

Objektorientiertes Software-Engineering

TIT99BPE/TIT99CPE
BA Mannheim

WS 2001/2
F. Schönleber

Organisatorisches

- Kurs 1: TIT99BPE 6.Studienhalbjahr
Termin Mo. 13.00 – 14.30
Raum: 037B

- Kurs 1: TIT99CPE 6.Studienhalbjahr
Termin Mo. 15.00 – 16.30
Raum: 038B

- Klausur
Termin

Ziel der Vorlesung

- Objektorientierte Ansätze verstehen und umsetzen
- Problemstellungen mit OO-Entwurfsmethoden lösen
- Vorgehensmodelle erkennen

Vorgehensweise

- Vorstellung von formalen Aspekten der OOSE
 - Schwerpunkt UML
- Begleitendes praktisches Beispiel mit einem OO-Tool (Rational Rose)
 - Geschäftsprozess Runners Inc.
- Softwaretechnische Beispiele in C++ oder Java

Beispiel Runners Inc. Projekt

- Geschäftsprozess modellieren
- Runners Inc.
 - Verkauft Schuhe an Jogger
 - Bestellungen werden per Telefon angenommen
 - Lieferung erfolgt per Post
 - Es gibt zwei Sorten von Schuhen „Trotter“ und „Sprinter“
 - Jedes Paar Schuhe gibt es in unterschiedlichen Grössen
 - Bisher wurde alles auf Papier bearbeitet
 - Jetzt soll automatisiert werden

Inhalt

1. Einführung in objektorientierte Methoden

- Geschichtliche Entwicklung
- Grundbegriffe der OOSE, Objekt, Klasse, Polymorphie, Einkapselung, Vererbung
- OOA
- OOD

2. Unified Modeling Language (UML)

- Geschichtliche Entwicklung
- Einführung
- Klassenmodelle
- Verhaltensmodelle

3. Analysemuster

4. Entwurfsmuster

5. Heuristiken

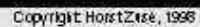
6. OO-Test

7. Vorgehensmodelle

- XP vs RUP vs V-Modell

8. Komponenten

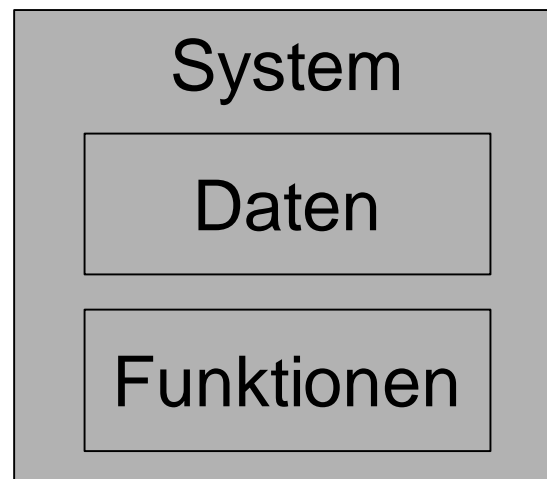
- COM vs Java Beans vs CORBA
- Microsofts .net Strategie



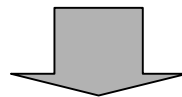
1 Einführung in die Objektorientierung

1.1 Historische Entwicklung

- Softwareentwicklung seit den 60er Jahren: Prozedurale Programmiersprachen (z.B.: FORTRAN, COBOL, Pascal, C)
 - Zerlegung der Aufgaben nach Funktionen
 - Daten werden getrennt von den Funktionen betrachtet
- Prozeßmodelle, die diese Vorgehensweise unterstützen:
 - SADT (Structured Analysis and Design Technique)
 - SA/SD (Structured Analysis and Structured Design)



- Probleme bei dieser Vorgehensweise:
 - Änderungen der Datenstrukturen führten zu Änderungen an vielen davon abhängigen Funktionen
 - Änderungen meist nicht lokal, sondern über das gesamte System verstreut
 - Softwareprojekte wurden immer größer und komplexer
- Beobachtung: Änderung der Funktionalität häufiger als Änderung der Anwendungsobjekte

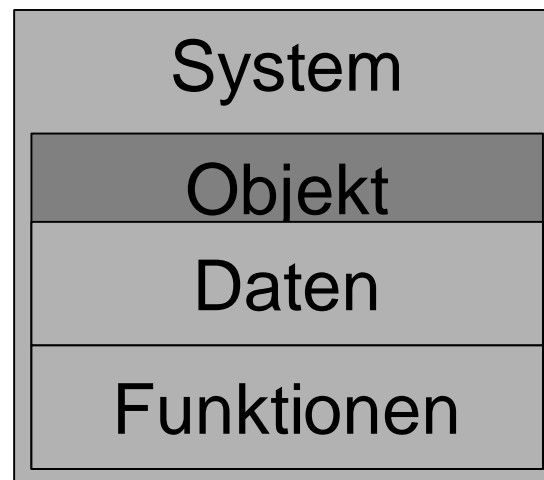


Da Systeme mit Funktionalität im Mittelpunkt entworfen wurden: Häufige Änderung des Gesamtsystems

- **Idee:**
 - Strukturierung von Software nicht nach Funktionen, sondern nach Objekten
 - (Objektorientierte) Objekte enthalten Daten und Funktionen (die mit diesen Daten arbeiten)

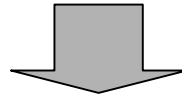
- **Zwischenstufe:**

Programmiersprachen, die die Idee der Datenabstraktion unterstützten
z.B.: Modula, Ada, CLU



- In den 80er Jahren: Aufkommen von objektorientierten Programmiersprachen
 - Smalltalk-80
 - verschiedene Lisp-Dialekte: Flavors, Loops, CLOS
 - Objective-C
 - C++
 - Eiffel
 - schon 1967: Simula
 - Nach 1990 JAVA
 - (Visual Basic, nicht echt objektorientiert)

- Zunächst vorherrschende Meinung:
 - Diese Sprachen bieten genügend Abstraktion, so daß Zwischenstufen im Entwicklungsprozeß (wie Analyse, Design) unnötig sind.
- Nach einiger Erfahrung (u. a. auch mit größeren Projekten): Auch beim objektorientierten Programmieren sind auf diese Vorgehensweise ausgerichtete Prozeßmodelle notwendig.



Ab Ende der 80er bis Mitte der 90er Jahre entstanden eine Vielzahl (mehr als 50) von objektorientierten Prozeßmodellen

- Objektorientierte Analyse und Design nach Booch
- Object Modeling Technique (OMT) nach Rumbaugh et al.
- OOSE nach Jacobson

1.2 Grundbegriffe der objektorientierten Softwareentwicklung

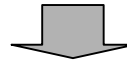
1.2.1 Objekt

A thing presented to or capable of being presented to the senses

- Kann nahezu alles sein!
- Ein Objekt besteht aus
 - Eigenschaften
 - Fähigkeiten
- Beispiel: ein Apfel, Hund, der Student Martin Müller, ein Kochrezept
 - Objekte besitzen eine Identität und sind klar unterscheidbar
- Im engeren, softwaretechnischen Sinne:
ein Objekt besteht aus:
 - Attributen mit zugehörigen Attributwerten sowie
 - Methoden, die auf dem Objekt ausgeführt werden können

Beispiel für ein Objekt

- (Objekt) Student



Paula
Geburtsdatum: 17.11.77 Einkommen: Stipendium Semester: 3.
lernen feiern

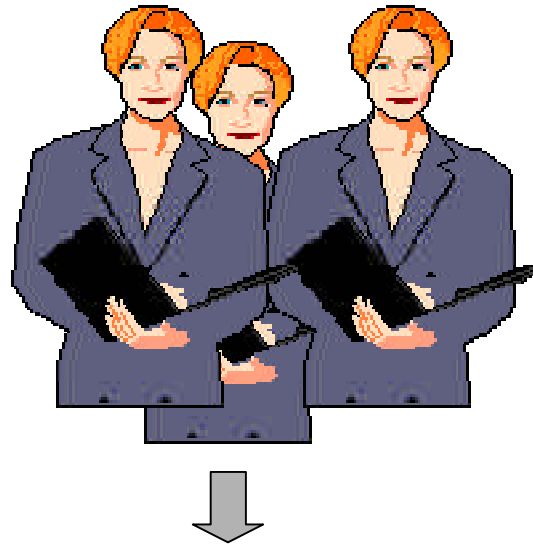
1.2.2 Klasse

- Eine Klasse beschreibt eine Gruppe von Objekten mit
 - gleichen Eigenschaften (Attributen)
 - gemeinsamem Verhalten (Operationen)
 - gemeinsamen Beziehungen zu anderen Objekten
 - gemeinsamer Semantik
- Beispiel Klasse: Student
- Objekte der Klasse Student:
 - der Student Martin Müller, die Studentin Anne Mayer
- Ein Objekt ist eine *Instanz* einer Klasse

Eine Klasse ist eine Schablone für ihre Objekte, die bestimmt, welche Struktur diese Objekte haben (Stempelmetapher)

Beispiel Modellierung einer Klasse

Klasse Student



Student
Geburtsdatum
Einkommen
Semester
lernen
feiern

1.2.3 Objektorientierte Sichtweise (nach Heeg)

