Softwaretechnologie II

Prof. Dr. U. Aßmann
Dr. Sebastian Richly
Technische Universität Dresden
Institut für Software- und Multimediatechnik
Gruppe Softwaretechnologie
http://st.inf.tu-dresden.de
WS 12/13-0.2, 06.10.12

Vorlesungen und Übungen

- **Vorlesung: Mi 14:50 WIL A 120**
  - 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 über Frau Heber kontakten, da emails an Prof. Aßmann oft nur verzögert beantwortet werden können
- **Vorlesung ist empfohlen für Jahr 3 (Bachelor und Diplom)**
  - Es werden wichtige Grundlagen für weitere Kurse eingeführt
- **Wichtigste Informationsquelle:**
  - http://st.inf.tu-dresden.de/teaching/swt2
  - http://st.inf.tu-dresden.de/ -> Teaching -> Softwaretechnologie II
- **Übungsleiter: Dr. Sebastian Richly**
  - Übungen können nur einen kleinen Teil der Vorlesung abdecken
  - Ab Woche 2
  - Semester ist in Komplexe aufgeteilt:
    - Ontologien
    - Anforderungsanalyse: ZOPP, Lasten- und Pflichtenheft
  - Testen Regressiontest
  - Reuseware
  - Model Driven Architecture

Übungsgruppen

- **Teilung der Übungsgruppen in kleine Gruppen á 4-5 Personen**
- **Zumeist 2-3 Woche Zeit zur Bearbeitung eines Komplexes**
- **Lösungen werden ins SVN eingecheckt und dann bewertet**
  - Nacharbeitung möglich
- **Alle Übungskomplexe müssen bearbeitet werden**
  - → Ansonsten Prüfung nur möglich als 2/0/0 Prüfung

Course Structure

Part IV: Selling Software
Business models

Part III: Product Line Engineering
Model-driven architecture
Feature modeling of product lines
Ontologies as constraint checkers

Part II: Design methods
Overview, Comparison of Design Methods with regard to Decomposition focus, Extensibility

Part I: The top-level of the V-model
Requirements analysis
Validation and Software Quality
Model quality: analysis, structuring

Part 0: Introductory Material
Engineering - Petri Nets

Content

- **Part 0 Introduction**
  - What is Engineering?
- **Part I The upper part of the V-Model: Analysis, Quality, Structuring, Validation**
  - Requirements Analysis
  - Testing
  - Reviews and Inspections
- **Part II: Design Methods**
  - Functional design
  - Action-oriented design
  - Transformative design
  - Formal methods
- **Part III: Product Lines**
  - Transformational design and MDA
- **Part IV: Selling Software**

Recommended Literature: Overview Books

- **We recommend one of (reading instructions can be followed in one of them):**
  - Helmut Balzert, Lehrbuch der Softwaretechnik, 2. Auflage. Heidelberg, 2000, ISBN 3-8274-0042-2 (deutsch)
  - Bernd Brügge, Allen H. Dutoit, Objektorientierte Softwaretechnik, Pearson Studium
  - L. A. Maciaszek, B. L. Liong. Practical Software Engineering. A Case Study Approach. Addision-Wesley. Modern book on SE, UML in action in several case studies.
- **Other good books, priority from top to bottom:**
  - Ghezzi, Jazayeri, Mandrioli. Fundamentals of Software Engineering. Prentice Hall. Nice fundamented book. No fuzz, concrete.
  - S. Pfleeger: Software Engineering – Theory and Practice. Prentice-Hall. Good book, not too deep, but broad.
  - Van Vliet: Software Engineering. Wiley.
  - R. Pressman. Software Engineering – A Practitioner's Approach. McGrawHill

Recommended Books on UML – Unified Modeling Language

- **UML is required. It is expected that you learn UML yourself from a good book.**
- **We recommend one of:**
  - Online documentation on www.omg.org/uml
  - H. Störrle. UML für Studenten. Addison-Wesley (cheap, good!).
  - Leszek A. Maciaszek. Requirements Analysis and System Design – Developing Information Systems with UML. Addison-Wesley. Excellent concept book.
  - Object Management Group (OMG). UML - Unified Modeling Language. 2.0.
- **Other excellent books:**
  - Ken Lunn. Software development with UML. Palgrave-Macmillan. Many case realistic studies.

Reference Books

- **R. Thayer, A. McGettrick. Software Engineering: A European Perspective. IEEE Press. Good collection of papers.**
- **M. Dorfman, R. Thayer. Software Engineering. IEEE Press. Good collection of papers.**
- **John McDermid. Software engineer's reference book. Butterworth-Heinemann. ISBN 0-7506-0813-7.**
- **A. Endres, D. Rombach. A Handbook of software and systems engineering - Empirical observations, laws and theories. Addison-Wesley. Very good collection of software laws. Nice!**

- **E. Gamma et. al, Design Patterns, Addison-Wesley, ISBN 0-201-63361-2.**
  - This standard reference book belongs to the bookshelf of every software engineer!
  - Buy this now, if you want to visit "Design Patterns and Frameworks".
- **Others**
  - Rumbaugh et.al. Object-oriented modelling and design. Prentice-Hall.
  - Booch. Object-oriented Analysis and Design. Addison-Wesley.
  - In German: Heide Balzert. Objektorientierten Systemanalyse. Spektrum der Wissenschaft.
  - Prieto-Diaz/Arango, Domain Analysis and Software Systems Modelling, IEEE Computer Society Press tutorial, ISBN 0-8186-8996-X, 1991

- **C. Szyperski: Component Software. Addison-Wesley**
- **K. Czarnecki, U. Eisenecker: Generative Programming. Addison-Wesley**
- **U. Aßmann. Invasive Software Composition. Springer.**

- B. W. Boehm, Software Risk Management, 1989
- F. Brooks, The Mythical Man-Month, Addison-Wesley, 1975
- G. Weinberg, The Psychology of Computer Programming, Computer Science Series, 1971.
- E. Yourdan: The Death March.
- P. Neumann: Computer Risks, Addison-Wesley 1995.
- David Thielen. The 12 simple secrets of Microsoft McGraw-Hill.
- Dana Sobel. Longitude. About John Harrison. Just a good book about an excellent engineer.
- Simon Singh. Fermat's last theorem. Just an excellent book about an excellent mathematician (Wiles) thinking excellently hard.

- J.L. Bentley, Programming Pearls, Addison-Wesley, 2. Auflage 1989, ISBN 0-201-10331-1
- J.L. Bentley, More Programming Pearls, Addison-Wesley, 1988, ISBN 0-201-11889-0
- J.L. Bentley, Writing Efficient Programs, Prentice-Hall, ISBN 0-13-970244-X, 1982

- Uwe Viggenschow. Objektorientiertes Testen und Testautomatisierun in der Praxis. Konzepte, Techniken und Verfahren. Dpunkt-Verlag, Heidelberg. www.oo-testen.de. Nice practical book on testing.
- P. Liggesmeyer. Software-Qualitätsmanagement. Verlag Spektrum der Wissenschaften, Heidelberg.
- Boris Beizer: System Testing and Quality Assurance,Van Nostrand Reinhold, New York, 1984, ISBN 0-442-21306-9
- Glenford J. Myers, The Art of Software Testing, 1979
- Nesi (ed.), Objective Software Quality, 1995, Springer LNCS 926, ISBN 3-540-59449-3
- N. Fenton, S.L. Pfleeger. Software Metrics – a rigorous and practical approach. PWS Publishing.

- Version control with subversion. http://svnbook.red-bean.com/, also available as paper book of O'Reilly
- Sommerville (ed.), Software Configuration Management, 5. ed., 1996
- David Whitgift, Methods and Tools for Software Configuration Management, Wiley, 1991, ISBN 0-471-92940-9

Miscellaneous Literature

- **On Writing:**
  - A. Franklin Parks, J. A. Levernier, I. Masters Hollowell. Structuring Paragraphs and Essays – A Guide To Effective Writing. Bedford/St. Martin's. www.bedfordstmartins.com. Very good book.
- **Fogler/LeBlanc, Strategies for Creative Problem Solving**

Warning: Remarks on the Nature of the Course

- **A University is unlike a high school**
  - You should not expect to get a book, and that's it
    1. Software Engineering is too broad for that, unfortunately
    2. The lectures have to focus on most important things
  - You should not expect to be an expert after the course
- **Find your way from the lecture slides into the books**
  - Follow the reading instructions
  - Learn the additional material and read the additional readings
  - Follow the exercises in the groups
- **Expect to learn 3-4 weeks for the oral exam**
  - Don't wait until 1 week before the exam! That's too late...
- **Be aware: you have not yet seen larger systems**
  - Middle-size systems start over 100KLOC

Remarks on the Nature of the Course

- **The purpose of lecturing is**
  - To give you a condensed insight on the most important topics, such that you do not waste too much time during reading
  - To give you pointers for future work, once you left the course
    - If you haven't got the pointer, you can waste years in darkness

Main Goals

- **Learn about "engineering" software**
  - Engineering attitudes
  - Technology, process, experiences, human conditions
  - What a software engineer may sell (services, products, product lines…)
- **Get as many ideas as possible (broad overview)**
  - NOT: technical in-depth teaching (this must be left to other courses)
- **Get an introduction into the main obstacle: from a set of requirements, how do I arrive at a system? (forward engineering)**
- **Learn about systematic methods for graph-based specifications**
  - Because almost all requirements and design notations are graph-based
  - Get hold on the complexity of a large specification
- **Learn about the behavioral language Petri Nets, and derivatives thereof**

The top level of the V-model: Requirements, Validation, Software Quality

- **Know about requirement specification**
- **Software Quality:**
  - Contract-based development
  - Know what inspections are
  - Know about maintenance problems
  - Know about basic testing concepts
- **Model quality**
  - Model analysis
  - Model structuring

Design

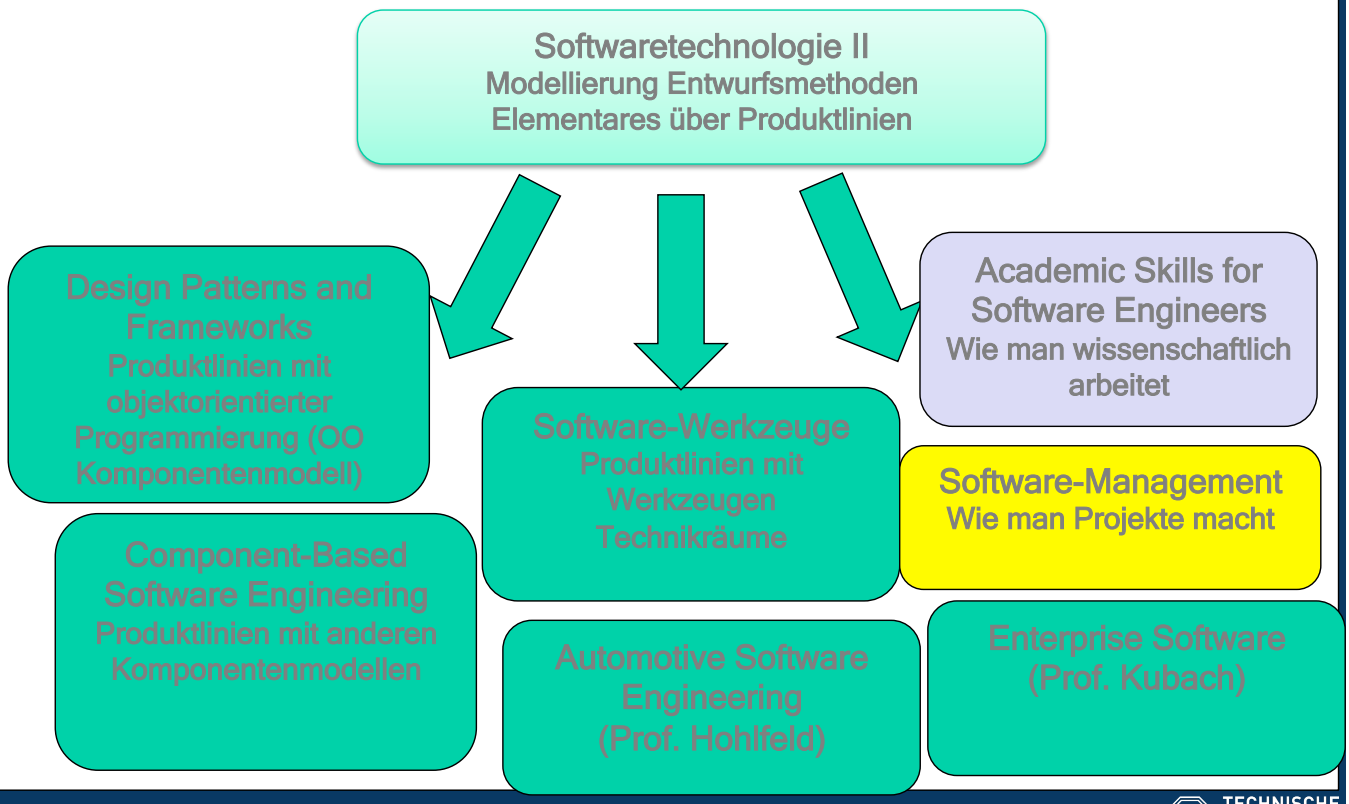
- **Know different forms of design methods**
  - functional, object-oriented, data-oriented
- **Know behavioral methods to generate code for verifiable specifications**
  - Petri nets
- **Get overview of software processes**
  - MDA, XP, V-model, ...
- **Know about "software architecture" and architectural styles**

Earning Money With Software

Business models
Markets
Product lines

Other Courses

- **Design Patterns and Frameworks (WS)**
  - Basic design patterns
  - Design patterns in frameworks
  - Role-based design
  - Composition of design patterns
  - Layered frameworks
- **Component-based Software Engineering (SS)**
  - Black-box component models (e.g., EJB)
  - Grey-box component models (e.g., Aspects)
  - Software composition
- **Software-Werkzeuge (SEW) (WS)**
  - Metamodelling, technological spaces, domain-specific languages
- **Academic Skills for Software Engineers (ACSE) (WS)**
- **Software-Management (SWM) (SS)**
- **Automotive Software Engineering (ASE, Prof. Hohlfeld, SS)**
- **Enterprise Software (ES, Prof. Kubach, SAP, WS)**



The End

- **http://st.inf.tu-dresden.de**