

***Einführung in die Informatik I:
Informatik-Systeme***

Prof. Bernd Brügge, Ph.D
Technische Universität München

Wintersemester 2000/2001

24.Oktober 2000

Überblick der Vorlesung

- ❖ Der Begriff des Systems
- ❖ Definition eines Informatik-Systems
- ❖ Wie entwickeln wir Informatik-Systeme?
 - Wie modellieren wir die Realität?
 - Beschreibungstechniken
- ❖ Aktivitäten bei der Entwicklung eines Informatik-Systems:
 - Analyse, Systementwurf, Detaillierter Entwurf, Implementation, Test
- ❖ Typische Aktivitäten bei der Implementation

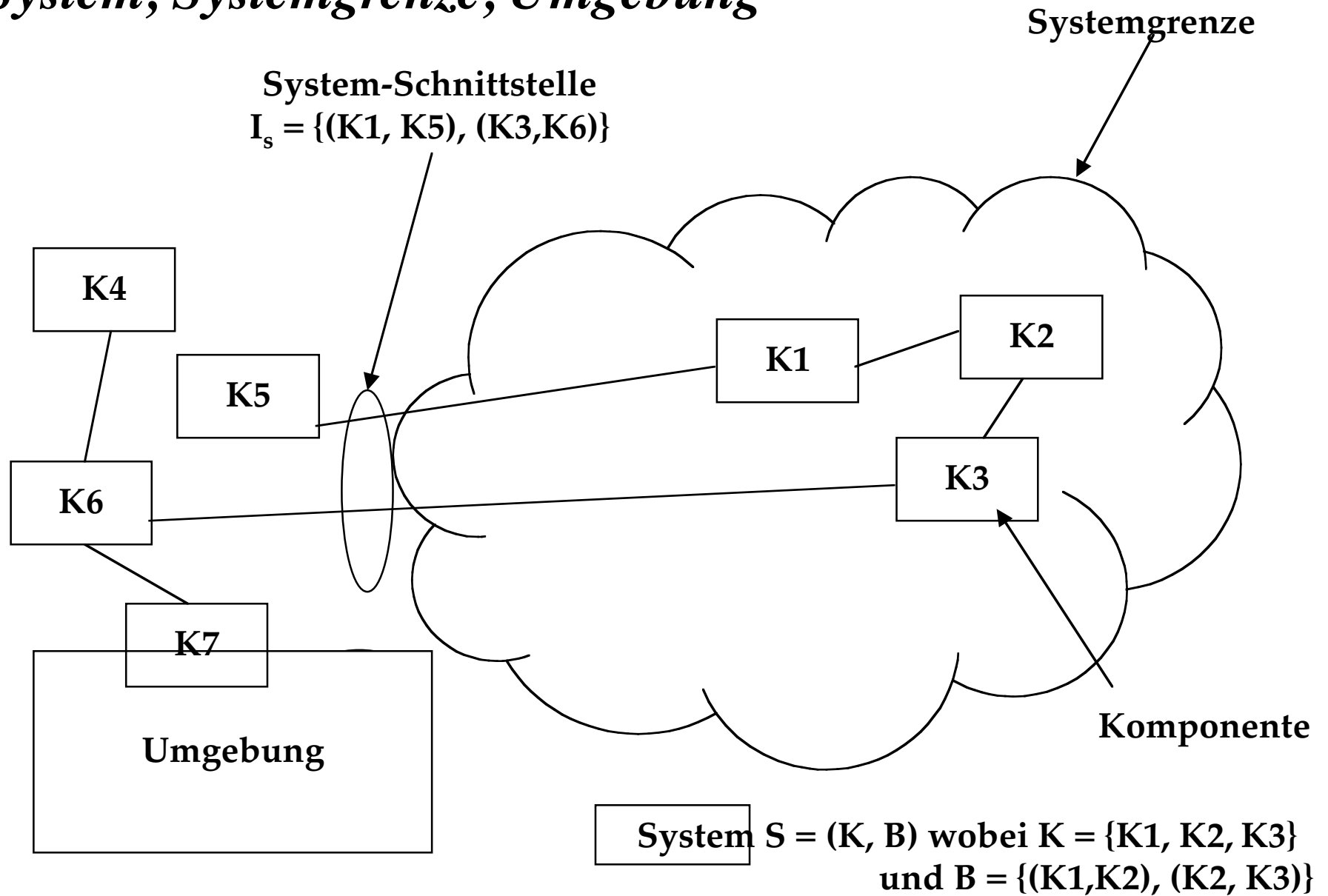
Der Begriff des Systems

- ❖ Definition eines Systems: **Unter einem System versteht man eine Menge von Gegenständen, die in einem gegebenen Bezugssystem in einem Zusammenhang stehen, und die Beziehungen zwischen diesen Gegenständen. Die Gegenstände heißen Bausteine oder Komponenten.**
- ❖ Jedes System hat eine Systemgrenze
 - Die Systemgrenze legt fest, welche Komponenten zu einem System gehören. Alle anderen Komponenten bezeichnet man als die Umgebung.
- ❖ Definition System-Schnittstelle: **Die Beziehungen, die über die Systemgrenze laufen, d.h.**
 - die Beziehungen zwischen Komponenten des Systems und Komponenten in der Umgebung.

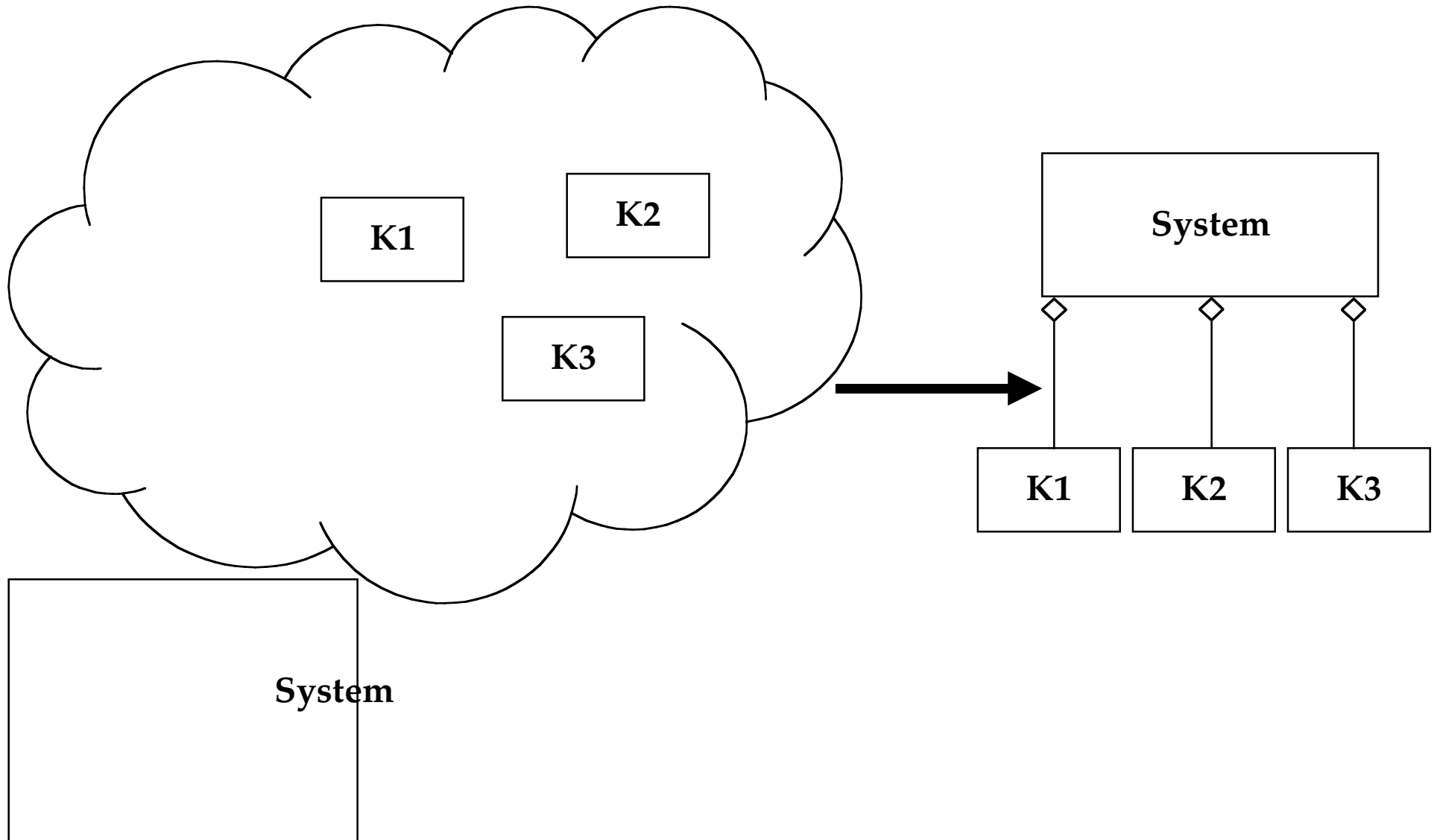
Was ist kein System?

- ❖ Ein Sandhaufen in einer Sandkiste ist kein System.
 - Er besteht aus einer Menge von Sandkörnern, die aber in keiner Beziehung zueinander stehen (zumindestens im gegebenen Bezugssystem).

System, Systemgrenze, Umgebung



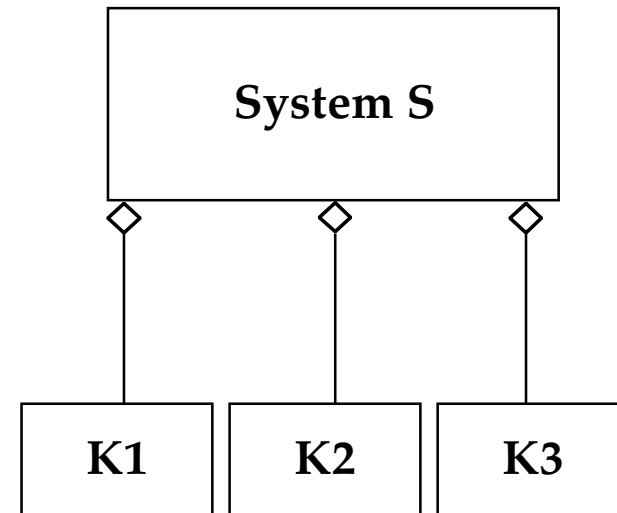
Statt der Wolke benutzen wir einen Graphen



Die Aggregations Beziehung

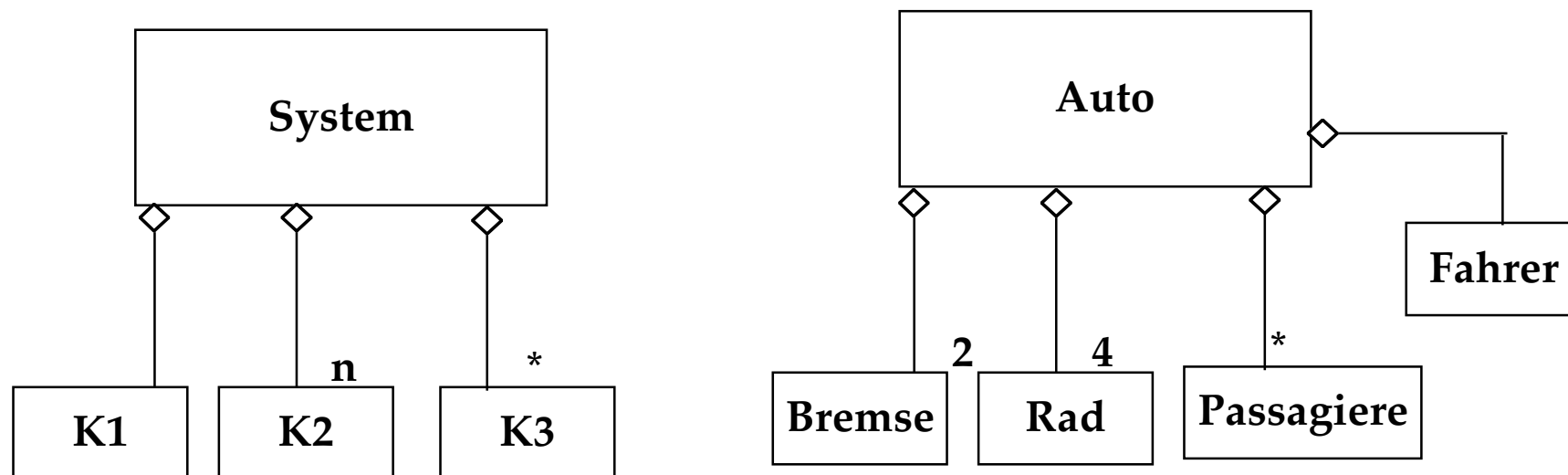
- ❖ Aggregation:
 - Ein Gegenstand *besteht aus* einem oder mehreren Gegenständen
 - Ein Gegenstand ist *Teil von* einem anderen Gegenstand
- ❖ **Aggregation zwischen zwei Gegenständen G1 und G2 wird durch eine Kante zwischen G1 und G2 mit einem Karo-Symbol an einem Ende bezeichnet**
- ❖ Leseweise:
 - Vom Karo-Symbol aus: „*besteht aus*“
 - Zum Karo-Symbol hin: „*ist Teil von*“

- ❖ S besteht aus K1, K2, und K3
- ❖ K3 ist Teil von S



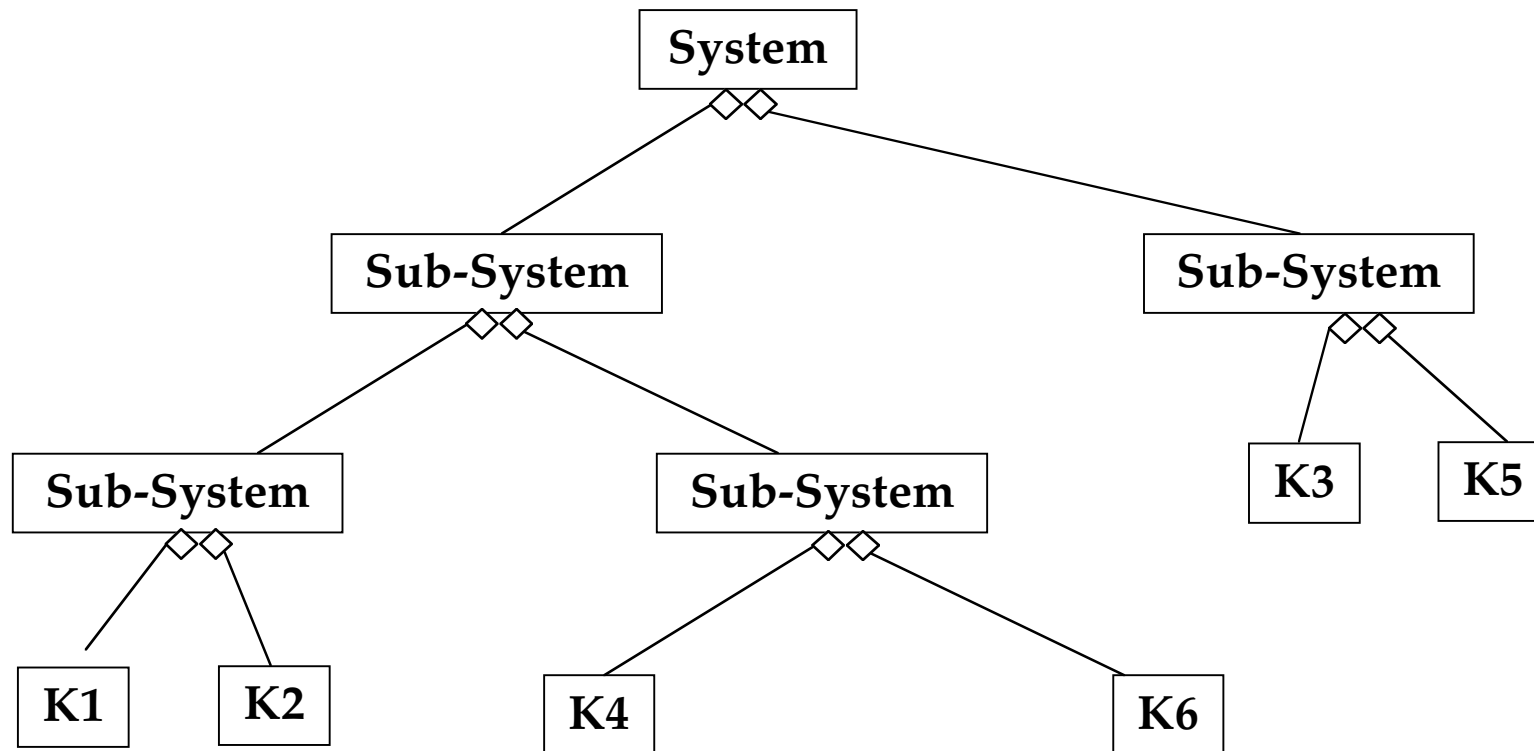
Modellierung der Vielfachheit in Aggregationen

- ❖ In der Aggregationsbeziehung kann die Vielfachheit von Komponenten angegeben werden.
- ❖ Eine natürliche Zahl am Ende der Kante bezeichnet die Anzahl der Komponenten. 2 besondere Fälle:
 - Vielfachheit 1 wird nicht explizit angezeigt,
 - Vielfachheit “viele” wird durch ein Sternchen * angezeigt.



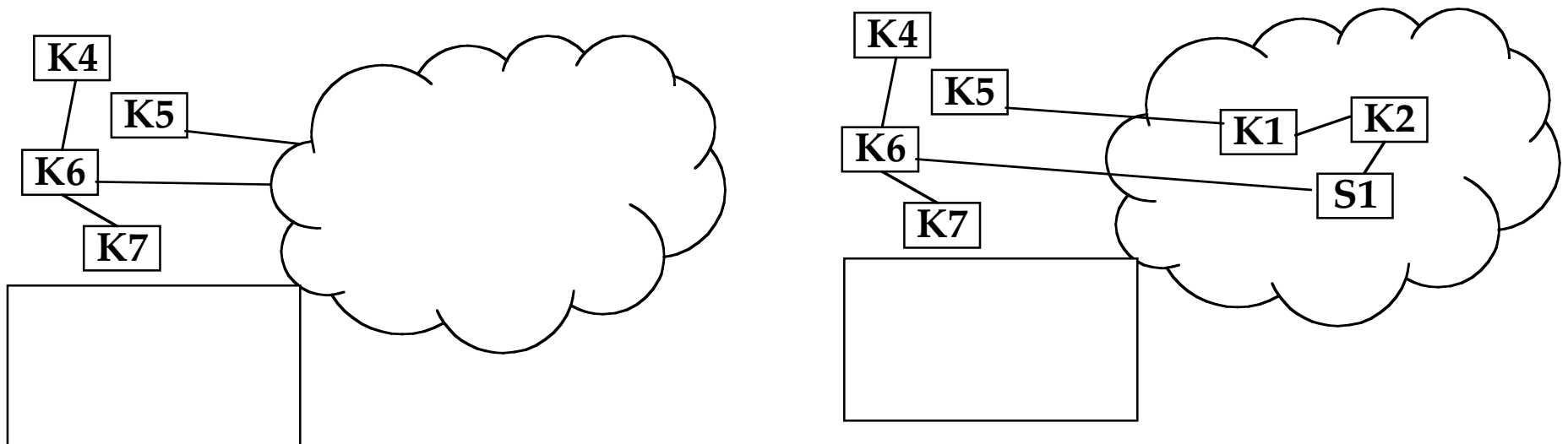
Der Systembegriff ist rekursiv

- ❖ Die Komponenten eines Systems können selbst wieder (Sub)Systeme sein.



Rekursion in Informatik-Systemen

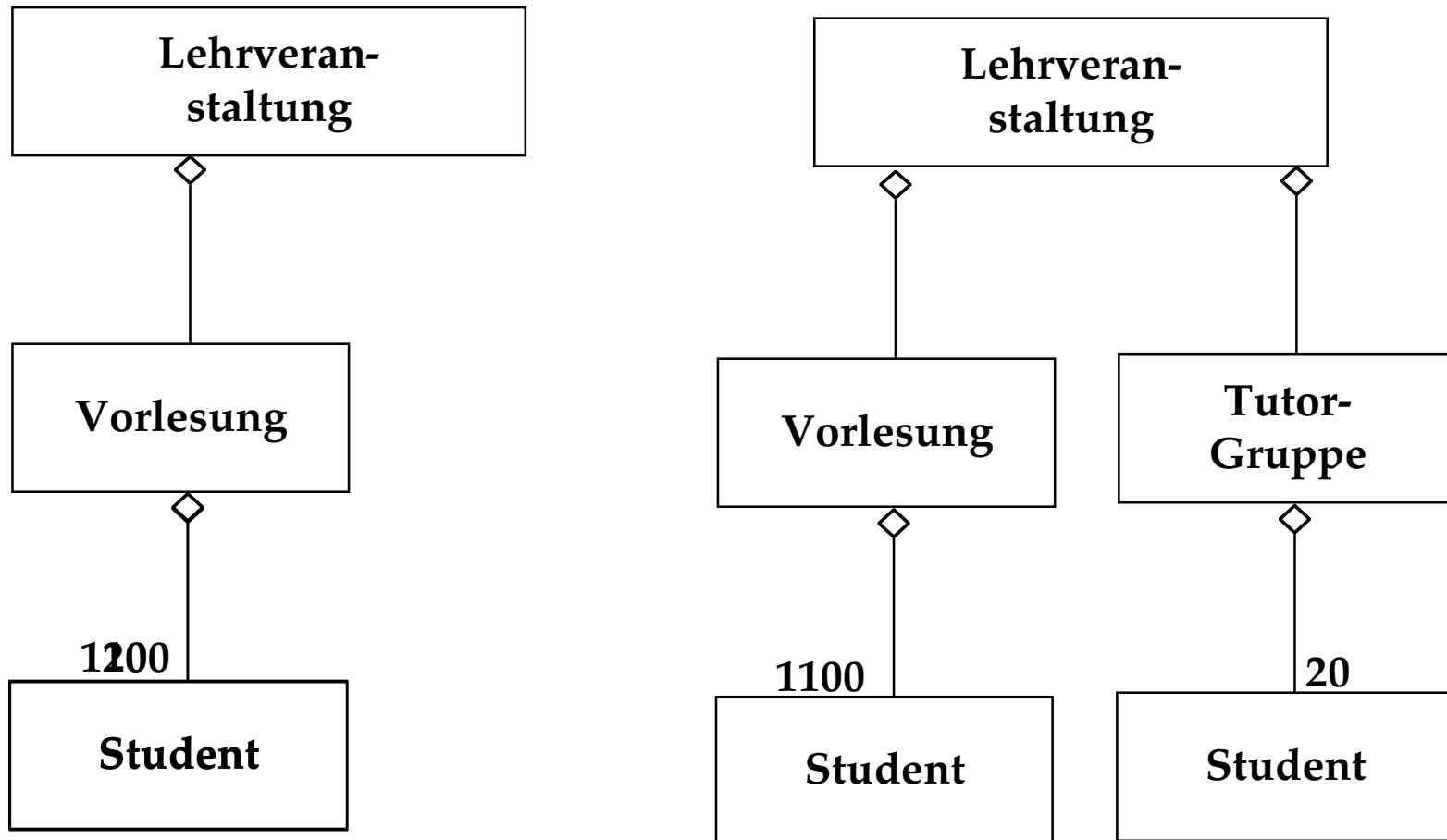
- ❖ Die Komponenten eines Systems können wieder selbst (Sub)Systeme sein. Wir unterscheiden 2 Fälle:
 - Das System ist ein schwarzer Kasten (**black box**): Das System wird als einzelner Gegenstand aufgefasst.
 - Das System ist ein weißer Kasten (**white box**): Für das Verständnis des Systems ist die Zusammensetzung in Subsysteme und Komponenten wichtig.



Rekursion

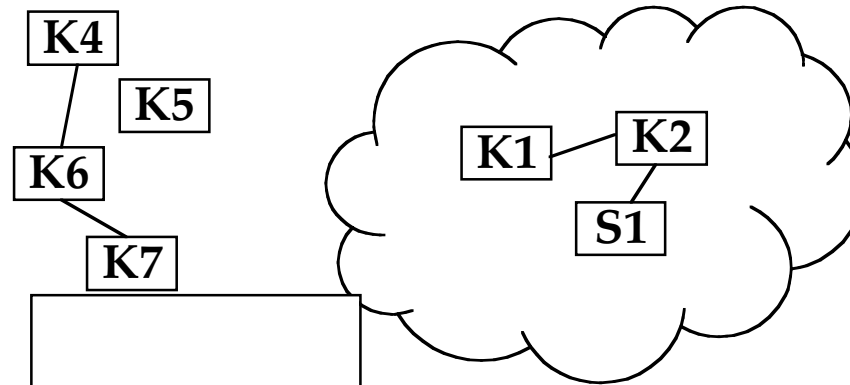
- ❖ Rekursion ist ein wichtiger Begriff in der Informatik, weil rekursive Systeme skalierbar sind:
 - Das Studium von Systemen im Kleinen liefert Erkenntnisse, die auf größere Systeme übertragbar sind.
- ❖ Beispiel: Vorlesung
 - Problem: Erreichbarkeit der einzelnen Studenten
 - Lösung: Einführung eines Subsystems Tutorgruppe

Beispiel: Studentenverwaltungssystem

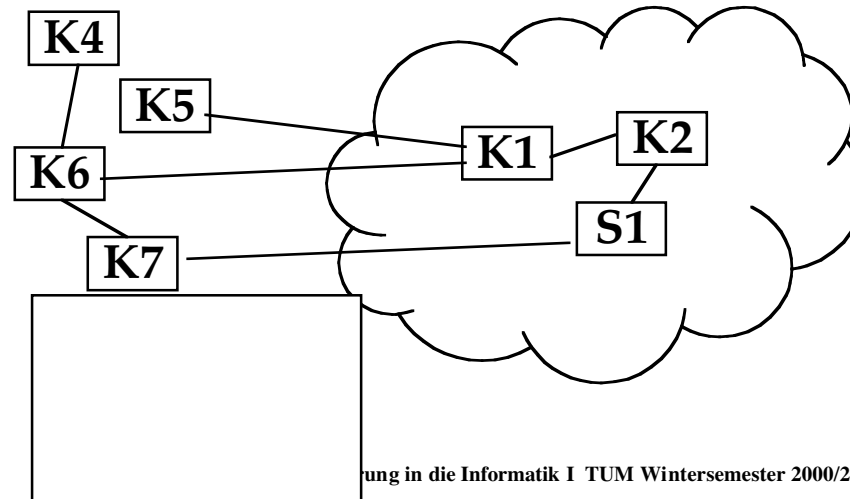


Offenes vs geschlossenes System

- ❖ Geschlossenes System: **Die Komponenten des Systems haben keine Beziehung zu den Komponenten der Umgebung.**



- ❖ Offenes System: **Die Komponenten des Systems haben eine Beziehung zu den Komponenten der Umgebung. Die Systemschnittstelle kann sich sogar ändern.**



Beispiele von offenen Systemen

- ❖ **Ein Buchhaltungssystem, das die neuesten Steuergesetze berücksichtigen muss.**
- ❖ **Ein Straßennavigationsystem, das die aktuellen Baustellen anzeigt.**
- ❖ **Eine existierende Datenbank, die ans Internet anzuschließen ist.**
- ❖ **Eine Studentenverwaltungssystem, das die neueste Prüfungsordnung berücksichtigen muß.**

Der Begriff „geschlossenes System“ ist eine Idealisierung. Wir benutzen geschlossene Systeme in Info 1 aus didaktischen Gründen. In der Praxis hat der Informatiker fast nur mit offenen Systemen zu tun.

Kategorisierung von Informatik Systemen

❖ Die Aufgaben, die wir mit Informatik-Systemen bearbeiten, haben wir in 5 Klassen eingeordnet:

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



Offene Systeme

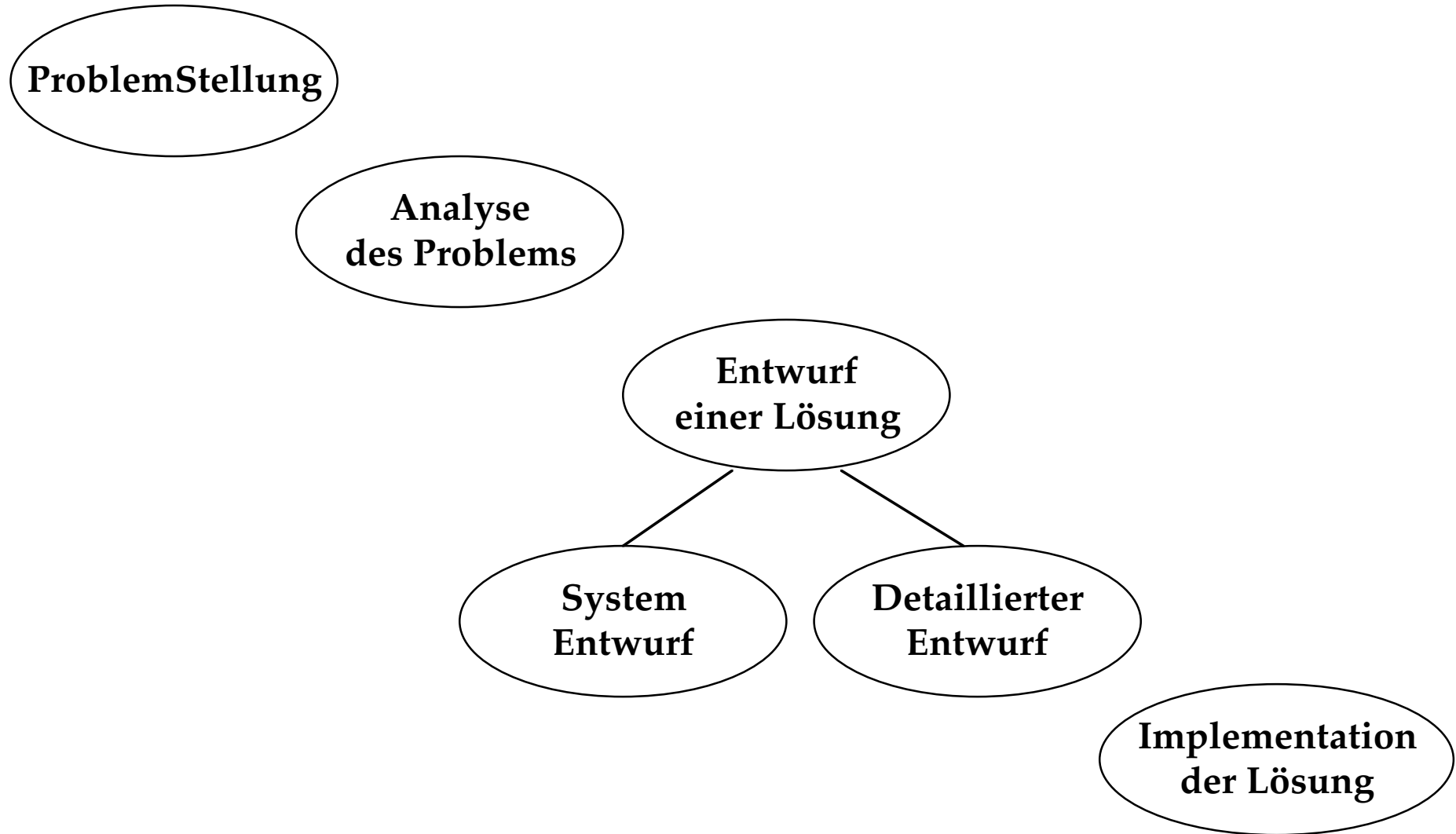
Informatik-Systeme

- ❖ Interaktive Systeme, Prozessüberwachung, eingebettete und adaptive Systeme bezeichnen wir als *reaktive Systeme*.
- ❖ Definition Reaktives System: **Ein reaktives System berechnet nicht ein fertiges Ergebnis $f(x)$ für einen Eingabewert x , sondern reagiert auf Ereignisse und Daten aus der Umgebung.**
- ❖ In Info I behandeln wir die Systemklassen “Berechnung von Funktionen” und “Interaktive Systeme”

Wie entwickelt man Informatik-Systeme?

- ❖ Anforderungen bei der Abbildung der Realität auf ein Modell
 - Funktionelle Anforderungen
 - ◆ Gewünschte Funktionalität
 - Nichtfunktionelle Anforderungen (auch Leistungsmerk-male genannt)
 - ◆ Antwortzeit, Anzahl der unterstützen Benutzer
- ❖ Unterscheidung von Entwicklungsaktivitäten:
 - Analyse, Entwurf, Implementation, Test, Lieferung

Aktivitäten bei der Entwicklung eines Informatik-Systems



Aktivitäten bei der Entwicklung eines Informatik-Systems

- ❖ Problemstellung:
 - Beschreibung des Problems und Ziele, Lösungsanforderungen
- ❖ Analyse:
 - Modellierung der Realität und Erstellung eines Modells
- ❖ Entwurf (Design):
 - Konstruktion einer Lösung, die die Machbarkeit des Modells demonstriert und den Anforderungen gerecht wird
 - ◆ **System-Entwurf:** Zerlegung der Lösung in Komponenten
 - ◆ **Detaillierter Entwurf:** Spezifikation der Schnittstellen
- ❖ Implementation:
 - Programmierung der Lösung mit einer Programmiersprache
- ❖ Test:
 - Überprüfung der Lösung: Sind die gewünschten Ziele erreicht?

Entwicklungsaktivitäten am Beispiel eines Studentenverwaltungssystems

❖ Problembeschreibung:

- Die Verwaltung von Studenten-Daten ist bisher ein Papier-basierter Prozess (Scheine, Zeugnisse)

❖ Ziel:

- Ein rechner-basiertes System (Informatik-System) für die Verwaltung
- Fakultät und Studenten sollen sich jederzeit einen Überblick über den "Studienstatus" des Studenten machen können:
 - ◆ *"Wieviele Punkte habe ich schon für den Bachelor-Grad erreicht?"*

Analyse-Aktivitäten

❖ Analyse:

- *Wer sind die Benutzer?* Studenten, Immatrikulationsamt Prüfer, Prüfungsausschuss, Prüfungsamt
- *Was sind die funktionellen Anforderungen, d.h. welche Funktionalität soll das System bereitstellen?*
 - ◆ Einloggen ins System
 - ◆ Eintragen neuer Studenten in ein ~~System~~
 - ◆ Suchen nach Studenten in das ~~System~~
 - ◆ Sortieren des Studentenverzeichnisses
 - ◆ Studienstatus von Studenten
- *Was sind die nichtfunktionellen Anforderungen?*
 - ◆ Die Antwortzeit des Systems soll kleiner als 1 Sekunde sein

Semesternoten, ECTS Punkte,
Scheine, Voraussetzungen für
Vorlesungen

Entwurfsaktivitäten...

- ❖ System-Entwurf:
 - Welch Subsysteme hat das System?
 - ◆ Benutzerschnittstelle, Datenbasis, Auswertungskomponente
 - Welche Schnittstellen stellen die Subsysteme bereit?
 - ◆ Einloggen, Sortieren, Suchen, Berechnung der Note für das Zeugnis
- ❖ Detaillierter Entwurf:
 - Definition der APIs (Detaillierte Schnittstellenbeschreibung für den Programmierer)
 - ◆ Genaue Definition der Operationen für Student, Studentenverzeichnis, Schein und Zeugnis
 - Entwurf/Wahl von Algorithmen und Datenstrukturen
 - ◆ Sortierroutinen: Bubblesort, Quicksort oder Mergesort?
 - ◆ Studentenverzeichnis: Menge, Sequenz oder Baum?

Schnittstellenbeschreibung

Studentenverwaltungssystem

- **Login (Name)**
- **Eintrag (Neuer Student, Studentenverzeichnis)**
- **Suche (Name, Studentenverzeichnis)**
- **Sortiere (Studentenverzeichnis, Suchkriterium)**
- **Studienstatus (Name, Studentenverzeichnis)**

- ❖ **Die Liste dieser Operationen heißt die Schnittstelle des Systems**
 - *Dienste* im Systementwurf
 - *API* (Application Programmer Interface) im detaillierten Entwurf.
- ❖ **Mathematisch ist das Studentenverwaltungssystem eine Algebra:**
 - *Die Schnittstelle heißt dann auch Signatur* (Details in der Vorlesung über Algebren und Termalgebren)
- ❖ **Der Benutzer eines Systems braucht nur dessen Signatur zu kennen, um es zu benutzen.**
 - **Kenntnisse über die Implementation sind nicht nötig.**

Implementations- und Testaktivitäten...

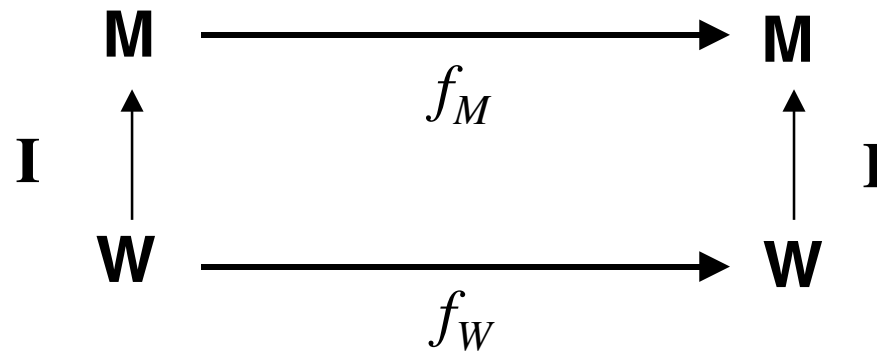
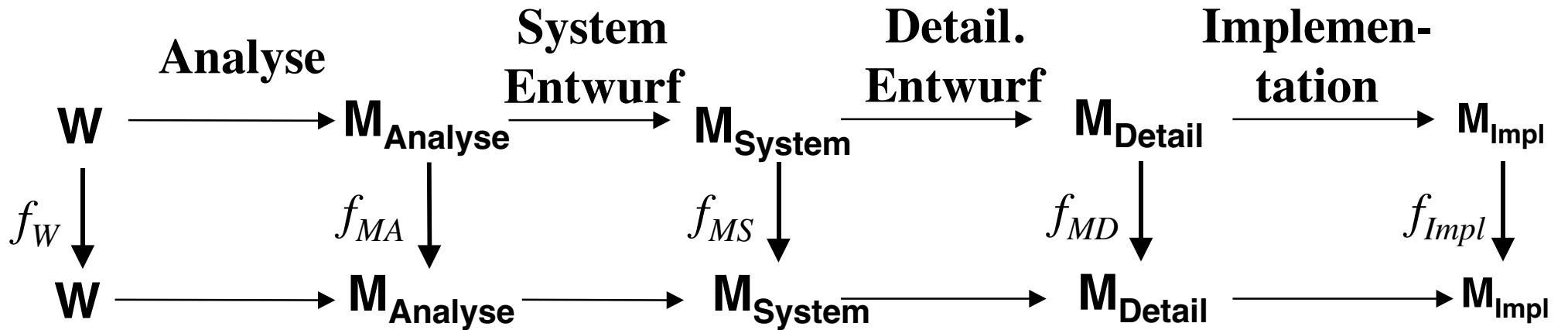
❖ Implementation:

- Wahl der Programmiersprache (z.B. Java für web-basiertes System)
- Schreiben der Datenstrukturen und Algorithmen,
- Editieren, Compilieren, Exekutieren

❖ Test:

- Entwurf von Tests. Was wird getestet? Bedienbarkeit, gewünschte Funktionalität (siehe Analyse), Fehlermeldungen.
- Wahl der Tester. Die 1100 Studenten in Info I!!

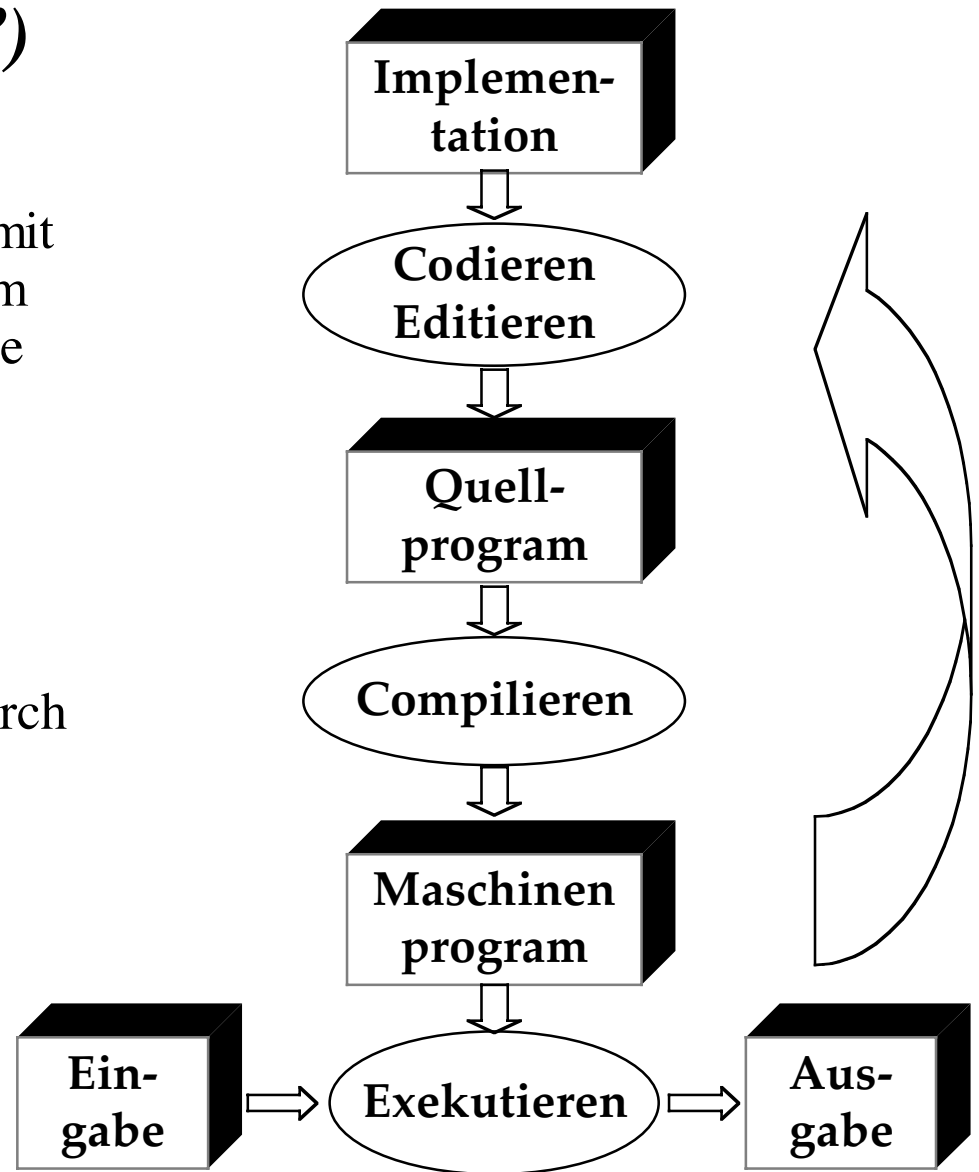
Entwicklungsaktivitäten sind Modeltransformationen



Subaktivitäten während Implementation und Test

(“edit-compile-execute cycle”)

- ❖ Codieren:
 - Übersetzung des detaillierten Entwurfs mit Hilfe eines Editors in ein Quellprogramm in der ausgewählten Programmiersprache
- ❖ Editieren:
 - Verbesserungen/Veränderungen am Quellprogramm
- ❖ Compilieren
 - Transformation des Quellprogramms durch einen Übersetzer (Compiler) in ein ausführbares Programm (Maschinenprogramm)
- ❖ Exekutieren
 - Exekution des Programs durch einen Interpreter auf einer Maschine.



Interaktion Modell <-> Realität

- ❖ **Wir benutzen Modellvorstellungen um die Realität zu steuern.**

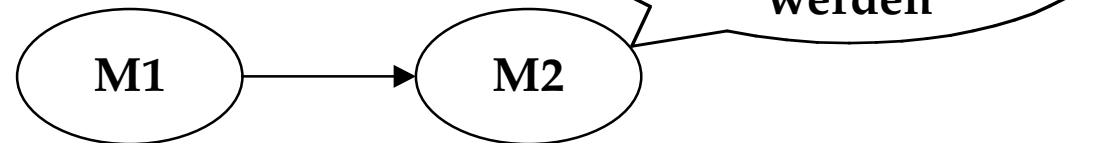
- ❖ **Beispiel: Ich komme abends nach nach Hause und gieße meinen Garten für 20 Minuten. Danach schalte ich den Gartenschlauch ab.**
 - **Mein Modell: Wenn mein Garten trocken ist, dann sind 20 Minuten Bewässerung gut genug, so dass die Pflanzen nicht verderben.**
 - **Die Realität: Ob das wahr ist, bleibt offen.**

- ❖ **Frage: Wie kann man in der Realität überprüfen, ob der Modellschluss stimmt? --> *Verifikation und Validation***

Wahrheitsgehalt von Modellen

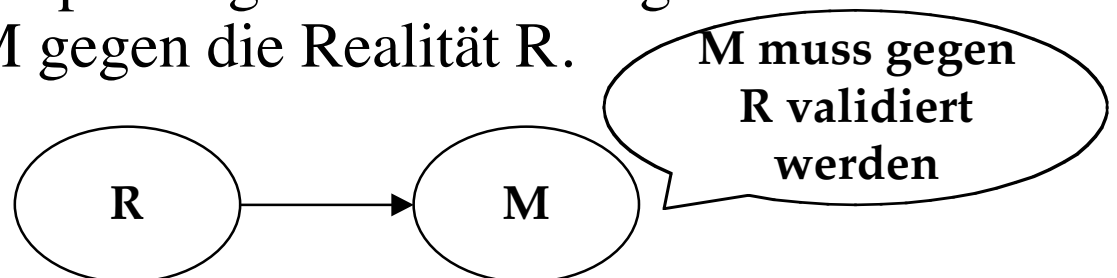
- ❖ Definition Spezifikation: **Eine Wirklichkeit, die unserer Gedankenwelt entstammt, oder ein Modell.**
- ❖ Die Übereinstimmung einer Spezifikation mit einem Modell lässt sich mit mathematischen und logischen Schritten prüfen.

– **Definition Verifikation:** Die Überprüfung des Wahrheitsgehaltes eines Modells M2 gegen eine Spezifikation M1.

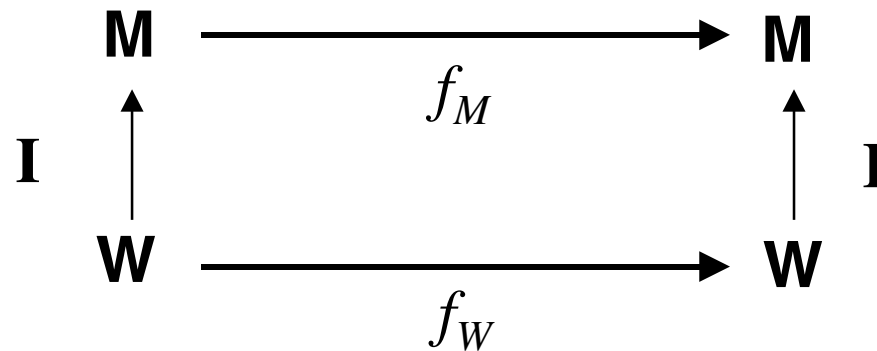
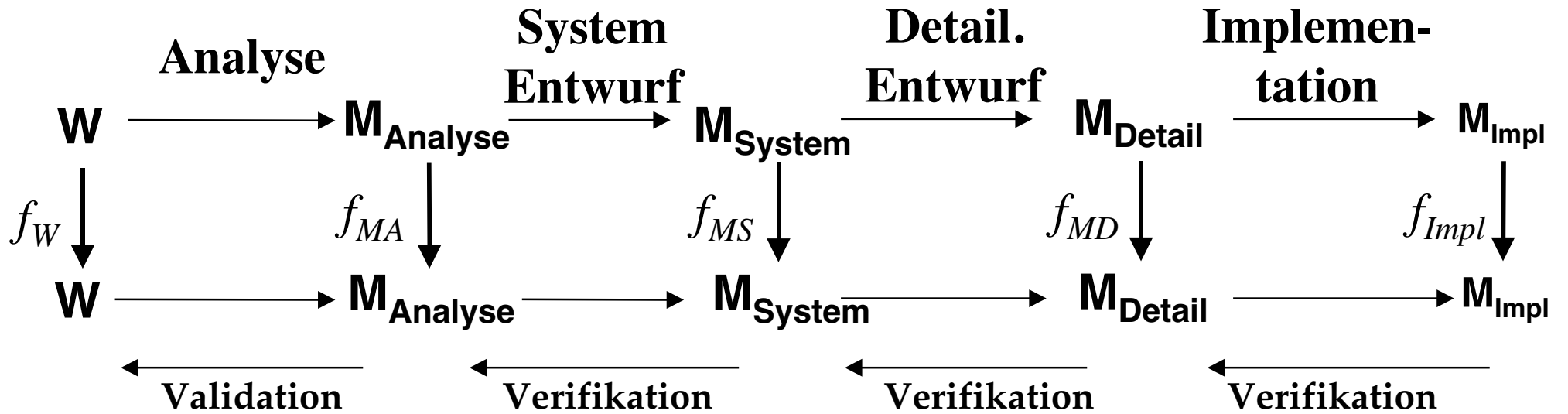


- ❖ Wenn die Realität gegeben ist und nicht selbst der Welt des Denkens entstammt, dann können wir den Wahrheitsgehalt eines Modells nur durch Experimente feststellen:

– **Definition Validation:** Überprüfung des Wahrheitsgehaltes eines Modells M gegen die Realität R.



Verifikation und Validation von Modellen



Methoden zur Validation von Modellen

- ❖ Bei der Validation muß das Modell mit einem Informatik-System realisiert werden, bevor man prüfen kann, ob es realitätsgetreu ist und die Anforderungen erreicht. Es gibt zwei Möglichkeiten:
 - Simulation
 - Prototypische Realisierung

Simulation

- ❖ Man realisiert einzelne Systemkomponenten und ihre Beziehungen untereinander soweit, dass man die gewünschten Aussagen über die Systemanforderungen an Einzelfällen überprüfen kann.
- ❖ Bei der Simulation wird die Funktion des simulierten Systems oft nicht berücksichtigt. Eine Simulation adressiert oft nur die nichtfunktionalen Anforderungen
- ❖ Beispiele:
 - Der Interrupthandler des Startsystems der Space Shuttle darf nicht mehr als 50 Mikrosekunden für einen Interrupt verwenden.
 - Das GPS Subsystem im Navigationssystem eines Autos muss zwischen minus 70 C und plus 100 C funktionieren.
- ❖ Wie kann man diese simulieren?

Prototypische Realisierung

- ❖ Adressiert die funktionalen Anforderungen des Systems.
 - Die nichtfunktionalen Anforderungen (Leistungsmerkmale) werden in einem Prototyp nicht adressiert.
- ❖ Das Ziel eines “guten Prototypen” ist die Überprüfung des Modells aus der Analyse.
 - Modellierungsfehler sollte man eher zu früh als zu spät entdecken.
 - Die Behebung eines Fehlers, der während oder am Ende der Analyse entdeckt wird kostet weniger als die Behebung während der Implementation.

Zusammenfassung

- ❖ Systembegriff, Umgebung, Schnittstelle
- ❖ Offenes vs. geschlossenes System
- ❖ Aktivitäten bei der Entwicklung eines Informatik-Systems
 - Analyse, Systementwurf, Detaillierter Entwurf, Implementation
- ❖ Aktivitäten bei der Implementation
 - Editieren, Compilieren, Exekutieren, Testen, Debuggen
- ❖ Unterschied Simulation und Prototyp