Grundlagen der Programmierung

Dr. Christian Herzog
Technische Universität München

Wintersemester 2006/2007

Kapitel 2: Informatiksysteme

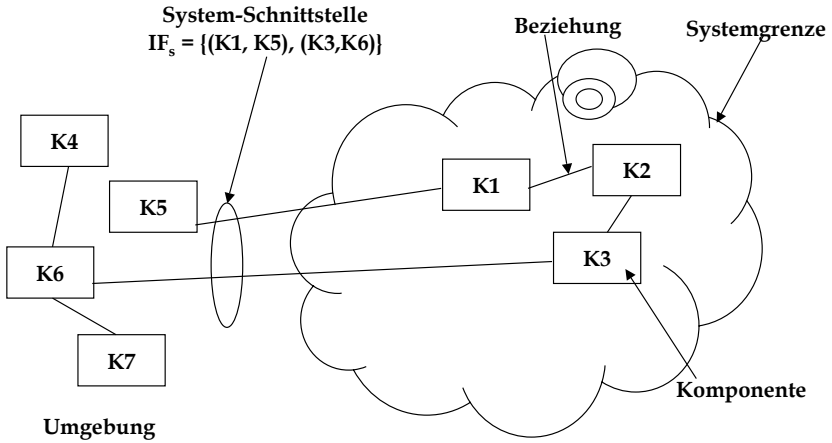
Überblick über Kapitel 2 der Vorlesung

- **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**
- **Verifikation und Validation von Modellen**

Der Begriff des Systems

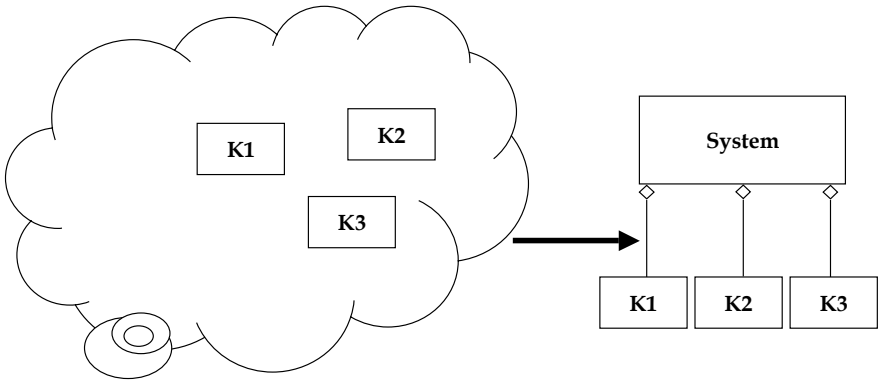
- **Definition eines Systems:** Unter einem System versteht man eine *Menge von Gegenständen*, die in einem gegebenen Bezugssystem in einem Zusammenhang stehen, *und die Beziehungen zwischen diesen Gegenständen*. Die Gegenstände heißen Bausteine oder Komponenten.
- **Jedes System hat eine Systemgrenze**
  - 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.

System, Systemgrenze, Umgebung



System S = (K, B) wobei K = {K1, K2, K3}
und B = {(K1,K2), (K2, K3)}

Statt der Wolke benutzen wir einen Graphen

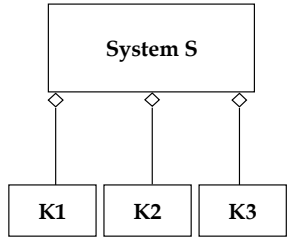


System

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 Rauten-Symbol an einem Ende bezeichnet.
- **Leseweise:**
  - Vom Rauten-Symbol aus: *„besteht aus“*
  - Zum Rauten-Symbol hin: *„ist Teil von“*

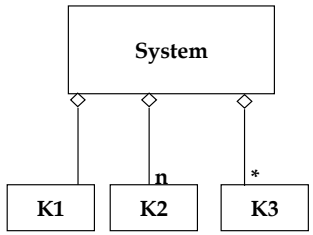
- S besteht aus K1, K2 und K3.
- K3 ist Teil von S.

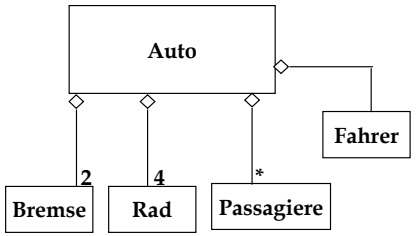


- Dies ist eine Darstellung in **UML** (*Unified Modeling Language*).
- Wir werden in der Vorlesung UML verwenden bzw. uns stark an UML anlehnen.

Modellierung der Vielfachheit in Aggregationen

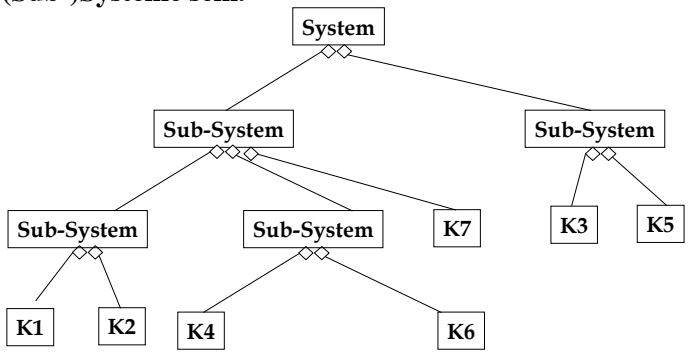
- In der Aggregationsbeziehung kann die Vielfachheit (Multiplizität) von Komponenten angegeben werden.
- Eine natürliche Zahl am Ende der Kante bezeichnet die Anzahl der Komponenten. 2 besondere Fälle:
  - Vielfachheit 1 wird nicht explizit angezeigt,
  - Vielfachheit “viele” wird durch ein Sternchen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





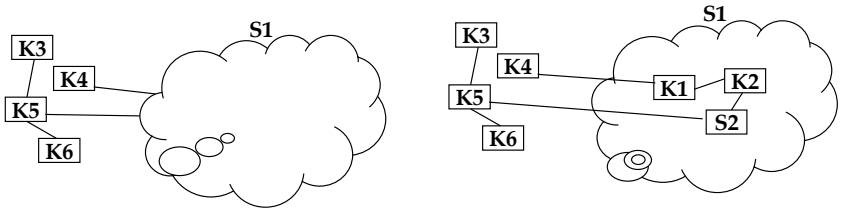
Der Systembegriff ist rekursiv

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



Rekursion in Informatik-Systemen

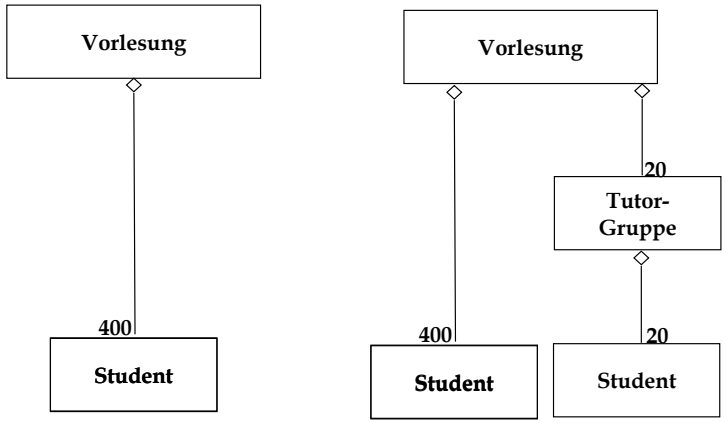
- **Die Komponenten eines Systems können selbst wieder (Sub-)Systeme sein. Wir unterscheiden 2 Fälle:**
  - Das Subsystem ist ein schwarzer Kasten (**black box**): Das Subsystem wird als einzelner Gegenstand aufgefasst.
  - Das Subsystem ist ein durchsichtiger Kasten (**white box**): Für das Verständnis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



Rekursion

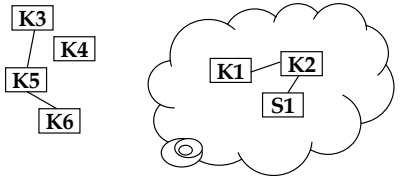
- **Rekursion ist ein wichtiger Begriff in der Informatik, weil rekursive Systeme skalierbar sind:**
  - Das Studium von Systemen im Kleinen liefert Erkenntnisse, die auf größere Systeme übertragbar sind.
- **Beispiel: Übung zu einer großen Vorlesung**
  - Problem: Erreichbarkeit der einzelnen Studenten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

Beispiel: Studentenverwaltungssystem

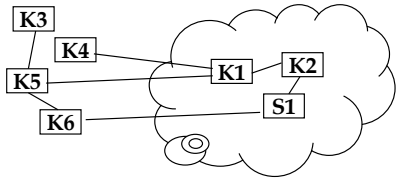


Offenes vs. geschlossenes System

- **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 Buchhaltungsystem, 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 im Rahmen dieser Vorlesung aus didaktischen Gründen. In der Praxis hat der Informatiker fast nur mit offenen Systemen zu tun.

Kategorisierung von Informatik Systemen

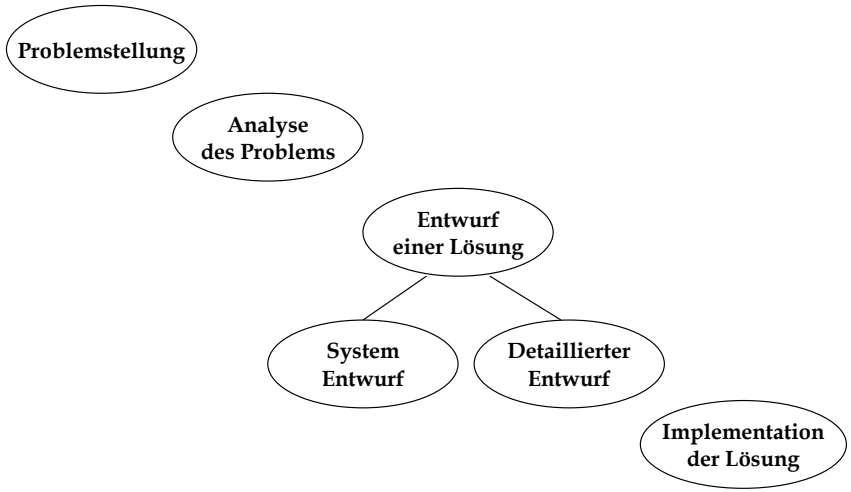
- **Die Aufgaben, die wir mit Informatik-Systemen bearbeiten, haben wir in 4 Klassen eingeordnet:**
  1. Berechnung von Funktionen
  2. Prozessüberwachung
  3. Eingebettete Systeme
  4. Adaptive Systeme

  } **Offene Systeme**

Wie entwickelt man Informatik-Systeme?

- **Anforderungen bei der Abbildung der Realität auf ein Modell**
  - Funktionale Anforderungen
    - Gewünschte Funktionalität
  - Nichtfunktionale Anforderungen (auch Leistungsmerkmale 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 der 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 in 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 papierbasierter Prozess (Scheine, Bescheide)
- **Ziel:**
  - Ein rechnerbasiertes System (Informatik-System) für die Verwaltung
  - Fakultät und Studenten sollen sich jederzeit einen Überblick über den "Studienstatus" des Studenten machen können:
    - *„Welche Scheine fehlen mir noch für die Zulassung zur Abschlussprüfung?“*

Analyse-Aktivitäten

- **Analyse:**
  - *Wer sind die Benutzer?* Studenten, Prüfer, Prüfungsausschuss, Prüfungsamt, Studiendekan
  - *Was sind die funktionalen Anforderungen,* d.h. welche Funktionalität soll das System bereitstellen?
    - Anmelden im System
    - Eintragen neuer Studenten in ein Studentenverzeichnis
    - Suchen nach Studenten im Studentenverzeichnis
    - Sortieren des Studentenverzeichnisses
    - Studienstatus von Studenten
  - *Was sind die nichtfunktionalen Anforderungen?*
    - Die Antwortzeit des Systems soll kleiner als 1 Sekunde sein

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 (*Application Programmer Interface,* detaillierte Schnittstellenbeschreibungen für den Programmierer)
    - Genaue Definition der Operationen für Student, Studentenverzeichnis, Schein und Bescheid
  - Entwurf/Wahl von Algorithmen und Datenstrukturen
    - Sortieralgorithmen: Bubblesort, Quicksort oder Mergesort?
    - Studentenverzeichnis: Menge, Sequenz oder Baum?

Schnittstellenbeschreibung

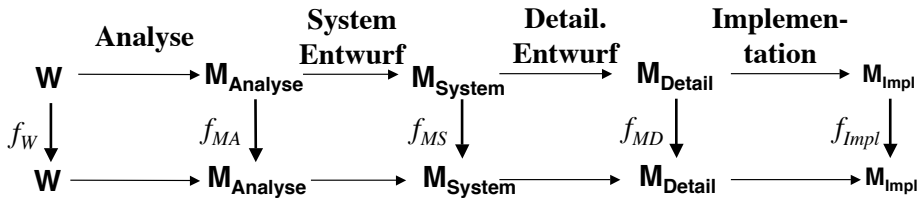
Studentenverwaltungssystem
Anmeldung (Name) Eintrag (Neuer Student, Studentenverzeichnis) Suche (Name, Studentenverzeichnis) Sortiere (Studentenverzeichnis, Suchkriterium) Studienstatus (Name, Studentenverzeichnis)

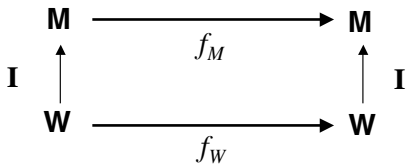
- Die Liste dieser Operationen heißt die Schnittstelle des Systems
  - *Dienste* im Systementwurf
  - *API* (*Application Programmer Interface*) im detaillierten Entwurf.
- Der Benutzer eines Systems braucht nur dessen Schnittstelle zu kennen, um es zu benutzen.
- Kenntnisse über die Implementation sind nicht nötig.

Implementations- und Testaktivitäten...

- **Implementation:**
  - Wahl der Programmiersprache (z.B. Java für web-basiertes System)
  - Schreiben der Datenstrukturen und Algorithmen,
  - Editieren, Compilieren, Ausführen
- **Test:**
  - Entwurf von Tests. Was wird getestet? Bedienbarkeit, gewünschte Funktionalität (siehe Analyse), Fehlermeldungen.
  - Wahl der Tester.

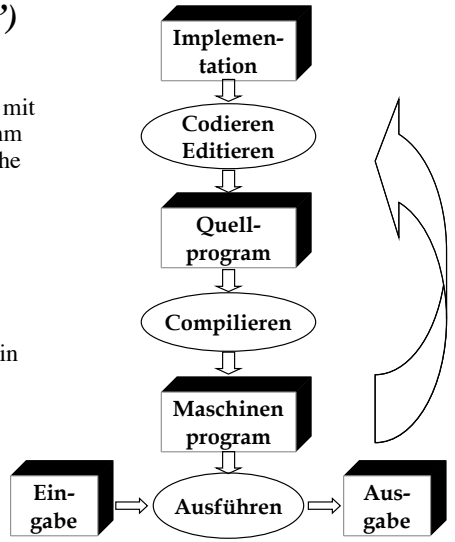
Entwicklungsaktivitäten sind Modelltransformationen





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 ausführ-bares Programm (Maschinenprogramm)
- **Ausführen**
  - Ausführen des Programms durch einen Interpreter auf einer Maschine.

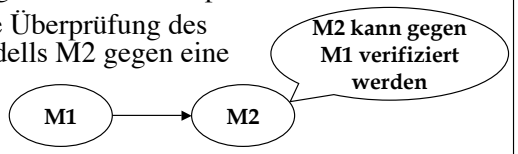


Interaktion Modell <-> Realität

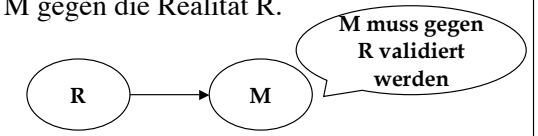
- Wir benutzen Modellvorstellungen um die Realität zu steuern.
- Beispiel: Ich komme abends nach Hause und gieße meinen Garten 20 Minuten lang. Danach schalte ich den Gartenschlauch ab.
  - **Mein Modell:** Wenn mein Garten trocken ist, dann sind 20 Minuten Bewässerung gut genug, so dass die Pflanzen 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

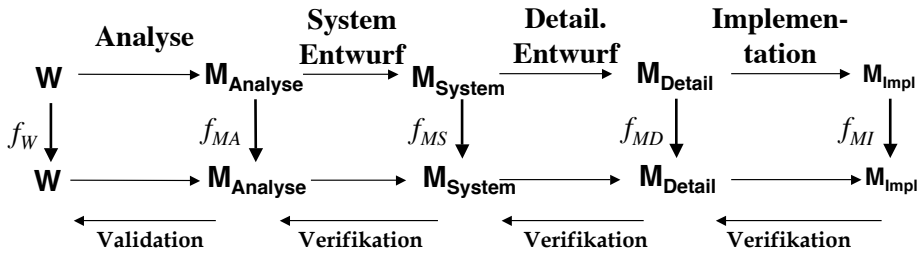
- **Definition Spezifikation:** Eine Wirklichkeit, die unserer Gedankenwelt entstammt, oder ein Modell.
- Die Übereinstimmung einer Spezifikation mit einem Modell lässt sich mit mathematischen und logischen Schritten prüfen.
  - **Definition Verifikation:** Die Überprüfung des Wahrheitsgehaltes eines Modells M2 gegen eine Spezifikation M1.

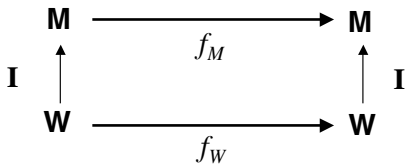


- Wenn die Realität gegeben ist und nicht selbst der Welt des Denkens entstammt, dann können wir den Wahrheitsgehalt eines Modells nur durch Experimente feststellen:
  - **Definition Validation:** Überprüfung des Wahrheitsgehaltes eines Modells M gegen die Realität R.



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 beschränkt sich oft nur auf 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

- **Beschränkt sich auf die funktionalen Anforderungen des Systems.**
  - Die nichtfunktionalen Anforderungen (Leistungsmerkmale) werden in einem Prototyp nicht berücksichtigt.
- **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, Ausführen, Testen
- **Unterschied Simulation und Prototyp**