relevanz der Idee können wir nicht eingehen.

Beispiel 2: Eine Auswahl fundamentaler Ideen der klassischen Physik könnte sein:

- **Kausalität**. Dieses Prinzip ("von nichts kommt nichts") bildet seit dem Altertum die Basis aller physikalischen (und nicht nur physikalischer) Beobachtungen. Kein physikalisches Ereignis geschieht ohne irgendeine äußere Ursache. Ein einmal in Bewegung gesetzter Körper behält Richtung und Geschwindigkeit immerfort bei, ein unbewegter Körper setzt sich nicht von selbst in Bewegung, eine Supernova entsteht nicht willkürlich. Eng verknüpft mit dem Kausalitätsprinzip ist die Idee des
- **Determinismus**. "Gleiche Ursachen haben gleiche Wirkungen". Auf dieser Idee beruhen alle physikalischen Experimente. Durch wiederholtes Herstellen gleicher oder leicht geänderter Ausgangsbedingungen versucht man funktionale Zusammenhänge zwischen Ursachen und Wirkungen herzuleiten. Experimente werden sinnlos, wenn das deterministische Prinzip nicht angenommen wird.
  Beide Prinzipien gelten jedoch nicht mehr im atomaren Bereich, wo Ereignisse auch spontan oder mit gewissen Wahrscheinlichkeiten ablaufen können oder man wegen der Heisenberg'schen Unschärferelation keine Beziehungen zwischen Ursache und Wirkung mehr messen kann.
- **Invarianz**. Die Idee der Invarianz spielt überall da (nicht nur in der Physik) eine Rolle, wo gewisse Eigenschaften von Objekten durch bestimmte Operationen erhalten bleiben. In der Physik wird Invarianz u.a. formalisiert durch den Energieerhaltungssatz und den Impulserhaltungssatz. Eine Invariante eines Objekts ist z.B. seine Masse (relativistische Effekte ausgenommen).
- **Extremalprinzip**. Dieses Prinzip macht Aussagen über das Verhalten von Systemen, wobei eine bestimmte Größe bei der Bewegung einen Extremwert (meist ein Minimum) annimmt. Typische Ausprägungen sind das Hertzsche Prinzip der geradesten Bahn und das Fermatsche Prinzip.

Beispiel 3: In vielen verschiedenen Wissenschaften und auch im täglichen Leben spielen reversible Operationen eine bedeutende Rolle, z.B.

in der Mathematik: Addition/Subtraktion, Differentiation/Integration;
in der Physik: Kraft/Gegenkraft, Energieaufwendung/-wiedergewinnung;
in der Chemie: Mischen/Trennen von Stoffen, Herstellen/Lösen einer Verbindung;
in der Informatik: Konstruktor/Selektor von Datenstrukturen, get/put bei Files;
in der Musik: Spiegelung von Musikteilen;
im Alltag: falten/entfalten von Papier, Schrank öffnen/schließen,
Inhalt eines Gefässes auf Gläser verteilen/zurückgießen,
Geschenk einpacken/auspacken, Licht einschalten/ausschalten.

Reversibilität scheint also so etwas wie ein Grundkonzept jeglicher Form des Denkens zu sein. Wir können hier von einer wissenschaftsübergreifenden fundamentalen Idee sprechen.

2.3 Wie findet man fundamentale Ideen?

Die Literaturhinweise zur Beantwortung dieser Frage sind noch dünner gesät als für die Klärung des Begriffs der fundamentalen Ideen selbst. Jedenfalls handelt es sich um keine leichte Aufgabe (lt. Bruner), denn man benötigt schon einen umfassenden Überblick über eine Wissenschaft, weniger, um einzelne Ideen als fundamental nachzuweisen, als vielmehr, um eine *vollständige* Kollektion aller fundamentalen Ideen der Wissenschaft zu gewinnen, deren Unabhängigkeit zu zeigen oder Hierarchien zu entdecken. Solide und praktisch durchführbar scheint mir vorerst nur folgendes Programm, das von den Inhalten einer Wissenschaft zu ihren Ideen abstrahiert:

1. Schritt: Man analysiert konkrete Inhalte einer Wissenschaft und ermittelt Beziehungen und Analogien zwischen ihren Teilgebieten (wg. Horizontalkriterium) sowie zwischen unterschiedlichen intellektuellen Niveaus (wg. Vertikalkriterium). So erhält man eine erste Kollektion von fundamentalen Ideen.

2. Schritt: Diese Liste verbessert und modifiziert man, indem man nachprüft, ob jede der Ideen auch eine lebensweltliche Bedeutung besitzt und im Alltag nachweisbar ist (Sinnkriterium).

3. Schritt: Anschließend versucht man die historische Entwicklung jeder Idee nachzuzeichnen. So gewinnt man evtl. weitere Gesichtpunkte und stabilisiert die Ideenkollektion. Hierzu beachte man den Vorschlag von Nievergelt (Abschnitt 2.2).

4. Schritt: Schließlich stimmt man die Ideen aufeinander ab und analysiert 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? Sind die Ideen "linear unabhängig"?

Nach ggf. iterativer Durchführung dieses Verfahrens ist man schließlich mit etwas Glück im Besitz einer soliden Sammlung von fundamentalen Ideen. 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, also immer Interpretationsfreiheiten lassen. Auch gibt es kein Kriterium, um nachzuweisen, daß eine Ideenkollektion vollständig ist. Diese Eigenschaft gewinnt sie allenfalls durch Diskussion und regelmäßigen Gebrauch innerhalb der Wissenschaft, aber auch durch Orientierung des Unterrichts an diesen Ideen im Sinne Bruner's.

2.4 Wie vermittelt man fundamentale Ideen?

Den entscheidenden Beitrag liefert hier Bruner selbst durch die Forderung, ein Unterricht, der sich an fundamentalen Ideen einer Wissenschaft orientiert, müsse nach dem **Spiralprinzip** organisiert werden. Er beschreibt dieses Prinzip so:

> "Der Anfangsunterricht ... sollte so angelegt sein, daß diese Fächer mit unbedingter intellektueller Redlichkeit gelehrt werden, aber mit dem Nachdruck auf dem intuitiven Erfassen und Gebrauchen dieser grundlegenden Ideen. Das Curriculum sollte bei seinem Verlauf wiederholt auf diese Grundbegriffe zurückkommen und auf ihnen aufbauen, bis der Schüler den ganzen formalen Apparat, der mit ihnen einhergeht, begriffen hat. ... Man muß noch viel über die 'Curriculum-Spirale' lernen, die auf höheren Ebenen immer wieder zu sich selbst zurückkommt." (S. 26/27).

Analysiert man die einzelnen Passagen dieser Beschreibung genauer, so entdeckt man drei Teilprinzipien, die gewissermaßen die tragenden Säulen des Spiralprinzips bilden [K89]: das Prinzip der Fortsetzbarkeit eines Themas (Zitat: "mit unbedingter intellektueller Redlichkeit"), das Prinzip der Präfiguration von Begriffen (Zitat: "Nachdruck auf dem intuitiven Erfassen und Gebrauchen dieser grundlegenden Ideen") und das Prinzip des vorwegnehmenden Lernens (Zitat: "wiederholt auf diese Grundbegriffe zurückkommen ..., bis der Schüler den ganzen formalen Apparat ... begriffen hat."). Wir wollen diese Prinzipien im folgenden an Beispielen erläutern [W81]:

- **Prinzip der Fortsetzbarkeit**. Die Auswahl und die Behandlung eines Themas an einer bestimmten Stelle des Curriculums soll nicht ad hoc, sondern so erfolgen, daß auf höherem Niveau ein Ausbau möglich ist. Zu vermeiden sind vordergründige didaktische Lösungen (auch durch Vermittlung von Halbwahrheiten), die später ein Umdenken und eine teilweise Zurücknahme von Aussagen erforderlich machen.

  Beispiel: Flußdiagramme erscheinen (mehr noch als Struktogramme) im Anfangsunterricht der Informatik besonders geeignet, um Algorithmen darzustellen und zu entwickeln. Sie verfügen über sehr wenige Darstellungsmittel und werden leicht begriffen. Schnelle Anfangserfolge im Unterricht sind garantiert. Der Nachteil dieses Ansatzes zeigt sich erst später: Er ist nicht ausbaufähig und zwingt zum Umdenken. Denn da in Flußdiagrammen keine Beschreibungsmittel für Schleifen vorgesehen sind,

gibt es kein adäquates Verfahren, um von Flußdiagrammen zu strukturierten Programmen (z.B. in PASCAL) zu gelangen. Bereits entwickelte Programme in Flußdiagrammdarstellung müssen umgeschrieben werden.

- **Prinzip der Präfiguration von Begriffen**. Die *symbolische* Darstellung eines Begriffs oder Konzepts und seine begrifflich-strukturelle Analyse sollen bereits auf niedriger Stufe durch Bilder (*ikonisch*) und Handlungen (*enaktiv*) vorbereitet werden. Bevor also ein Begriff in allen Einzelheiten und Auswirkungen theoretisch analysiert werden kann, sollte er zunächst in Gebrauch genommen werden, damit die Schüler eine intuitive Vorstellung erhalten.

  Beispiele:

  1) Statt die Syntax einer Programmiersprache oder ihrer Sprachelemente jeweils explizit einzuführen, kann man die einzelnen Konstrukte zunächst "einfach benutzen". An Beispielen wird die Syntax durch Analogschluß klar.
  2) Man kann das Prozedurkonzept an einer bestimmten Stelle des Curriculums explizit einführen. Alternativ verwendet man vorher bereits implizit Standardprozeduren und -funktionen. So entsteht bei den Schülern bereits eine Vorstellung von den Abläufen bei Aufruf und Parameterübergabe.

- **Prinzip der vorwegnehmenden Lernens**. Die Behandlung eines Wissensgebietes soll nicht aufgeschoben werden, bis eine endgültig-abschließende Behandlung möglich erscheint, sondern ist bereits auf früheren Stufen in einfacher Form einzuleiten.

  Beispiel: Innerhalb der Informatik benötigt man zur Verifikation von Programmen einen relativ hohen technischen Aufwand, den man in der Schule allenfalls in den letzten Klassen der Oberstufe zur Verfügung hat. Zuvor kann und sollte man jedoch schon auf einfacher Stufe Grundzüge vermitteln, z.B.

  = durch Plausibilitätsbetrachtungen zur Termination von Schleifen auf argumentativer oder halbformaler Ebene.

    Beispiel: Man ermittle mit den Schülern durch Plausibilitätsbetrachtungen eine hinreichende Bedingung für die Termination von Schleifen. Hinreichend ist z.B. folgende Bedingung: Findet man eine Funktion von der Menge der in der Schleife vorkommenden Variablen in die Menge der ganzen Zahlen, deren Funktionswert mit jedem Schleifendurchlauf echt kleiner wird, gleichzeitig aber eine feste untere Schranke nicht unterschreitet, so terminiert die Schleife. Die Funktion ist häufig nicht leicht zu finden, besonders dann nicht, wenn reelle Variablen beteiligt sind.

  = durch Aufzeigen des Zusammenhangs zwischen Verifikation und Testen.

    Beispiel: Mögliche Vorgehensweisen hierzu können sein:

    1) Darstellung unterschiedlicher Korrektheitsniveaus von Programmen:
       - syntaktische Korrektheit eines Programms.

- Korrektheit bezgl. Einsatzbereich, d.h., das Programm liefert für typische Eingabewerte des Einsatzbereichs korrekte Ergebnisse.
- Das Programm liefert für irgendwelche Eingaben (meist "Spielbeispiele") korrekte Ausgaben.
- Korrektheit für alle zulässigen (auch boshaft gewählte) Eingaben; dieses Niveau fordert man i.a. von Programmen; alle übrigen Programme sind nahezu wertlos.
- Korrektheit für *alle* Eingaben.

2) Vermittlung des Unterschieds zwischen Verifikation und Testen:
   Verifikation = Mathematischer Nachweis, daß ein Programm für alle zulässigen Eingaben korrekt ist.
   Testen = Wahl einer *endlichen* Teilmenge der Menge der Eingabewerte; arbeitet das Programm für diese Testmenge korrekt, so schließt man (besser: so hofft man), daß es auch für alle übrigen zulässigen Eingaben korrekt ist.
3) Motivation durch Aufzeigen aktueller Anforderungen an die Korrektheit von Programmen, etwa so: "Computer werden zunehmend auch in kritischen Bereichen eingesetzt (z.B. Steuerung von Eisenbahnen, Signalanlagen, medizinischen Apparaten oder Kraftwerken). Da hier besonders hohe Anforderungen an die Korrektheit der verwendeten Programme gestellt werden müssen, verlangen die Sicherheitsüberprüfungen (z.B. Technische Überwachungsvereine) heute meist eine Verifikation der beteiligten Programme für alle möglichen Eingabewerte.
   Beispiel: Zwischen den beiden gut 1 km entfernten Bauteilen der Universität Dortmund verläuft eine Hänge-Bahn (H-Bahn). Die H-Bahn transportiert vollautomatisch Fahrgäste hin und her. Es gibt keine Fahrzeugführer. Die Sicherheit des Systems wurde vom TÜV-Rheinland überprüft. Die Überprüfung bezog sich auch auf die Steuerungsprogramme. Sie wurden stufenweise in eine PASCAL-ähnliche Sprache übersetzt und anschließend anhand der Spezifikation verifiziert."

= durch ausgewählte kleine Programme, deren Semantik nicht unmittelbar ersichtlich ist, aber durch Testen und Nachdenken schließlich exakt ermittelt werden kann.
Beispiel: Das folgende pathologische Programm eignet sich zur Motivation exakter Semantikdefinitionen:

```
program p(input,output);
var x: integer;
function q(var x: integer): integer;
begin
      q:=x*x; x:=x+1
end;
begin
      read(x); write(x+q(x)+x+q(x))
end.
```

Auf die Eingabe 3 erwartet man die Ausgabe 32. Drei verschiedene Rechnersysteme des Autors liefern jedoch den Wert 33. Diese Ausgabe ist durch die

Sprachbeschreibung nicht gedeckt. Offenbar macht die Implementierung hier mangels einer genauen Semantikdefinition, "was sie will".

2.5 Vorteile des Brunerschen Konzepts

Bruner sieht vor allem folgende Vorteile:

- Der Unterrichtsgegenstand wird faßlicher, wenn man seine Grundlagen versteht.
- Fundamentale Ideen schaffen Beziehungsnetze. Sie prägen Details, die man unzusammenhängend schnell wieder vergessen würde, eine verbindende Struktur auf. Über sie können Einzelheiten wieder rekonstruiert werden (Verdichtung von Information).
- Fundamentale Ideen unterstützen den nichtspezifischen Transfer. Durch ihre Allgemeinheit können viele Fälle als Spezialfälle behandelt werden.
- Da fundamentale Ideen den Stoff vertikal gliedern, verringern sie den Abstand zwischen elementarem und fortgeschrittenem Wissen. Dahinter verbirgt sich die Überzeugung

  > "... daß geistige Tätigkeit überall dieselbe ist, an den Fronten des Wissens ebenso wie in einer dritten Klasse. ... Der Unterschied liegt im Niveau, nicht in der Art der Tätigkeit." [B60].

  Die Tätigkeiten eines Wissenschaftlers und eines Volksschülers unterscheiden sich also nicht prinzipiell, da beide die gleichen fundamentalen Ideen, jedoch auf unterschiedlichem Niveau, verwenden.

Weitere Vorzüge:

- Ein Vorteil, der sich im Kurssystem auszahlen kann: Da die fundamentalen Ideen im Zentrum des Unterrichts stehen, kann man nahezu an jeder Stelle abbrechen, ohne die Schüler mit einem sinnlosen Fragment alleine zu lassen (vgl. [J78]).
- Während fundamentale Ideen eine längerfristige Gültigkeit besitzen (vgl. Zeitkriterium), sind Details schnell wieder veraltet. Dies trifft vor allem auf die Informatik zu, so daß das Brunersche Konzept hier besonders schlagkräftig erscheint.
- Es gibt bisher keine Philosophie der Informatik. Hier kann eine Zusammenstellung von fundamentalen Ideen als Einstieg dienen und helfen, das "Wesen" von Informatik zu klären und sie gegen andere Wissenschaften abzugrenzen.

3 Fundamentale Ideen der Informatik

Bei der Entwicklung einer Kollektion von fundamentalen Ideen der Informatik werden wir uns im wesentlichen an das in Abschnitt 2.3 vorgeschlagene Vorgehen halten. 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 (zu alternativen Ideenvorschlägen s. vor allem [K89] und [CS86,D84,L86]).

Zum 1. Schritt: Analyse *konkreter* Inhalte und Ermittlung von Beziehungen und Analogien zwischen Teilgebieten und intellektuellen Niveaus. Welche konkreten Inhalte der Informatik soll man 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 diesen Entwicklungsprozeß auf Vorkommen von fundamentalen Ideen zu analysieren. Dies ist Thema des folgenden Abschnitts.

3.1 Softwareentwicklung

Die Grundlage für die Untersuchung von Fragestellungen bei der Softwareentwicklung bildet der Software life cycle (Standardform s. Abb. 4 [CS88]). Pfeile markieren jeweils Tätigkeiten, zu deren Abschluß das in Rechtecken eingetragene Produkt vorliegt.

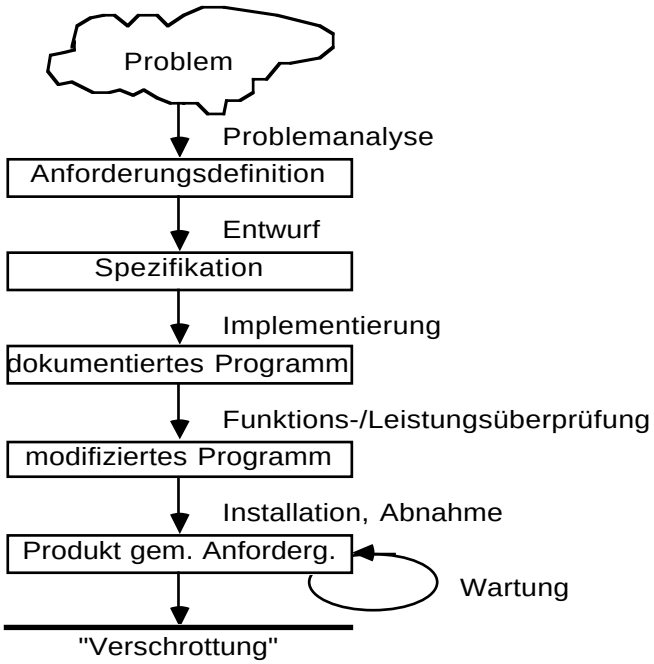


Abb. 4: Software life cycle

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.
  Als Darstellungsmittel für diese Phase eignen sich z.B. Datenfluß- oder SADT-Diagramme.

- Sollkonzeptentwicklung: Festlegung der Anforderungen an die zu erstellende Software durch Angabe u.a. des Benutzermodells, der Basismaschine, der Benutzermaschine usw. Hierbei werden noch keine Aussagen zur Realisierung gemacht.
- 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 berechenbar sind?) und ökonomisch vertretbar sind. Das Ergebnis dieser Studie führt entweder zur Aufgabe des Projekts, zur Revision der Anforderungen oder zur Durchführung des Projekts.
- Projektplanung: Erstellung von Zeitplänen, Zusammenstellung von Teams, Verteilung des Personals und der übrigen Hilfsmittel.

Das gesamte Ergebnis der Problemanalyse wird in der Anforderungsdefinition festgehalten, die Bestandteil des Vertrags zwischen Auftraggeber und Softwareproduzent wird.

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 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. Die Durchführbarkeitsanalyse bezieht sich dabei 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, entwickeln die Programmierer 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. 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 für jedes Modul die Funktion, die Schnittstellen und Hinweise zur Anwendbarkeit, sowie ein Gesamtüberblick über die Abhängigkeit der einzelnen Module untereinander enthalten sind. Die Spezifikation kann graphisch (z.B. durch die HIPO-Methode) oder sprachlich mit Spezifikationssprachen oder abstrakten Datentypen formuliert werden. Zur Beschreibung der Modulhierarchie und der gegenseitigen Abhängigkeiten eignen sich z.B. Bäume bzw. gerichtete Graphen.

Im Zentrum der Entwurfsphase steht die Idee der *Modularisierung* 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 *Parameterkonzept* enthält und die es gestattet, 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 die Module 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. Das Ergebnis der Implementierung ist ein dokumentiertes Programm.

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, sog. Paradigmen, z.B. *Divide-and-Conquer*, *Branch-and-Bound* usw., zur Verfügung. Bei der Programmierung ist *strukturiert* vorzugehen. Damit die Module 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 des fertigen Programms oder seiner Teile auf *Korrektheit*. Formal oder durch Testeingaben werden die Programmteile *partiell* und *total* verifiziert. Bei nebenläufigen Programmen spielen hierbei auch Begriffe wie *Konsistenz* und *Fairness* eine Rolle.
Leistungsüberprüfung. Nach der Korrektheitsüberprüfung müssen Leistungsmessungen durchgeführt werden. Im Zentrum dieser Phase steht, ebenfalls unter dem Qualitätsaspekt, 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 Idee der *Algorithmisierung* ist bereits explizit erwähnt worden, die beiden anderen treten nur implizit auf, die Idee der *Sprache* und die Idee der *strukturierten Zerlegung*. Beide Ideen wollen wir im folgenden genauer unter die Lupe nehmen, da sie auch Anhaltspunkte für eine Reihe weiterer Ideen liefern.

Sprache. Nicht nur in 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 besteht, z.B. beim VLSI-Entwurf oder beim Entwurf logischer Schaltungen. Dieses Vorgehen bietet u.a. folgenden Vorteil: Es vereinheitlicht die Sichtweise, denn jedes Problem reduziert sich dann auf ein Problem über Wörtern; die Manipulation von Sprachen und Wörtern ist andererseits wiederum gut erforscht.

Beispiel: Eine Rechnerarchitektur veranschaulicht man sich häufig durch ein Ebenenmodell (Abb. 5). Jeder Ebene wird ein anderes Sprachniveau zugeordnet, mit dem die Objekte und Aktionen der Ebene adäquat beschrieben werden können. Der Benutzer arbeitet üblicherweise auf der obersten Ebene. Jede seiner Aktionen wird schrittweise in das jeweils tiefer liegende Sprachniveau und schließlich auf die Hardware transformiert.

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*.

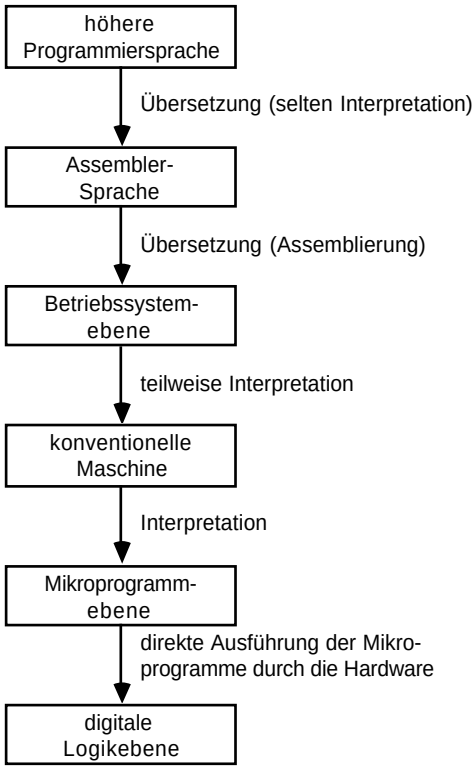


Abb. 5: Ebenenmodell der Rechnerarchitektur

Zerlegung. Die Zerlegung konkretisiert sich 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.

Offenbar können wir bei der Idee der Zerlegung einen horizontalen und einen vertikalen Aspekt unterscheiden, wobei die Hierarchisierung (Abb. 6) den vertikalen und die Modularisierung (Abb. 7) den horizontalen beschreibt. Die hierarchische Modularisierung entsteht dann als Mischform aus diesen beiden Grundformen (Abb. 8).

Die Idee der Hierarchisierung finden wir auch noch in vielen anderen Zusammenhängen: Ebenenmodelle der Rechnerarchitektur (s.o.), Sprachhierarchien (wichtigster Vertreter ist hier die Chomsky-Hierarchie), Maschinenmodelle, Komplexitäts- und Berechenbarkeitsklassen, virtuelle Maschinen, ISO-OSI-Ebenenmodell.

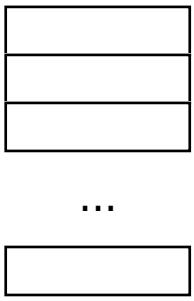


Abb. 6: Hierarchisierung

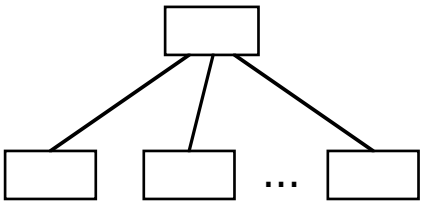


Abb. 7: Modularisierung

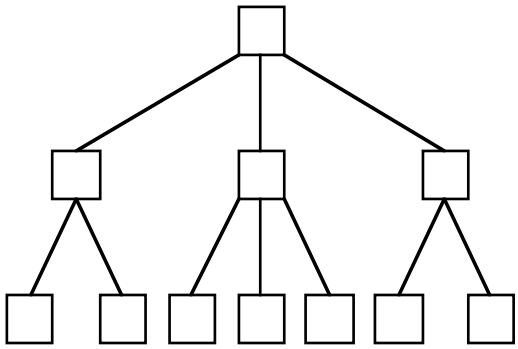


Abb. 8: Hierarchische Modularisierung

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.

Beispiel: 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 endlichen 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. Die

Fundamentalität der Orthogonalisierung erkennt man an ihrer vielfältigen Anwendbarkeit innerhalb und außerhalb der Informatik (vgl. Horizontal- und Sinnkriterium), z.B. bei

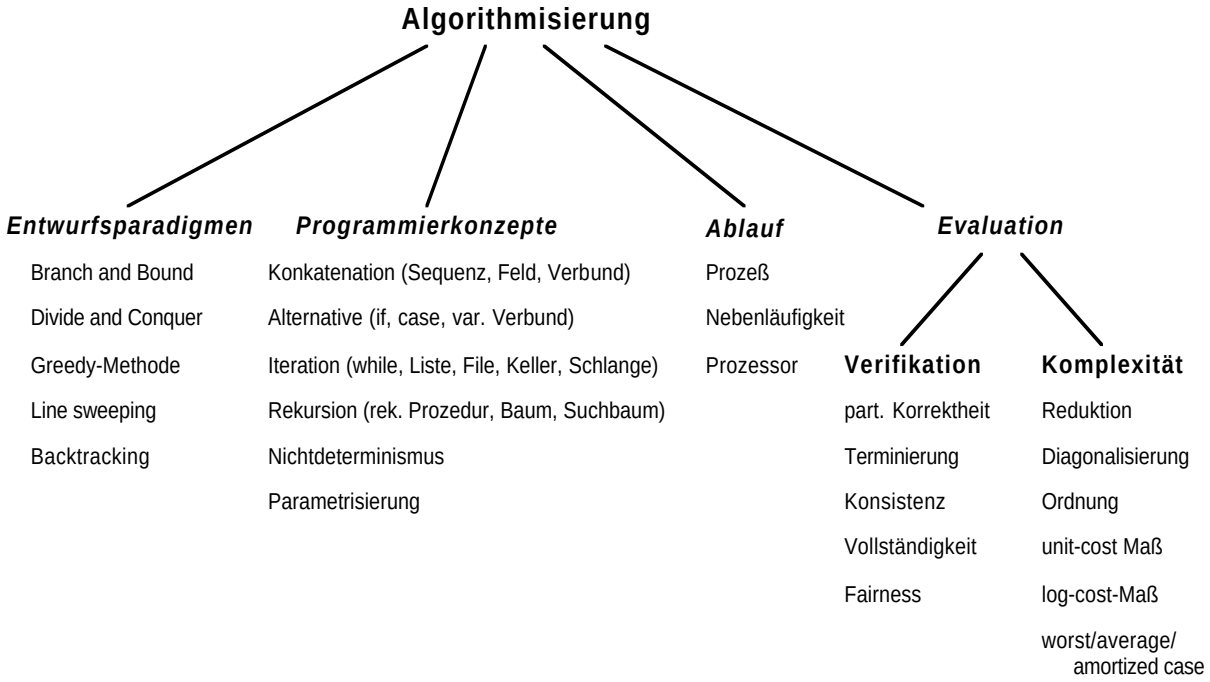
- Programmiersprachen. Zentrales Prinzip der Sprache ALGOL68 ist der Orthogonalentwurf: Es gibt einige wenige elementare Daten- und Anweisungstypen, die jedoch in beliebiger Weise kombiniert werden dürfen. Zum Beispiel sind in ALGOL68 Felder ebenso wie alle anderen Datentypen als Ergebnistypen von Funktionen zugelassen. 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. Die Dyck-Sprache bildet gewissermaßen eine Basis der 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. "Master"-Ideen sind wie erwähnt Algorithmisierung, 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.

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 Methode zur Realisierung von Hierarchien. Man erkennt also, daß die Ideen teilweise in vielfältiger Weise miteinander verflochten sind. Eine exakte Zuordnung und scharfe Abgrenzung untereinander ist kaum möglich.



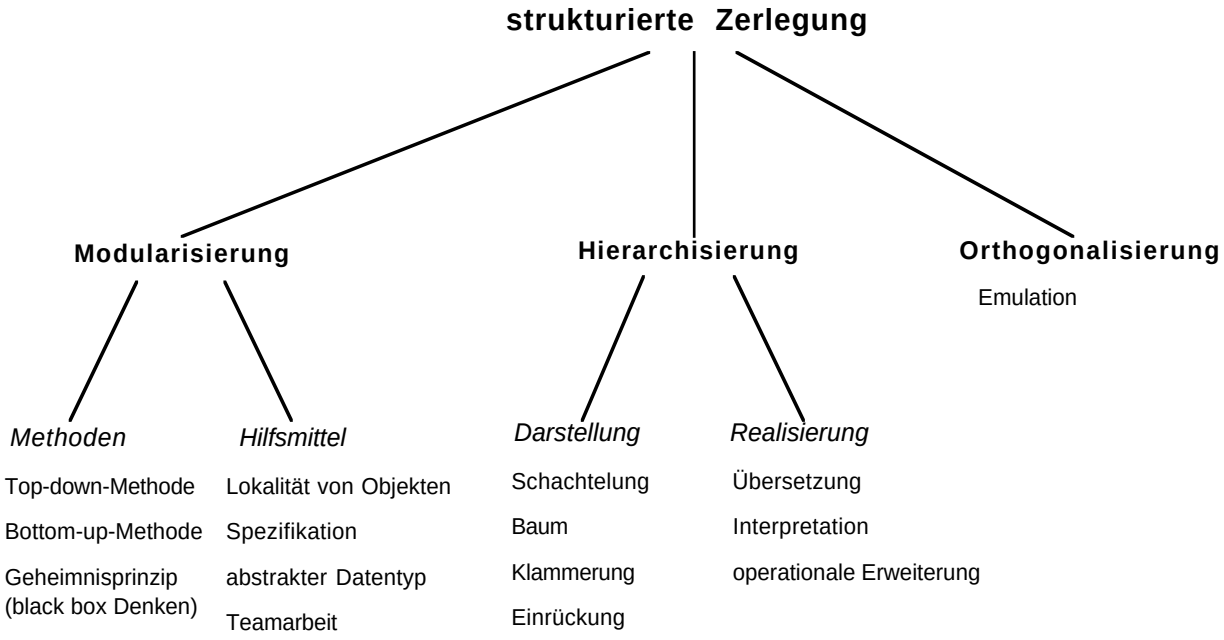


Abb. 9: Fundamentale Ideen der Informatik

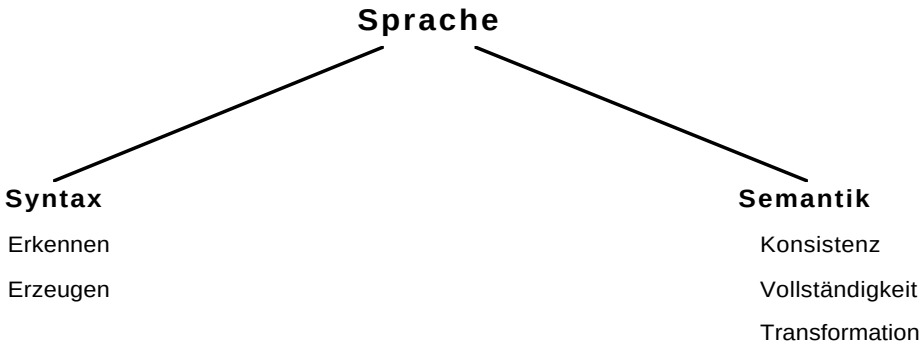


Abb. 9: Fundamentale Ideen der Informatik (Fortsetzung)

3.3 Fundamentalitätsnachweis

Nun wollen wir einzelne ausgewählte Ideen der Kollektion beispielhaft auf ihre Fundamentalität hin überprüfen. Maßstab bilden naturgemäß die vier Kriterien der Definition. Die Masterideen weist man leicht als fundamental nach. Sinn- und Horizontalkriterium hatten wir in Abschnitt 3.2 bereits skizzenhaft verifiziert.

Bei den folgenden Ideen beschränken wir uns zum Nachweis des Vertikalkriteriums darauf, jeweils Unterrichtsgegenstände für die Primarstufe (P), die Sekundarstufen I (S1) und II (S2) zu skizzieren, mit denen die entsprechenden Ideen auf den jeweiligen intellektuellen Niveaus vermittelt werden können. Auf das Zeitkriterium gehen wir en bloc am Schluß des Abschnitts ein.

Divide-and-Conquer-Methode.

Horizontalkriterium: Anwendung bei Sortier- und Suchverfahren, bei allen Problemen im Zusammenhang mit der Datenstruktur "Baum", bei der Matrixmultiplikation, bei algorithmischer Geometrie, bei der Separation von planaren Graphen.

Vertikalkriterium:

(P) Ein Kind kann Pappkarten der Größe nach sortieren, indem es die Karten auf mehrere Mitschüler aufteilt und die zurückgelieferten sortierten Stapel mischt. Zahlen können nach der Methode des binären Suchens erraten werden.

(S1) Verfahren der algorithmischen Geometrie, z.B. zur Bestimmung der konvexen Hülle können zur Vertiefung herangezogen werden.

(S2) Komplexitätsbetrachtungen für allgemeine Divide-and-Conquer-Verfahren; Aufstellen einer Rekurrenz für die Laufzeit und Ermitteln ihrer Lösung.

Sinnkriterium: Das Verfahren erscheint im Alltag z.B. bei allen Formen der hierarchisch organisierten Arbeitsteilung oder bei verschiedenen Suchverfahren: Ein Kind hat etwas verloren. Andere Kinder helfen ihm beim Suchen. Jedes Kind sucht in einem bestimmten Abschnitt.

Worst case-Analyse.

Horizontalkriterium: Worst case-Analysen werden angestellt bei Algorithmen bezgl. Zeit und Speicherplatz, bei Zeitplänen für Projekte, zur Fehlerabschätzung bei der Gleitpunktarithmetik oder bei stochastischen Algorithmen.

Vertikalkriterium:

(P) Worst case-Überlegungen können mit Fragen der Form beginnen: Wie lange brauche ich für meinen Schulweg im schlimmsten Fall, wenn alle Busse Verspätung haben, Glatteis ist, alle Ampeln rot sind, ...? Oder: Wieviele Fragen muß ich höchstens stellen, um beim Zahlenraten nach dem binären Suchverfahren die erdachte Zahl zu ermitteln?

(S1) Hier kann sich eine mehr formale Darstellung anschließen, etwa durch Beziehung der Laufzeit auf die Länge der Eingabe und Ermitteln des für jede Länge schlimmsten Falls.

(S2) Formale Definition der Worst case-Laufzeit und Nachweis von unteren Schranken für die Worst case-Laufzeit.

Sinnkriterium: Im Alltag trifft man Worst case-Überlegungen häufig bei Risikoabschätzungen, z.B. bei der Finanzierung eines Eigenheims (Wie hoch darf der Kredit sein, damit ich die Belastung auch im schlimmsten Fall (Arbeitslosigkeit) noch tragen kann?), bei der Festlegung des größten anzunehmenden Unfalls GAU in Atomkraftwerken, beim Sicherheitsabstand zwischen Autos (Der Abstand sollte so

groß sein, daß der Nachfolger auch bei absichtlicher Vollbremsung (schlimmster Fall) des Vorgängers noch bremsen kann).

Abstrakter Datentyp.

Horizontalkriterium: Bei allen Formen der Spezifikation von Objekten, bei denen die Operationen und deren Eigenschaften im Vordergrund stehen und zunächst noch kein Bezug zur Implementierung besteht, findet man die Idee der abstrakten Datentypen.

Vertikalkriterium:

(P) Die natürlichen Zahlen kann man als abstrakten Datentyp mit der Konstanten 1 und den Operationen +1 und -1 darstellen. Analog für die sog. Klötzchenwelt: Auf einem Tisch liegen Bauklötze, die man aufeinander stapeln kann. Operationen sind das Aufeinanderlegen von Klötzen und die Abfragen, ob auf einem Klotz noch ein weiterer liegt bzw. ob ein Klotz direkt auf dem Tisch liegt. Kann man mit diesem Modell jede beliebige Konfiguration von Klötzchentürmen herstellen (Idee der Vollständigkeit)?

(S1) Die beiden unter (P) aufgeführten Beispiele können hier präzisiert werden. Fragen zur Konsistenz und Vollständigkeit eines abstrakten Datentyps können sich anschließen. Welche Gesetze gelten für die Operationen der Klötzchenwelt?

(S2) Auf diesem Niveau kann man eine formale Notation für abstrakte Datentypen einführen und umfangreichere Beispiele (Keller, Schlange, File) behandeln. Überlegungen zur Implementierung abstrakter Datentypen können folgen.

Sinnkriterium: Ein mit abstrakten Datentypen vergleichbares Vorgehen findet man bei jedweder Konstruktion von Maschinen, indem man spezifiziert, welches Verhalten die Maschine besitzen soll. Besonders markant ist dieses Prinzip bei öffentlichen Ausschreibungen, bei denen lediglich das "abstrakte" Verhalten eines Objekts ohne Bezug zu herstellerspezifischen "Implementierungen" genannt werden darf.

Eine detailliertere Ausarbeitung zur exemplarischen Vermittlung der fundamentalen Idee der Terminierung auf drei unterschiedlichen Schulstufenniveaus ist in Vorbereitung [S92]. [K89] enthält mehrere Unterrichtseinheiten zur Vermittlung diverser fundamentaler Ideen der Informatik im Mathematikunterricht der Primarstufe.

Nun bleibt noch zu zeigen, wie sich die Ideen in der historischen Entwicklung der Informatik darstellen (Zeitkriterium). Einige wichtige Stationen der ersten 30 Jahre Informatik ab 1950 zeigen die Abb. 10 bis 12.

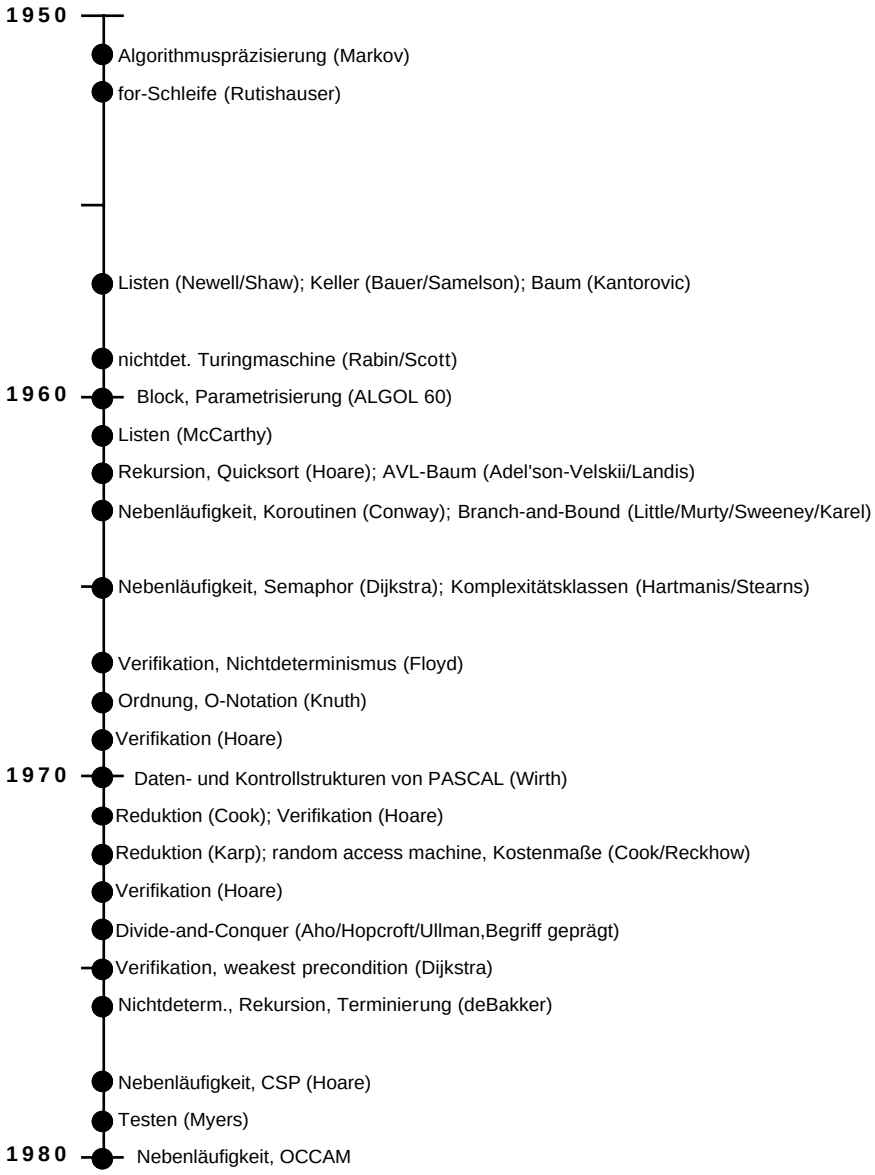


Abb. 10: Historische Entwicklung der Algorithmisierung und zugehöriger Ideen

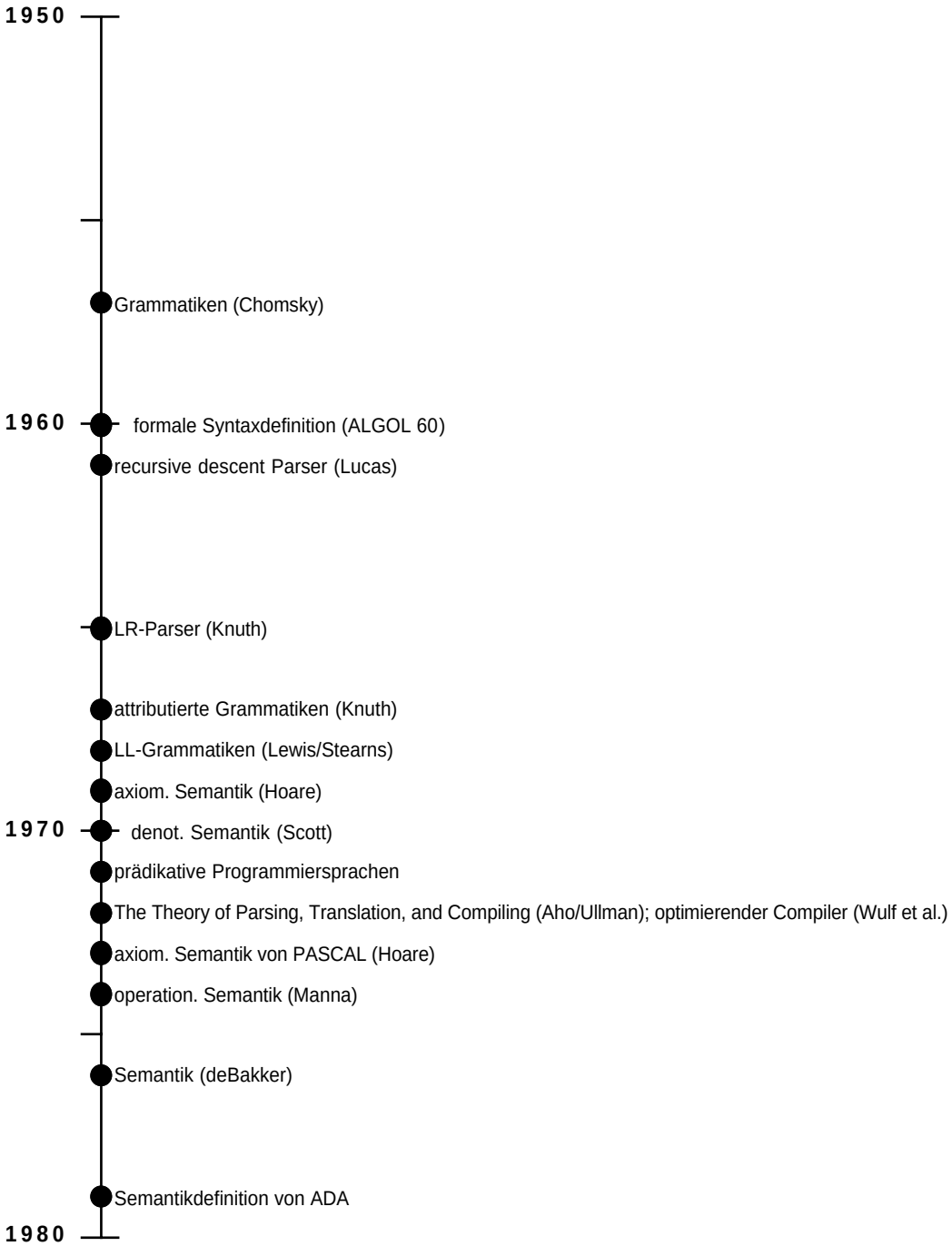


Abb. 11: Historische Entwicklung der Sprache und zugehöriger Ideen

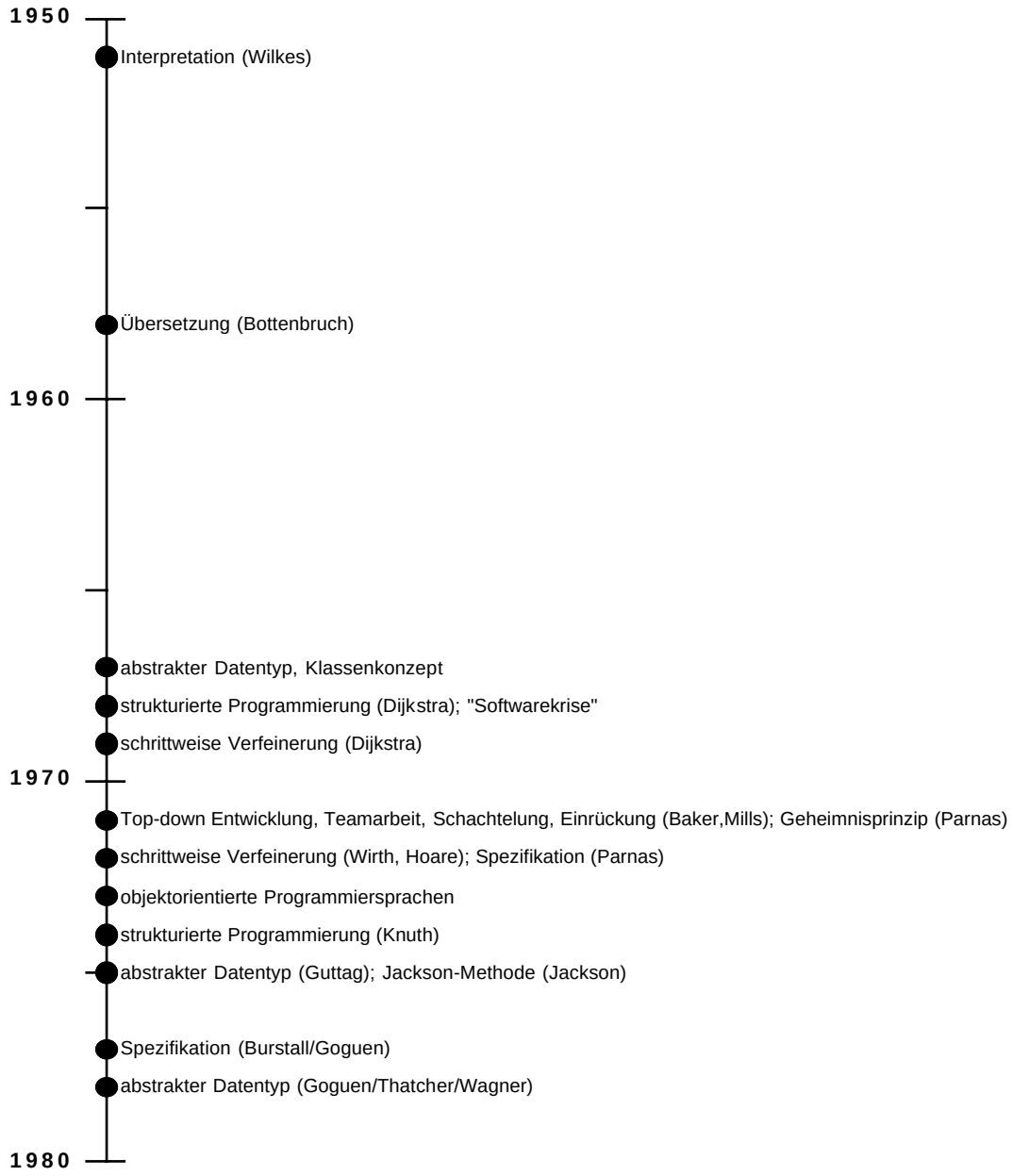


Abb. 12: Historische Entwicklung der Zerlegung und zugehöriger Ideen

4 Schlußbemerkungen

Wir haben in dieser Arbeit das von Bruner entwickelte Unterrichtsprinzip der Orientierung an fundamentalen Ideen auf die Informatik übertragen und zur Diskussion gestellt. Nach den bisherigen Überlegungen erscheint es gerade für den Informatikunterricht als besonders tragfähiges Konzept, weil sich die Schule dadurch auf absehbare Zeit dem Innovationsdruck der Wissenschaft Informatik entzieht, ohne jedoch inhaltlich ins Abseits zu geraten.

Um dieses Erscheinungsbild abzusichern, wären u.a. folgende weitergehenden Untersuchungen hilfreich:

1) Entwicklung von Curricula für Informatik, bei denen fundamentale Ideen im Zentrum des Unterrichts stehen.
2) Erarbeitung von geeigneten Beispielen für diverse kognitive Niveaus, die gewisse Ideen besonders deutlich erkennen lassen; Entwicklung zugehöriger Unterrichtseinheiten für verschiedene Schulstufen und Erprobung der Einheiten im Unterricht.
3) Der vorgestellte Ideenkatalog basiert zwar weitgehend auf objektiver Beobachtung und Analyse informatischer Aktivitäten und Denkweisen, dennoch ist er notwendigerweise (wie oben erwähnt) in erheblichem Maße subjektiv geprägt. Dieser Katalog ist zu verfeinern und durch Diskussion innerhalb der "scientific community" abzusichern.

Danksagung. Herrn Prof. Dr. V. Claus danke ich für die Anregung zu dieser Arbeit und für zahlreiche Diskussionen über fundamentale Ideen.

Literatur

[B60] Bruner, J.S.: "The process of education", Cambridge Mass. 1960 (dt. Übers.: "Der Prozeß der Erziehung", Berlin 1970)
[CS86] Claus, V.; Schwill, A.: "Informatikkenntnisse von Jugendlichen, untersucht am Beispiel der drei Bundeswettbewerbe Informatik", Informatik-Spektrum 9 (1986) 270-279
[CS88] Claus, V.; Schwill, A.: Duden Informatik, Bibliographisches Institut AG, Mannheim 1988
[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
[G77] Gärtner, H.: "Lehrplan Biologie - Analyse und Konstruktion", Sample Verlag 1977
[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
[J78] Jung, W.: "Zum Begriff einer mathematischen Bildung - Rückblick auf 15 Jahre Mathematikdidaktik", mathematica didactica 1 (1978) 161-176
[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
[L86] Lockemann, P.C.: "Konsistenz, Konkurrenz, Persistenz - Grundbegriffe der Informatik?", Informatik-Spektrum 9 (1986) 300-305
[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, LNCS 438
[S81] Schmidt, H.J.: "Fachdidaktische Grundlagen des Chemieunterrichts", Vieweg Verlag 1981
[S79] Schreiber, A.: "Universelle Ideen im mathematischen Denken - ein Forschungsgegenstand der Fachdidaktik", mathematica didactica 2 (1979) 165-171
[S83] Schreiber, A.: "Bemerkungen zur Rolle universeller Ideen im mathematischen Denken", mathematica didactica 6 (1983) 65-76
[S82] Schweiger, F.: "Fundamentale Ideen der Analysis und handlungsorientierter Unterricht", Beitr. zum Mathematikunterricht (1982) 103-111
[S92] Schwill, A.: "Drei Unterrichtsbeispiele zur Terminierung von Programmen", in Vorbereitung (1992)
[S70] Spreckelsen, K.: "Strukturelemente der Physik als Grundlage ihrer Didaktik", Naturwiss. im Unterr. 18 (1970) 418-424
[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

[W29] Whitehead, A.N.: "Die Gegenstände des mathematischen Unterrichts", Neue Sammlung 2 (1962) 257-266 (Originalarbeit von 1929)
[W81] Wittmann, E.C.: "Grundfragen des Mathematikunterrichts", Vieweg Verlag 1981