

# OO und UML

Jie Tang

FerienAkademie 2002

Sarntal Italien

OO und UML  
Jie Tang

# Überblick

- ◆ Software Engineering kurz gefasst
- ◆ Vorgehensweise von OO zum Software Engineering
- ◆ Ausblick

OO und UML  
Jie Tang

# Software Engineering kurz gefasst

- ◆ kurze Geschichte
- ◆ Ziel vom SE
- ◆ Eigenschaft und Anforderung vom SE
- ◆ Bestandteile vom Software Engineering(SE)

OO und UML  
Jie Tang

# Kurze Geschichte

*The whole trouble comes from the fact that there is so much tinkering with software. It is not made in a clean fabrication process, which it should be.*

*What we need, is software engineering.*

*📖 Professor F.L.Bauer TU München*

OO und UML  
Jie Tang

## Kurze Geschichte

- ◆ Der Grund – Software Crisis  
Wegen des Scheiterns vieler grosser Softwareprojekte sprach man von Softwarekrise
  1. falsches oder nutzloses System gebaut
  2. schlechte Softwarequalität
  3. Kosten und/oder Deadline nicht eingehalten

OO und UML  
Jie Tang

## Kurze Geschichte

- ◆ Einführung des Begriffs „Softwareengineering“ im Jahr 1968 in Garmisch-Patenkirchen bei einer NATO Konferenz

*„The phrase 'software engineering' was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of **theoretical foundations and practical disciplines**, that are traditional in the established branches of engineering.“*

-P.Naur und B.Randell

OO und UML  
Jie Tang

## Ziel vom SE

- Erzeugen von Software,
- ◆ die wie beabsichtigt funktioniert,
  - ◆ die zuverlässig funktioniert,
  - ◆ und die planmässig und mit vorhersehbaren und angemessenen Kosten abzuliefern ist.

OO und UML  
Jie Tang

## Eigenschaft und Anforderung

- ◆ Wir wissen:
  - Software ist komplex
  - Software unterläuft beständigen Änderungen
- ◆ Wir brauchen:
  - Software, die korrekt ist
  - Software, die robust ist
  - Software, die erweiterbar ist
  - Software, die wiederverwendbar ist.

OO und UML  
Jie Tang

# Bestandteile des SE

## ◆ Prozesse

Unter Prozess verstehen wir den Ablauf und die Pflege der Softwareentwicklung.

- Rational Unified Process
- Agile Development Process
- .....

## ◆ Methodik

Wie betrachten (modellieren) wir die Welt ?

- Strukturierte Denkweise
- Objekt-orientierte Denkweise
- Aspekt-orientierte Denkweise
- .....

OO und UML  
Jie Tang

# OO-Methodik

## ◆ Definition

## ◆ Grundkonzepte der OO

## ◆ Fallstudie

## ◆ Vergleich mit E-R Modell

## ◆ Vergleich mit strukturierter Vorgehensweise

## ◆ Vorteile und Nachteile

## ◆ UML

OO und UML  
Jie Tang

# Definition

## ◆ Es ist schwer, einheitliche Definitionen zu finden. Jeder hat seine eigene Interpretation:

- Object Orientation is a technique for developing models. (in OOSE von I.Jacobson)
- Object Orientation is a method which bases the software system on modules deduced from the types of objects it manipulates (in OOSC von B.Meyer)

OO und UML  
Jie Tang

# Definition

## ◆ Meine Definition:

- Objekt Orientierung basiert auf dem Objekt (statisch), das die kleinste Softwarekomponente ist, und beschreibt die Interaktion von Objekten (dynamisch).

## ◆ Funktionalität

- 1. OO ist ein Modellierungstool
- 2. OO ist eine methodische Vorgehensweise

OO und UML  
Jie Tang

# Grundkonzept

- ◆ Alles ist ein Objekt.
  - Objekte modellieren Eigenschaften und Verhalten unserer Welt
  - Objekte setzen sich aus Daten und Funktionen zusammen
  - Objekte, die gleiche Merkmale besitzen, fassen wir zu einer höheren Abstraktionseinheit – Klassen zusammen
  - Klasse ist eine Implementierung eines Abstract Data Type (ADT)
  - Objekte sind Instanzen von Klasse

OO und UML  
Jie Tang

# Techniken

- ◆ Geheimnisprinzip (Information Hiding)
- ◆ Vererbung (Inheritance)
- ◆ Polymorphismus und dynamische Bindung

OO und UML  
Jie Tang

# Geheimnisprinzip

- ◆ Kapselung
  - Objekte fassen 2 Arten von Informationen zusammen: Daten und Funktionen.
    - Früher separat
  - Verantwortung an die Objekte, die Informationen zu verwalten
  - Beherrschbarkeit der Komplexität

OO und UML  
Jie Tang

# Geheimnisprinzip

- ◆ Das Protokoll zwischen Objekten
  - Blackbox Betrachtung
- ◆ Was und Wie – separation of concerns
  - 2 Anliegen von Objekten
    - Was: Was das Objekt bietet (Client View)
    - Wie: Wie realisiert das Objekt (Supplier View)

OO und UML  
Jie Tang

## Geheimnisprinzip

- ◆ Mißverständnis?
- ◆ Noch nicht weit genug
  - Vorbereitung auf Änderungen
  - Entwurf von Modulen, die Änderungen abschirmen
  - „On the Criteria To Be Used in Decomposing Systems into Modules“ – David L. Parnas

OO und UML  
Jie Tang

## Vererbung

- ◆ Vererbung befähigt eine Klasse, Verhalten und Daten seiner Instanzen als eine Obermenge von anderen Klassen zu definieren.
- ◆ Erweiterung oder Spezialisierung?
  - Modulsicht
  - Typsicht
- ◆ Ermöglicht Wiederverwendung
- ◆ Verhindert Redundanz, entdeckt Ähnlichkeiten

OO und UML  
Jie Tang

## Polymorphismus und dynamische Bindung

- ◆ Eine Tatsache Zwei Sichten
  - Polymorphismus
    - Statische Ersetzbarkeit aus Typsicht
    - Alter Freund – Sichtweise auf Programmierenebene
  - Dynamische Bindung
    - Dynamische Ersetzbarkeit aus Typsicht
- ◆ Nochmals Geheimnisprinzip
- ◆ Ein mächtiges Mittel

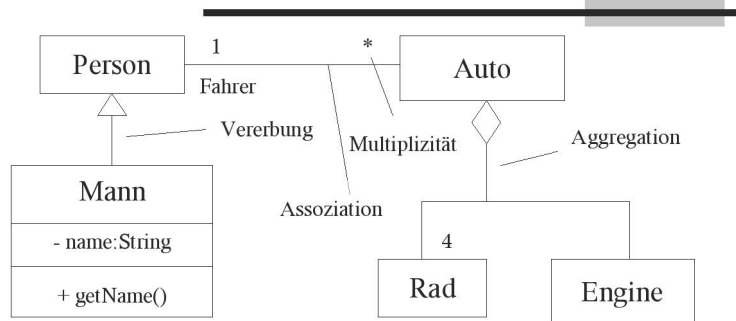
OO und UML  
Jie Tang

## Fallstudie (Acht Damenproblem)

- ◆ Einführung in UML Klassendiagramme
  - UML 1997 eingeführt von G.Booch, J.Rumbaugh und I.Jacobson
  - Die Deutung vom Name
    - Eine Modellierungssprache
    - Unterdrückung vor ihm existierenden OO Notationen
    - Gilt jetzt als Standard von OMG für OO Modellierung
  - Klassendiagramm
    - Strukturierte Darstellung von den Relationen zwischen Klassen
    - Dynamisches System vs. Statisches Klassendiagramm

OO und UML  
Jie Tang

# Klassendiagramm

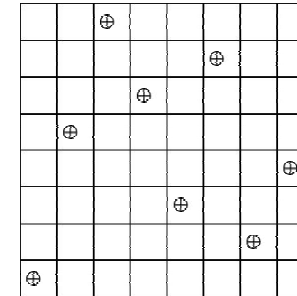


OO und UML  
Jie Tang

# Fallstudie

## ◆ Das Achtdamen Problem

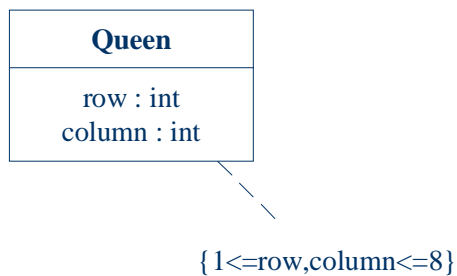
- Was wir haben ist



OO und UML  
Jie Tang

# Analyse

- ◆ Welches Objekt haben wir gesehen ?



OO und UML  
Jie Tang

# Design

## ◆ Eine Lösung konzipieren

- Was wir intuitiv wissen ist:
  - Keine 2 Damen dürfen in einer Spalte auftreten
  - Das Gleiche gilt auch für die Reihe
- Vereinfachen wir auf:
  - Jede Dame ist eine Spalte zugewiesen
  - Wir brauchen für jede Dame nur die Zeilenpositionen bestimmen

OO und UML  
Jie Tang

# Design

- ◆ Idee
  - Wir lassen die Damen selbst die Lösung finden
- ◆ Wir kombinieren die Damen

D1 ← D2 ← D3 ← D4 ← D5 ← D6 ← D7 ← D8

OO und UML  
Jie Tang

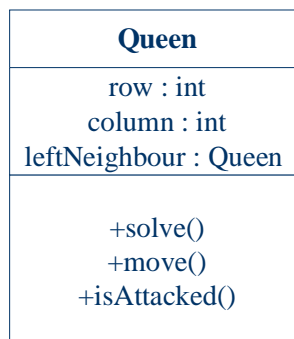
# Design

- ◆ Was bedeutet das für uns
  - Jede Dame kann sagen, ob sie von ihren linken Nachbarn attackiert wird
  - Wir brauchen nur diese Abfragevorgänge iterieren, bis die rechteste Dame nicht mehr von anderen Damen attackiert wird.
  - Wenn wir Glück haben, dann ist das Problem gelöst!

OO und UML  
Jie Tang

# Design

- ◆ Das Klassendiagramm



OO und UML  
Jie Tang

# Implementierung

- ◆ <snippet>

```
public Queen(int column, Queen leftNeighbour) {
    this.column = column; this.leftNeighbour = leftNeighbour; this.row = 1;
}
public void solve() {
    while( neighbor != null && isAttacked() ) {
        if( ! advance() ) return false;
    }
    return true;
}
public static void main(String[] args) {
    Queen lastQueen = null;
    for(int i=1;i<9;i++) {
        lastQueen = new Queen(i,lastQueen) ; lastQueen.solve();
    }
}
```
- ◆ </snippet>

OO und UML  
Jie Tang

## Fallstudie

- ◆ Zusammenfassung
  - Keine Datenstruktur, um die Position darzustellen
  - Kein grosser Kontrollblock, um die Damen zu bewegen
  - Stattdessen
    - Durch die Objektkommunikation versuchen die Damen selbst das Problem zu lösen!
    - Dezentralisier Kontrollblock
    - Wir modellieren die Welt in einer anderen Weise

OO und UML  
Jie Tang

## Vergleich mit E-R Modell

- ◆ Modellierung allgemein
  - Aufbauen einer Abstraktion für ein System, die sich nur auf interessante Aspekte fokussiert und irrelevante Details ignoriert.
  - Ein paar Bemerkungen
    - Modellierung ist ein Mittel zur Komplexitätsminderung.
    - Das Ziel von Modellierung ist Aufbauen der Modelle, die verständlich sind. In der Realität ist das oft nicht möglich. Deswegen ist bei der Modellierung divide and conquer notwendig (Abstraktionsstufen)

OO und UML  
Jie Tang

## Vergleich mit E-R Modell

- ◆ Gleiche Weltanschauung
  - Entity Sets vs. Klasse
  - Statische Relation vs. Assoziation und Kommunikation
- ◆ Hierarchisches System
  - „is a“ vs Vererbung
  - Ist reale Welt wirklich hierarchisch?
  - Eine subtile Unterscheidung

OO und UML  
Jie Tang

## Vergleich mit E-R Modell

- ◆ Wo sind die Attribute?
  - Sowohl Entity Set als auch Relationship hat Attribute
  - Nur Klassen haben Attribute, man kann aber explizit eine Assoziationsklasse einführen
- ◆ Die Daten-Identifizierung
  - Primary key (PK) vs. Objekt Identität
- ◆ Der Anwendungsbereich
  - Datenmodell vs. Allgemeines Systemmodell

OO und UML  
Jie Tang

## Vergleich mit E-R Modell

- ◆ Zusammenfassung
  - OO betrachte die statischen und dynamischen Aspekte des Systems, E-R nur die statischen
  - E-R hat ein wohldefiniertes mathematisches Modell. Leider hat OO noch keins
  - OO Modellierung ist viel komplizierter
  - OO Modellierung kann in machen Fällen das E-R Modell ersetzen
    - Objektorientierte Datenbank vs. Relationale Datenbank

OO und UML  
Jie Tang

## Vergleich mit Strukturierung

- ◆ Methodik allgemein
  - Methodische Vorgehensweise ist auf alle Phasen des Softwareprozesses verteilt. Ist somit auch der Leitfaden des Software Engineerings.
  - 4 Phasen eines einfachen Softwareprozesses
    - Systemspezifikation und Analyse
    - Design
    - Implementation
    - Test und Dokumentation

OO und UML  
Jie Tang

## Vergleich mit Strukturierung

- ◆ Analysephase
  - Spezifikation: What is the Software supposed to do?
  - Analyse: Applikationsdomäne modellieren
    - Strukturierung: Daten und Funktionen seperat modellieren
      - ◆ Daten – E-R Modell etc.
      - ◆ Funktionen – DatenFlussModell etc.
    - OO:
      - ◆ Finden der Objekte
      - ◆ Organisieren der Objekte
      - ◆ Beschreiben wie die Objekte interagieren
      - ◆ Definieren der Operationen von Objekten

OO und UML  
Jie Tang

## Vergleich mit Strukturierung

- ◆ Designphase
  - Design: Bietet die Lösung, die Computer verstehen
    - Strukturierung: das berichtigte Top-Down Design
      - ◆ Funktionen haben zentrale Bedeutung
        - Sie nehmen die Inputdaten,berechnen und erzeugen die OutputDaten
      - ◆ Warum es nicht geht
        - premature ordering durch datafluss approach
    - OO: Umwandeln des AnalyseObjekts zum DesignObjekt
      - ◆ Schnittstellenspezifikation
      - ◆ Der sogenannte Shopping List Approach

OO und UML  
Jie Tang

# Vergleich mit Strukturierung

## ◆ Implementationsphase

- Go to Programming
  - Strukturierung : Das Abschaffen des GoTo Statements
    - ◆ Befürwortet 3 Arten von Programmierung
      - Sequentiell
      - Bedingung - Enumeration
      - Schleife - Induktion
    - ◆ Verbessert die Softwarequalität in gewissen Maßen
  - OO : das OO Konzept entschlossen zu verwenden

OO und UML  
Jie Tang

# Vorteile und Nachteile

## ◆ Vorteile

- OO gibt uns einen neuen Blick auf die Welt, nämlich eine Entität mit Daten und Funktionen zusammen. Das betrachte ich als revolutionär.
- OO erlaubt uns, Ähnlichkeiten zu entdecken.(Vererbung)
- OO gibt uns eine neue Denkweise zur Lösung eines Problems durch den sogenannten shopping list approach.
- OO ermöglicht uns, ein locker gekoppeltes System modulweise aufzubauen (durch das Information Hiding Prinzip).
- OO ermöglicht uns, ein gut erweiterbares System aufzubauen (durch Polymorphie und Dynamische Bindung).
- Ein nahtloser Übergang zwischen allen Phasen ist bei OO möglich

OO und UML  
Jie Tang

# Vorteile und Nachteile

## ◆ Nachteile

- Objekt Orientierung ist gut, aber es ist nicht so gut
  - OO ist schwer beherrschbar
  - Es gibt auch die Risiken, die Techniken zu missbrauchen.
- Kein formales mathematisches Modell vorhanden
  - Was wir haben, sind nur Konzepte, Principles und Richtlinien
  - Reicht das für uns?

OO und UML  
Jie Tang

# UML

## ◆ UML ist eine OO Notation

## ◆ Anwendungsfalldiagramm

- Eine Vorgehensweise des Requirements Engineerings
- Szenario und Anwendungsfall
  - Anwendungsfall besteht aus Szenarien
  - Szenario besteht aus Interaktionsbeschreibung zwischen Benutzer und System (Wieder mal Client und Supplerview)

OO und UML  
Jie Tang

# Anwendungsfalldiagramm

## ◆ Ein bekanntes kleines Beispiel

- Ein Szenario für Online Shopping
  - Der Kunde stöbert durch den Katalog
  - Der Kunde wählt einen Artikel zum Kauf aus
  - Der Kunde geht zur Kasse
  - Der Kunde gibt seine Versand-Information an
  - Der Kunde gibt Kreditkartendaten an
  - Das System autorisiert den Verkauf
  - Das System bestätigt den Verkauf
  - Das System sendet eine Bestätigung per E-Mail an den Kunden

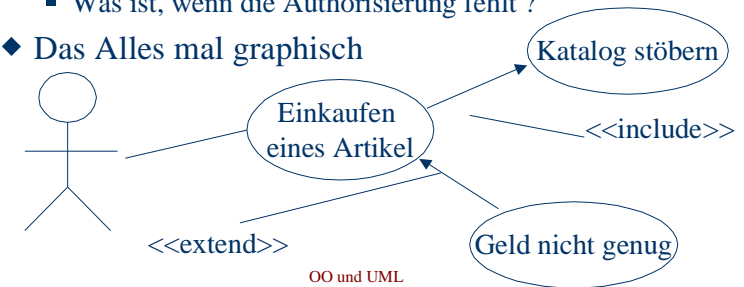
OO und UML  
Jie Tang

# Anwendungsfalldiagramm

## ◆ Es könnte aber noch mehr sein

- Was ist, wenn der Kunde nicht genug Geld hat ?
- Was ist, wenn die Authorisierung fehlt ?

## ◆ Das Alles mal graphisch



OO und UML  
Jie Tang

# Anwendungsfalldiagramm

## ◆ Ein paar Bemerkungen

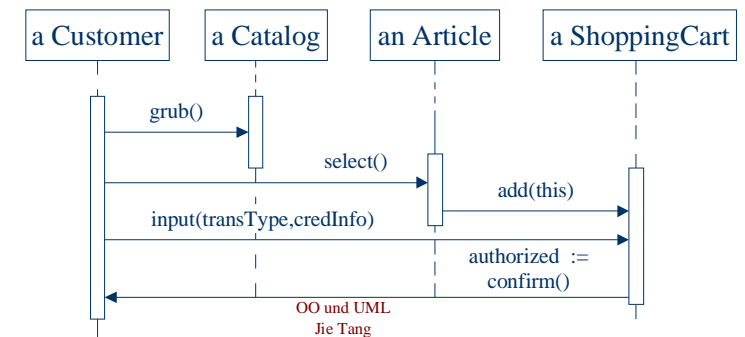
- Grundidee :
  - Korrektes Software basiert nur auf korrekter Spezifikation
  - Anwendungsfall ist informal, aber er
    - ◆ erlaubt bessere Kommunikation mit den Kunden, was ein Schlüssel für korrekte Software ist.
    - ◆ erlaubt schnelles Finden der Objekte, was wichtig für die spätere Phase ist.
  - Graphisch oder textuell?
  - Brauchen wir noch formale Spezifikationen?

OO und UML  
Jie Tang

# Interaktionsdiagramm

## ◆ Software ist dynamisch

## ◆ Das Beispiel nochmal



OO und UML  
Jie Tang

# UML

- ◆ UML hat mehr
  - Zustandsdiagramme
  - Aktivitätsdiagramme
  - Packagediagramme
  - .....

OO und UML  
Jie Tang

# Der Ausblick

- ◆ Is there a Silver Bullet?  
*Building software will always be hard. There is inherently no silver bullet.*  
*F.Brooks in „No Silver Bullet“*
- Weil Die Eigenschaften vom Software von Natur aus ist,
  - complexity
  - conformity
  - changibility
  - invisibility
- ◆ Hat er Recht oder ist er zu pessimistisch?

OO und UML  
Jie Tang

# Der Ausblick

- ◆ Prozess oder Methodologie?
  - Was ist wichtiger für uns?

OO und UML  
Jie Tang

# Der Ausblick

- ◆ formal oder informal  
*„The phrase 'software engineering' was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of **theoretical foundations and practical disciplines**, that are traditional in the established branches of engineering.“*
- Inwieweit ist die Theorie für Software Engineering interessant?

OO und UML  
Jie Tang