# Neue Methoden in der Vermittlung von Betriebssystemkenntnissen

**Hans-Georg Eßer**

Dagstuhl-Seminar
29.06.–01.07.2009

# Hans-Georg Eßer

## Kurz-CV in 60 Worten

- Studium in Aachen (Mathematik und Informatik)
  - Mathe-Diplom 1997, DA in theor. Informatik
  - Info-Diplom 2005, DA in prakt. Informatik (Betreuer: Felix Freiling)
- 1999 ein Jahr Doktorand am ZAM, Forschungszentrum Jülich
- seit 2000 Redakteur einer Linux-Zeitschrift (München)
- WS 2006/07, SS 2008 und SS 2009: Lehrauftrag „Betriebssysteme“ an der Hochschule München (FH)
- seit 04/2008 externer Doktorand am PI1

# „Neue Methoden in der Vermittlung von Betriebssystem-Kenntnissen“

**Aktuelle Untersuchung:**

- 4C/ID (Four Components Instructional Design, van Merriënboer)
  - Instruktionsdesign-Modell für komplexe Themen
- angewandt auf die Themen Dateisysteme und Speicherverwaltung
  - beide Themen kombiniert in einem gemeinsamen „Kapitel“
  - Test im laufenden Sommersemester an der Hochschule München (Vorlesung Betriebssysteme I; endet nächste Woche)
  - geplante Auswertungen: Fragebogen, Klausurergebnisse

# Das 4C/ID-Modell (1/3)

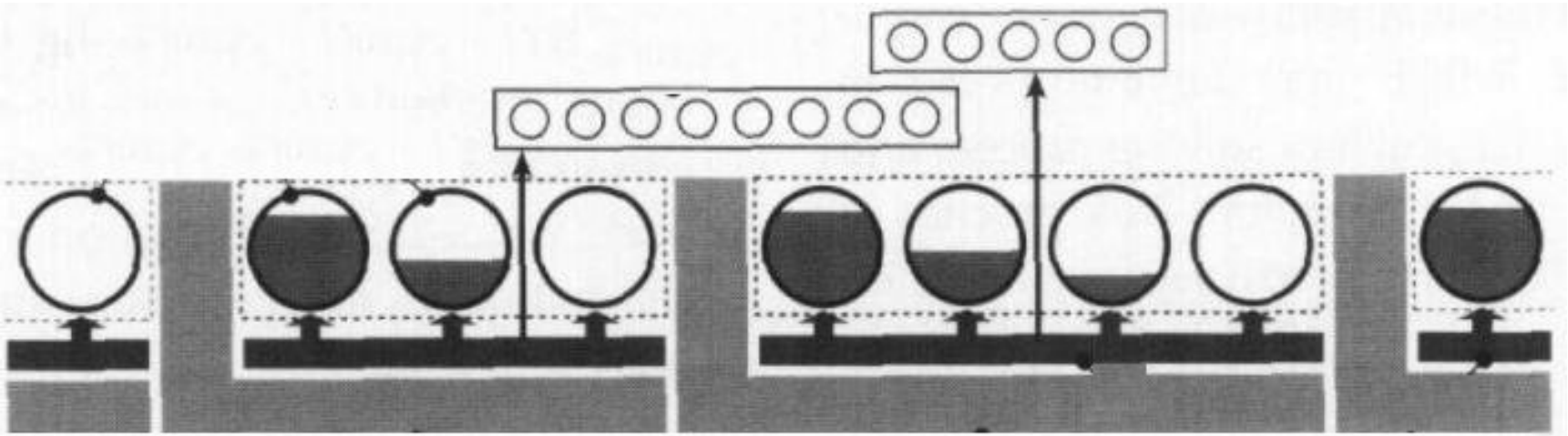
**Ansätze:**

- **“whole task” learning** – keine Zerlegung in unvollständige “sub tasks”
- **“constituent skills”** – Teilfertigkeiten, die für ein “complex skill” nötig sind, aber nicht isoliert geübt werden sollten
- **“shrinking support”** – Schrittweise die Unterstützung zurücknehmen
- **“variability principle”** – *ABC ABC ABC* statt *AAA BBB CCC*

# Das 4C/ID-Modell (2/3)

**Vier Komponenten:**

- Learning Tasks (organisiert in Task Classes)
- Supportive Information
- Just-in-time Information
- Part-Task Practice



# Das 4C/ID-Modell (3/3)

**recurrent vs. non-recurrent skills:**

- **recurrent:** Muster, die immer wieder auftauchen
- **non-recurrent:** Anspruchsvolle Aufgaben, die von Situation zu Situation verschieden sind

# Umsetzung von 4C/ID (1/2)

**Konzentration auf "variability principle"**

- Drei Task Classes:
  - Zusammenhängende Speicherzuordnung
  - Nicht-zshgd. Speicherzuordnung, einfache Modelle
  - Nicht-zshgd. Speicherzuordnung, fortgeschrittenes Modell (Paging)
- Supportive Information: Die Vorlesung selbst (Theorie, Beispiele); das Skript
- Just-in-time Information: Einführung in Python, ausführliche und kommentierte Beispielfunktionen in Rumpfprogrammen
- Part-Task Practice: —

# Umsetzung von 4C/ID (2/2)

## Beispiele für Tasks:

**Task Class I: Contiguous Allocation**
This task class teaches the simplest model that can be used for organizing memory or disk space: contiguous allocation, i. e. allocation of areas without holes.

Task I.1: *Worked-out example:* Partitioning of memory into a number of memory partitions with fixed, equal size.

Task I.2: *Completion task:* The same for disk space; given: the idea that disk space is to files what memory is to processes.

Task I.3: *Conventional task:* What are disadvantages of this concept (fixed number of files/processes, fixed maximum filesize/ process memory usage) and how can this approach be improved? (allow for different sizes of partitions and as a second step allow for dynamic partitioning)

# Gliederung Speicher/Dateisysteme (1/2)

| | |
|---|---|
| Grundlagen | Konkurrenz um Ressourcen (Speicher, Platte)<br>Dateien: erzeugen, öffnen, seek, append, read/write<br>Prozesse: Speicher reservieren, vergrößeren, freigeben<br>Verwaltung freier Speicher/Plattenplatz |
| Einführende Beispiele | Dateisysteme von CP/M, MS-DOS<br>Erweiterter Speicher unter MS-DOS |
| Gliederung | Zusammenhängende / nicht zshgd. Speicherzuordnung<br>Fragmentierung |
| Zusammenhgd. Sp. | Partitionen fester, gleicher Größe<br>Partitionen fester, verschiedener Größe<br>dynamische Partitionierung<br>Verwaltung freier Speicher/Plattenplatz: Bit Map, Linked List<br>Speicherzuteilung: First-Fit, Best-Fit, . . . , Buddy System<br>Besonderheiten Speicher: Code-Verschiebung, Speicherschutz |

# Gliederung Speicher/Dateisysteme (2/2)

| | |
|---|---|
| Nicht-zshgd. Speich. | Aufteilung in Blöcke / Seiten(-rahmen)<br>Beispiele: CP/M-Dateisystem, fiktives Speichersystem<br>leicht verbessert: Index-Allokation, teilweise zshgd.<br>leicht verbessert: Segmentierung, teilweise zshgd.<br>Dateisystem: Mehrstufige Indirektion<br>Exkurs: Linux-Dateisystem: VFS, Ext2 |
| Paging | Seiten/Rahmen, virtueller Adressraum<br>Adressübersetzung<br>Lokalitätsprinzip<br>Beschleunigung: TLB, invertierte Seitentabellen, mehrstufiges P.<br>Speicherschutz<br>Page-Fault-Behandlung, Seitenersetzung, Strategien<br>Swapping |

# Was sich für parallele Darstellung eignet

- Was Speicher für Prozesse ist, ist Plattenplatz für Dateien (Plattenplatz als Ressource der Datei betrachten)
- Lokalitätsprinzip – in beiden Bereichen relevant, wenn auch aus untersch. Gründen (Disk: Seek-Times, Memory: z.B. TLB)
- Disk-Blöcke über mehrstufige Indirektion;
  Paging mit aufgeteilten Page tables
- Probleme bei zusammenhängender Speicherzuteilung
- Index-Allokation $\Leftrightarrow$ Segmentierung
- interne Fragmentierung
- externe Fragmentierung
- Verwaltung freier Plattenplatz/freier Speicher

# Übungsaufgaben im Praktikum

- Dateisystem ähnlich CP/M in Python programmieren (gegeben: Code, der Disk-Image erzeugt und Blöcke lesen/schreiben kann); Verwaltung der FAT und Free Block List
- Buddy-System, Beispielrechnung mit verschiedenen Anforderungen und Freigaben
- Buddy-System in Python programmieren (gegeben: Rumpf-Speicherverwaltung; zu programmieren sind i. W. `anfordern()` und `freigeben()`.)
- Memory-Mapped-Files: Verwendung von UNIX-`mmap()` in Python
- Indirektion in Dateisystemen, Rechenaufgabe
- Dateisysteme: UNIX-Attribute und Timestamps (`chmod`, `ls`)
- Dateisysteme: Erweiterte Attribute in Ext3 (`attr`, `get/setfattr`)
- Paging, Anzahl und Größe von Seitentabellen

# Auswertungen

**Was haben die Änderungen gebracht?**

- Fragebogen am Semesterende
- Vergleich der Klausurergebnisse

Abhängig von der Auswertung

$\Rightarrow$ vielleicht ein erstes Ergebnis;

$\Rightarrow$ Thema weiter verfolgen?

# Fragebogen

1. Die abwechselnde Behandlung der beiden Themen finde ich sinnvoll.
2. Der häufige Wechsel zwischen Eigenschaften eines Dateisystems und Eigenschaften der Speicherverwaltung war für mich verwirrend.
3. Durch die Kombination der Themen konnte ich gut erkennen, dass sich viele Konzepte aus dem einen Gebiet auf das andere übertragen lassen
4. Da es einige Konzepte (z. B. Paging in der Speicherverwaltung und Indirektion bei den Dateisystemen) gibt, die kein direktes Gegenstück im jeweils anderen Thema haben, ist die parallele Behandlung unnütz.
5. Die Implementierungsaufgaben im Praktikum haben mir geholfen, die Konzepte aus der Vorlesung zu vertiefen.
6. Die Implementierungsaufgaben waren ...
   viel zu leicht / zu leicht / genau richtig / zu schwer / viel zu schwer
7. Das begleitende Skript zu Dateisystemen/Speicherverwaltung ...
   war hilfreich / war nicht hilfreich / habe ich nicht verwendet

# Vergleich der Klausurergebnisse

Vergleiche . . .

- mit den Ergebnissen vom letzten Semester
- mit den Ergebnissen der aktuellen Klausur aus der Parallelveranstaltung
- Berechnung: Punkte für Dateisysteme/Speicherverwaltung in Relation zu den Gesamtpunkten

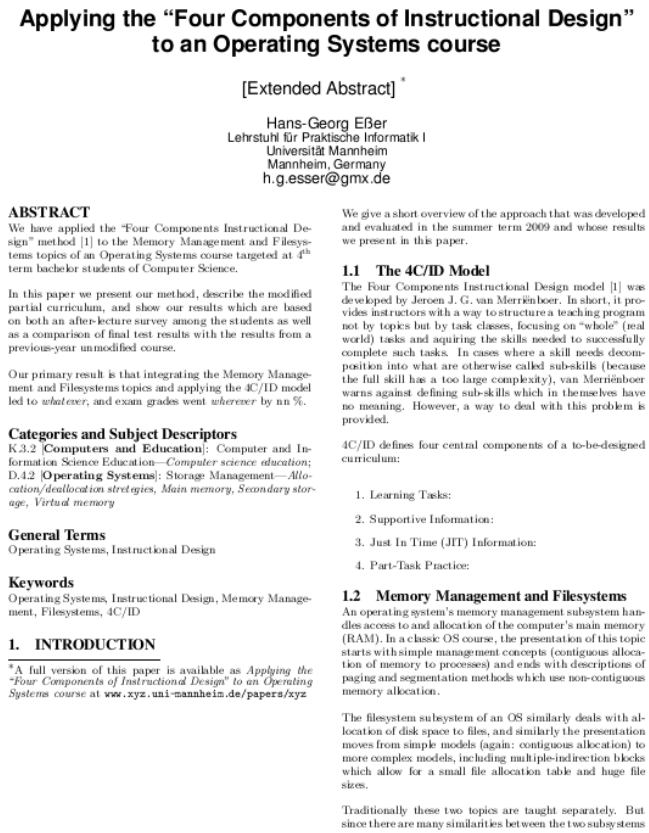
# Was ist im letzten Jahr passiert?

- Zertifikat Hochschullehre Bayern (06/2008–11/2008, 140 Stunden Didaktikkurse)
- Vorlesung „Betriebssysteme“: Speicherverw. & Dateisysteme nach 4C/ID überarbeitet, begleitendes Skript, begleitende Praktikumsaufgaben (03/2009–07/2009)
- Lektüre:
  - van Merriënboer, Clark & de Croock. Blueprints for complex learning: The 4C/ID-model (2002)
  - van Merriënboer & Kester. The Four-Component Instructional Design Model: Multimedia Principles in Environments for Complex Learning (2005)
  - van Merriënboer & Kirshner. Ten Steps to Complex Learning (2007)
  - und diverse weitere Publikationen rund um 4C/ID sowie Didaktik-Skripte der Fernuni Hagen

# Wie geht es weiter?

## Nächste Schritte

- Auswertung meiner aktuellen Vorlesung
  ⇒ evtl. Paper für SIGCSE 2010 (ACM Technical Symposium on Computer Science Education), in Vorbereitung
  `http://www.sigcse.org/sigcse2010/`
- falls aktuelle Auswertungen erfolgversprechend
  ⇒ Ausweitung der 4C/ID-Umsetzung auf weitere BS-Themen, für nächste Vorlesung (voraussichtlich im SS 2010)
- Mehr Didaktikgrundlagen (Fernuni Hagen)



**Applying the "Four Components of Instructional Design" to an Operating Systems course**

[Extended Abstract] *

Hans-Georg Eßer
Lehrstuhl für Praktische Informatik I
Universität Mannheim
Mannheim, Germany
h.g.esser@gmx.de

**ABSTRACT**

We have applied the "Four Components Instructional Design" method [1] to the Memory Management and Filesystems topics of an Operating Systems course targeted at 4th term bachelor students of Computer Science.

In this paper we present our method, describe the modified partial curriculum, and show our results which are based on both an after-lecture survey among the students as well as a comparison of final test results with the results from a previous-year unmodified course.

Our primary result is that integrating the Memory Management and Filesystems topics and applying the 4C/ID model led to *whatever*, and exam grades went *wherever* by nn %.

**Categories and Subject Descriptors**
K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Computer science education*; D.4.2 [**Operating Systems**]: Storage Management—*Allocation/deallocation strategies, Main memory, Secondary storage, Virtual memory*

**General Terms**
Operating Systems, Instructional Design

**Keywords**
Operating Systems, Instructional Design, Memory Management, Filesystems, 4C/ID

**1. INTRODUCTION**

*A full version of this paper is available as *Applying the "Four Components of Instructional Design" to an Operating Systems course* at `www.xyz.uni-mannheim.de/papers/xyz`

We give a short overview of the approach that was developed and evaluated in the summer term 2009 and whose results we present in this paper.

**1.1 The 4C/ID Model**

The Four Components Instructional Design model [1] was developed by Jeroen J. G. van Merriënboer. In short, it provides instructors with a way to structure a teaching program not by topics but by task classes, focusing on "whole" (real world) tasks and aquiring the skills needed to successfully complete such tasks. In cases where a skill needs decomposition into what are otherwise called sub-skills (because the full skill has a too large complexity), van Merriënboer warns against defining sub-skills which in themselves have no meaning. However, a way to deal with this problem is provided.

4C/ID defines four central components of a to-be-designed curriculum:

1. Learning Tasks:
2. Supportive Information:
3. Just In Time (JIT) Information:
4. Part-Task Practice:

**1.2 Memory Management and Filesystems**

An operating system's memory management subsystem handles access to and allocation of the computer's main memory (RAM). In a classic OS course, the presentation of this topic starts with simple management concepts (contiguous allocation of memory to processes) and ends with descriptions of paging and segmentation methods which use non-contiguous memory allocation.

The filesystem subsystem of an OS similarly deals with allocation of disk space to files, and similarly the presentation moves from simple models (again: contiguous allocation) to more complex models, including multiple-indirection blocks which allow for a small file allocation table and huge file sizes.

Traditionally these two topics are taught separately. But since there are many similarities between the two subsystems