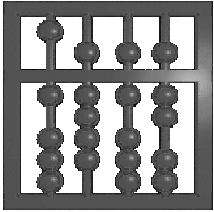


***Einführung in die Informatik II  
Vergleich von Programmierstilen  
und abschließende Bemerkungen***

Prof. Bernd Brügge, Ph.D  
Institut für Informatik  
Technische Universität München

Sommersemester 2001

25. Juli 2001



# Das Institut für Informatik sucht ab WS01/02 **Studentische Hilfskräfte als Tutorinnen/Tutoren**



u.a. für die Übungen zu den Vorlesungen

**„Einführung in die Informatik I, III, Diskrete Strukturen 1, Konkrete Mathematik“**

Motivierten Studenten mit dann abgeschlossenem Vordiplom bietet sich die Möglichkeit, Grundlagenwissen aus der Sicht der Wissensvermittlung und im Kontakt mit Mitarbeitern des Instituts zu vertiefen.

## Gegenleistung:

- **Bezahlung: DM 5.000,-**
- **Vergabe eines „Seminarscheins für überfachliche Grundlagen“ (benötigt für DHP)**
- **Ausstellung eines Zertifikats über die Tätigkeit**

## Aufwand, am Beispiel Info1:

- **3 \_ Stunden Tutorübung pro Woche in der Vorlesungszeit**
- **1 Stunde Teilnahme an der wöchentlichen Tutorenkonferenz**
- **Korrektur der Haus- und Programmieraufgaben der Studenten der betreuten Gruppe**
- **Vorbereitungszeit**

Für die Vorlesungen Info1, Info3 und DS1 gilt ein gleicher Aufwand.

Bei anderen Vorlesungen mit gfs. geringerem Arbeitsaufwand erfolgt ein angemessener Abschlag bei der Bezahlung.

**Bitte wenden Sie sich bei Interesse per e-mail an**

**Dr. Werner Meixner, [meixner@in.tum.de](mailto:meixner@in.tum.de)**

# *Programmierstile*

- ❖ Funktionale Programmierung
- ❖ Imperative Programmierung
- ❖ Objekt-basierte Programmierung
- ❖ Objekt-orientierte Programmierung
- ❖ Ereignis-basierte Programmierung
- ❖ Regel-basierte Programmierung (nicht in Info II besprochen)
- ❖ ...

# *Programmierstil, System-Kategorien und Sprachniveaus*

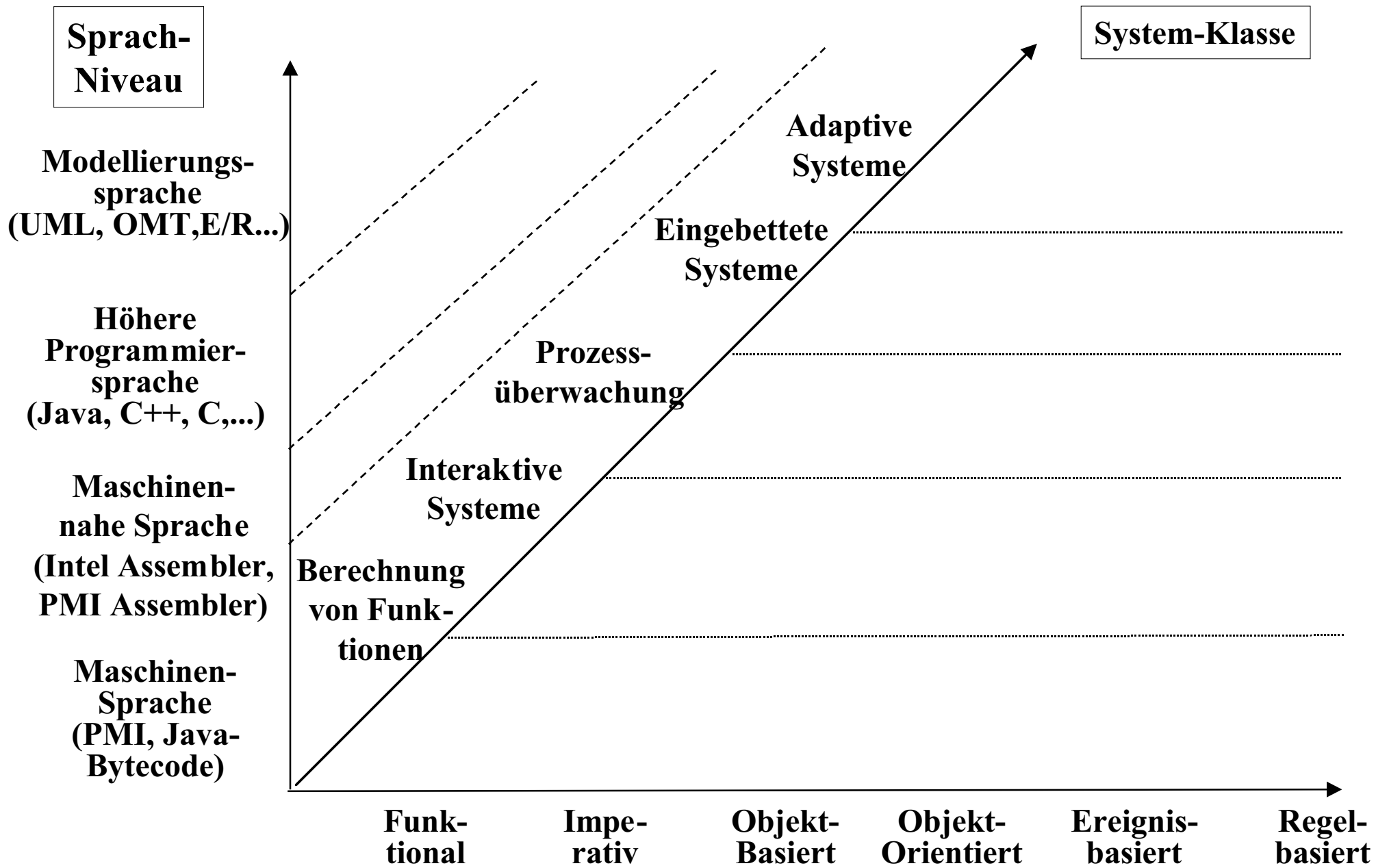
- ❖ Der Programmierstil ist nur eine von vielen Dimensionen, um Aspekte eines Informatik-Systems zu beurteilen. Andere Aspekte:
  - ◆ System-Kategorie
  - ◆ Sprachniveau

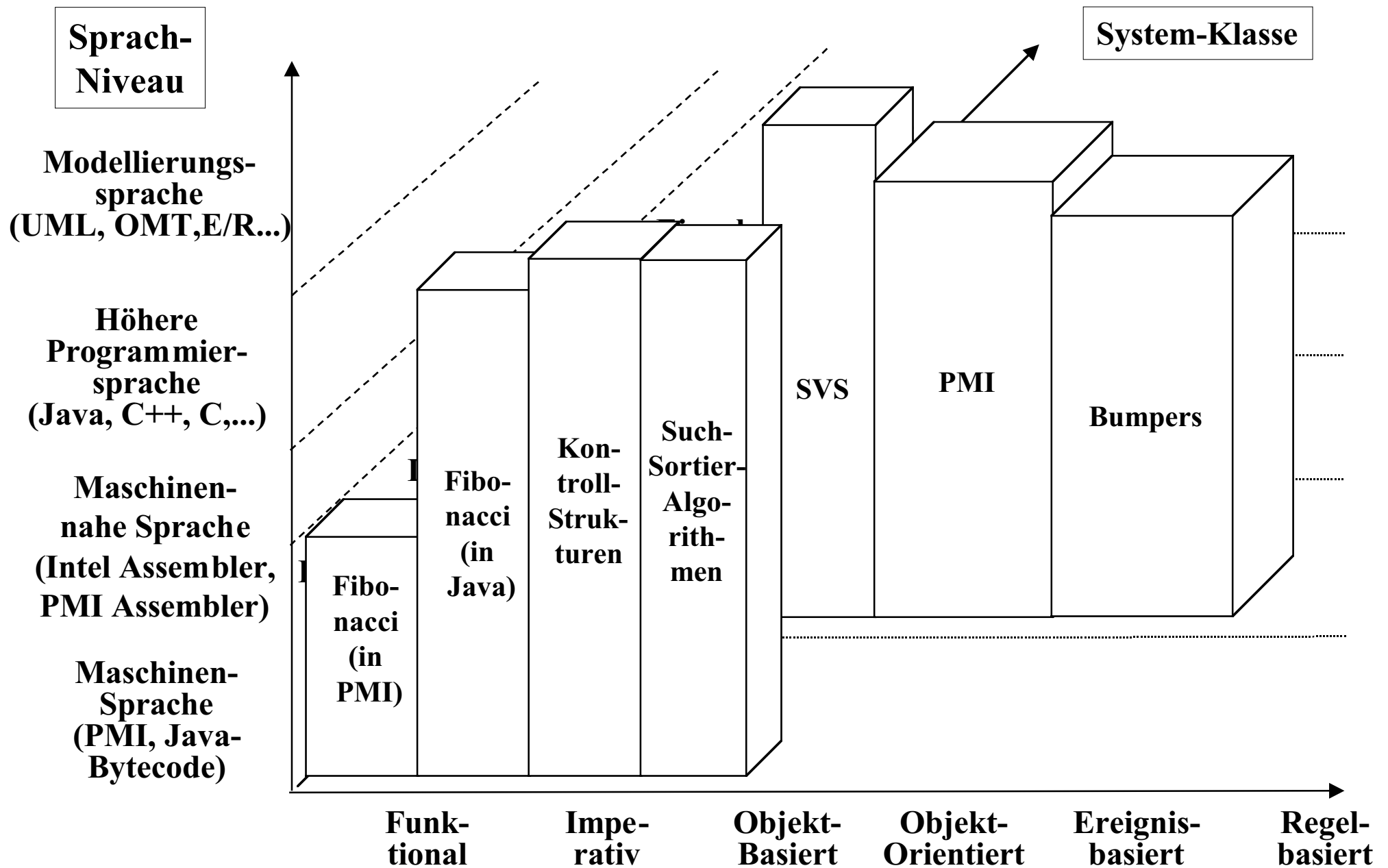
## ❖ System-Kategorie

1. Berechnung von Funktionen
2. Interaktive Systeme
3. Prozeßüberwachung
4. Eingebettete Systeme
5. Adaptive Systeme

## ❖ Sprachniveau

- ◆ Modellierungssprache
- ◆ Höhere Programmiersprache
- ◆ Maschinennahe Programmiersprache
- ◆ Maschinensprache





# *Dimensionen von Programmierstilen*

- ❖ **Elemente:** Was sind die wesentlichen Bestandteile des Stils?
- ❖ **Sicherheit:** Sind Seiteneffekte möglich?
- ❖ **Effizienz:** Wie hoch ist die Laufzeiteffizienz?
- ❖ **Nachweis der Korrektheit:** Wie leicht ist es zu beweisen, dass das Programm das Spezifikationsmodell korrekt implementiert?
- ❖ **Nachweis der Terminierung:** Wie leicht ist es, zu beweisen, dass das Programm anhält?
- ❖ **Enkapsulierbarkeit:** Erlaubt der Stil Zugriffsschutz (z.B. durch Sichtbarkeitsregeln)
- ❖ **Erweiterbarkeit:** Unterstützt der Programmierstil die Einführung von neuen Typen?
- ❖ **Lesbarkeit:** Wieweit kann man das Programm durch Lesen des Quelltexts verstehen?
- ❖ **Wiederverwendbarkeit:** Wie leicht kann man das Programm bei der Lösung anderer Probleme einsetzen?

# *Funktionaler Programmierstil*

- ❖ **Wesentliche Elemente:** Funktionsanwendung, Fallunterscheidung, Rekursion
- ❖ **Sicherheit:** hoch, keine Seiteneffekte
- ❖ **Effizienz:** für jeden Aufruf wird eine Inkarnation (Aktivierungssegment) angelegt (gute Effizienz erfordert i.a. aufwendige Optimierung)
- ❖ **Nachweis Korrektheit :** partielle Korrektheit durch strukturelle Induktion
- ❖ **Nachweis Terminierung:** Terminierungsfunktion
- ❖ **Enkapsulierung:** Gering (unproblematisch, da keine global zugänglichen Daten zu schützen sind)
- ❖ **Erweiterbarkeit:** Schwierig
- ❖ **Lesbarkeit** OK
- ❖ **Wiederverwendbarkeit:** Gering (etwas besser bei Verwendung von Funktionsbibliotheken)

# *Imperativer Programmierstil*

- ❖ **Wesentliche Elemente:** Zuweisung, Kontrollstrukturen, Anweisungssequenzen, Programmzustand
- ❖ **Sicherheit:** nicht sehr hoch, weil Seiteneffekte auf Variablen/Zustand möglich
- ❖ **Effizienz:** in Schleifen wird auf eine Variable mehrfach zugewiesen, Zwischenergebnisse können gespeichert werden (i.A. gute Effizienz)
- ❖ **Nachweis Korrektheit:** Durch Zusicherungen (Hoare-Kalkül)  
Sehr aufwendig schon für kleine Programme
- ❖ **Nachweis Terminierung:** Schwierig
- ❖ **Enkapsulierbarkeit:** Gering
- ❖ **Erweiterbarkeit:** Schwierig
- ❖ **Lesbarkeit:** OK
- ❖ **Wiederverwendbarkeit:** Gering

# *Objekt-basierter Programmierstil*

- ❖ **Wesentliche Elemente:** Imperativer Programmierstil + Klassenkonzept
- ❖ **Sicherheit:** Seiteneffekte durch Klassenkonzept auf lokale Variablen/Zustand einschränkbar
- ❖ **Effizienz:** in Schleifen wird auf eine Variable mehrfach zugewiesen, Zwischenergebnisse können gespeichert werden (i.A. gute Effizienz)
- ❖ **Nachweis Korrektheit:** Entwurf durch Vertrag, aufwendig
- ❖ **Nachweis Terminierung:** Schwierig
- ❖ **Enkapsulierbarkeit:** Hoch
- ❖ **Erweiterbarkeit:** Schwierig
- ❖ **Lesbarkeit** OK
- ❖ **Wiederverwendbarkeit:** Nicht sehr hoch

# *Objekt-Orientierter Programmierstil*

- ❖ **Wesentliche Elemente**                      Objekt-basierter Stil + Vererbung, Polymorphismus
- ❖ **Sicherheit:**                                      Seiteneffekte durch Klassenkonzept auf lokale Variablen/Zustand einschränkbar
- ❖ **Effizienz:**                                        mittelmäßig (i.A. geringere Effizienz als beim imperativen Stil wegen dynamischem Polymorphismus)
- ❖ **Nachweis Korrektheit:**                      Entwurf durch Vertrag, aufwendig
- ❖ **Nachweis Terminierung:**                    Schwierig
- ❖ **Enkapsulierbarkeit:**                        Hoch
- ❖ **Erweiterbarkeit:**                              Sehr gut
- ❖ **Lesbarkeit**                                        OK
- ❖ **Wiederverwendbarkeit:**                    Sehr gut

# *Ereignis-basierter Programmierstil*

- ❖ **Wesentliche Elemente**      Kontrollfluss durch Ereignisse,  
keine Steuerung durch ein Hauptprogramm
- ❖ **Sicherheit:**                      gefährdet, weil Seiteneffekte auf  
Variablen/Zustand möglich
- ❖ **Effizienz:**                          hoch (wie beim imperativen Stil)
- ❖ **Nachweis Korrektheit:**          sehr schwierig wegen fehlendem Kontrollfluss
- ❖ **Nachweis Terminierung:**      Äußerst schwierig (aber im allgemeinen  
Terminierung gar nicht erwünscht :-)
- ❖ **Enkapsulierbarkeit:**              Gering
- ❖ **Erweiterbarkeit:**                  Sehr gut
- ❖ **Lesbarkeit**                          Schlecht
- ❖ **Wiederverwendbarkeit:**          Mittelmäßig

# Übersichtstabelle

	Funktional	Imperativ	Objekt- basiert	Objekt- orientiert	Ereignis- basiert
Sicherheit	+	-	0	0	-
Effizienz	-	+	0	0	+
Nachweis der Korrektheit	+	-	0	0	-
Nachweis der Terminierung	+	-	-	-	--
Enkapsulierung	-	-	+	+	-
Erweiterbarkeit	-	-	0	+	+
Lesbarkeit	0	0	0	0	-
Wiederverwend- barkeit	-	-	0	+	0

Legende: + (hoch/einfach), 0 (mittel), - (gering/schwierig)

# *Evaluierung*

## ❖ **Audio- und Videoprobleme:**

- ◆ Gelöst, aber...

## ❖ **Skriptverfügbarkeit:**

- ◆ Lernen Sie eine Mitschreibetechnik

## ❖ **Schwierigkeit:** Zu schwierig, zu leicht

- ⇒ Koordination mit TGI, HM

## ❖ **Standort während der Vorlesung:**

- ◆ Hinter dem Laptop, linker HS

## ❖ **Folien:** Zu viele, gerade richtig

- ◆ Extrahieren ist eine wichtige wissenschaftliche Tätigkeit und *Fähigkeit*

## ❖ **Publikum:** Stoppt das Schwatzen

- ◆ Photos, Handies, Nebengespräche reduzieren

## *Ein letzter Tip: Prüfungsvorbereitung*

- ❖ Ein Skript ist das Rohmaterial, aber noch kein Kondensat für die Prüfungsvorbereitung!
- ❖ Destillieren Sie die wesentlichen Punkte aus dem Skript
  - ◆ Erarbeiten Sie eine Zusammenstellung der Konzepte
- ❖ Arbeiten Sie nochmals möglichst viele Übungsaufgaben durch
- ❖ Spielen Sie (mental) eine Prüfungssituation durch:
  - ◆ Was kann passieren?
- ❖ Arbeiten Sie mit anderen zusammen!

**Viel Glück im weiteren Studium!**