Softwaretechnologie Ankündigungen

Prof. Dr. rer. nat. Uwe Aßmann
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
Version 13-0.3, 25.11.13

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

Lehrstuhl 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. Bitte auch über Frau Heber kontakten, da emails an Prof. Aßmann oft nur verzögert beantwortet werden können
- Übungsleitung:
  - Dr. Birgit Demuth, Nöthnitzer Str. 46, 2. OG, Raum 2085
- Wichtigste Informationsquelle:
  - http://st.inf.tu-dresden.de/ -> Teaching -> Softwaretechnologie
  - http://st.inf.tu-dresden.de/teaching/swt

Navigation im Web:
Fakultät Informatik
Institut Software- und Multimediatechnik
Professur Softwaretechnologie
Teaching
Softwaretechnologie

Vorlesung und Übungen

- **Vorlesung "Softwaretechnologie":** Konzepte, Überblickswissen zu:
  - Objektorientiertes Programmieren (OOP, aber *keine vollständige* Einführung in Java)
  - Objektorientierter Modellierung (OOM)
    - Objektorientierte Analyse (OOA) + Objektorientiertes Design (OOD)
- **Übungen "Softwaretechnologie":**
  - Praktische Anwendung von Modellierungstechniken und Java
  - Grundlage für Praktikum “Softwaretechnologie” im 4. Semester
  - Achtung: Ohne regelmässigen Besuch der Übungen ist der Erfolg bei Klausur und Praktikum unwahrscheinlich!
- **Leistungsnachweis:**
  - **Klausur** (120 Minuten) zu Semesterende (**Prüfung** für INF, MEDINF**,** WINF, **Schein** für IST)

Voraussetzungen des Praktikums INF-B-320

- Die Kenntnisse, die in INF-B-310 erworben werden, sind, siehe Modulhandbuch, Voraussetzung zur Teilnahme am Praktikum “Softwaretechnologie” INF-B-320 im 3. Semester (Bachelor und Diplom INF, Bachelor MEDINF)
  - Die erfolgreiche Teilnahme an INF-B-320 ohne die vollen Kenntnisse von INF-B-310 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 INF-B-320 wegen mangelnder Leistungen ausscheiden, schädigt er seine Gruppe
- **Vorsicht:** Das Praktikum kann nicht jedes Semester durchgeführt und absolviert werden!
  - Es sind nicht genügend Ressourcen vorhanden, das Praktikum semesterlich durchzuführen
  - Die Klausur Softwaretechnologie kann nach jedem Semester geschrieben werden; das Praktikum nicht!

Die BaFöG-Falle

- **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

Übungen

- Ab erster Woche
- Bitte dringend noch in JExam in Übungsgruppen eintragen!
- Übungswoche läuft jeweils von Mo bis Fr (in Synchronisation mit der Vorlesung)

Klausurstatistik

- Beispiel aus einem der vorigen Jahre
  - 616 Studenten, 544 nehmen an erster Klausur im WiSe teil
  - 39% bestanden
- Wiederholungsklausur im SoSe
  - 101 Studenten, 1% bestanden
- Hauptproblem: Viele Studenten können nicht mehr programmieren. Gängige Vorurteile:
  - “Ich bin Medieninformatiker – ich brauche nicht programmieren”
    - Fehler: die meisten Medienanwendungen (Websites, alle Spiele) sind komplexe Programme
  - “Ich werde Softwarearchitekt oder Manager – ich brauche nicht programmieren”
    - Fehler: Architekten, die nicht mauern können, taugen nichts
    - [Beispiel: Microsoft bestallt keinen zum Manager, der nicht die technischen Vorkenntnisse mitbringt]

Klausur – Regelung

- Es gibt eine Bestehensregel:
- Die Klausur besitzt 2 Teile, die beide bestanden werden müssen
  - **Teil 1**: Objektorientierte Modellierung mit UML (45 Punkte, 20 Punkte zum Bestehen nötig)
  - **Teil 2**: Objektorientiertes Programmieren mit Java (45 Punkte, 20 Punkte zum Bestehen nötig)
- Daraus folgt, dass Programmierkenntnisse wesentlich zum Bestehen der Klausur sind.

- Achtung: Man verlasse sich nicht auf die Struktur der vergangenen Klausuren. Es wird sicher variiert werden!

Selbsttests mit dem Praktomaten

- Im Laufe des Kurses werden wir Informationen zum **Praktomaten** veröffentlichen
  - Webbasiertes Selbstlern-System,
  - in das Java-Programme eingetippt werden können
  - das Stil und Übersetzbarkeit prüft
  - und automatisch Tests mit Testdatensätzen anwendet
- Frühes Feedback über Ihre Programmierfähigkeiten möglich!
  - Die Erfahrung der letzten Jahre zeigt, dass fleissige Benutzung des Praktomaten das Bestehen der Klausur erleichtert.
  - Der Praktomat ist eine Chance für Sie, nutzen Sie sie!
- IST-ler ohne FRZ-Account bitte bei sebastian.richly@tu-dresden.de melden

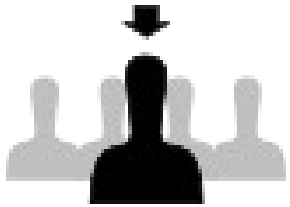
http://praktomat.inf.tu-dresden.de/

Für Wirtschaftsinformatiker

- Anfertigung eines Übungsbeleg
- 3 selbst gewählte Aufgaben aus der Rubrik "Previous Exams" des Praktomaten (http://praktomat.inf.tu-dresden.de/)
  - Das sind Aufgaben aus alten Klausuren, die Ihre Java-Programmierfähigkeiten und Ihr Verständnis für Design Pattern prüfen.
- Achtung: Sie sollten sich an diese Aufgaben erst ab Übung U07 wagen.
- Zuvor empfehlen wir Ihnen, die einfachen Praktomatsaufgaben zu lösen, auf die in den Übungen hingewiesen wird.

Ziel: Die Universität bildet Problemlöser aus

- Die Universität ist keine Schule, sondern eine Bildungsanstalt:
  - Sie setzt *selbständige Aktivität* voraus und will *Problemlöser* ausbilden
    - Sie bekommen kein Buch vorgelesen, und das war's
    - Sie sollen von den Folien aus den Weg in die selbstständige Literaturerarbeitung finden (Problemlösen!)
- Sie will *Lernliebhaber* und *Literaturfresser* ausbilden
    - Beachten Sie trotzdem die Lese-Anweisungen, die angegeben werden. Es werden pro Woche 2-4 Kapitel zu lesen sein!
    - Steigern Sie also Ihr persönliches Lesetempo!
    - Im Laufe des Studiums sollten Sie lernen, 8 Stunden am Tag zu lernen
- .. die die Natur des Fachs meistern können (*Meisterschaft*)
    - Softwaretechnologie ist leider zu umfangreich. Die Vorlesung muss sich auf die wichtigsten Punkte konzentrieren
- .. die selbständig lebenslang lernen können (*Profis*)
    - Sie haben noch keine grösseren Systeme gesehen
    - Sie sollten nicht erwarten, nach dem Kurs ein Experte zu sein
    - Sie müssen selbständig weiterkommen



Blooms Taxonomie des Lernens

- [Wikipedia, Lernziele] Die 6 Stufen im kognitiven Bereich lauten:
- **Lehrlingsschaft**
  - **Kenntnisse / Wissen:** Kenntnisse konkreter Einzelheiten wie Begriffe, Definitionen, Fakten, Daten, Regeln, Gesetzmäßigkeiten, Theorien, Merkmalen, Kriterien, Abläufen; Lernende können Wissen abrufen und wiedergeben.
  - **Verstehen:** Lernende können Sachverhalt mit eigenen Worten erklären oder zusammenfassen; können Beispiele anführen, Zusammenhänge verstehen; können Aufgabenstellungen interpretieren.
- **Gesellenschaft**
  - **Anwenden:** Transfer des Wissens, problemlösend; Lernende können das Gelernte in neuen Situationen anwenden und unaufgefordert Abstraktionen verwenden oder abstrahieren.
  - **Analyse:** Lernende können ein Problem in einzelne Teile zerlegen und so die Struktur des Problems verstehen; sie können Widersprüche aufdecken, Zusammenhänge erkennen und Folgerungen ableiten, und zwischen Fakten und Interpretationen unterscheiden.
  - **Synthese:** Lernende können aus mehreren Elementen eine neue Struktur aufbauen oder eine neue Bedeutung erschaffen, können neue Lösungswege vorschlagen, neue Schemata entwerfen oder begründete Hypothesen entwerfen.
- **Meisterschaft**
  - **Beurteilung:** Lernende können den Wert von Ideen und Materialien beurteilen und können damit Alternativen gegeneinander abwägen, auswählen, Entschlüsse fassen und begründen, und bewusst Wissen zu anderen transferieren, z. B. durch Arbeitspläne.

Sehr empfohlen für die Technik des wiss. Arbeitens

- Stickel-Wolf, Wolf. Wissenschaftliches Arbeiten und Lerntechniken. Gabler. Blau. Sehr gutes Überblicksbuch für Anfänger.
- Stary, Kretschmer: Umgang mit wissenschaftlicher Literatur. Cornelsen. Sehr gutes Buch zum Thema “Lesen”.
- **Kurs “Vorbereitung von Abschlussarbeiten/Forschungskolleg Softwaretechnologie”**,
  - 5. Semester (Sommersemester), Dienstag, 14:50-16:20, Beginn 9.4. E00

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.
  - Falls du dich nicht recht konzentrieren kannst, versuche, auf ausgedruckten Folien Anmerkungen zu machen.
- Zuhause nach der Vorlesung
  - Gleiche deine Notizen mit den ausgedruckten Folien ab.
  - Erweitere die Folien um Anmerkungen.
  - Suche die Buchkapitel, die empfohlen wurden
  - Versuche herauszufinden, was aus der Vorlesung im Buch behandelt wird und was nicht (selektives Lesen von Kapiteln).
  - Schreibe eine Liste von Fragen auf (wiki, blog, Papier)

Während des Semesters:

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

Anleitung zum Unglücklichsein

- Besuche Übung nur unregelmässig
- Surfe während Vorlesung
- Probiere Java-System erst im Juni aus
- Ignoriere den Praktomat
- Leihe kein Buch aus, lese nichts
- Warte mit Lernen bis 2 Wochen vor der Klausur (ST ist ja so einfach...)
  - Achtung: es gibt i.d.R. nur **eine** Wiederholungsklausur (sächs. Hochschulgesetz)
- Verschiebe die Klausur auf WS
  - Teilnahme am Praktikum erst ein Jahr später möglich

Verhältnis von ST-Vorlesung und dem Praktikum im Wintersemester

- ST-Vorlesung gibt einen Überblick, aber bereitet nicht speziell für das Praktikum vor
  - Das Praktikum enthält einen kompletten Durchgang durch einen Entwicklungszyklus
  - Semi-realistisch bis realistisch (oft industrielle Kunden)
- Es lohnt, beides intensiv zu betreiben. Wer sich zum Beispiel um's Programmieren herummogeln will,
  - wird große Lücken in seiner beruflichen Praxis haben
  - und es bei Bewerbungen schwer haben (Programmierkenntnisse werden vorausgesetzt)
- Wer aber mitprogrammiert, hat viel Gewinn

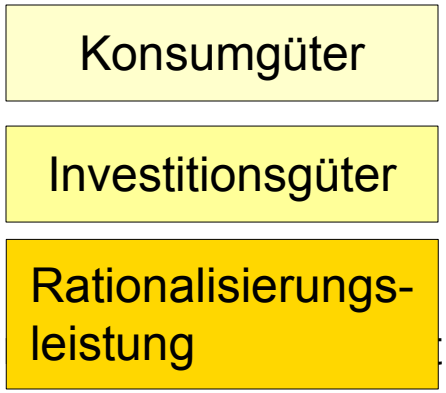
Wie erreicht man Bildung?

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

Softwaretechnologie
Ziele und Inhalt

Warum ist Softwaretechnologie wichtig?

- 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
  - Warum so wenig Europäer aktiv in neuen Anwendungen?
- Als Rationalisierungsindustrie ist die IT besonders den Schwei :
  - Tal 1993/94, Boom 1997-2000, Tal 2001-03, Boom 2007-heute
  - Viele Firmen in DD suchen momentan gute Softwareingenieure!



Warum sind *gute* Softwaretechnologen so wichtig?

- 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**
  - Lernen 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

Zwei Gruppen von Kenntnissen

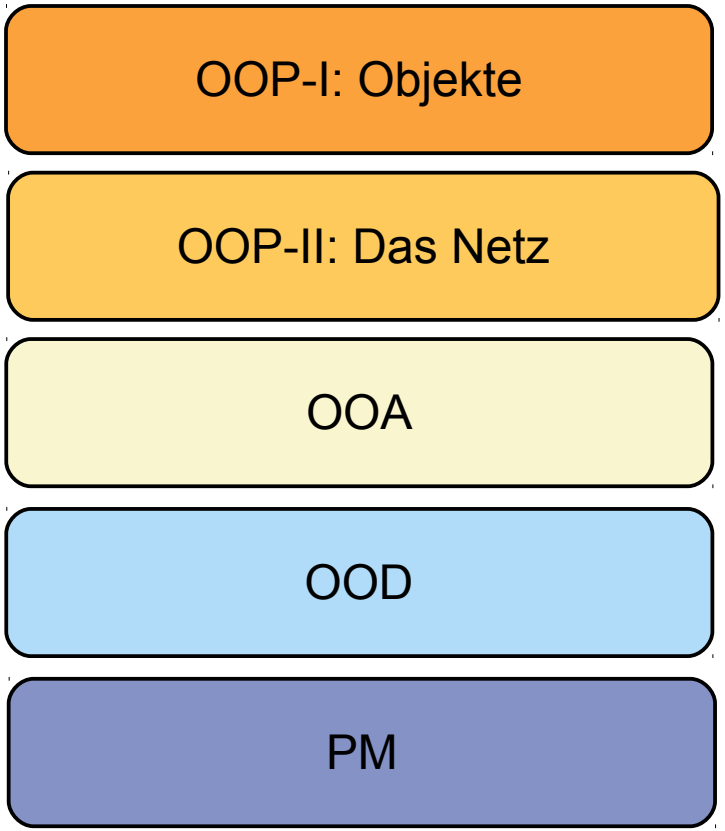
- "Software engineering" beinhaltet Wissen über:
  - Softwaretechnologie (Software-Techniken)
    - Systemanalyse
    - Systementwurf
    - Systemimplementierung
    - Systemwartung
  - Software-Prozesse
    - Entwicklungszyklus
    - Lebenszyklen
    - Projektmanagement
    - Konfigurationsmanagement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

Schwerpunkt der Vorlesung liegt auf Techniken

- **Objektorientierte Programmierung** mit Java
  - Prinzipien der Objektorientierung
  - Bibliotheken und Frameworks
  - Java
    - dominiert die neu gestarteten Softwareprojekte in vielen Anwendungsgebieten
    - ist das Vorbild für aktuelle Entwicklungen in der Industrie (C#, .net)
  - Professionelle arbeitsteilige Herstellung von Software
    - Vorbereitung auf ein umfangreiches Team-Praktikum
- **Objektorientiertes Modellieren** mit der Unified Modeling Language (UML)
  - Objektorientierte **Systemanalyse** (OOA)
  - Objektorientierter **Entwurf** (OOD)
    - Softwarearchitektur
    - Entwurfsmuster
- Vertiefung Prozesswissen in Vorlesung “Softwaremanagement” (Hauptstudium)

Phasen und Meilensteine der Vorlesung

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



Softwaretechnologie
Literatur

Begleitende Literatur

- 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
- Das **Vorlesungsbuch** von Pearson: Softwaretechnologie für Einsteiger. Vorlesungsunterlage für die Veranstaltungen an der TU Dresden. Pearson Studium, 2009. 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. 2004.
- Java
  - Helmut Balzert. Objektorientierte Programmierung mit Java 5. Elsevier, München. 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.

Weiterführende Literatur zum Programmieren

- Stefan Middendorf, Rainer Singer, Jörn Heid: Java – Programmierhandbuch und Referenz für die Java-2-Plattform Standard Edition, 3. Auflage, dpunkt Oktober 2002. Dicke Referenz
- Eclipse Intro: http://www.eclipse.org/whitepapers/eclipse-overview.pdf
- Wenn Sie noch mehr programmieren können/möchten:
  - 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.
  - Insgesamt: C. Heinisch, F. Müller, J. Goll: Java als erste Programmiersprache. Teubner.
- Andrew Hunt, David Thomas. The pragmatic programmer. Addison-Wesley
  - Deutsch: Der Pragmatische Programmierer. Hanser-Verlag.
  - Sehr schönes Buch mit “Gesetzen des Programmierens”.

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
- Online-Documentation bei der OMG (kostenlos) www.omg.org/uml
- Dan Pilone, Neil Pitman. UML 2.0 in a nutshell. Free ebook download http://it-ebooks.info/book/154/. O'Reilly Media, ISBN: 978-0-596-00795-9, 2005
- 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

Weiterführende Literatur zum Gebiet Softwaretechnologie

- *Weiterführende Literatur zum Gebiet Softwaretechnologie. Können Sie anschaffen, wenn Sie ST-II hören wollen*
- 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.
- Bernd Brügge, Alan H. Dutoit. Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java. Pearson Studium/Prentice Hall.
- Leszek A. Maciaszek. Requirements Analysis and System Design – Developing Information Systems with UML. Addison-Wesley.

Appendix

Bildung und Meisterschaft

- Christopher Fifiield. 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.

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