Module title		Module code
Global Software Engineering		
Person responsible for the module	Faculty	
Prof. Felix Schwägerl	Computer Science and Mathematics	

Semester taught according to the curriculum	Level of study	Module type	Credit value
3.	2.	mandatory	8

Mandatory requirements
At least 30 credits from the 1st study stage
Recommended previous knowledge
Programming 1 and Programming 2

Content
see next page

Assigned submodules

Nr.	Submodule title	Teaching hours	Credit value
1.	Global Software Engineering	6 SWS	8

Submodule	Submodule abbreviation
Global Software Engineering	**GSE**

Responsible person	Faculty
Prof. Felix Schwägerl	Computer Science and Mathematics
Lecturer	Availablilty of module
Prof. Dr. Carsten Kern Prof. Felix Schwägerl	only in winter semester

Teaching method
Seminar teaching with exercises (4 SWS) and practical course (2 SWS)

Semester taught according to the curriculum	Teaching hours	Teaching language	Credit value
3.	6 SWS	english	8

Study hours required

Hours in attendance/lectures	Hours for self-study
90h	150h

Method of assessment
Written exam: 90 minutes

Content

- Basics of software engineering (motivation, definitions, ethics, role of models)
- Phases, disciplines, and processes (phase models, iterative, spiral model, V model)
- Agile software development (manifesto, principles, Scrum, empirical process improvement)
- Requirements engineering (definitions, gathering techniques, attributes, templates)
- Object-oriented analysis (use case models, domain models, behavior/interaction models)
- Software architecture (views, evaluation criteria, architectural styles, documentation)
- Fine-grained design (refinement, implementation in Java, design principles, design patterns)
- Testing (regression, refactoring, unit tests, code coverage, test-driven development)
- Quality assurance (verification/validation, coverage, continuous integration, acceptance tests)
- Deployment and maintenance (delivery, software evolution, predictive maintenance)
- DevOps engineering (continuous deployment, containers, infrastructure as code, monitoring)
- Project management and planning (risk management, team management, cost estimation)
- Software version management (revision logs, branching, tagging, conflict resolution)
- Global software development (motivations, socio-technical challenges, methods, tools)

Learning objectives: Subject competence

After successful completion of the submodule, students are able to,

- Know and reproduce the ways of thinking and procedures of software engineering (1).

- Express awareness about the importance, difficulties and possibilities of software engineering and its disciplines (1).
- Select, tailor, and improve the software development process suitable for a specific project or product (2).
- Use standardized modeling notations on an adequate level of detail and utilize models' ability to break down software engineering tasks by abstracting from requirements, software, and hardware (3).
- Document the results of requirements engineering, object-oriented analysis and fine-grained design using adequate language, terms, and formalisms (2).
- Systematically specify, design, implement, verify, and deliver a software system with limited extent using suitable engineering methodologies and an object-oriented programming language like Java (3).
- Apply appropriate software quality assurance metrics, methods, and tools to existing systems or systems under development (2).
- Select and apply suitable methods and tools for project management, software maintenance, and software version management (3).
- Explain (1) and classify (2) the specific challenges, methods, and tools occurring in international, intercultural, and interdisciplinary software engineering teams.

Learning objectives: Personal competence

After successful completion of the submodule, students are able to,

- Understand how the specifics of global software development impact each discipline of software engineering (1).
- Theoretically know how to collaborate with clients or managers to gather software requirements and help them make informed business decisions based on technical facts (1).
- Ask the crucial questions for being able to select adequate methods and tools for each discipline of software engineering (2).
- Assess analysis, design and implementation artifacts produced by team members according to well-defined criteria and communicate constructive feedback effectively and adequately (2).
- Coordinate the activities of software engineering teams and deal with challenges such as stress, motivation, or conflicts (2).Adopt different roles in software engineering teams with different responsibilities therein (3).

Teaching materials offered

Copies of slides, exercises, code examples, materials from case studies, templates

Teaching media

Laptop, beamer, blackboard

Literature
• Ian Sommerville: Software Engineering, 10th edition, Pearson, 2016 • Ian Sommerville: Engineering Software Products, Pearson, 2021 • Ken Schwaber, Jeff Sutherland: Scrum Guide, Creative Commons, 2020 • Klaus Pohl, Chris Rupp: Requirements Engineering Fundamentals, RockyNook, 2015 • Grady Booch et al.: Object-Oriented Analysis and Design with Applications, 3rd edition, Addison-Wesley, 2007 • Mark Richards, Neil Ford: Fundamentals of Software Architecture, O’Reilly, 2020 • Erich Gamma et al: Design Patterns, Addison-Wesley, 2009 • Robert C. Martin: Clean Code, Prentice Hall, 2009 • Shekhar Gulati, Rahul Sharma: Java Unit Testing with JUnit 5, Apress, 2017 • Paul Ammann, Jeff Offutt: Introduction to Software Testing, Cambridge University Press, 2016 • Gene Kim et al.: The DevOps Handbook, IT Revolution Press, 2016 • Scott Chacon, Ben Straub: Pro Git, Apress, 2014 • Pierre Bourque, Dick Fairley: Software Engineering Body of Knowledge (SWEBOK), v3, IEEE Computer Society, 2014 • James D. Herbsleb: Global software engineering in the age of GitHub and Zoom. J. Softw. Evol. Process. 35(6), 2023 [and previous work referenced by the author]

The numbers in brackets indicate the levels to be reached: 1 - understanding 2 - ability 3 - understand and application