Valid from 2024.HS

Module description: Software Engineering 1

Module Code	t.BA.IT.SWEN1.19HS
ECTS Credits	4
Language of Instruction/Examination	German
Organizational Unit	InIT
Module Coordinator	Kurt Bleisch
Legal Framework	The module description is part of the legal basis in addition to the general academic regulations. It is binding. During the first week of the semester a written and communicated supplement can specify the module description in more detail.
Module Characteristic	Type 2a 4 consecutive lecture lessons per semester week and class
Module Description	This module provides you with the necessary analysis and design skills to develop and realize larger and more complex software applications.User and domain research, use cases and domain modeling are applied to analyze a problem. The elicited requirements for a software system are systematically specified and tested. Derived from the requirements, techniques for the design of a suitable software architecture and the implementation in an object-oriented design are taught. Proven architecture and design patterns are applied and emphasis is placed on high software quality (extensibility, maintainability). Standardized notations (such as UML) are used for modeling and communicating the results from analysis, software architecture and design.

Module description: Software Engineering 1

Module Content	**This module provides the necessary analysis and design skills to develop and implement larger and more complex software application.** **Analysis competence includes above all the will and ability to communicate and cooperate with clients and future system users and to quickly familiarize oneself with new application contexts. Students must be able to recognize familiar problems in the application context and be familiar with the associated solution patterns. They recognize inconsistencies and can deal with unclear requirements. Complex domains can be modelled and large application problems can be broken down into subproblems using suitable interfaces.** **Design competencies include the ability to design hardware and software systems that fully meet the requirements. Abstraction is as essential as a solid knowledge of software architecture. Central to the design is the implementation of non-functional requirements such as security, performance, scalability, maintainability, extensibility and reliability.** **Introduction and Software Development Processes (4 lessons)** **Overview of software development process models and their home grounds (waterfall, iterative-incremental and agile)** **Process and artifacts in an iterative-incremental, use-case-driven and architecture-centric software development process** **Requirements Analysis (16 lessons)** **Introduction in Usability and UX(contextual inquiry, personas and scenarios, UI sketching & prototyping)** **Elicitation and communication of functional requirements with use cases (UML use case diagram, use case specification)** **Elicitation and communication of non-functional requirements (quality requirements, constraints)** **Modelling of the user's technicality and terms (domain model) and introduction to Domain Driven Design (DDD, conceptual UML class diagram)** **Software Architecture and Design (36 lessons)** **Design and modeling of a software architecture (4+1 view, represented by UML package diagram, UML deployment diagram)** **Introduction to Clean Architecture (SOLID principles, layer architecture, onion architecture)** **Use case realization and class design (Responsibility Driven Design (RDD), UML class diagram, UML sequence diagram, UML communication diagram, UML state diagram, UML activity diagram)** **Design with Design Patterns (GoF: Factory, Singleton, Adaptor, Bridge, Composite, Decorator, Facade, Proxy, Chain of Responsibility, Observer, State, Strategy, Visitor)** **Various in-depth topics such as: Distributed systems, GUI architectures, persistence, framework design** **Practical Training** **As a practical training, the students solve exercises tailored to the topic of the lecture. The practical training is an integral part of the lecture in this module type (2 lessons lecture and 2 lessons exercises).**

Module description: Software Engineering 1

Prerequisite Knowledge

-

Learning Objectives (Competences)

Students...	Competencies	Taxonomies
You can use standardized notations (such as UML) for modeling and communicating artifacts in the software development process.	F, M	K3
You can define a software system and systematically collect and communicate the functional requirements with use cases as well as quality requirements and constraints.	M, F	K3
You can grasp the user's terminology using suitable procedures and condense it into a domain-specific terminology (domain model).	M, F	K3
Based on the requirements, you can design a suitable software architecture and an object-oriented design for the components of the domain logic contained therein.	F, M	K3
For a given, iterative-incremental software development process, you can explain the process and artifacts for developing an object-oriented software application.	F, M	K2
You can use proven analysis, architecture and design patterns adequately for a problem.	M, F	K3, K4

Performance Assessment

End-of-module exam	Assessment	Length (min.)	Weighting	Form
written exam	Grade	90	70	acc. to module agreement

Performance assessment during the semester	Assessment	Length (min.)	Weighting	Form
Quiz *Quiz based on MC questions*	Grade	15	5	acc. to module agreement
Quiz *Quiz based on MC questions*	Grade	15	5	acc. to module agreement
Evaluated exercises	Grade		20	acc. to module agreement

Classroom Attendance Requirement

None

Learning material

Comments