

# Component-Based Software Engineering (CBSE)

## 0. Announcements

Dr.-Ing. Sebastian Götz  
Technische Universität Dresden  
Institut für Software- und  
Multimediatechnik

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

04.04.2018

Based on Slides by Prof. Uwe Aßmann

# Master's Courses (Hauptstudium)

2

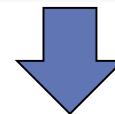
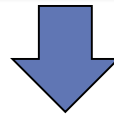
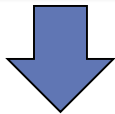
Component-Based Software Engineering (CBSE)

## Softwaretechnologie I (Bachelor)

(SS: Prof. Aßmann)

## Softwaretechnologie II (Bachelor)

(WS: Dr. Götz)



### Future-Proof Software Systems (Dr. Furrer)

(WS: Prof. Furrer)

### Design Patterns and Frameworks

(WS: Dr. Götz)

### Software-Management

How to manage software projects (SS)  
(SS: Dr. Demuth)

### Software Reengineering

(SS: Harry Sneed)

### Component-Based Software Engineering

(SS: Dr. Götz)

### Software as a Business

(WS: Prof. Aßmann)

### Automotive Software Engineering

(SS: Dr. Conrad)

### Requirements Engineering und Testen

(WS: Dr. Demuth)

### Academic Skills in Computer Science

(SS: 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, IEEE Explorer, etc.
  - Slides on “Secondary Literature” contain useful but optional literature
- ▶ Exercise with Dr. Thomas Kühn
  - No exercise this week.
  - Exercises will start next week.
- Oral exams usually in September, so that you have enough time to learn
  - For exchange students, other individual dates are possible
- To register for the exam
  - Write an email to [katrin.heber@tu-dresden.de](mailto:katrin.heber@tu-dresden.de)
  - Specify the module you want to be tested in



# Reading Along the Lectures

## Component-Based Software Engineering (CBSE)

- ▶ 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

## Component-Based Software Engineering (CBSE)

- ▶ During the course, read the following papers, if possible, in sequential order.
  - ▶ Every week, read about 1 paper (3-4h work)
  - ▶ Course web site
- ▶ Side note
  - ▶ 30 LP can be interpreted as a full position (40h/week) for the whole semester
  - ▶ This course captures 6 LP → 8h/week
  - ▶ This leaves **5h/week** for self-study! (1.5h lecture, 1.5h exercise)

## Papers

- ▶ [McIlroy68] D. McIlroy. Mass-produced Software Components. 1st NATO Conference on Software Engineering.
  - ▶ <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>  
(Pages 79 – 87)
- ▶ Others will be announced.



# Obligatory Literature

## Component-Based Software Engineering (CBSE)

- ▶ [GOF, Gamma95] 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 you can be read:
  - [Freeman04] E. Freeman, E. Robson, B. Bates, K. Sierra. Head First Design Patterns: A Brain-Friendly Guide. O'Reilly Media, Inc., 2004.
- ▶ [Völter06] Markus Völter, Thomas Stahl, Jorn Bettin, Arno Haase, Simon Helsen, Krzysztof Czarnecki: **Model-Driven Software Development: Technology, Engineering, Management.** Wiley 2006.
  - ▶ Read Chapter 2



# Be Aware – There Will Be Pain!

## Component-Based Software Engineering (CBSE)

- ▶ 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 understood all parts of the course.
- ▶ 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)

- ▶ 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

## Component-Based Software Engineering (CBSE)

### 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

### 4. Gray-box composition systems

- Composition filters
- Generic programming
- View-based programming
- Aspect-oriented programming
- Invasive Software Composition

### 5. Applications of composition

- Robotics
- Mobile Applications

# Main Goals

## Component-Based Software Engineering (CBSE)

- ▶ 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

Component-Based Software Engineering (CBSE)

