

Fachdidaktische Vorgehensweisen im Vergleich

Markus Schneider
Institut für Informatik
Technische Universität München
markus.schneider@in.tum.de



Übersicht

Der Vergleich von Vorlesungen zur Einführung in die Informatik:

- *Wie könnte er durchgeführt werden*
- *Wie könnte eine Instanz aussehen, an der eine Vorlesung gemessen wird*

Didaktische Prinzipien der inhaltlichen Vorlesungsgestaltung

Exemplarischer Vergleich zweier Vorlesungen:

- *Thematische Struktur*
- *Fachdidaktische Struktur*
- *Bewertung*

Empirische Analyse von Einführungsvorlesungen



Grundsätzliches zum Vergleich fachdidaktischer Vorgehensweisen

Ein Vergleich kann nur in sehr geringem Maße die Ergebnisse der Abschlußklausuren heranziehen, da

- *unterschiedliches Vorlesungskonzept*
- *unterschiedliche Prüfungsaufgaben*
- *unterschiedliche Probanden*

Ein Vergleich

- *erfordert eine absolute Instanz, d.h. einen allgemein fachlich-didaktische Strukturierung, an der die einzelnen Vorlesungen gemessen werden können*
- *beschränkt sich hier größtenteils die fachliche Vorgehensweise*



Fachlich-didaktische Strukturierungen in der Informatik ?

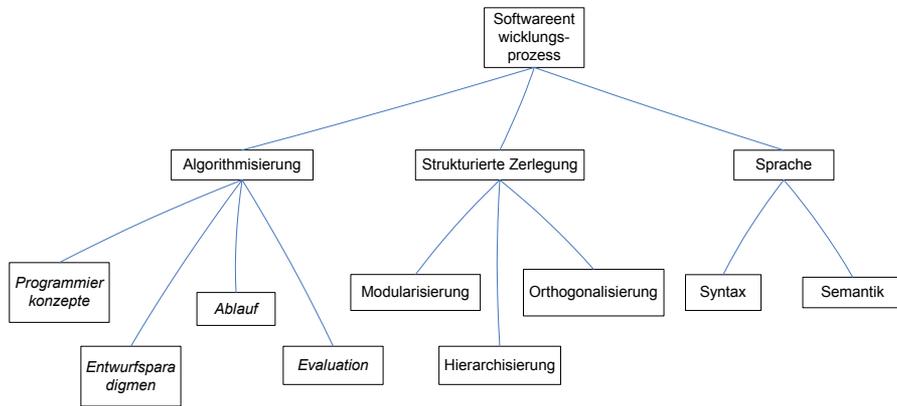
Dagstuhl 2004:

- *allgemein anerkannte, inhaltliche Lehrstandards existieren in der Informatik nur in Ansätzen,*
- *diese Ansätze sind weder für den Schul-, noch Hochschulbereich gefestigt*

Fundamentale Ideen (Bruner J.S. 1960):

- *Grundprinzip: Lehre soll sich in erster Linie an den Strukturen (= fundamentale Ideen) der zugrundeliegenden Wissenschaft orientieren*
- *Fundamentale Ideen wurden erfolgreich in der Mathematik- und Physikdidaktik eingesetzt*
- *Eine fundamentale Idee sollte*
 - *Eine Vielzahl von Anwendungsbereichen tangieren (Horizontalkriterium)*
 - *Auf unterschiedlichen Komplexitätsniveau thematisierbar sein (Vertikalkriterium)*
- *Beispiel aus der Informatik: Rekursion*

Fundamentale Ideen der Informatik (nach Schwill)



Vermittlung fundamentaler Ideen: Allgemeine didaktische Prinzipien

Spiralprinzip

- *Prinzip der Fortsetzbarkeit: Die Auswahl eines Themas sollte später einen Ausbau auf einem höheren Niveau ermöglichen.*
 - Beispiel: Rekursive Datenstrukturen zunächst anhand funktionaler Sprachen und später über Zeigerstrukturen mit imperativen Sprachen
- *Prinzip der Präfiguration (Intuition vor Exaktheit); der genauen Behandlung eines Themas sollte eine Phase intuitiven Lernens/Lehrens vorangehen.*
 - Beispiel: Anstatt eine Sprache anhand ihrer genauen Syntax einzuführen, empfiehlt es sich, die Sprache einfach zu verwenden
- *Prinzip des vorwegnehmenden Lernens: Ein Thema sollte nicht erst dann behandelt werden, wenn es mit wissenschaftlich notwendiger Genauigkeit thematisierbar ist*
 - Beispiel: Komplexitätsanalyse zunächst informell, später exakt



Vermittlung fundamentaler Ideen

Prinzip der Gleichgewichtung der Ideen: **Horizontales Vorlesungsdesign**

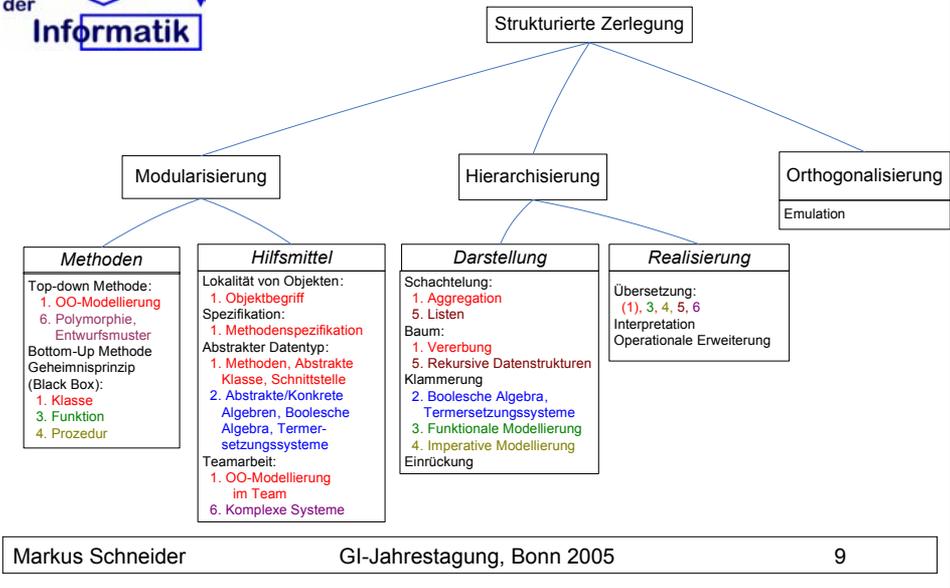
- Ein zentrales Ziel des ersten Studienjahres ist die Erlangung einschlägiger Erfahrungen im praktischen Erstellen (einfacher) Informatiksysteme.
- Hierzu ist es notwendig, dass die Studenten möglichst schnell mit den zentralen fundamentalen Ideen konfrontiert werden
- Die Auswahl der Lehrinhalte sollten deshalb Thematiken favorisieren, die es ermöglichen, den Baum der fundamentalen Ideen zügig in möglichst großer Breite zu traversieren
- Das (spiralartige) Vertiefen nur einiger weniger Ideen sollte diesem Ziel nachgeordnet sein.



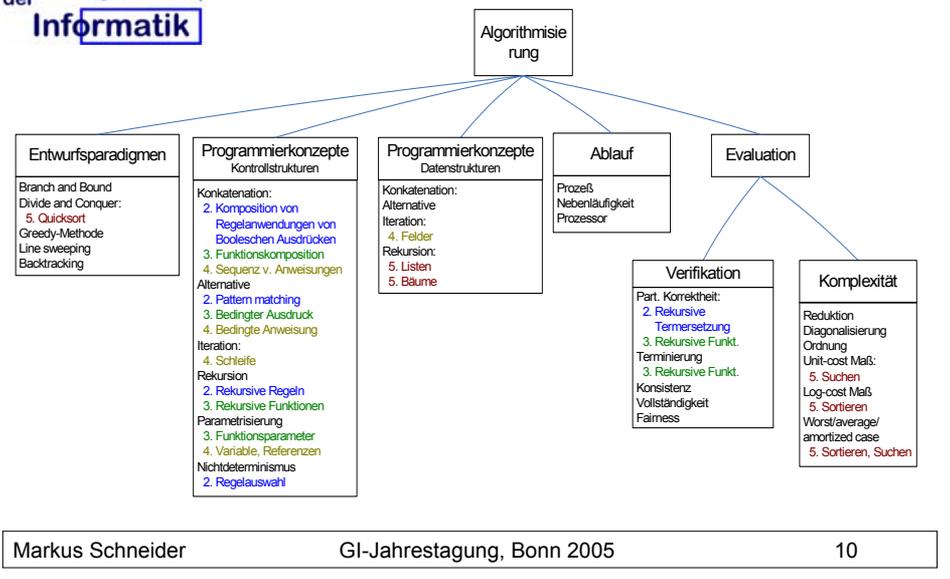
Vorlesung WS 2000/2001: Thematische Grobgliederung

Thema	Konzepte	Sprache
1. OO-Modellierung	Objekt, Klasse, Aggregation, Vererbung, Schnittstelle	UML, Java
2. Algebren	Abstrakte und konkrete Algebren, Algorithmusbegriff, Textersetzungssysteme, Boolesche Algebren, Termersetzungssysteme,	
3. Funktionales Programmieren	Funktionales Modellieren, Rekursion, bedingter Ausdruck, Korrektheit, Fixpunktsemantik	
4. Imperatives Programmieren	Zuweisung, Schleife, imperatives Programmieren eingebettet in OO-Techniken, Reihung	
5. Rekursive Datenstrukturen	Listen, Bäume, Algorithmen auf rekursiven Datenstrukturen	
6. OO-Programmierung	Vererbung, Abstrakte Klassen, Polymorphismus	

Vorlesung WS 2000/2001:



WS 2000/2001





Vorlesung WS 2000/2001: Didaktische Analyse

Von den fundamentalen Ideengruppen der Informatik werden im Detail thematisiert:

- *Modularisierung: Hilfsmittel, Methoden*
- *Hierarchisierung: Darstellung, Realisierung*
- *Algorithmisierung: Programmierkonzepte, Verifikation*

Vorläufig unberücksichtigt oder nur marginal tangiert bleiben

- *Orthogonalisierung*
- *Entwurfsparadigmen*
- *Aspekte des Ablaufs*
- *Komplexität (höchstens informell)*

Sprache:

- *Die Vorlesung beschränkt sich auf UML und Java; Probleme der Semantik bleiben Randthemen*



Vorlesung WS 2000/2001: Didaktische Analyse

Wertung:

- *Die starke Betonung der Programmierkonzepte, Modularisierung und Hierarchisierung ist im 1. Semester unerlässlich (Grundlagen)*
- *Die Thematisierung der Entwurfsparadigmen und Komplexitätsaspekte erfolgt auf einem gemäß Spiralprinzip angemessenen Niveau (Fortsetzbarkeit, Präfiguration)*
- *Orthogonalisierungs- und Ablaufthematiken werden nicht angesprochen (Eine angemessene Einführung in diese Ideengruppen dürfte mit den verfügbaren Mitteln schwer möglich sein)*



Vorlesung WS 2000/2001: Didaktische Analyse

Wertung:

- *Es treten genau die Probleme auf, die aus der Lernpsychologie (Konstruktivismus nach Mandl) bekannt sind*
 - Hohe Komplexität der Probleme und Begrifflichkeiten durch geringen Abstraktionsgrad
 - Praktischer Nutzen der UML-Modellierung ist zunächst nur schwer zu erkennen
 - Langer Weg bis zur Realisierung echter Systeme
- *Der systemnahe Ansatz erfordert die detaillierte Diskussion von Einzelheiten der Modellierung:*
 - Algorithmisierungsaspekte können erst spät angesprochen werden
 - Die Idee der strukturierten Zerlegung wird setzt auf einem sehr hohen kognitiven Niveau ein (Fortsetzbarkeit, Präfiguration ?)
 - Gleichgewicht der Ideen ist nicht gegeben:



Vorlesung WS 2000/2001: Didaktische Analyse

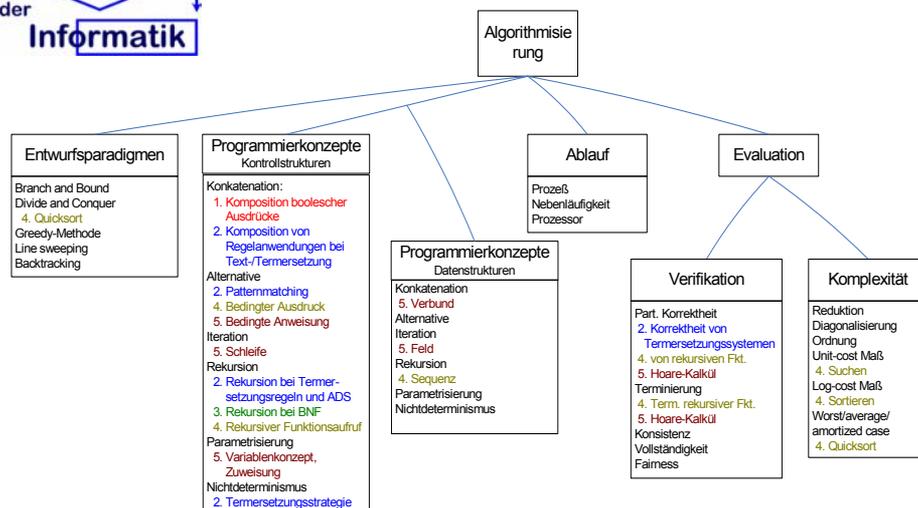
Weiteres:

- *Verwendung einer einzigen Programmiersprache (insbesondere Java):*
 - Idee der Orthogonalisierung kann nur schwer vermittelt werden
 - Das Wesentliche der einzelnen Konzepte wird verwischt (Beispiel: Rekursion, Iteration) oder ist nicht lehrbar (Pattern-matching)
- *Umfang:*
 - Für einen Studenten, der sich erstmals mit den Themen der Vorlesung auseinandersetzt, dürfte der Umfang zu hoch sein (Reine Folienvorlesung).

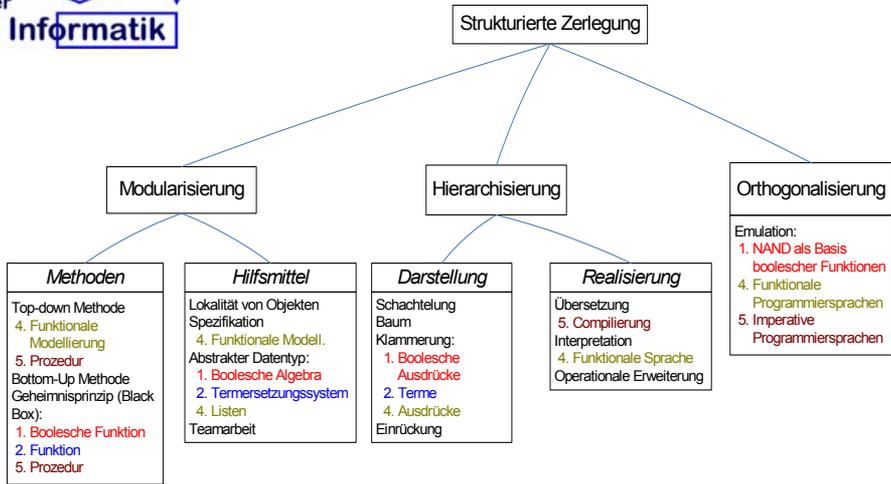
Vorlesung WS 2001/2002: Thematische Grobgliederung

Thema	Konzepte	Sprache
1. Information und Representation	Interpretation von Information, Boolesche Algebra, Interpretation von Termen	Gofor
2. Algorithmen und Algebren	Algorithmusbegriff, Textersetzungssysteme, Rechenstrukturen, Algebraische Spezifikation, Termersetzungssysteme, Zentrale algebraische Strukturen	
3. Programmiersprachen	BNF, Syntax, Semantik	
4. Funktionales Programmieren	Funktionales Modellieren, Alternative, Rekursion, einfache rekursive Algorithmen über Zahlen und Listen, Semantik, Korrektheit	
5. Imperatives Programmieren	Zuweisung, Schleife, Prozedur, Hoare-Kalkül, Reihenungen, Referenzen	Pascal

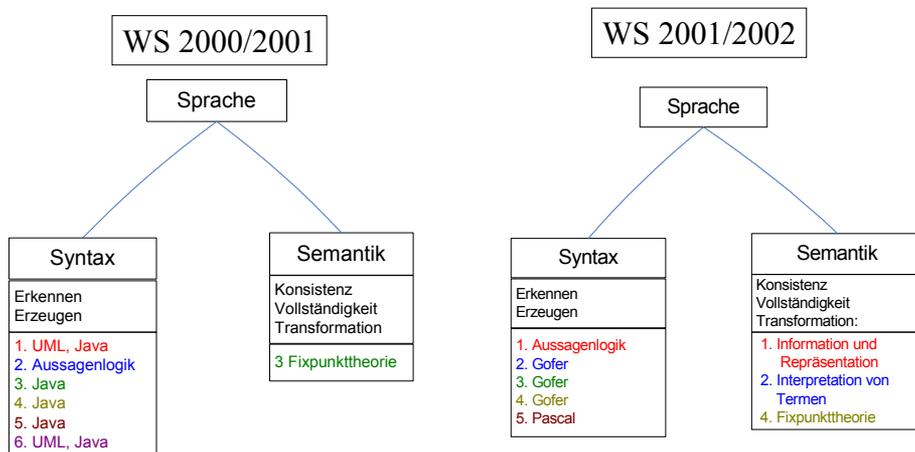
Vorlesung: WS 2001/2002



Vorlesung: WS 2001/2002



Vergleich: Sprache





Vorlesung WS 2001/2002: Didaktische Analyse

Grundsätzliches (im Vergleich zu WS 2000/2001):

- *Die Schwerpunkte der Vorlesung liegen wiederum im Bereich der*
 - Programmierkonzepte
 - Modularisierung
 - Hierarchisierung
- *Auf didaktisch angemessenem Niveau werden auch folgende Themenbereiche behandelt*
 - Verifikation (tiefer als im WS 2000/2001) und Komplexität
 - Entwurfsparadigmen
- *Idee der Orthogonalisierung wird jedoch auf propädeutischen Niveau bereits angesprochen*
- *Insbesondere die Ideen der strukturierten Zerlegung werden in größerer Breite thematisiert*



Vorlesung WS 2001/2002: Didaktische Analyse

Grundsätzliches (im Vergleich zu WS 2000/2001):

- *OO-Thematik wird auf spätere Vorlesungen verlegt (2. Semester)*
- *Das Vorlesungsdesign ist vor allem im Bereich der strukturierten Zerlegung hinsichtlich der Ideenabdeckung horizontaler angelegt*
- *Gleichzeitig werden jedoch zentrale Konzepte anhand verschiedener Themenbereiche im Sinne des Spiralprinzips vertieft*
- *Bessere vertikale und horizontale "Traversierung des Ideenbaumes"*
- *Der unter praktischen Gesichtspunkten wichtige Ideenkomplex der Programmierkonzepte wird gleich von Beginn an tangiert.*
- *Unterschiedliche Konzeptgruppen werden anhand unterschiedlicher Sprachen thematisiert*
- *Die Behandlung konkreter Programmierkonzepte erfolgt jedoch auch hier relativ spät*



Kumulative Analyse der Klausurergebnisse (TU-München: 2000 – 2004)

Thema	Schwierigkeitsgrad		
	Wiederholung	Geringer Transfer	Hoher Transfer
Strukturelle Zerlegung	75%	44%	-
Text/ Termersetzungssysteme	57%	41%	-
Funktionales Programmieren	57%	36%	28%
Imperatives Programmieren	46%	-	26 %
Objekt-orientierte Programmierung	56%	-	-
Machinennahes Programmieren	56%	38%	-
Verifikation/Korrektheit	-	38%	-



Beobachtungen

Aufgaben zur strukturellen Zerlegung wurden vergleichsweise gut gelöst

Aufgaben zur Algorithmisierung, insbesondere Programmierkonzepte, wurden deutlich schlechter gelöst, wobei folgende absteigende Reihe zu beobachten ist:

- *Text-/Termersetzungssysteme*
- *Funktionale Programmierung*
- *Imperative Programmierung*
- *Objektorientierte Programmierung*

Bei echten Transferfragen hat der Großteil der Studierenden nur geringe Erfolgchancen



Folgerung

Algorithmisierung muss möglichst bald im Verlaufe des Semesters thematisiert werden

Strukturierungsaspekte sollten zumindest zu Beginn des Semesters geringeres Gewicht erhalten

Die Algorithmisierung sollte mit einer syntaktisch einfachen, jedoch mächtigen Modellierungstechnik durchgeführt werden

- *Termersetzung*
- *Funktionale Sprachen*

Im weiteren könnten Modellierungstechniken verwendet werden, die technisch aufwendiger sind: z.B. Java, etc.