Softwaretechnologie
0. Ankündigungen

Prof. Dr. rer. nat. Uwe Aßmann
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
Version 2020-1.1, 25.06.20

Zugehörig zu Modulen
INF-B-310, INF-D-240
Kenntnisse sind Voraussetzung zur Teilnahme für Modul INF-B-320

2.4.20
Neue Hinweise auf elektronisch verfügbare Bücher

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Professur Softwaretechnologie

- Vorlesungen:
  - Prof. Dr. Uwe Aßmann, Nöthnitzer Str. 46, 2. OG, West, Raum 2087
  - Sekretariat mometan nicht besetzt
  - Sprechstunde Do, 11:00-13:00.
  - Kanal/URL: https://st-lab-ci.inf.tu-dresden.de/UASprechstunde (vorläufig)
  - Bitte anmelden durch email softwaretechnologie@tu-dresden.de
- Übungsleitung:
  - Dr. Birgit Demuth, Nöthnitzer Str. 46, 2. OG, Raum 2085, birgit.demuth@tu-dresden.de
- Wichtigste Informationsquelle:
  - https://tu-dresden.de/ing/informatik/smt/st

    Studium > Lehrveranstaltungen > Softwaretechnologie
  - Kurs in opal.tu-dresden.de im Aufbau

Vorlesung und Übungen

https://tu-dresden.de/ing/informatik/smt/st/studium/lehrveranstaltungen?leaf=2&lang=de&subject=414&embedding_id=47eddfa7c5a54ed5be49042aff35a31b

- **Vorlesung "Softwaretechnologie":**
  - Objektorientiertes Programmieren (OOP)
    - “außerhalb von Prozeduren”, also fortgeschrittenes Java
  - Objektorientierter Softwareprozess
    - Objektorientierte 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)
- **Übungen "Softwaretechnologie": (via Opal)**
  - Praktische Anwendung von Modellierungstechniken, UML 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

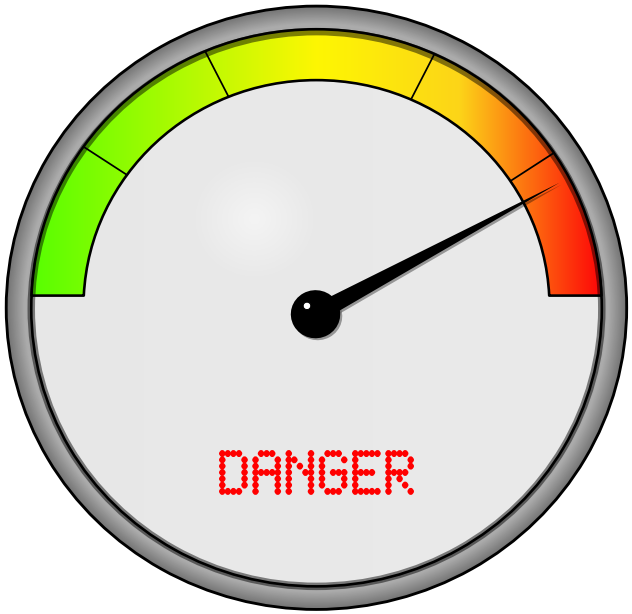
- **Das Praktikum "Softwaretechnologie-Projekt" im 3. Semester ist ein wesentlicher Bestandteil der Bachelor und Diplom-Ausbildung**
  - Es wird ein kompletter, praktischer, anspruchsvoller Softwareentwicklungsprozess in der Gruppe durchgeführt
- **Die Kenntnisse, die hier erworben werden, sind, siehe Modulhandbuch, Voraussetzung zur Teilnahme am Praktikum "Softwaretechnologie-Projekt"**
  - Die erfolgreiche Teilnahme am Praktikum ohne die vollen Kenntnisse von Softwaretechnologie ist sehr unwahrscheinlich
  - 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

- Zur Vorbereitung für das Praktikum gibt die ST-Vorlesung einen Überblick über das Gebiet Softwaretechnologie
  - 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)*

Die BaFöG-Falle im 5. Semester

- **Beachte**: Wer im 3. Semester das Praktikum INF-B-320 nicht erfolgreich abschließen kann, kann die 6 Leistungspunkte nicht beim BaFöG am Ende des 4. Semesters geltend machen.
- Ein Bachelor-Student muss am Ende des 4. Semester erstmals seinen Studienfortschritt dokumentieren, um weiterhin BaFöG zu erhalten.
- Nachzuweisen sind 100 von 120 LP (für 4 Semester).
- **Achtung**: Wird das Praktikum nicht im 3. Semester bestanden, kann es erst im 5. wiederholt werden



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 Opal 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 einen **Java-Lernraum** an. Nähere Informationen folgen auf Webseite.

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 werde 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

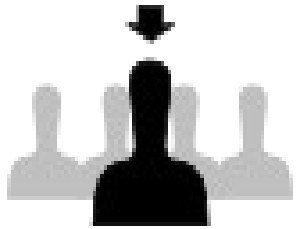
https://inloop.inf.tu-dresden.de/

Wie man erfolgreich studiert

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Ziel: Die Universität bildet Problemlöser aus

- Programmieren ist Problemlösen!
- Die Universität ist keine Schule, sondern eine **Bildungsanstalt**, die Problemlöser ausbildet:
  - Sie setzt *selbständige Aktivität* voraus und lehrt:
    - Probleme von Menschen erkennen und präzise definieren
    - selbstständig Wege zur Lösung eines Problems finden
  - Dazu lehrt sie *selbstständiges Lernen*
    - kein Standardstoff: Sie bekommen kein Buch vorgelesen, und das war's
    - selbstständige Literaturerarbeitung von den Folien aus
- Sie will selbständige *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.

- **Kurs "Academic Skills in Software Engineering (ASiSE)",** mit Teil "**Vorbereitung von Abschlussarbeiten/Forschungskolleg Softwaretechnologie"**
  - Sommersemester
  - https://tu-dresden.de/ing/informatik/smt/st/studium/lehrveranstaltungen?leaf=2&lang=en&subject=418&embedding_id=47eddfa7c5a54ed5be49042aff35a31b
  - Managed in Opal
    https://bildungsportal.sachsen.de/opal/auth/RepositoryEntry/23071653920

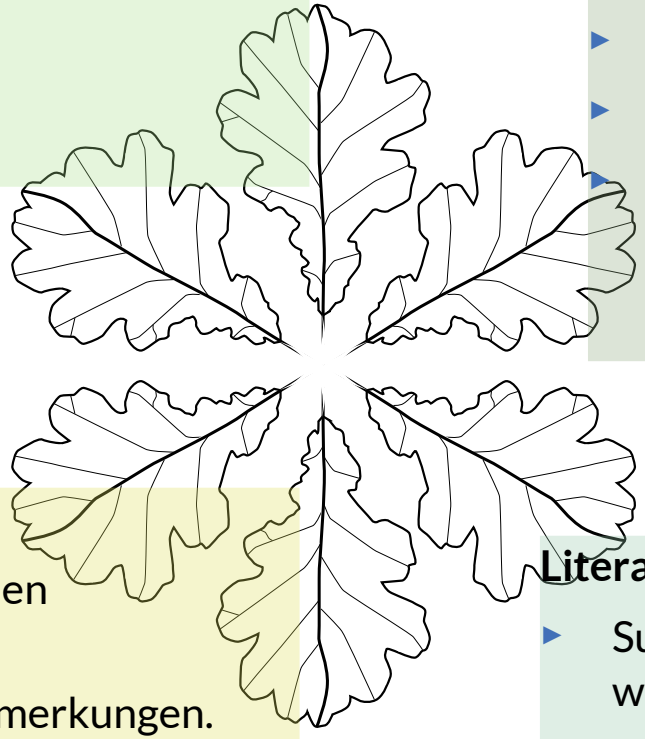
Wie man die Lehrveranstaltung erfolgreich absolviert

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).

Wie man richtig sein Studium plant

- Hauptseite der Fakultät mit Studienordnungen, wichtig:
  - https://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_informatik/smt/st/studium?leaf=2&lang=de&subject=313
- Lade die Studienordnung herunter (Bachelor)
  - https://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_informatik/studium/studiengaenge#s1
- Module Bachelor (ähnlich für MINF-Bachelor und Diplom)
- Studienablaufplan Bachelor

Eine Hilfe: Das Studientagebuch von Webler

- Dr. Wolff-Dietrich Webler. Instrument “Studientagebuch”. Universität Bielefeld. Fakultät Soziologie/ Interdisziplinäres Zentrum für Hochschuldidaktik
  - http://www.hochschulwesen.info/inhalte/anlage-webler-hsw-3-2002.pdf
  - Libreoffice-Calc Sheet auf Vorlesungsseite unter
  - Material/studientagebuch.ods
- Allgemein: Tagebuch als Methode
  - http://methodenpool.uni-koeln.de/download/tagebuchmethode.pdf

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)
    - Videokonferenzssysteme

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], https://www.gehalt.de/
  - Obere 10%: 50592 Euro
  - Median: 48629 Euro
  - untere 10%: 42900 Euro
  - Projektleiter: 80000 Euro
- https://www.heise.de/developer/meldung/Studie-Erfahrene-Entwickler-verdienen-im-Durchschnitt-63-000-Euro-in-Deutschland-4037246.html
- Arbeitsplätze wird es auf lange Sicht in Europa hauptsächlich für den Software-Architekten und Projektleiter geben
  - Programmieren, Testen, ... wird oft nach Indien oder China ausgelagert
  - Daher muss der Software-Werker ein guter Softwaretechnologe werden, dessen Produktivität höher liegt als die der Konkurrenz

Wichtige Fähigkeiten des Softwareingenieurs

- 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?

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

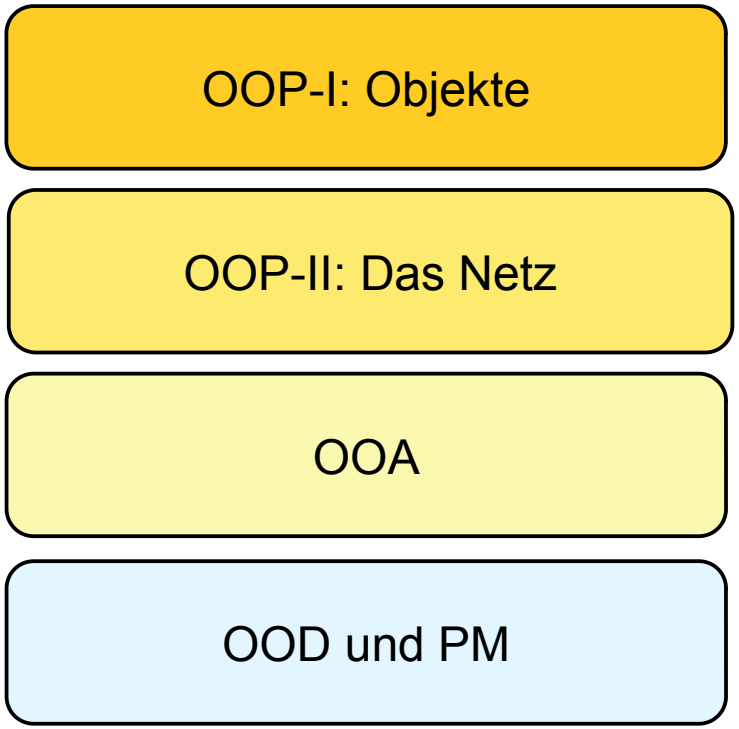
- Softwareingenieure sind **teamfähig**
  - Die meiste Software wird in Teams erstellt
  - Daher wird das Softwaretechologie-Projekt in Teams erstellt

Zwei Gruppen von Kenntnissen

- "Software Engineering" beinhaltet Wissen über:
  - Softwaretechnologie (Software-Techniken)
    - Systemanalyse -> Anforderungen
    - Systementwurf
    - Systemimplementierung
    - Systemtest
    - Systemwartung
  - Software-Entwicklungsprozesse
    - Entwicklungszyklus
    - Teamfähigkeit
    - 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
  - Kanäle
  - 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

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

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)
  - https://katalog.slub-dresden.de/id/0-1669474089/#detail
  - 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.

Softwaretechnologie - Nur Mut!

Alpenüberquerung von Anna Schaffelhuber

- Paralympics Siegerin im Monoski
  - 5malig in Sotschi 2014
  - 2malig in Pjöngchang 2015
  - https://de.wikipedia.org/wiki/Anna_Schaffelhuber
- https://www.youtube.com/watch?v=7uvDVvcZoI0
- https://www.youtube.com/watch?v=dxEn1d0Tg60
- Grenzenlos - Alpenüberquerung mit dem Fahrrad 2017 mit Paralympics-Star Anna Schaffelhuber und Peter Schlickenrieder (Naturbahnrodel-Silbermedaillengewinner)
  - Interview mit Schlickenrieder
    https://www.youtube.com/watch?v=DczADz4amtw
  - https://www.peter-schlickenrieder.de/vortraege/
- https://www.youtube.com/watch?v=R2Kpw_C7EOQ

Weiterführende Literatur

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

Weiterführende Literatur zum Java-Programmieren

- Java-Bücher:
  - Helmut Balzert. Objektorientierte Programmierung mit Java 5. Elsevier,. www.w3l.de
  - D. Ratz et al: Grundkurs Programmieren in Java. Hanser-Verlag, 2006
    - Band 1: Der Einstieg in die Programmierung und Objektorientierung,
    - Band 2: Einführung in die Programmierung kommerzieller Systeme.
  - C. Heinisch, F. Müller, J. Goll: Java als erste Programmiersprache. Teubner.
  - D. Ratz, J. Scheffler, D. Seese, J. Wiesenberger. Grundkurs Programmieren in Java. Hanser-Verlag
  - Hans-Peter Habelitz. Programmieren lernen mit Java: Aktuell zu Java 7 - Keine Vorkenntnisse erforderlich. Galileo Computing
- Wenn Sie noch mehr programmieren können/möchten:
  - eBook: Arnold V. Willemer. Java : Der Sprachkurs Für Einsteiger und Individualisten. Wiley
    - https://ebookcentral.proquest.com/lib/slub/detail.action?docID=1221172&query=java#goto_toc
  - Imperatives Programmieren, Rekursion:
    - D. Boles. Programmieren spielend gelernt mit dem Java-Hamster-Modell. Teubner.
  - Wenn Sie schon imperativ programmieren können:
    - D. Boles, C. Boles: Objekt-orientierte Programmierung spielend gelernt mit dem dem Java-Hamster-Modell. Teubner.

Weitere Literatur

- Eclipse Intro: http://www.eclipse.org/whitepapers/eclipse-overview.pdf
- Andrew Hunt, David Thomas. The pragmatic programmer. Addison-Wesley
  - Deutsch: Der Pragmatische Programmierer. Hanser-Verlag.
  - Sehr schönes Buch mit “Gesetzen des Programmierens”.
- Achtung, neuer e-Book-Service unserer Bibliothek SLUB:
  - http://www.dbod.de/db/start.php?database=ebl_ebl (DBoD)
  - https://www.slub-dresden.de/recherche/datenbanken/erweitertes-angebot-an-e-medien-waehrend-covid-19/
    (Alle)
  - http://rzblx10.uni-regensburg.de/dbinfo/warpto.php?bib_id=slub&color=32&titel_id=10762&url=http%3A%2F%2Fwww.dbod.de%2Fdb%2Fstart.php%3Fdatabase%3Debl_ebl
    (ProQuest)
- Marc Andreesen. Software is eating the world. Wall Street Journal.
  http://online.wsj.com/news/articles/SB10001424053111903480904576512250915629460

Weiterführende eBooks zu UML und OO-Modellierung

- eBook: Martina Seidl, Marion Brandsteidl, Christian Huemer, and Gerti Kappel. UML @ Classroom : Eine Einführung in die objektorientierte Modellierung
  - https://ebookcentral.proquest.com/lib/slub/detail.action?docID=1049728&query=object-oriented+Programming
- eBook: Bernd Oestereich, Christian Weiss, Claudia Schröder, and Tim Weilkiens. Objektorientierte Geschäftsprozessmodellierung mit der UML. dpunkt.
  - https://ebookcentral.proquest.com/lib/slub/detail.action?docID=1095655&query=UML
- Online-Documentation bei der OMG (kostenlos) www.omg.org/uml

Weiterführende Literatur zu UML und OO-Modellierung

- Bernd Oestereich. Die UML-Kurzreferenz 2.3 für die Praxis. 5., überarbeitete Auflage 2009. I, 186 S., broschiert, Oldenbourg, ISBN 978-3-486-59051-7
- Martin Hitz, Gerti Kappel: UML@Work, dpunkt-Verlag
- G. Booch, J. Rumbaugh, I. Jacobson: The Unified Modeling Language User Guide, Addison-Wesley 1999.
- Bernhard Lahres, Gregor Rayman. Praxisbuch Objektorientierung- Von den Grundlagen zur Umsetzung. Galileo Computing. Schönes Buch über OO, nicht auf Java fixiert, breit angelegt.
- Bernd Oestereich: Objektorientierte Softwareentwicklung mit der Unified Modeling Language, Oldenbourg-Verlag
- Ken Lunn. Software development with UML. Palgrave-Macmillan. Viele realistische Fallstudien
- Ebook: Dan Pilone, Neil Pitman. UML 2.0 in a nutshell. http://it-ebooks.info/book/154/. O'Reilly Media, ISBN: 978-0-596-00795-9, 2005
- Free pdf: Trygve Reenskaug, P. Wold, O.A. Lehne. Working with Objects. The OORam Software Engineering Method. Web edition. http//www.uio.no/~trygver.
  - Reenskaug co-invented the MVC design pattern in 1978.

Weiterführende Literatur zum Gebiet Softwaretechnologie

- Jochen Ludewig and Horst Lichter. Software Engineering: Grundlagen, Menschen, Prozesse, Techniken
  - https://ebookcentral.proquest.com/lib/slub/detail.action?docID=1024264&query=UML
- *Weiterführende Literatur zum Gebiet Softwaretechnologie. Können Sie anschaffen, wenn Sie ST-II hören wollen*
- Bernd Brügge, Alan H. Dutoit. Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java. Pearson Studium/Prentice Hall. (Teile sind im "Softwaretechnologie für Einsteiger" vorhanden)
- Helmut Balzert: Lehrbuch der Software-Technik, 2 Bände, Spektrum Akademischer Verlag 2000 und 1998. Umfassendes Kompendium.
- Ghezzi, Jazayeri, Mandrioli. Fundamentals of Software Engineering. Prentice Hall. Sehr gutes, fundamentales, weiterführendes Buch. Klar. Starke Kost.
- S. Pfleeger: Software Engineering – Theory and Practice. Prentice-Hall. Gutes Buch, breit angelegt.
- Leszek A. Maciaszek. Requirements Analysis and System Design – Developing Information Systems with UML. Addison-Wesley.

Appendix

Literatur zu Bildung

- Geistesgeschichte Deutschlands und Preußens 1700-1930:
  - Herbert Meschkowski. Jeder nach seiner Facon. Berliner Geistesleben 1700-1800. Piper-Verlag.
  - Herbert Meschkowski. Von Humbold bis Einstein. Berlin als Weltzentrum der exakten Wissenschaften. Piper-Verlag. Deckt 1820-1930 ab

Wie erreicht man Schönheit?

- Aaron Rosand spielt die Melodie in Max Bruch's Violinkonziert II, Satz 1, Min. 3
- https://www.youtube.com/watch?v=CVWb-xYiYHE

A violin can sing a melody better than the piano can,
and melody is the soul of music.
[Max Bruch, in Fifield]

Schönheit, Bildung und Meisterschaft

- Christopher Fifield. Max Bruch: His life and works.
  - Alle, die Melodien lieben, sollten dieses Buch lesen, um zu erfahren, wie man Meisterschaft in dem erreichen kann, wofür man Interesse und Leidenschaft hat. Die Geschichte von Max Bruch und seiner Meisterschaft für Melodien ist eng verknüpft mit dem Schicksal der Deutschen im 19. und 20. Jahrhundert, mit der Entwicklung von Kunst und Wissenschaft in der Zeit um den Ersten Weltkrieg. Vielleicht das allergrößtes deutsche Werk für ein Solo-Instrument ist sein 3. Violinkonzert, Beethoven und Beatles hin oder her. Bruch ist ein Genuss und wird noch in 500 Jahren einer der größten deutschen Komponisten sein! …Lesen, hören, bilden, um Sonnenuntergänge intensiver zu sehen und Vögel zwitschern zu hören, …
  - Tipp: von der Oma zu Weihnachten mit dem 3. VK schenken lassen (amazon) und dann abwechselnd mit ihr hören.

Bruch is a music architect. He architects a piece, much in the sense you should be able to do writing, speaking, or programming.
He THINKS about his works. He can create tension over a full violin concert, e.g., VC III or Serenade. Prepare yourself to do the same.

Warum problemlösende Softwareingenieure gut für die Gesellschaft sind..

- [Charles M. Horton. Opportunities in Engineering, 1920, by Harper & Brothers. Free Gutenberg Book. http://www.gutenberg.org/ebooks/24681]
- Lawyers and politicians have successfully dominated our government from its beginning, with a single beautiful exception in George Washington at one end and another admirable exception in Woodrow Wilson at the other. Washington was a civil engineer, and Wilson, while trained as a lawyer, was an educator. In between these two men there may have fallen a scattering of others who were not lawyers or politicians; the writer is not sure. **Of one thing he is sure, however, and that is that engineers in the future will dominate politics to the betterment of the nation as a whole.** For engineers are idealists--otherwise they would never have entered upon an engineering career--and idealism has come, as it were, into its own again. The man of vision of a wholesome aspect, the man who can so completely forget himself in his work of service as to engage in tasks whose merits nobody save himself and those pursuing like tasks can or will understand--which is pre-eminently the engineer--is the one man best fitted to administrate in public affairs. More important still than this statement is the fact that the world at large is beginning to realize the truth of it. Engineers as a body stand poised upon the rim of big things. Nor will they as a body stoop to the petty in politics, once they are fairly well launched in active participation of civic affairs. ..

Warum problemlösende Softwareingenieure gut für die Gesellschaft sind

- Neither their training nor their outlook, based upon their training, will permit it. **For engineers, more than any other group of professional men, are given to "see true." And seeing true, being, as it is, the essence of a full life, is what is needed in our public administrators.**
- **Engineers belong in civic affairs. The world of humanity needs men of their stamp in high places.** Humanity needs men in control of state and national affairs who would hold the interests of humanity sacred. Engineers are such men. Not that engineers more than any other professional men are sprouting wings--not that. But engineers do see things in their true light--cannot see them in any other light than the one imposed by the law of mathematics, which is that two and two make four, never five or three--and this involuntarily would admit of decisions and grant graces from the point of view of absolute truth, which is, of course, the point of view of humanity--the greatest good for the greatest number. With such men occupying high places in the nation's affairs, the world of men and mankind would leap forward ethically and spiritually at a pace in keeping with the pace at which civilization has progressed under the impetus of engineering thought since the days of Watt. Nobody can deny _that_ progress. Nobody could well deny the fact that ethical progress under engineering guidance would be equally great.