

<b>Modulbezeichnung</b>	<b>Softwaretechnik</b>
<b>Modulbezeichnung (eng.)</b>	Software Engineering
<b>Semester (Häufigkeit)</b>	4 (jedes Sommersemester)
<b>ECTS-Punkte (Dauer)</b>	5 (1 Semester)
<b>Art</b>	Pflichtmodul
<b>Studentische Arbeitsbelastung</b>	35 h Kontaktzeit + 115 h Selbststudium
<b>Voraussetzungen (laut BPO)</b>	keine
<b>Empf. Voraussetzungen</b>	Sichere Anwendung von Hochsprachen wie Java, C#, etc.
<b>Verwendbarkeit</b>	BOMI, BOWI, BIPV
<b>Prüfungsart und -dauer</b>	Klausur 2 h oder mündliche Prüfung
<b>Lehr- und Lernmethoden</b>	Multimedial aufbereitetes Online-Studienmodul zum Selbststudium mit zeitlich parallel laufender Online-Betreuung und regelmäßigen virtuellen Lehrveranstaltungen
<b>Modulverantwortliche(r) (HSEL/VFH)</b>	C. Wunck / S. Edlich (BHT)

## **Qualifikationsziele**

Nach dem erfolgreichen Abschluss des Studienmoduls, sind die Studierenden in der Lage:

- softwaretechnische Kenntnisse in Projekte und in die Projektarbeit zu übertragen und anzuwenden
- Anforderungsermittlung und Verwaltung eigenständig durchzuführen
- informationstechnische Sachverhalte grafisch darzustellen
- tragfähige IT-Architekturen zu entwerfen und zu gestalten
- zu entscheiden und abzuwägen, wann welches (bestimmtes) Vorgehensmodell besser geeignet ist als ein anderes
- Requirements Engineering im Rahmen der Projektarbeit einzusetzen und zu erklären.
- die Hauptprobleme der Softwareentwicklung durch Analyse und Berücksichtigung der wichtigsten Anforderungsmerkmale zu identifizieren.
- im Rahmen der Analyse - Pflichten- und Lastenheft, Use-Cases und Requirements einzuordnen und zu erstellen.
- den geeigneten Einsatz von UML zu beurteilen und UML praktisch an einem eigenen Projekt anzuwenden und die kritische Nutzung dieser Industriesprache zu berücksichtigen.
- zu beurteilen welche UML-Diagramme in welcher Reihenfolge anzuwenden sind, um ein Modellierungsziel zu erreichen
- die Bedeutung der Architektur im Designprozess zu erklären und diese auf Projekte anzuwenden und zu begründen
- Werkzeuge für das systematische und objektorientierte Testen einzusetzen und selber Tests zu entwerfen
- die Möglichkeiten und Grenzen des Refactoring zu erklären und unter Eclipse oder einer anderen IDE anzuwenden, u.a. durch identifizieren von Bad Code Smell
- die Funktionen des Buildmanagements mit ANT praktisch einzusetzen
- die Konzepte des Versions- und Fehlermanagements zu erklären und die bekanntesten Systeme praxisnah zu verwenden
- die Bedeutung von Metriken als Qualitätsmaß praktisch zu beurteilen und Basismetriken zu berechnen
- Codemetriken und deren Werkzeuge zu gebrauchen, bspw. Architekturmetriken und deren Visualisierung
- das Entwurfsmuster Dependency Injection unter Verwendung unterschiedlicher Frameworks in Projekten zu nutzen.

## **Lehrinhalte**

LE01 Einführung in die Softwaretechnik  
LE02 Vorgehensmodelle / agile Modelle  
LE03 Requirements Engineering  
LE04 Analyse  
LE05 Unified Modeling Language  
LE06 Objektorientiertes Design  
LE07 Objektorientierte Architekturen  
LE08 Objektorientiertes Testen und Test-Driven Development  
LE09 Refactoring  
LE10 Buildmanagement  
LE11 Versions- und Fehlermanagement  
LE12 Software- und Architekturmetriken  
LE13 Dependency Injection

**Literatur**

Balzert, Lehrbuch der Softwaretechnik Oesterreich, Analyse und Design mit UML 2.5  
Christ Rupp, Requirements Engineering Balzert, Lehrbuch der Objektmodellierung  
Ian Sommerville, Softwaretechnik (Global Edition)  
Jeckle, UML 2 glasklar

**Lehrveranstaltungen****Dozenten/-innen****Titel der Lehrveranstaltung**

P. Bartels

Softwaretechnik