Softwaretechnologie
Ankündigungen

Prof. Dr. rer. nat. Uwe Aßmann
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
03.04.17
editiert durch Dr. Birgit Demuth

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Professur Softwaretechnologie

- Vorlesungen:
  - Prof. Dr. Uwe Aßmann, Nöthnitzer Str. 46, 2. OG, Raum 2087
  - Katrin Heber, Sekretärin. 0351 463 38 463
  - Sprechstunde Do, 11:00-13:00. Bitte bei Frau Heber anmelden.
  - Email katrin.heber@tu-dresden.de
- Übungsleitung:
  - Dr. Birgit Demuth, Nöthnitzer Str. 46, 2. OG, Raum 2085
- Wichtigste Informationsquelle:
  - https://tu-dresden.de/ing/informatik/smt/st

    Studium > Lehrveranstaltungen > Softwaretechnologie

Vorlesung und Übungen

- **Vorlesung "Softwaretechnologie":** Konzepte, Überblickswissen zu:
  - Objektorientiertes Programmieren (OOP), aber keine vollständige Einführung in Java
  - Objektorientierter Modellierung (OOM)
    - Objektorientierte Anforderungsanalyse (OOA) + Objektorientiertes Design (OOD)
  - **Achtung:** Folien erscheinen sukzessive. Zur Vorlesungsvorbereitung können auch die von letztem Jahr benutzt werden (normalerweise noch verlinkt)
- **Hörsaalübung "Softwaretechnologie" (OOSE):** Fr, 13:00, HSZ/03
  - "Vorrechnen" von Aufgaben, Vorbereitung Softwarepraktikum im WS
- **Übungen "Softwaretechnologie":**
  - Praktische Anwendung von Modellierungstechniken und Java
  - Grundlage für Praktikum "Softwaretechnologie" im 3. Semester
  - Achtung: Ohne regelmässigen Besuch der Übungen ist der Erfolg in Klausur und Praktikum unwahrscheinlich!
- **Prüfung:** Klausur (120 Minuten) zu Semesterende (Prüfung für INF, MINF, WINF, IST, Nebenfach Informatik)

Voraussetzungen für das Softwarepraktikums

- **Die Kenntnisse, die hier erworben werden, sind, siehe Modulhandbuch, Voraussetzung zur Teilnahme am Praktikum "Softwaretechnologie-Projekt" im 3. Semester.**
  - Die erfolgreiche Teilnahme am Praktikum ohne die vollen Kenntnisse von Softwaretechnologie ist sehr unwahrscheinlich, da ein kompletter, praktischer, anspruchsvoller Softwareentwicklungsprozess in der Gruppe durchgeführt wird
  - Ein Teilnehmer mit unzureichenden Kenntnissen in Java oder UML schädigt seine Gruppe durch mangelnde Leistungen
  - Muss ein Teilnehmer aus dem Gruppenpraktikum wegen mangelnder Leistungen ausscheiden, schädigt er seine Gruppe
- **Vorsicht:** im Praktikum scheidet man aus, wenn man die Meilensteine nicht absolvieren kann.
- **Vorsicht:** Das Praktikum kann nur im Wintersemester durchgeführt und absolviert werden!

Verhältnis von ST-Vorlesung und dem Praktikum im Wintersemester

- ST-Vorlesung gibt einen Überblick, bereitet aber nicht speziell für das Praktikum vor
  - Das Praktikum enthält einen kompletten Durchgang durch einen Entwicklungszyklus
  - Semi-realistisch bis realistisch (auch industrielle Kunden)
- Es lohnt, beides intensiv zu betreiben. **Programmieren heißt Realisieren**
  - Wer sich das Programmieren sparen will, wird große Lücken in seiner beruflichen Praxis haben und seine Ideen nicht wirklich realisieren können
  - es bei Bewerbungen schwer haben, denn Programmierkenntnisse werden vorausgesetzt und Realisierer werden gesucht
- Wer aber mitprogrammiert, hat viel Gewinn
- **Parallel: Lehrveranstaltung "Programmierung"**
  - Programmieren *innerhalb von Operationen (Methoden)*
  - Hier: programmieren *außerhalb von Operationen (Methoden)*

Hypothese von Robert France (2012)

A good modeler is a good programmer,
a good programmer is not always a good modeler!

Modellierung erfordert Kenntnisse und Erfahrungen in der

- Programmierung
- Abstraktion

Abstraktionsfähigkeiten verbessern die SE-Fähigkeiten!

Programme von Programmierern mit guten Abstraktionsfähigkeiten sind von signifikant besserer Qualität!

Übungen

- Ab erster Woche, also ab HEUTE!
- Bitte dringend in jExam in Übungsgruppen eintragen!
- Übungswoche läuft jeweils von Mo bis Fr (in Synchronisation mit der Vorlesung)
- An Feiertagen fallen die Übungen aus – deshalb bitte in diesen Wochen andere Übungen besuchen!
- Wir bieten zusätzlich zu den Übungen jeden Freitag, beginnend ab dem 7.4.17, 3. DS in APB E046 einen **Java-Lernraum** an.

Klausurstatistik

- Beispiel aus dem Studienjahr 2016/17
  - SS: 72% bestanden (das war ein sehr guter Jahrgang)
  - WS (Wiederholungsklausur): 22% bestanden
- Hauptproblem: Viele Studenten können nicht programmieren. Gängige Vorurteile:
  - *"Ich bin Medieninformatiker – ich brauche nicht zu programmieren"*
    - Fehler: die meisten Medienanwendungen (Websites, Spiele, Informationssysteme, Apps) sind komplexe Programme
  - *"Ich werde Softwarearchitekt oder Manager – ich brauche nicht programmieren"*
    - Fehler: Architekten, die nicht mauern können, taugen nichts
    - [Beispiel: Microsoft bestellt keinen zum Manager, der nicht die technischen Vorkenntnisse mitbringt]

Es sind substantielle Java-Programmierkenntnisse nötig, um die Klausur zu bestehen.

Selbsttests mit INLOOP

- Wir empfehlen die Arbeit mit dem **INLOOP**-Lernsystem
  - INLOOP: INteractive Learning-center for Object-Oriented Programming
  - Webbasiertes Selbstlern-System,
  - in das Java-Programme eingetippt werden können
  - das Stil und Übersetzbarkeit prüft
  - und automatisch Tests mit Testdatensätzen ausführt
- Frühes Feedback über Ihre Programmierfähigkeiten möglich!
  - Die Erfahrung der letzten Jahre zeigt, dass die fleissige Benutzung des Praktomaten (ab SS 2016 INLOOP) das Bestehen der Klausur erleichtert.
  - INLOOP ist eine Chance für Sie, nutzen Sie sie!
- Bei Problemen bitte über “Auditorium” melden

https://inloop.inf.tu-dresden.de/

Ziel: Die Universität bildet Problemlöser aus

- Die Universität ist keine Schule, sondern eine **Bildungsanstalt**:
  - Sie setzt *selbständige Aktivität* voraus und lehrt *Problemlösen*
    - Probleme von Menschen erkennen und präzise definieren
    - selbstständig Wege zur Lösung eines Problems finden
  - Selbstständiges Lernen wird gelehrt
    - kein Standardstoff: Sie bekommen kein Buch vorgelesen, und das war's
    - selbstständige Literaturerarbeitung von den Folien aus
- Sie will *Lernliebhaber* und *Literaturfresser* ausbilden
  - Beachten Sie die Lese-Anweisungen, die angegeben werden. Es werden pro Woche 2-4 Kapitel zu lesen sein
  - Steigern Sie also Ihr persönliches Lesetempo
  - Leseleistung: Im Laufe des Studiums sollten Sie lernen, 8 Stunden am Tag zu lernen

Sehr empfohlen für die Technik des wiss. Arbeitens im Studium

- **Stickel-Wolf, Wolf. Wissenschaftliches Arbeiten und Lerntechniken. Gabler.** Sehr gutes Überblicksbuch für Anfänger.
- Stary, Kretschmer: Umgang mit wissenschaftlicher Literatur. Cornelsen. Sehr gutes Buch zum Thema “Lesen”.
- **Kurs “Academic Skills in Computer Science (ASiCS)”,** mit Teil **“Vorbereitung von Abschlussarbeiten/Forschungskolleg Softwaretechnologie”**
  - Sommersemester
  - http://st.inf.tu-dresden.de/teaching/asics
  - Dienstag, 13:00-15:30, APB/E010
  - Donnerstag, 16:40-18:10, APB/E001

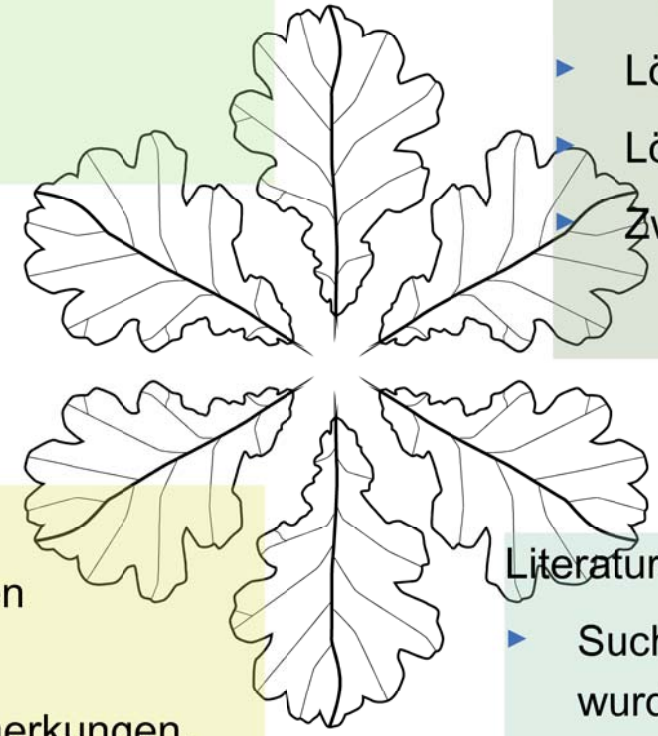
Wie man die Lehrveranstaltung erfolgreich absolviert

Starte mit der Vorlesung

- Höre einfach zu.
- Schreibe auf einem leeren Blatt mit, um das Gehörte in eigenen Worten auszudrücken.
- Zeichne Mindmaps und concept maps
- Falls du dich nicht recht konzentrieren kannst, versuche, auf ausgedruckten Folien Anmerkungen zu machen.

Während des Semesters:

- Erstes Lesen (nur das nötigste)
  - Beantworte Fragen, soweit als möglich
- Rede mit FreundIn
  - Diskutiere Fragen.
- Löse alle Übungsaufgaben
- Löse die INLOOP-Aufgaben
- Zweites Lesen, auf Klausur vorbereitend (erschöpfendes Lesen)



Zuhause nach der Vorlesung

- Gleiche deine Notizen mit den ausgedruckten Folien ab.
- Erweitere die Folien um Anmerkungen.
- Schreibe eine Liste von Fragen auf (wiki, blog, Papier)

Literaturarbeit (am Freitag)

- Suche die Buchkapitel, die empfohlen wurden
- Versuche herauszufinden, was aus der Vorlesung im Buch behandelt wird und was nicht (selektives Lesen von Kapiteln).

Anleitung zum Unglücklichsein

- Besuche Übung nur unregelmässig
- Surfe während Vorlesung
- Probiere Java-System erst im Juni aus
- Ignoriere INLOOP
- Leihe kein Buch aus, lese nichts
- Konzentriere dich auf andere Kurse, die schwerer erscheinen
- Warte mit Lernen bis 2 Wochen vor der Klausur (ST ist ja so einfach...)
  - Achtung: es gibt nur zwei Wiederholungsklausuren (sächs. Hochschulgesetz)
- Verschiebe die Klausur auf WS

Softwaretechnologie
Ziele und Inhalt

Warum ist Softwaretechnologie wichtiger als andere Technologien?

- Softwaretechnologie ist eine Schlüsselindustrie, da eine *Rationalisierungsindustrie*
  - Die Wohlfahrt eines Landes hängt von der Produktivität ab
  - Nach wie vor entstehen völlig neue Anwendungen in unvorhergesehenen Märkten
    - Google, Google Earth, Video Google
    - Ebay, Amazon
    - Bioinformatik, Bauinformatik
    - Maschineninformatik (Virtual Engineering)
    - Digital Pen and Paper

Konsumgüter

Investitionsgüter

Rationalisierungs-industrie

Software ist die größte gesellschaftsverändernde Kraft heute. (Anonymous)

Software is eating the world. (Marc Andreesen)
http://online.wsj.com/news/articles/SB10001424053111903480904576512250915629460

Warum sind *gute* Softwaretechnologen so wichtig?

- Als Rationalisierungsindustrie ist die IT besonders den Schweinezyklen ausgesetzt:
  - Tal 1993/94, Boom 1997-2000, Tal 2001-03, Boom 2007-heute
  - Viele Firmen in DD suchen momentan gute Softwareingenieure!
- Einstiegsgehalt pro Jahr brutto [Quelle IX 1/2005]
  - Obere 10%: 50592 Euro
  - Median: 48629 Euro
  - untere 10%: 42900 Euro
  - Projektleiter: 80000 Euro
- Arbeitsplätze wird es auf lange Sicht in Europa hauptsächlich für den Software-Architekten und Projektleiter geben
  - Programmieren, Testen, ... wird nach Indien oder China ausgelagert
  - Wollen Sie mit 45 arbeitslos sein?
- Daher muss der Software-Werker ein guter Softwaretechnologe werden, dessen Produktivität höher liegt als die der Konkurrenz

Fähigkeiten des *guten* Softwareingenieurs

- Gute Softwareingenieure **wissen, wie man lernt** (lernen zu lernen)
  - Und das lebenslang
  - Gute Softwareingenieure kennen ihre Lern-Grenzen, -Stärken und Schwächen:
  - Was kann ich wie schnell lernen? [Komplexprüfungen]
  - Wie gut kann ich schätzen?
  - Wie gut kann ich in Abstraktionen denken?

- Gute Softwareingenieure **gewinnen Erfahrung**
  - Lerne jedes Jahr eine neue Modellier- und Programmiersprache
  - Lerne Projekte kennen (Prozess- und Produktmanagement)
  - Lerne so viele Ideen kennen als möglich

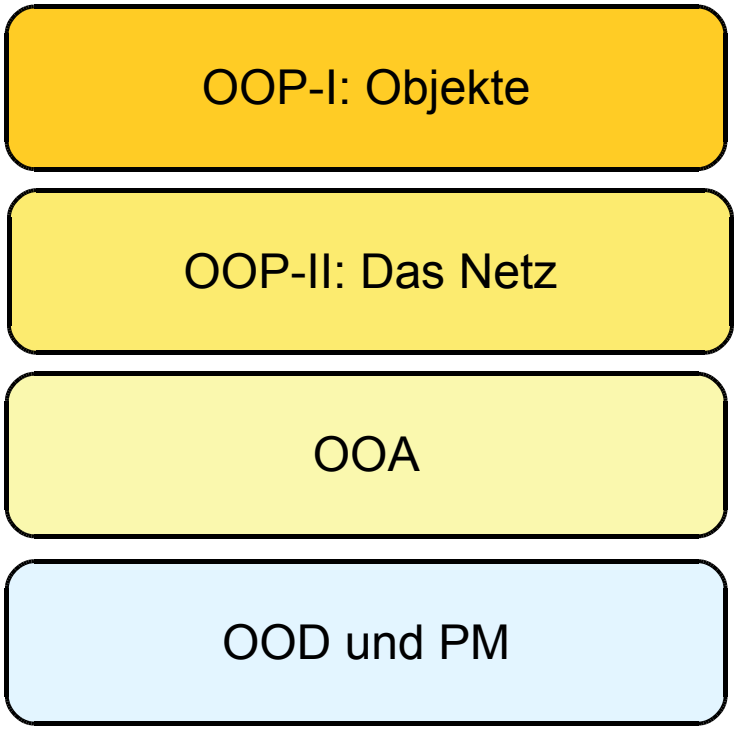
- Gute Softwareingenieure sind **teamfähig**
  - Die meiste Software wird in Teams erstellt
  - Daher wird das Softwaretechologie-Projekt in Teams erstellt

Zwei Gruppen von Kenntnissen

- "Software Engineering” beinhaltet Wissen über:
  - Softwaretechnologie (Software-Techniken)
    - Systemanalyse
    - Systementwurf
    - Systemimplementierung
    - Systemwartung
  - Software-Prozesse
    - Entwicklungszyklus
    - Lebenszyklen
    - Projektmanagement
    - Konfigurationsmanagement
    - Qualitätsmanagement

Phasen und Meilensteine der Vorlesung

- **Objektorientieres Programmieren (OOP)**
- Teil I: Java I – Objekte und Klassen
  - Grundlegende Kenntnisse in Java und jUML
  - Objekte, Klassen, Vererbung, Polymorphie, CRC-Karten
  - Java starten, APIs lesen können, Tests durchführen können
- Teil II: Java II – Das Objektnetz
  - Generics, Collections, GUI
  - Entwurfsmuster, Frameworks
- **Objektorientieres Modellieren (OOM)**
- Teil III: **Objektorientierte Analyse (OOA)**
  - Balzert-Methodik, UML
  - Dynamische Modellierung mit Zustandsmaschinen
- Teil IV: **Objektorientiertes Design (OOD)** und **Projektmanagement (PM)**
  - Software-Architektur
  - Projektmanagement



Softwaretechnologie
Literatur

Das Vorlesungsbuch von Pearson

- Das Anschaffen von Büchern lohnt sich für die Softwaretechnik, weil
  - das Gebiet sehr breit ist und man immer auf Bücher als Nachschlagewerke zurückgreifen muss. Das Lernen von Folien alleine genügt nicht

- Softwaretechnologie für Einsteiger. Vorlesungsunterlage für die Veranstaltungen an der TU Dresden. Pearson Studium, 2014.
  - ausleihbar in der Lehrbuchsammlung sowie Präsenz-Exemplar im DrePunct. Jeweils unter ST 230 Z96 S68(2)
  - Erhältlich bei **Thalia** in Dresden
- Enthält ausgewählte Kapitel aus:
  - UML: Harald Störrle. UML für Studenten. Pearson 2005. Kompakte Einführung in UML 2.0.
  - Softwaretechnologie allgemein: W. Zuser, T. Grechenig, M. Köhle. Software Engineering mit UML und dem Unified Process. Pearson.
  - Bernd Brügge, Alan H. Dutoit. Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java. Pearson Studium/Prentice Hall.

Weiterführende Literatur

- Webseite der Lehrveranstaltung Softwaretechnologie unter
  - Literatur
  - Web-Links (Online-Bücher)