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

Übungsgruppen

- Teilung der Übungsgruppen in kleine Gruppen à 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

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 kontaktieren, da emails an Prof. Aßmann oft 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 Regressionstest
    - Reuseware
    - Model Driven Architecture

Course Structure

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

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

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

Part IV: Selling Software
  - Business models
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):
  - Bernd Brügge, Allen H. Dutoit, Objektorientierte Softwaretechnik, Pearson Studium
- Other good books, priority from top to bottom:

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!).
- Other excellent books:
  - Ken Lunn. Software development with UML. Palgrave-Macmillan. Many case realistic studies.

Reference Books
Analysis and Design

  - 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.

Component-Based Design

- C. Szyperski: Component Software. Addison-Wesley
- K. Czarnecki, U. Eisenecker: Generative Programming. Addison-Wesley

Project Management

- B. W. Boehm, Software Risk Management, 1989
- F. Brooks, The Mythical Man-Month, Addison-Wesley, 1975
- E. Yourdan: The Death March.
- David Thieles. The 12 simple secrets of Microsoft McGraw-Hill.

Implementation

Testing and Quality


Configuration Management

- Version control with subversion. http://svnbook.red-bean.com/, also available as paper book of O'Reilly

Miscellaneous Literature

- On Writing:
  - 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
    - Software Engineering is too broad for that, unfortunately
    - 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

TU Dresden, Prof. U. Aßmann
Announcements
Folie 21 von 24

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)

TU Dresden, Prof. U. Aßmann
Announcements
Folie 22 von 24

Master’s Courses
(weiteres Hauptstudium)

Softwaretechnologie II
Modellierung Entwurfsmethoden
Elementares über Produktlinien

TU Dresden, Prof. U. Aßmann
Announcements
Folie 23 von 24

Software-Werkzeuge
Produktlinien mit Werkzeugen
Technikerkurse

Software-Management
Wie man Projekte macht

Academic Skills for
Software Engineers
Wie man wissenschaftlich arbeitet

Component-based Software Engineering
Produktlinien mit anderen Komponentenmodellen

Automotive Software Engineering
(Prof. Hohlfeld)

Enterprise Software
(Prof. Kubach)

The End

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

TU Dresden, Prof. U. Aßmann
Announcements
Folie 24 von 24