Begriffliche Strukturen der Informatik: Ein empirischer Zugang

Markus Schneider

Didaktik der Informatik – Technische Universität München
Boltzmannstr. 3
85748 Garching
markus.schneider@in.tum.de

Abstract: Die begrifflichen Strukturen der Informatik sind nicht nur von rein akademischen Interesse, sondern sie bilden auch die theoretische Grundlage der Entwicklung von Curricula an Schule und Hochschule. Da ist zum einen die in den letzten Jahren in den Blickpunkt didaktischer Arbeit gerückte Frage nach Standards für das Fach Informatik und zum anderen die Konzeption von Studienabläufen im Rahmen des neu eingeführten Bachelor/Master-Systems. In der vorliegenden Arbeit sollen insbesondere solche begrifflichen Strukturen identifiziert werden, die vom informatischen Anfangsunterricht an der Schule bis zu Einführungsvorlesungen an der Hochschule auftreten. In einer konzeptionellen Analyse von Übungsaufgaben an Schule und Hochschule werden die darin enthaltenen fundamentalen Ideen der Informatik identifiziert. Dadurch lassen sich Ideen identifizieren, die sich durch die gesamte Lehre der Informatik ziehen. Die Arbeit schließt mit einer kritischen Reflexion zu Details der fundamentalen Ideen.

1 Begriffliche Strukturierung: Wozu?

Welchen Nutzen hat eine wissenschaftliche Disziplin und nicht zuletzt auch deren Fachdidaktik von der Kenntnis der jeweiligen inhaltlichen Strukturen? Die begriffliche Strukturierung ist für jede Wissenschaft zum einen aus rein fachsystematischen Gründen von entscheidender Bedeutung. Insbesondere die Informatik, die mitunter als Strukturwissenschaft bezeichnet wird [We71], darf ihre eigene Strukturierung keinesfalls zurückstellen.

Darüber hinaus gibt es für einen in der Didaktik der Informatik Tätigen jedoch auch praktischere Gründe, sich mit der inhaltlichen Struktur seines Faches zu beschäftigen: Aufgaben wie die Gestaltung und der Entwurf von Vorlesungen bzw. Seminaren, oder die Konzipierung von Unterrichtsabläufen sind Herausforderungen, mit denen ein Lehrender tagtäglich konfrontiert ist. Zwar wird dem Lehrenden an der Schule die Grobplanung durch Lehrpläne erleichtert, jedoch bleibt die Feinstrukturierung letztlich jedem selbst überlassen. Für diese Arbeit benötigt jeder Lehrer eine begriffliche Strukturierung, die die fachlichen und didaktischen Abhängigkeiten auf unterschiedlichem Komplexitätsniveau, von der Grundschule bis hin zur Lehre an der Hochschule dokumentiert.

2 Vorhandene Klassifikationen

Selbstverständlich existieren bereits Klassifikationen der wichtigsten Begriffe der Informatik. Einige zentrale sollen hier angesprochen und bewertet werden. Dabei wird zwischen Klassifikationen unterschieden, die sich primär an den verschiedenen Spezialdisziplinen der Informatik orientieren und denen, die übergeordnete Prinzipien als entscheidendes Kriterium verwenden. Entsprechend werden erstere im weiteren als domänenorientierte und letztere als prinzipienorientierte Klassifikationen bezeichnet.

2.1 Domänenorientierte Klassifikationen

ACM-Klassifikation: Diese Klassifikation ([Ne98]) ist eine Taxonomie für Begriffe der Informatik, die mit der Version aus dem Jahre 1998 einen vorläufigen Endzustand erreicht hat. Sie wurde bereits im Jahre 1964 in einer ersten Version veröffentlicht, woraus eine entsprechend umfangreiche und nahezu alle Bereiche der Informatik umfassende Klassifikation resultierte. Die Intention zur Erstellung dieses Klassifikationsschemas war jedoch keine didaktische, sondern eher eine bibliographische. In der Version von 1982 nennt sich die Klassifikation noch „Computing Reviews Classification System“, was darauf hindeutet, das man in erster Linie ein System zur Klassifikation der in der Informatik auftretenden wissenschaftlichen Literatur im Auge hatte. Folglich wird die oberste Ebene der Taxonomie durch Begriffe repräsentiert, die stellvertretend für die verschiedenen Spezialdisziplinen der Informatik stehen: „Hardware“, „Software“, „Data“, „Theory of Computing“ um nur einige zu nennen. Leider lassen sich dann viele übergeordnete Begriffe, wie beispielsweise die Rekursion, nur schwer in die Klassifikation einfügen: Sie findet sich auf einer relativ tiefen Ebene des Klassifikationsbaumes bei den Progammierkonstrukten, obwohl Rekursion sicherlich bei zahlreichen anderen Begriffen der Informatik, wie etwa bei Design-Pattern, formalen Sprachen oder rekursiven Datenstrukturen eine Rolle spielt (vgl. z.B. [Sc02]).

Domänenorientierte Klassifikation von P. Denning [De99]: Eine weitere domänenorientierte Klassifikation wurde von P. Denning vorgeschlagen [De99]. Es handelt sich hier um eine zweidimensionale Klassifikation: Fachlich-inhaltliche Kategorien, wie Algorithmen und Datenstrukturen, Programmiersprachen, Netzwerke, AI, Datenbanken bilden eine Dimension. Orthogonal dazu steht eine Kategorisierung in unterschiedliche Abstraktionsniveau: Theorie, Abstraktion, Design. Im Falle der inhaltlichen Kategorie „Programmiersprachen“ sind Konzepte wie λ-Kalkül oder Turing-Maschinen der Theorie, die verschiedenen Programmierparadigmen (objektorientiert, funktional, etc.) der Abstraktion und die konkreten Programmiersprachen der Designebene zugeordnet.

Resümee: Das entscheidende Defizit domänenorientierter Klassifikationen ist offensichtlich: Durch Aufspaltung der Inhalte in die verschiedenen Teildisziplinen fällt es dem Lernenden schwer, Konzepte, die in allen Teilbereichen auftreten, als solche zu erkennen. Lehrstrategien, die ausschließlich auf derartigen Klassifikationen aufbauen, bergen somit die Gefahr, isoliertes und punktuelles Wissen aufzubauen, anstatt die Begriffe zu vernetzen und übergeordnete Strukturen zu bilden.

2.2 Prinzipienorientierte Klassifikationen

Eine vollkommen andere Strukturierungsstrategie weisen prinzipienorientierte Klassifikationen, wie die so genannten fundamentalen Ideen von A. Schwill [Sc93] oder die „Great Principles“ von P. Denning [De03], [De06] auf.

Fundamentale Ideen: Ohne auf die Details der fundamentalen Ideen einzugehen, seien hier die entscheidenden Elemente vorgestellt, da diese in dieser Arbeit von zentraler Bedeutung sind: Der Softwareentwicklungsprozess als eine der zentralen Aufgaben der Informatik bildet den Ausgangspunkt der „Ideenfindung“. Diesem werden die so genannten „Masterideen“ Algorithmisierung, Strukturierte Zerlegung und Sprache zugeordnet. Die Masterideen wiederum enthalten verschiedene Ideengruppen wie etwa „Entwurfsparadigmen“ oder „Ablauf“. Die einzelnen Ideengruppen werden schließlich mit den eigentlichen Ideen „aufgefüllt“. So enthält etwa die Ideengruppe der Programmierkonzepte die Ideen: Konkatenation, Alternative, Iteration, Rekursion, Parametrisierung und Nichtdeterminismus. Innerhalb der Ideengruppe „Hierarchisierung“ werden zwei weitere Untergruppen namens „Darstellung“ und „Realisierung“ eingeführt. Diese enthalten die Ideen der Schachtelung, des Baumes, der Klammerung und der Einrückung bzw. Compilierung und Interpretation. Die Masteridee der „Sprache“, die ursprünglich lediglich die Ideengruppen „Syntax“ und „Semantik“ beinhaltete, wurde von E. Modrow weiterentwickelt [Mo06].

Great Principles: Einen völlig anderen Weg als in der Arbeit aus dem Jahre 1999 [De99] beschreitet P. Denning in den Publikationen über „Great Principles of Computing“ [De03], [De06]. Zentrales Element der Klassifikation sind die so genannten „Great Principles“, die sich in verschiedenen Teilgebieten der Informatik und innerhalb eines Gebietes an zahlreichen Stellen wieder finden lassen [De06]; so sind etwa die P–NP Problematik, der Algorithmusbegriff, oder die verschiedenen Zeitkomplexitäten zu den Great Principles zu zählen. Dabei verwendet er für ein derartiges „Great Principle“ eine Definition, die der Definition der fundamentalen Idee, wie sie von Bruner [Br60] gegeben wird, verblüffend ähnlich ist, wie wohl Denning sich nicht darauf bezieht. Demzufolge ist die Liste der „Great Principles“ mit den fundamentalen Ideen von Schwill durchaus vergleichbar, wenn gleich sie nicht identisch ist. Allerdings scheint die endgültige Formulierung aller Prinzipien noch nicht abgeschlossen zu sein, denn Denning hat diese bisher nur elektronisch als vorläufige Liste veröffentlicht [De06]. Des weiteren ist die Anzahl der Prinzipien im Vergleich zum Schwillschen Ideenkatalog sehr umfangreich und eine detaillierte Beschreibung, wie die Auswahl der Prinzipien erfolgt, nicht verfügbar.

2.3 Folgerung

Wegen der oben angesprochenen Problematik mit domänenorientierten Klassifikationen, verbleiben aus didaktischer Sicht als Arbeitsgrundlage für eine Klassifikation der zentralen Begriffe der Informatik in erster Linie die prinzipienorientierten Klassifikationen. Da jedoch der Ansatz von Denning und insbesondere dessen Ideenkatalog noch nicht abgeschlossen ist, konzentriert sich diese Arbeit im Weiteren auf die Schwillschen Ideen. Der Schwillsche Ansatz, wie auch dessen Modifikationen nach Modrow und nicht zuletzt die „Great Principles“ von Denning werfen jedoch eine entscheidende Frage auf: Handelt es

sich bei den fundamentalen Ideen empirisch nachweisbar um übergeordnete Prinzipien der Informatik? Erfüllen Sie tatsächlich die Brunersche Forderung nach Fundamentalität [Br60]? Kritische Aufsätze zu den fundamentalen Ideen gibt es durchaus: Etwa von R. Baumann [Ba98], der nicht nur Teile des Katalogs der Ideen, sondern auch deren hierarchische Anordnung ablehnt. Jedoch wurde bisher noch nicht systematisch und empirisch untersucht, welche der verschiedenen Ideen den Brunerschen Kriterien genügen. Hierzu will diese Arbeit durch die detaillierte Untersuchung der Masterideen „Strukturierte Zerlegung“ und „Algorithmisierung“ einen Beitrag leisten. Insbesondere werden die Weiterentwicklungen von Modrow aus Platzgründen nicht diskutiert.

3 Fundamentale Ideen in Schule und Hochschule

3.1 Fundamentale Ideen in einführenden Vorlesungen zur Informatik

Während die fundamentalen Ideen ursprünglich für eine Verwendung im Schulbereich konzipiert wurden, soll hier auch untersucht werden, ob sie auf inhaltliche Strukturen der Hochschuldidaktik anwendbar sind. Betrachtet werden hierzu die Vorlesungen zur Einführung in die Informatik I aus den Jahren 2001 bis 2004 an der TU München [Br01, Bru02, Kn03, Se04]. Dabei wird analysiert, inwieweit die Lernziele von Vorlesungen im Grundstudium der Informatik durch die fundamentalen Ideen repräsentiert werden. Unter der Annahme, dass die Übungsaufgaben der Vorlesungen die Lernziele der Vorlesung repräsentieren, werden diese Aufgaben auf ihre Beziehung zu den fundamentalen Ideen analysiert.

Folgendes Beispiel einer Übungsaufgabe zur Vorlesung [Bru03] veranschaulicht die prinzipielle Vorgehensweise, wie einer Übungsaufgabe die entsprechende fundamentale Idee zugeordnet wird:

> **Übungsaufgabe:** Ein Buch ist charakterisiert durch den Namen des Autors, den Buchtitel und die Verlagsdaten. Die Verlagsdaten umfassen den Namen des Verlages, die Nummer der Auflage und das Erscheinungsjahr.
> a) Stellen Sie die Klasse Buch in UML dar und definieren Sie sie in Java. Geben Sie dabei auch eine Konstruktordefinition an.
> b) Es soll ein Objekt der Klasse Buch erzeugt werden, das das für die Vorlesung als weiterführende Literatur empfohlene Buch von Professor Brügge repräsentiert.
> B. Brügge und A. Dutoit: *”Object-Oriented Software-Engineering: Using UML, Patterns and Java”* 2. Auflage, Prentice Hall (2003)
> Stellen Sie das Objekt in UML dar und instanziieren Sie es in Java. Das Objekt soll mit dem Namen `bruеggeObjektorientierung` bezeichnet werden.

Da die fundamentalen Ideen nicht direkt die Begriffe der objektorientierten Modellierung enthalten, kann die Zuordnung nicht durch Extrahieren bestimmter Worte erfolgen. Vielmehr müssen die einzelnen Fragestellungen der Aufgabe herausgearbeitet und einer fundamentalen Idee zugeordnet werden. Obige Aufgabe ergibt folgende Zuordnung von Teilproblemen und fundamentalen Ideen:

- Aus der angegebenen Spezifikation müssen sinnvolle Klassen abgeleitet werden → Top-Down, Black-Box
- Den Klassen müssen sinnvolle Attribute zugeordnet werden → Parametrisierung
- Die Klassen sind als neue Datentypen zu interpretieren → Abstrakter Datentyp
- Es sind konkrete Objekte zu instanziieren → Lokalität von Objekten
- Es ist eine Implementierung in Java durchzuführen → Übersetzung

In derartiger Weise wurden nun alle Übungsaufgaben der Vorlesungen [Br01, Bru02, Kn03, Se04] analysiert. Die verschiedenen Ideen wurden hierbei nur tabellarisch aufgeführt, d.h. die hierarchische Ordnung blieb unberücksichtigt. Die Tabellen 1und 2 zeigen die Ergebnisse dieser Analyse. Die verschiedenen Einträge geben hierbei wieder, wie oft eine bestimmte Idee in den Übungsaufgaben der verschiedenen Vorlesungen thematisiert wurde. Aus Gründen der Übersicht sind hierbei Ideen, die der Masteridee „Strukturierte Zerlegung“ und solche, die der Masteridee „Algorithmisierung“ zugeordnet sind, in unterschiedlichen Tabellen aufgeführt.

3.2 Masteridee „Strukturierte Zerlegung“

	2001/2002	2002/2003	2003/2004	2004/2005
Top-down	0	7	> 10	> 10
Bottom-up	5	2	4	1
Black-box	> 10	> 10	> 10	> 10
Lokalität von Objekten	1	0	5	7
Spezifikation	4	5	2	4
Abstrakter Datentyp	> 10	> 10	> 10	> 10
Teamarbeit	> 0	> 0	> 0	> 0
Schachtelung	> 10	5	0	2
Baum	1	4	4	4
Klammerung	> 10	4	6	0
Einrückung	0	0	0	0
Übersetzung	0	0	> 10	> 10
Interpretation	> 10	> 10	> 10	0
Operationale Erw.	0	0	0	0
Emulation	1	0	0	0

Tabelle 1: Ideengruppe „Strukturierte Zerlegung“ in Einführungsvorlesungen zur Informatik

Tabelle 1 zeigt, dass über die Jahre hinweg nahezu jede Idee der Gruppe „Strukturierte Zerlegung“ thematisiert wird. Einzig die Idee der Operationalen Erweiterung wird von keiner der Vorlesungen tangiert. Zwar trifft dies auch auf die Einrückung zu, jedoch handelt es sich bei der Einrückung um eine Technik, die unausgesprochen bei vielen Implementierungen eingesetzt wird, wenn sie auch zur Lösung nicht notwendig ist. Darüber hinaus dürfte es sich bei Klammerung, Schachtelung und Einrückung um verwandte Begriffe zur Repräsentation von Baumstrukturen handeln. Insofern könnte man diese

drei Ideen auch als spezielle Repräsentationen von Bäumen ansehen und mit diesen zu einer Idee zusammenfassen.

Darüber hinaus fällt auf, dass manche Ideen in der einen Vorlesung häufig, in der anderen dagegen überhaupt nicht auftreten, beispielsweise die Idee des „Top-down-Entwurfs". Diese Idee tritt in der Vorlesung [Br01] nicht, in der Vorlesung [Bru03] dagegen sehr häufig auf. Grund hierfür ist das Design der Vorlesung. Die Vorlesung [Br01] startet mit Boolescher Algebra, thematisiert anschließend ausführlich die Elemente des funktionalen Paradigmas und gelangt schließlich zu imperativen Strukturen. Eine derartige Strategie legt auch für die zugehörigen Übungen eine Vorgehensweise von kleinen Funktionen zu großen Systemen und damit den Bottom-Up-Entwurf nahe. Andererseits wurde in [Bru03] ein systemorientierter Ansatz gewählt; objektorientierte Modellierung bestimmte von Beginn an die Vorlesung. Konsequenterweise wurden auch die entsprechenden Übungsaufgaben in diesem Sinne konzipiert, die Top-Down-Strategie ist deshalb das bestimmende Entwurfsschema. Die oben angegebene Übungsaufgabe zur Modellierung und Implementierung eines Buches ist ein Beispiel hierfür.

Eine Bemerkung sollte noch zu der Idee der Emulation, die der Ideengruppe der Orthogonalisierung zu zurechnen ist, gemacht werden. Diese Idee wird in den 4 Vorlesungen nur ein einziges Mal angesprochen und auch da, genau genommen, nur am Rande. In allen anderen Vorlesungen werden Probleme aus dem Bereich der Orthogonalisierung dagegen nicht thematisiert

3.3 Masteridee „Algorithmisierung"

Wenden wir uns nun der Umsetzung von Ideen aus dem Bereich der Algorithmisierung zu (Tabelle 2). Die Ideen, die die klassischen Kontrollstrukturen repräsentieren (Alternative, Konkatenation, Rekursion bzw. Iteration) sowie die Ideen der Parametrisierung werden sehr häufig thematisiert. Andererseits werden die Ideen, die der Gruppe der Entwurfsparadigmen und des Ablaufs zu zuordnen sind nur am Rande oder gar nicht angesprochen. Verifikations- und Komplexitätsaspekte werden ebenfalls nicht in allen Vorlesungen diskutiert.

	2001/2002	2002/2003	2003/2004	2004/2005
Branch & Bound	0	0	0	0
Divide & Conquer	1	0	2	1
Greedy	0	0	0	0
Line-Sweeping	0	0	0	0
Backtracking	0	3	0	1
Konkatenation	> 10	>10	>10	>10
Alternative	> 10	>10	>10	>10
Iteration	5	2	> 10	> 10
Rekursion	>10	> 10	> 10	> 10

Parametrisierung	> 10	> 10	> 10	> 10
Nichtdeterminismus	1	1	2	0
Nebenläufigkeit	0	0	0	3
Prozess	0	0	0	0
Prozessor	0	0	0	0
Part. Korrektheit	6	1	3	0
Terminierung	3	1	2	0
Konsistenz	0	0	0	0
Vollständigkeit	0	0	0	0
Fairness	0	0	0	0
Reduktion	0	0	0	0
Diagonalisierung	0	0	0	0
Ordnung	1	1	3	0
Unit-/ Log - cost	2	0	0	0
Worst/average case	1	0	0	0

Tabelle 2: Ideengruppe „Algorithmisierung“ in Einführungsvorlesungen der Informatik

4 Fundamentale Ideen im Informatikunterricht

Im Bereich der Schulcurricula ist eine Einschränkung erforderlich: Exemplarisch wird der Lehrplan der 6. und 7. Jahrgangsstufe des Pflichtfaches Natur und Technik an bayerischen Gymnasien, in dem zu einem Drittel der Unterrichtszeit Informatik gelehrt wird, herangezogen. Auch hier sind die Übungsaufgaben Grundlage der Ideenanalyse. Konkret werden die Aufgaben der offiziell genehmigten Unterrichtswerke „Informatik I“ [Fr04] und „Ikarus: Natur und Technik“ [Br04] herangezogen. Zur Erläuterung der prinzipiellen Vorgehensweise auch hier eine Beispielaufgabe (Ikarus, II/1 Aufgabe 4):

> **Attribute von Ländern:** Vergleiche in einer Tabelle die Länder Deutschland, Österreich, USA und China bezüglich der Attribute Kontinent, Größe, Einwohnerzahl, Binnenland und Hauptstadt. Schlage dazu in einem Lexikon oder Atlas nach. Wähle ein weiteres Land deiner Wahl und trage es mit allen Werten in die Tabelle ein**.**

Diese Übungsaufgabe ist hinsichtlich der Begriffe durchaus mit der in Abschnitt 3 erwähnten Modellierungsaufgabe vergleichbar. Auch hier müssen sinnvolle Objekte definiert und Attribute festgelegt werden. Top-Down-Zerlegung, Black-Box-Denken, Parametrisierung und Lokalität von Objekten tritt also auch hier auf, wenngleich diese Begriffe nicht explizit thematisiert werden. Andererseits wird der Klassenbegriff noch nicht tangiert, weshalb hier die Idee des „Abstrakten Datentyps“ noch keine Rolle spielt.

Das Resultat der Analyse der Übungsaufgaben der genannten Schullehrbücher zeigt Tabelle 3.

	Ikarus	Informatik I
Top-down	> 20	> 20
Bottom-up	0	0
Black-box	> 20	> 20
Lokalität von Objekten	> 20	> 20
Spezifikation	> 0	> 0
Abstrakter Datentyp	> 20	> 20
Teamarbeit	> 0	> 0
Schachtelung	> 10	> 10
Baum	> 10	> 10
Klammerung	0	0
Einrückung	0	0
Übersetzung	0	0
Interpretation	> 10	> 10
Operationale Erweiterung	0	0
Orthogonal./Emulation	0	0

Tabelle 3: Strukturierte Zerlegung in Jahrgangstufe 6/7 an bayerischen Gymnasien in Natur und Technik / Informatik

Offensichtlich gibt es auch hier Ideen, die in keiner Aufgabe thematisiert werden. Dazu gehören die Idee des Bottom-Up-Entwurfs, die Ideen der Klammerung, der Einrückung und der Übersetzung, sowie die Ideen der Operationalen Erweiterung und der Orthogonalisierung. Die Idee „Teamarbeit“ wurde mit „>0“ bewertet, da implizit bei vielen Aufgaben Gruppenarbeit erwartet wird. Ähnlich wurde Spezifikation behandelt, wenn Methoden informell beschrieben, jedoch nur eine Modellierung des Methodenkopfes verlangt ist.

Auf die Angabe der Häufigkeitstabelle für die Ideengruppe der Algorithmisierung wurde hier aus Platzgründen verzichtet. Es ergibt sich eine Tabelle, die im Vergleich zu Tabelle 2 stark „ausgedünnt“ ist: Es werden lediglich in der Ideengruppe der Programmierkonzepte die Ideen „Konkatenation“, „Alternative“ und „Iteration“ konkretisiert. Alle anderen Ideen werden nicht angesprochen.

5 Folgerungen

Welche Konsequenzen lassen sich aus den Überlegungen von Abschnitt 3 und 4 ziehen? Offensichtlich gibt es fundamentale Ideen, die von Beginn des Informatikunterrichts bis zur Hochschule wiederholt in unterschiedlichem Kontext diskutiert werden. Im Falle der Masteridee „Algorithmisierung“ sind es die Ideen Konkatenation, Alternative und Itera-

tion, bei der Masteridee „Strukturierte Zerlegung“ die Ideen Top-down/Bottom-Up-Methode, Geheimnisprinzip, Lokalität von Objekten, Spezifikation, Abstrakter Datentyp, Teamarbeit, Schachtelung, Klammerung und Baum (wobei letztere zu einer Idee zusammengefasst werden könnten). Diese Ideen werden auch in den weiteren Jahrgangstufen des Bayerischen Gymnasiums im Schulfach Informatik thematisiert. Für die Jahrgangsstufen 9 und 10 wird dies vom Autor dieser Arbeit in [Sc06] gezeigt. Diese Ideen bilden somit rote Fäden, die sich durch die gesamte Lehre in der Informatik ziehen, und genügen dem von Bruner [Br60] für Ideen geforderten vertikalen Kriterium. Allerdings gibt es auch Ideen, die in den ersten Jahren des Faches Informatik (Jahrgangstufe 6/7) an bayerischen Gymnasien nur implizit oder gar nicht tangiert werden. Dazu gehören etwa die gesamten Ideen zur Evaluation, Orthogonalisierung, die meisten Entwurfsparadigmen und Ablaufaspekte.

Problematisch gestaltete sich die Zuordnung des Themenbereichs „Objektmodellierung durch Klassen- und Objektdiagramme“. Grundsätzlich müsste er der „Strukturierte Zerlegung“ zuzuordnen sein. Bei der Vernetzung der Klassen ergeben sich jedoch im Allgemeinen nichthierarchische Strukturen, weshalb derartige Graphen nicht der Ideengruppe „Hierarchisierung“ zuzuordnen sind, während der Baum als spezieller Graph sich in dieser Gruppe befindet! Sollten vielleicht auch Ideen für allgemeine Graphen in die Gruppe „Hierarchisierung“ aufgenommen werden und die Benennung dieser Gruppe verallgemeinert werden?

Schließlich gibt es wichtige Konzepte, die nicht oder nur schwer im Baum der Ideen unterzubringen sind: Dazu gehört zunächst der Problembereich „Information und Repräsentation“: Bereits die erste Lehreinheit des bayerischen Curriculum für Informatik aber auch die oben diskutierte Einführungsvorlesung [Bru03] widmen sich dieser Problematik. Auch hierbei handelt es sich um einen roten Faden, der in einer begrifflichen Strukturierung des Faches repräsentiert sein müsste.

Literaturverzeichnis

[Ba98] Baumann, R.: Fundamentale Ideen der Informatik – gibt es das?; in „Informatische Bildung in Deutschland, Perspektiven für das 21 Jahrhundert“, Koerber, B. und Peters I.

[Br01] Broy, M.: Einführung in die Informatik I, WS 2001/2002 http://www4.in.tum.de/lehre/vorlesungen/info1ws01/tutoruebung.html (29.7.2006)

[Br04] Brichzin, P. et al.: Ikarus, Natur und Technik 6/7, Oldenbourg Schulbuchverlag 2004

[Br60] Bruner, J.S.: The process of education, Cambridge Mass. 1960 (dt. Übers.: "Der Prozeß der Erziehung", Berlin 1970)

[Bru03] Brügge, B.: Einführung in die Informatik I, WS 2003/2004 http://wwwbruegge.in.tum.de/twiki/bin/view/Lehrstuhl/Informatik1TutorWiSe2003 (29.7.2006)

[De99] Denning, P.: Computer Science: The Discipline, Encyclopedia of Computer Science, A. Ralston and D. Hemmendinger, eds., Nature Publishing Group, 2000, pp. 405-419.

[De03] Denning, P.: Great Principles of Computing, Comm. ACM, Nov. 2003, pp. 15-20.

[De06] Denning, P.: A Preliminary Listing of Great Principles, http://cs.gmu.edu/cne/pjd/GP/gp_list.html (29.7.2006)

[Fr04] Frey, E., Hubwieser, P., Winhardt, F.: Informatik I: Objekte, Strukturen, Algorithmen, Ernst Klett Verlag 2004

[Kn03] Knoll, A.: Einführung in die Informatik I, WS 2002/2003 http://atknoll1.informatik.tu-

muenchen.de:8080/tum6/lectures/courses/ws0203/infot1 (27.7.2006)
[Mo06] Modrow, E.: Zur Ordnungswirkung fundamentaler Ideen der Informatik am Beispiel der theoretischen Schulinformatik, Informatica Didactica 6 (2006)
[Ne98] Coulter, N.: ACM Computing Classification System, http://www.acm.org/class/1998/ (15.3.2006)
[Se04] Seidl, H.: Einführung in die Informatik I, WS 2004/2005; http://wwwseidl.informatik.tu-muenchen.de/ (29.7.2006)
[Sc93] Schwill, A.: Fundamentale Ideen der Informatik. Zeitschrift für Didaktik der Mathematik, 1993/1
[Sc02] Schneider, M.: Rekursive Strukturen in Einführungsvorlesungen der Informatik; (2002) Lecture Notes in Informatics, Volume 22, pp. 77
[Sc06] Schneider, M.: Functional modelling, fundamental ideas and threads in the subject informatics, ISSEP 2006, Vilnius (Litauen)
[We71] Weizsäcker von, C.F.: Die Einheit der Natur, 1971 Deutscher Taschenbuch Verlag