Component-Based Software Engineering (CBSE) 0. Announcements

Prof. Dr. Uwe Aßmann

Technische Universität Dresden

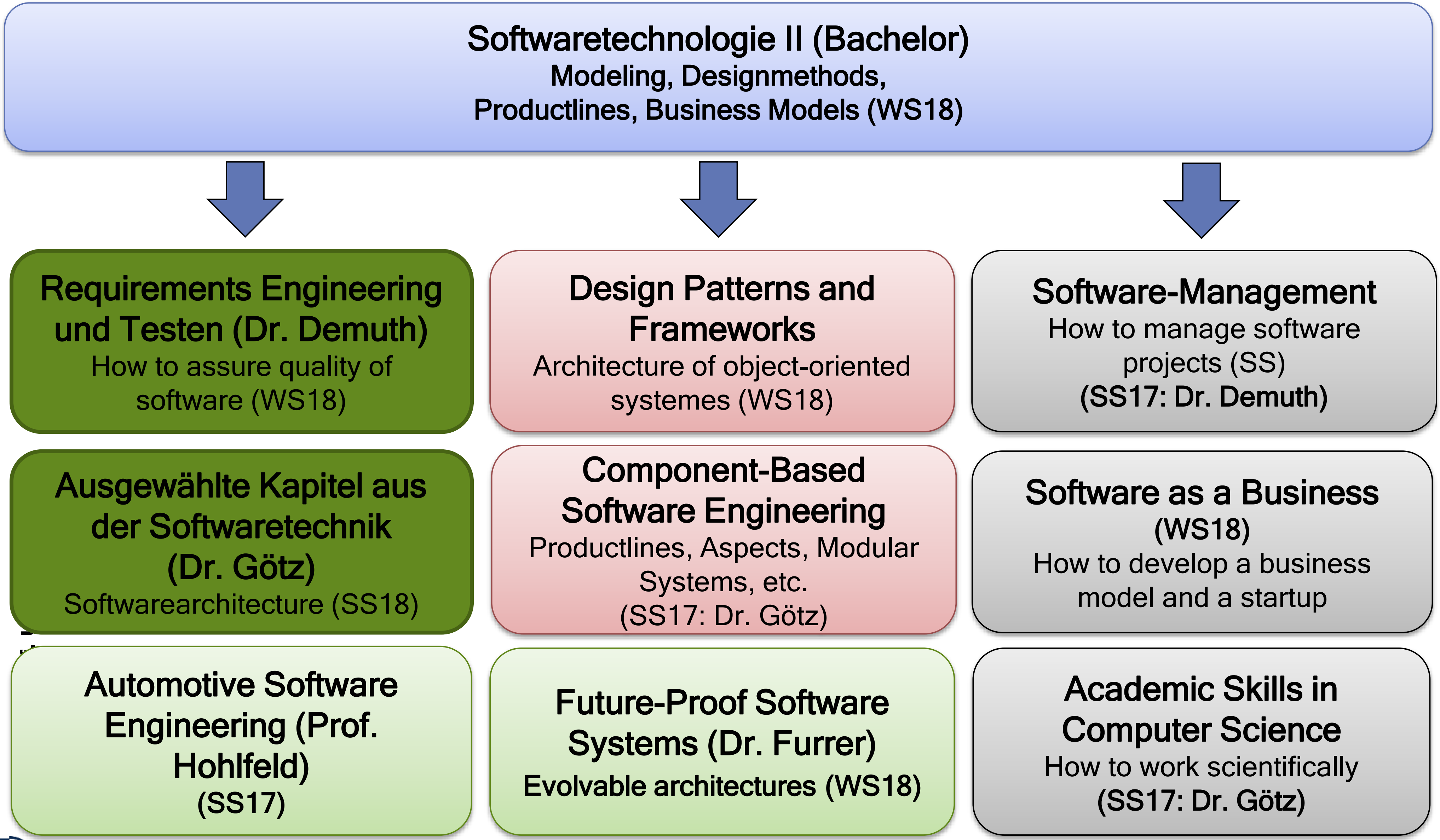
Institut für Software- und
Multimediatechnik

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

21.02.2017

Lecturer: Dr. Sebastian Götz

Master‘s Courses (Hauptstudium)



Softwaretechnologie II (Bachelor)
Modeling, Designmethods,
Productlines, Business Models (WS18)

Requirements Engineering und Testen (Dr. Demuth)
How to assure quality of software (WS18)

Design Patterns and Frameworks
Architecture of object-oriented systemes (WS18)

Software-Management
How to manage software projects (SS)
(SS17: Dr. Demuth)

Ausgewählte Kapitel aus der Softwaretechnik (Dr. Götz)
Softwarearchitecture (SS18)

Component-Based Software Engineering
Productlines, Aspects, Modular Systems, etc.
(SS17: Dr. Götz)

Software as a Business (WS18)
How to develop a business model and a startup

Automotive Software Engineering (Prof. Hohlfeld)
(SS17)

Future-Proof Software Systems (Dr. Furrer)
Evolvable architectures (WS18)

Academic Skills in Computer Science
How to work scientifically
(SS17: Dr. Götz)

Elements of the Course

- Lecturing
  - Do not miss one, they should give you a short and concise overview of the material
- Reading
  - Slides on “Obligatory Literature” require you to read papers from the web
    - TU Dresden has subscription to ACM Digital Library and IEEE Explorer
  - Slides on “Secondary Literature” contain useful but optional literature
- Exercise with Christian Piechnick
  - Exercise sheets are handed out every week, with some breaks
    - You have one week to solve them on your own
    - After that, solutions will be explained in the Exercise
    - Group work!
- Oral exams (20 min) usually in September, so that you have enough time to learn
  - For exchange students, other individual dates are possible

Reading Along the Lectures

- Unfortunately, the course is not covered by any book
  - About 60% is covered by the blue book “Invasive Software Composition”
  - Most of the rest on classical component systems by Szyperski in the book “Component Software. Beyond object-oriented computing. Addison-Wesley.”
- You have to read several research papers, available on the internet
  - Marked by “Obligatory Literature”
- Secondary Literature is non-mandatory, but interesting reading. Can be done during the course

Obligatory Literature

- During the course, read the following papers, if possible, in sequential order.
  - Every week, read about 1 paper (3-4h work)
  - Course web site
- [ISC] U. Aßmann. Invasive Software Composition. Springer, 2003.
- C. Szyperski. Component software. Beyond object-oriented computing. Addison-Wesley. Bestseller on classical component systems.

Papers

- [McIlroy68] D. McIlroy. Mass-produced Software Components. 1st NATO Conference on Software Engineering.
- [Dami95] Laurent Dami. Functions, Records and Compatibility in the Lambda N Calculus in Chapter 6 of "Object-oriented Software Composition". http://scg.unibe.ch/archive/oosc/PDF/Dami95aLambdaN.pdf
- CORBA. Communications of the ACM, Oct. 1998. All articles. Overview on CORBA 3.0.
- Others will be announced.

Recommended Literature

- Oscar Nierstrasz, Dennis Tsichritzis. Object-oriented Software Composition. Web book. http://scg.unibe.ch/archive/oosc/download.html
- I. Forman, S. Danforth. Meta-objects in SOM-C++. Very good book on meta object protocols and meta object composition.
- Journal Software - Tools and Techniques. Special Edition on Componentware, 1998. Springer. Good overviews.
- R. Orfali, D. Harkey: Client/Server programming with Java and CORBA. Wiley&Sons. Easy to read.
- CORBA. Communications of the ACM, Oct. 1998. All Articles.

Recommended Literature

- [GOF, Gamma] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns. Addison-Wesley 1995. Standard book belonging to the shelf of every software engineer.
  - The book is called GOF (Gang of Four), due to the 4 authors
- Alternatively to GOF can be read: [Remark: If you have already studied GOF intensively, do not read these]
  - A. Tesanovic. What is a pattern? Paper in Design Pattern seminar, IDA, 2001. Available at home page.
  - On Composite, Visitor: T. Panas. Design Patterns, A Quick Introduction. Paper in Design Pattern seminar, IDA, 2001. Available at home page.
  - P. Pop. Creational Patterns. Paper in Design Pattern seminar, IDA, 2001. Available at home page.

Less Important

- K. Czarnecki, U. Eisenecker. Generative programming . Addison-Wesley 2000. Good overview on aspects, but not on components
- F. Griffel. Componentware. dpunkt-Verlag. In German. A lot of material.

Be Aware – There Will Be Pain!

- This course is not like a standard course, it is research-oriented
  - It treats rather advanced material, the concept of graybox engineering
- No single book exists on all of that at all
  - ISC covers about 60%
  - Please, collaborate!
  - Read the articles
  - Ask questions!
  - Do the exercise sheets
- The exam can only be passed successfully, if you have visited all lectures and solved all exercise sheets
- Learn continuously! One week before the exam is too late!
- Be aware: most likely, you have not yet seen larger systems
  - Middle-size systems start over 100KLOC

The Positive Side – Why Should You Visit this Course

- Component-based software engineering (CBSE) is the generalization of object-oriented software engineering (OOSE)

- If you follow carefully,
  - You will discover an exciting world of graybox composition, a new way to *extend* software
  - You will know how to arrange **software reuse** in your company, because component models and composition are the enabling technologies
  - You will know why many companies fail in arranging a **product line**

- The gain is worthwhile the pain!

Component-based Software Contents and Goals

Course Content

1. Basics	•Introduction •Metamodelling •Component repositories
2. Simple black-box composition systems	•UML Business components •Transparency problems and connectors •CORBA •EJB
3. Architecture Systems	•ArchJava •Web services •Contract checking in SPEEDS HRC
4. Grey-box composition systems	•Composition filters •Generic programming •View-based programming •Aspect-oriented programming •Invasive Software Composition
5. Universal composition	•Rebinding and recomposition •Transconsistent composition •Staged composition
6. Applications of composition	•Document compostion •Software Ecosystems

Main Goals

- Understand the notion of a ***component***
  - With explicitly stated dependencies (in/out interfaces)
- Understand the concept of a ***component model***
  - Frameworks and product lines work with various different component models
  - Variability, extensibility, and gluing are three central goals
  - There are other central concepts for component models than classes and objects
- Understand ***composition techniques***
  - different times of composition
  - dynamic composition
  - Understand connectors as role models plus protocol
- Understand ***composition systems***
  - Understand grey-box, fragment-based composition
  - why it introduces new forms of static extensibility
  - why other static component models are special cases of it

The End