



Fakultät Informatik und
Wirtschaftsinformatik

Technische Hochschule
Würzburg-Schweinfurt

Modulhandbuch **Bachelor Software Engineering (B. Eng.)**

Wintersemester 2026
Sommersemester 2026



Inhalt

1. Semester.....	3
Algebra	4
Datenbanken	6
Grundlagen Informatik	8
IT-Projektmanagement und BWL	10
Introduction to Software Engineering	12
Programmieren 1	14
2. Semester.....	16
Algorithmen und Datenstrukturen	17
Analyse und Design	19
Netzwerke	21
Programmieren 2	23
Projekt 1	25
Stochastik	27
3. Semester.....	29
Backend Systems	30
Data Science	32
Professional Skills	35
Projekt 2	37
Softwarequalität	39
System-Oriented Programming	41

1. Semester

Modulprofil

Prüfungsnummer

1510600

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Wintersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Prof. Dr. Andreas Keller

Dozierende

Prof. Dr. Andreas Keller

Verwendbarkeit

BSED

Studiensemester

1. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

keine

Empfohlene Voraussetzungen

Beherrschung der Schulmathematik

Inhalte

Allgemeine Grundlagen:

- Körper der reellen Zahlen
- Prinzip der vollständigen Induktion
- Einführung in den Körper der komplexen Zahlen

Lineare Algebra:

- Vektorräume (lineare Unabhängigkeit, Basis und Dimension)
- Matrizen (Rechnen mit Matrizen, Spur und Determinante, Rang einer Matrix)
- Lineare Gleichungssysteme
- Gaußscher Algorithmus
- Lineare Abbildungen
- Eigenwerte und Eigenvektoren
- Diagonalisierung
- Spektralsatz

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

keine

Art der Prüfung

Schriftliche Prüfung (sP) gemäß
§ 23 APO

Dauer/Form der Prüfung

90 Minuten

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

keine

Lernergebnisse

- Die Studierenden kennen grundlegende mathematische Konzepte und Begriffe, die in Informatik und Technik Anwendung finden.
- Die Studierenden verstehen die Bedeutung mathematischer Methoden und deren Rolle in der Entwicklung von Informatikanwendungen.
- Die Studierenden wenden mathematische Techniken an, um praktische Probleme in der Informatik und Technik zu lösen.
- Die Studierenden analysieren mathematische Probleme, um geeignete Lösungsstrategien zu identifizieren und zu entwickeln.
- Die Studierenden bewerten verschiedene Lösungsansätze und deren Effektivität im Kontext spezifischer mathematischer Herausforderungen.
- Die Studierenden erstellen komplexe mathematische Modelle, die in realen Anwendungen der Informatik und Technik genutzt werden können.

Literatur

- Gramlich, Günter: Lineare Algebra – Eine Einführung; Fachbuchverlag Leipzig im Carl Hanser Verlag 2021
- Hartmann, Peter: Mathematik für Informatiker; Vieweg + Teubner, Wiesbaden 2019
- Papula, Lothar: Mathematik für Ingenieure und Naturwissenschaftler 1 und 2; Vieweg + Teubner; Wiesbaden 2024
- Schubert, Matthias: Mathematik für Informatiker; Vieweg + Teubner, Wiesbaden 2012
- Strang, Gilbert: Lineare Algebra; Springer-Verlag, Berlin/Heidelberg/New York 2003

Modulprofil

Prüfungsnummer

1510400

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Wintersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Michael Rott

Dozierende

Michael Rott

Verwendbarkeit

BSED

Studiensemester

1. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

keine

Empfohlene Voraussetzungen

keine

Inhalte

Das Modul vermittelt die grundlegenden Konzepte und Techniken der Datenbankentwicklung. Es werden das relationale Datenmodell und die Relationen-Algebra als theoretische Grundlagen vorgestellt. Ein Schwerpunkt liegt auf der Datenbankmodellierung, insbesondere der Erstellung von Entity-Relationship-Modellen (ER-Modelle) und deren Überführung in relationale Schemata unter Berücksichtigung von Normalformen. Einführung in die Sprache SQL, einschließlich der Datenmanipulation, Datenabfrage sowie der Definition von Schemata und der Transaktionsverwaltung. In praktischen Übungen und semesterbegleitenden Projekten wird die Datenbankentwicklung und -administration geübt.

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

keine

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Mündliche Prüfung

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

keine

Lernergebnisse

- Die Studierenden können grundlegende Konzepte der Datenpersistenz und die Unterschiede zwischen persistenten und nicht-persistenten Daten erläutern.
- Die Studierenden können die zentralen Begriffe der relationalen Datenbanken, wie Relation, Primärschlüssel, Fremdschlüssel und Normalisierung, definieren.
- Die Studierenden verstehen die Relationale Algebra und können einfache Operationen darauf anwenden.
- Die Studierenden können den Zusammenhang zwischen konzeptioneller, logischer und physischer Datenmodellierung erklären und deren Bedeutung für die Datenbankentwicklung begründen.
- Die Studierenden sind in der Lage, Entity-Relationship-Modelle (ERM) für gegebene Anwendungsfälle zu erstellen und diese in relationale Schemata zu überführen.
- Die Studierenden können SQL-Abfragen zur Datenmanipulation (DML) und Schema-Definition (DDL) formulieren und ausführen.
- Die Studierenden können bestehende Datenbankschemata analysieren und hinsichtlich Redundanz, Konsistenz und Normalformen bewerten.
- Die Studierenden sind in der Lage, fachliche Informationsbedarfe zu analysieren und daraus geeignete Datenstrukturen und Abfragen abzuleiten.

Literatur

- Michael Kofler (2024). Datenbanksysteme - Das umfassende Lehrbuch (2. Auflage). Bonn: Rheinwerk Verlag GmbH
- Kemper, A., & Eickler, A. (2015). Datenbanksysteme – Eine Einführung (10. Auflage). München: De Gruyter Oldenbourg Verlag
- Elmasri, R., & Navathe, S. B. (2015). Grundlagen von Datenbanksystemen (7. Auflage). München: Pearson Studium
- Garcia-Molina, H., Ullman, J. D., & Widom, J. (2013). Database Systems: The Complete Book (2nd ed.). Upper Saddle River, NJ: Pearson
- Saake, G., Sattler, K.-U., & Heuer, A. (2011). Datenbanken – Konzepte und Sprachen (3. Auflage). München: Pearson Studium

Modulprofil

Prüfungsnummer

1510500

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Wintersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Prof. Dr. Peter Braun

Dozierende

John Zhong Wang

Verwendbarkeit

BSED

Studiensemester

1. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

keine

Empfohlene Voraussetzungen

keine

Inhalte

Das Modul vermittelt die Grundlagen der Informatik für Studierende außerhalb der Kern-Informatik.

- Information, Informationsgehalt, Informationscodierung, Darstellung von Zahlen und Zeichen, Codierung von Text, Datumsangaben, Farbinformationen
- Binärarithmetik, Boole'sche Algebra und Logikgatter
- Modelle und Modellbildung als grundlegendes Prinzip in der Informatik, Abstraktion, Reduktion, Dekomposition, Aggregation
- Beschreibung von Datenstrukturen mit der erweiterten Backus-Naur-Form
- Modellierung dynamischer Systeme und ihre Beschreibung mit endlichen Automaten und Zustandsdiagrammen
- Formale Sprachen, reguläre Grammatiken und das Wortproblem
- Weitere Automatenmodelle: Moore und Mealy Automaten
- Der Begriff des Algorithmus, Berechenbarkeit, Halteproblem, Funktionsweise und Programmierung von Turing-Maschinen
- Grundlegende Algorithmen zum Suchen und Sortieren
- Geschichte der Hardwareentwicklung
- Aufbau und prinzipielle Arbeitsweise eines Computers und Mikroprozessors, Von-Neumann Architektur, Moore'sches Gesetz
- Aufbau und Funktionsweise des Internet und World Wide Web
- Einführung in die Sprachen HTML und Markdown
- Aufbau von verteilten Systemen, Client-Server, Peer-to-Peer, Blockchain, Git
- Geschichte der Künstlichen Intelligenz, Verfahren des maschinellen Lernens, Regression, Funktionsweise von neuronalen Netzen
- Datenschutz und Ethik in der Informatik

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

keine

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Mündliche Prüfung

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

keine

Lernergebnisse

- Die Studierenden erinnern grundlegende Begriffe der Informationsverarbeitung und können den Informationsgehalt von Nachrichten messen.
- Die Studierenden verstehen Codierungen von Daten und grundlegende Methoden zur Modellbildung innerhalb der Informatik.
- Die Studierenden wenden Methoden zur Beschreibung von Datenstrukturen an.
- Die Studierenden analysieren einfache dynamische Systeme und beschreiben diese mit Zustandsdiagrammen.
- Die Studierenden erstellen reguläre Grammatiken und lösen das Wortproblem mit Hilfe von endlichen Automaten.
- Die Studierenden analysieren einfache Problemstellungen und erstellen Lösungen mit Hilfe von Turing-Maschinen.
- Die Studierenden erinnern wichtige Punkte der Geschichte der Informatik.

Literatur

- Gumm, Heinz-Peter; Sommer, Manfred: Einführung in die Informatik, 10. Auflage, Oldenbourg, 2012.
- Ernst, Hartmut; Schmidt, Jochen; Beneken, Gerd: Grundkurs Informatik: Grundlagen und Konzepte für die erfolgreiche IT-Praxis, 8. Auflage, Springer Verlag, 2023

Modulprofil

Prüfungsnummer

1510100

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Wintersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Prof. Dr. Eva Wedlich

Dozierende

Prof. Dr. Eva Wedlich

Verwendbarkeit

BSED

Studiensemester

1. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

keine

Empfohlene Voraussetzungen

keine

Inhalte

Folgende Themen werden behandelt:

Einführung Projekt und Projektmanagement

- Projektorganisation
- Projektplanungsprozess
- Projektkalkulation
- Projektsteuerung und –überwachung
- Projektabschluss

Grundbegriffe der Betriebswirtschaftslehre:

- Der Betrieb
- Die betriebswirtschaftlichen Produktionsfaktoren
- Betriebswirtschaftliche Ziele
- Betriebswirtschaftliche Kennzahlen

Konstitutive Entscheidungen eines Betriebes:

- Standortwahl:
- Rechtsformen:

Betriebswirtschaftliche Funktionen:

- Beschaffung und Einkauf
- Marketing und Vertrieb

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

keine

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Mündliche Prüfung

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

keine

Lernergebnisse

- Die Studierenden können grundlegende Begriffe und Konzepte des IT-Projektmanagements sowie der Betriebswirtschaftslehre wiedergeben.
- Die Studierenden wenden Methoden zur Entwicklung von Projektplänen und zur Durchführung einer Standortanalyse an praktischen Beispielen an.
- Die Studierenden analysieren projektbezogene und betriebswirtschaftliche Daten.
- Die Studierenden beurteilen die Effizienz von Projektsteuerungsmechanismen und treffen fundierte Entscheidungen zur Projektkalkulation und -überwachung.
- Die Studierenden evaluieren alternative Standorte und strategische betriebswirtschaftliche Entscheidungen.
- Die Studierenden entwickeln eigenständig Lösungsstrategien für komplexe IT-Projektmanagementszenarien und betriebswirtschaftliche Herausforderungen.

Literatur

- Vahs, D.; Schäfer-Kunz, J.: Einführung in die Betriebswirtschaftslehre; 9. Auflage; Schäffer-Poeschel, Stuttgart, 2025
- Wöhe, G.: Einführung in die allgemeine Betriebswirtschaftslehre; 28. Auflage; Vahlen; München, 2023
- Hanke, D.: Die 10 wichtigsten Methoden im Projektmanagement: Der schnelle, verständliche und bewährte Einstieg ins klassische Projektmanagement, 2022
- Timinger, H.: Modernes Projektmanagement: Mit traditionellem, agilem und hybridem Vorgehen zum Erfolg, Wiley, 2024

Modul: 1510300

Introduction to Software Engineering

Modulprofil

Prüfungsnummer

1510300

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Wintersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Englisch

Organisation

Modulverantwortung

Prof. Dr.-Ing. Tobias Fertig

Dozierende

Prof. Dr. Isabel John,

Prof. Dr.-Ing. Tobias Fertig

Verwendbarkeit

BSED

Studiensemester

1. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

none

Empfohlene Voraussetzungen

none

Inhalte

Software engineering covers all phases of software development, from the initial idea to the tested and delivered system. This module introduces the fundamental principles and concepts of software engineering, focusing on systematic development processes, requirements engineering, and basic modeling techniques. It provides the foundation for advanced modules on Analysis and Design and Software Quality.

Fundamentals of Software Engineering

- Objectives and principles of software engineering
- Characteristics and challenges of software projects
- Overview of software development activities and processes (agile vs. traditional)

Introduction to software (requirements) modeling with UML

- Introduction to UML Diagrams: Class Diagrams, Use Case Diagrams, Activity Diagrams, Sequence Diagrams

Software Development Processes & Team Collaboration

- Roles in software projects (e.g., Product Owner, Developer, Tester)
- Fundamentals of project organization and documentation
- Cost and benefit estimation in software projects
- Agile processes (e.g., Scrum)

Quality Aspects

- Introduction to software quality (transition to "Software Quality")

Prüfung

**Verpflichtende Voraussetzung
gemäß SPO für die Teilnahme
an der Prüfung**
none

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Portfolio

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Englisch

**Voraussetzung für die Vergabe
von Leistungspunkten**
none

Lernergebnisse

- Students are able to explain the principles of software engineering and assess their relevance.
- Students compare and justify the application of different software development processes.
- Students collect, model, and specify systematically software requirements using UML.
- Students use UML diagrams to structure and analyze software systems.
- Students evaluate software projects in terms of feasibility, cost, and quality.
- Students understand fundamental quality assurance methods.

Literatur

- Sommerville, Ian: Software Engineering, Pearson, 2018
- Oestereich, Bernd: Analyse und Design mit UML 2.5, Oldenbourg, 2013/2020
- Rupp, Chris: UML glasklar, Hanser, 2012
- McLaughlin: Objektorientierte Analyse und Design von Kopf bis Fuß, O'Reilly, 2017
- Kecher, Christoph; Hoffmann-Elbern, Ralf; Will, Torsten T.: UML 2.5: Das umfassende Handbuch, Rheinwerk Computing, 2021

Modulprofil

Prüfungsnummer

1510200

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Wintersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Prof. Dr. Tristan Wimmer

Dozierende

Prof. Dr. Tristan Wimmer,
John Zhong Wang

Verwendbarkeit

BSED

Studiensemester

1. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

keine

Empfohlene Voraussetzungen

keine

Inhalte

Dieses Modul zielt darauf ab, Studierenden die Grundlagen der Programmierung mithilfe der Programmiersprache Python beizubringen. Es stellt die Grundkonzepte von Programmiersprachen und Programmierparadigmen vor und schafft die Basis für weitere Module im Studiengang Software-Engineering.

Folgende Themen werden behandelt:

- Elementarer Datentypen, Datenstrukturen und Operatoren
- Kontrollstrukturen: Schleifen und bedingte Anweisungen
- Programmieren mit Funktionen
- Einführung in die objektorientierte Programmierung
- Einführung in das Konzept der Vererbung
- Einführung in das Exception Handling

Neben diesen Themen werden in diesem Modul die geeigneten Strukturierungsmöglichkeiten von Code, sowie Dokumentationsmöglichkeiten für einen sauberen und gut lesbaren Programmierstil, aufgezeigt. Desweiteren wird den Studierenden gezeigt, wie sie Problemen am besten begegnen und lösen.

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

keine

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Portfolio

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

keine

Lernergebnisse

- Die Studierenden identifizieren und benennen die grundlegenden Datentypen, Datenstrukturen und Operatoren und wenden sie in der Programmiersprache Python an.
- Die Studierenden erläutern, wie Kontrollstrukturen wie Schleifen und bedingte Anweisungen den Ablauf von Programmen steuern und wie diese in Python implementiert werden.
- Die Studierenden schreiben einfache Python-Programme, die Funktionen und Parameterübergaben nutzen, um spezifische Aufgaben zu lösen und wenden dabei das Prinzip Divide & Conquer an.
- Die Studierenden verstehen objektorientierter Programmierung anzuwenden, um durch Kapselung die Struktur und Wartbarkeit eines Programms zu verbessern.
- Die Studierenden entwerfen und implementieren für eine spezifische Anforderung ein objektorientiertes Programm in Python und nutzen dabei die grundlegenden Prinzipien der Vererbung nutzt.
- Die Studierenden wenden Exception Handling für fehlerhafte Eingaben und Datentypinkompatibilität an.

Literatur

Häberlein, Tobias. Programmieren Mit Python: Eine Einführung in Die Prozedurale, Objektorientierte Und Funktionale Programmierung. 1st ed. 2024. Berlin, Heidelberg: Springer Berlin Heidelberg, 2024. <https://doi.org/10.1007/978-3-662-68678-2>.

2. Semester

Modulprofil

Prüfungsnummer

1511100

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Sommersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Prof. Dr.-Ing. Tobias Fertig

Dozierende

Prof. Dr.-Ing. Tobias Fertig

Verwendbarkeit

BSED

Studiensemester

2. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

keine

Empfohlene Voraussetzungen

Keine

Inhalte

Die Veranstaltung behandelt verschiedene komplexere Algorithmen und Datenstrukturen der Informatik in Theorie und praktischer Anwendung. Zur Implementierung der Lösungen können beliebige Programmiersprachen eingesetzt werden. Es werden exemplarisch die folgenden Themenschwerpunkte in Theorie und Praxis behandelt:

- Algorithmusbegriff, Datenstrukturen
- Stacks, Queues, Listen (mit Optimierungen)
- Graphen & verschiedene Algorithmen auf Graphen
- Verschiedene Bäume mit jeweiligen Vor- und Nachteilen
- Hashmaps und Sondierungsstrategien
- Monte-Carlo- und Las-Vegas-Algorithmen
- Evolutionäre Algorithmen
- Verschlüsselungsalgorithmen und Datenschutz
- Dezentrale Software und Blockchain-Datenstrukturen

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

Keine

Art der Prüfung

Schriftliche Prüfung (sP) gemäß
§ 23 APO

Dauer/Form der Prüfung

90 Minuten

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

Keine

Lernergebnisse

- Die Studierenden benennen und charakterisieren grundlegende Datenstrukturen und Algorithmen, einschließlich graph- und baumbasierter Verfahren.
- Die Studierenden erläutern typische Einsatzszenarien von Algorithmen anhand praxisnaher Beispiele.
- Die Studierenden wählen für gegebene Problemstellungen geeignete Datenstrukturen und Algorithmen aus.
- Die Studierenden analysieren die Leistung und Skalierbarkeit eingesetzter Algorithmen.
- Die Studierenden implementieren Algorithmen in einer Programmiersprache und testen deren Funktionalität.
- Die Studierenden bewerten verschiedene Lösungsansätze im Hinblick auf Effizienz und Anwendbarkeit.
- Die Studierenden entwickeln eigenständig algorithmische Lösungen für spezifische Anwendungsfälle.

Literatur

- Saake, Gunter; Sattler, Kai-Uwe: Algorithmen und Datenstrukturen, eine Einführung mit Java; 6. überarb. Aufl.; dpunkt-Verlag; Heidelberg, 2020
- Cormen, T., Leiseren, C., Riverest, R., Stein, C.: Algorithmen – Eine Einführung; 3. Aufl.; Oldenburg Verlag, 2010
- Fertig, Tobias; Schütz, Andreas. Blockchain the Comprehensive Guide. Rheinwerk Computing, 2024

Modulprofil

Prüfungsnummer

1510900

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Sommersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Prof. Dr. Isabel John

Dozierende

Prof. Dr. Isabel John,

Prof. Dr.-Ing. Anne Heß

Verwendbarkeit

BSED

Studiensemester

2. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

keine

Empfohlene Voraussetzungen

- Introduction to Software Engineering
- Programmieren 1

Inhalte

Dieses Modul baut auf den Grundlagen aus "Introduction to Software Engineering" auf und vermittelt Methoden und Techniken zur systematischen Analyse und dem strukturierten Entwurf von Softwaresystemen. Studierende lernen, komplexe Anforderungen zu analysieren, Softwarearchitekturen zu entwerfen und modellgestützte Entwicklungsansätze anzuwenden. Folgende Themen werden behandelt:

Anforderungsanalyse und Spezifikation

- Erhebung, Dokumentation und Validierung von Anforderungen
- Anwendung von Use Cases, User Stories und Szenarien
- Techniken zur Strukturierung und Priorisierung von Anforderungen

Grundlagen des Design

- SOLID-Prinzipien und Clean Code Design
- Design Patterns (z. B. Factory, Singleton, Observer)
- Modellierung und Entwurf von Softwarearchitekturen mit UML
- Datenmodellierung und Schnittstellenentwurf
- Detaillierte UML-Modellierung (z. B. Komponentendiagramme, Sequenzdiagramme, Deployment Diagramme)
- Einführung in Architekturmuster wie MVC und Standardarchitekturen (Schichten, Client-Server, Microservices)
- Abstraktionsebenen: Vom logischen zum physischen Design

Werkzeuge und Dokumentation

- Nutzung von CASE-Tools zur Modellierung
- Dokumentationstechniken für Architekturentscheidungen

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

Keine

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Portfolio

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

Keine

Lernergebnisse

- Die Studierenden kennen verschiedene Aspekte der Anforderungsanalyse
- Die Studierenden analysieren Anforderungen systematisch, strukturieren sie und dokumentieren diese.
- Die Studierenden konzipieren Softwarearchitekturen basierend auf Entwurfsmustern.
- Die Studierenden entwerfen Datenbank- und API-Schnittstellen für Anwendungen.
- Die Studierenden dokumentieren und präsentieren Architekturentscheidungen begründet.
- Die Studierenden kennen unterschiedliche Architekturmuster und wenden sie gezielt an.
- Die Studierenden beherrschen Methoden des objektorientierten Designs im Softwareentwurf.

Literatur

- Sommerville, Ian: Software Engineering, Pearson, 2018
- Oestereich, Bernd: Analyse und Design mit UML 2.5, Oldenbourg, 2013/2020
- Rupp, Chris: UML glasklar, Hanser, 2012
- McLaughlin: Objektorientierte Analyse und Design von Kopf bis Fuß, O'Reilly, 2017

Modulprofil

Prüfungsnummer

1511000

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Sommersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Englisch

Organisation

Modulverantwortung

Prof. Dr. Rolf Schillinger

Dozierende

Prof. Dr. Rolf Schillinger

Verwendbarkeit

BSED

Studiensemester

2. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

Keine

Empfohlene Voraussetzungen

Grundlagen Informatik

Inhalte

In this module, students gain a foundational understanding of networking technologies, their relevance for software engineering, and key security aspects of interconnected systems.

The course follows a layered approach, focusing on the four layers of the Internet Protocol Suite (TCP/IP) as the foundation of modern networking. Key topics include:

- Foundations: Basic networking concepts, ISO/OSI reference model as the conceptual grounding for modern networking
- TCP/IP concepts: Link layer and its protocols like Ethernet (IEEE 802.3) and Wi-Fi (IEEE 802.11), auxiliary protocols like (R)ARP, NDP, Internet layer with IPv4, IPv6, ICMP, IPsec, and routing basics (IGP, BGP), Transport layer with TCP and UDP, short introduction to QUIC, Application protocols with HTTP, HTTPS, DNS, and DHCP
- Networking devices, for example, NICs, hubs, switches, bridges, and routers
- Tools like Wireshark, network simulation via GNS3
- Mobile networking technologies such as 4G, LTE, and 5G
- Security considerations across all networking layers like ARP spoofing, IPsec, TLS attacks, firewalls, IDS and IPS

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

Keine

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Portfolio

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Englisch

Voraussetzung für die Vergabe von Leistungspunkten

Keine

Lernergebnisse

- Students will be able to describe and explain the principles of layered network architectures
- Students will be able to define and explain the key functions of each of TCP/IP's layers
- Students will be able to design a scalable best-practice internetworking architecture for a fictional global company, incorporating appropriate link layer protocols, network hardware, subnetting, and routing protocols
- Students will be able to construct and operate a medium-sized TCP/IP network in a virtual lab environment
- Students will be able to evaluate a given network's structure and assess its effectiveness in meeting specified functional or performance requirements

Literatur

- Tanenbaum, A. S., Wetherall, D. J., & Feamster, N. (2021). Computer networks (6th Global ed.). Pearson Education.
- Kurose, J. F., & Ross, K. W. (2021). Computer networking: A top-down approach (8th ed.). Pearson.
- Stevens, W. R., & Fall, K. R. (2011). TCP/IP illustrated, Volume 1: The protocols (2nd ed.). Addison-Wesley Professional.
- Stallings, W. (2017). Network security essentials: Applications and standards (6th ed.). Pearson.
- Grigorik, I. (2013). High performance browser networking: What every web developer should know about networking and web performance. O'Reilly Media.

Modul: 1510800

Programmieren 2

Modulprofil

Prüfungsnummer

1510800

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Semester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Prof. Dr. Peter Braun

Dozierende

Prof. Dr. Peter Braun

Verwendbarkeit

BSED

Studiensemester

2. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

keine

Empfohlene Voraussetzungen

Programmieren 1

Inhalte

Das Modul "Programmieren 2" baut auf den im ersten Semester erworbenen Kenntnissen in Python und objektorientierter Programmierung (OOP) auf und führt in die Programmiersprache Kotlin ein. Dabei werden sowohl die Sprachkonzepte von Kotlin als auch moderne Softwareentwicklungsprinzipien vermittelt. Die Schwerpunkte des Moduls sind:

- Syntax und Konzepte von Kotlin: Variablen, Kontrollstrukturen, Funktionen und Lambdas
- Objektorientierte Programmierung in Kotlin: Klassen, Objekte, Vererbung, Interfaces und Datenklassen
- Funktionale Programmieransätze: Higher-Order-Funktionen, Immutability und Collections-APIs
- Kotlin-spezifische Konzepte: Null-Safety, Extension Functions, Smart Casts und Coroutines
- Testgetriebene Entwicklung (TDD) mit Kotlin und Unit-Tests
- Fehlerbehandlung mit Exceptions und idiomatischen Ansätzen in Kotlin
- Einführung in moderne Softwareentwicklung mit Kotlin: Nutzung von Build-Tools (Gradle), Abhängigkeitsmanagement, einfache Anwendung von Design Patterns
- Projektarbeit: Umsetzung einer praxisnahen Anwendung unter Verwendung von Kotlin
- Praktische Übungen und Projekte begleiten die theoretischen Inhalte, sodass die Studierenden eigenständig Kotlin-Programme entwickeln und deren Qualität durch Tests und Code-Reviews verbessern können.

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

keine

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Portfolio

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

keine

Lernergebnisse

Nach erfolgreichem Abschluss des Moduls sind die Studierenden in
der Lage:

- Grundlegende und fortgeschrittene Kotlin-Sprachkonzepte zu verstehen und anzuwenden.
- Objektorientierte Prinzipien in Kotlin zu nutzen und eigene Klassenhierarchien zu entwerfen.
- Funktionale Programmierparadigmen in Kotlin anzuwenden, um sauberen und effizienten Code zu schreiben.
- Null-Sicherheit und Typsysteme von Kotlin zu verstehen und effektiv einzusetzen.
- Testgetriebene Entwicklung (TDD) und Unit-Testing mit Kotlin durchzuführen, um robuste Software zu entwickeln.
- Effiziente Fehlerbehandlung unter Nutzung idiomatischer Kotlin-Techniken zu implementieren.
- Kotlin-Projekte mit modernen Build-Tools (z. B. Gradle) zu verwalten und Abhängigkeiten sinnvoll einzubinden.
- Eine praxisnahe Softwarelösung in Kotlin zu konzipieren, zu implementieren und zu dokumentieren.
- Bestehenden Kotlin-Code zu analysieren und auf Qualität, Lesbarkeit und Effizienz zu überprüfen.
- Die Unterschiede und Gemeinsamkeiten zwischen Python und Kotlin zu reflektieren und für unterschiedliche Anwendungsfälle abzuwägen.

Literatur

- Kofler, Michael. Kotlin: Das Umfassende Handbuch. 1. Auflage. Bonn: Rheinwerk, 2021.
- Szwillus, Karl. Kotlin: Einstieg Und Praxis. 1. Auflage. Frechen: mitp, 2020.

Modul: 1510700

Projekt 1

Modulprofil

Prüfungsnummer

1510700

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Sommersemester

SWS

1

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 0 Std.

Selbststudienzeit: 135 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Projekt

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Prof. Dr. Peter Braun

Dozierende

Prof. Dr. Peter Braun,

Prof. Dr. Isabel John,

Michael Rott,

Prof. Dr. Rolf Schillinger,

Prof. Dr.-Ing. Tobias Fertig,

Prof. Dr. Frank-Michael Schleif,

Prof. Dr.-Ing. Sebastian

Biedermann,

Prof. Dr. Tristan Wimmer,

Prof. Dr.-Ing. Anne Heß

Verwendbarkeit

BSED

Studiensemester

2. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

Keine

Empfohlene Voraussetzungen

Programmieren 1, IT-Projektmanagement und BWL, Introduction to Software Engineering, Grundlagen Informatik

Inhalte

Die Studierenden entwickeln in einer Gruppe mit drei bis vier Personen eine Konsolenanwendung gemäß vorgegebener Anforderungen. Die Anwendung soll einen nicht-trivialen Algorithmus enthalten (z. B. Such-, Sortier-, Graphen- oder Optimierungsalgorithmen) und mit Dateien arbeiten. Die Software besteht aus mehreren Modulen, jedoch ohne eine umfassende Softwarearchitektur. Zu den zentralen Themen gehören:

- Modularisierung des Codes
- Effiziente Implementierung eines Algorithmus
- Nutzung von Versionskontrolle (z. B. Git)
- Grundlegende Code-Dokumentation (README, Funktionskommentare)
- Einfache Tests zur Validierung des Algorithmus
- Einfache Aufgabenverwaltung und -verteilung im Team

Die Studierenden werden während der Projektbearbeitung kontinuierlich von einer Dozentin oder einem Dozenten betreut. Die Betreuung erfolgt durch regelmäßige Treffen, in denen der Arbeitsfortschritt besprochen wird, sowie durch begleitende Zwischenpräsentationen, in denen die Entwicklung des Projekts reflektiert und weiterentwickelt wird.

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

Keine

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Präsentation, Dokumentation

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

Keine

Lernergebnisse

Nach dem erfolgreichen Abschluss des Moduls haben die
Studierenden die folgenden Kompetenzen:

- Die Studierenden erinnern die grundlegenden Konzepte der Softwareentwicklung und Versionskontrolle.
- Die Studierenden verstehen die Prinzipien modularer Softwareentwicklung und die Strukturierung von Projekten.
- Die Studierenden wenden Git und Versionskontrolle in einem Teamkontext an.
- Die Studierenden analysieren nicht-triviale Algorithmen hinsichtlich Laufzeit und Effizienz.
- Die Studierenden bewerten verschiedene Implementierungsansätze für eine gegebene Problemstellung.
- Die Studierenden erstellen eine kleine, modulare Konsolenanwendung nach vorgegebenen Anforderungen.
- Die Studierenden erstellen eine technische Präsentation zur Vorstellung ihrer Lösung.

Literatur

Wird in Abhängigkeit von den konkreten Projekten und verwendeten Technologie in der Veranstaltung bekannt gegeben.

Modulprofil

Prüfungsnummer

1511200

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Sommersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Prof. Dr. Patrik Stilgenbauer

Dozierende

Prof. Dr. Patrik Stilgenbauer

Verwendbarkeit

BSED

Studiensemester

2. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

keine

Empfohlene Voraussetzungen

Algebra: Grundkenntnisse der linearen Algebra (insb. lineare Gleichungssysteme, Matrizenalgebra, Vektorräume, Skalarprodukt), Aussagen- und Mengenalgebra, Kombinatorik.

Programmieren I: Programmierlogik, Entwurf einfacher Algorithmen. Schulkenntnisse aus der Analysis vorausgesetzt (Differential- und Integralrechnung).

Inhalte

- Deskriptive Statistik: Grundbegriffe, Häufigkeitsverteilungen, Lageparameter, Streuungsparameter, lineare Korrelations- und Regressionsrechnung.
- Wahrscheinlichkeitstheorie: Ergebnismenge, Ereignisse, Wahrscheinlichkeitsbegriff von Kolmogorow, bedingte Wahrscheinlichkeit und Unabhängigkeit, diskrete und stetige Zufallsvariablen, Erwartungswert und Varianz, Gesetz der großen Zahlen, Binomialverteilung, hypergeometrische Verteilung, Poissonverteilung, Exponentialverteilung, Normalverteilung, Summen von Zufallsvariablen, zentraler Grenzwertsatz.
- Schließende Statistik: Punkt- und Intervallschätzungen, Signifikanztests.
- Anwendung und Visualisierung stochastischer Methoden durch Programmierbeispiele in Python oder R.

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

Keine

Art der Prüfung

Schriftliche Prüfung (sP) gemäß
§ 23 APO

Dauer/Form der Prüfung

90 Minuten

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

Keine

Lernergebnisse

Nach Abschluss des Moduls sind die Studierenden in der Lage:

- grundlegende Konzepte und Begriffe der Wahrscheinlichkeitsrechnung und Statistik sicher anzuwenden,
- statistische Kennzahlen und Wahrscheinlichkeiten zu berechnen, zu interpretieren und zur Beschreibung von Daten einzusetzen,
- typische Wahrscheinlichkeitsverteilungen auszuwählen und auf geeignete, praxisnahe Problemstellungen anzuwenden,
- stochastische und statistische Fragestellungen mit logisch-strukturiertem Denken zu analysieren und lösungsorientiert zu bearbeiten,
- die Methoden als Grundlage für weiterführende Anwendungen in Softwareentwicklung, Data Science und Machine Learning einzusetzen.

Literatur

- Bamberg, G., Baur, F. und Krapp, M.: Statistik, De Gruyter Oldenbourg, 2022.
- Bourier, G.: Beschreibende Statistik, Springer Gabler, 2025.
- Bourier, G.: Wahrscheinlichkeitsrechnung und schließende Statistik, Springer Gabler, 2018.
- Dreiseitl, S.: Mathematik für Software Engineering, Springer Vieweg, 2018.
- Henze, N.: Stochastik für Einsteiger, Vieweg + Teubner, 2011.
- Kurt, N.: Stochastik für Informatiker, Springer Vieweg, 2020.
- Teschl, G. und Teschl, S.: Mathematik für Informatiker - Band 2 (Analysis und Stochastik), Springer Vieweg, 2014.

3. Semester

Modulprofil

Prüfungsnummer

1511600

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Wintersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Englisch

Organisation

Modulverantwortung

Prof. Dr. Peter Braun

Dozierende

Prof. Dr. Peter Braun

Verwendbarkeit

BSED

Studiensemester

3. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

None

Empfohlene Voraussetzungen

Programmieren 1 und Programmieren 2

Inhalte

- Introduction to distributed systems, client-server, and peer-to-peer systems.
- Software architectures for backend systems (3-tier, hexagonal, monolithic vs. micro-service, event-driven)
- Frameworks to implement backend systems (e.g. Spring)
- Advanced database techniques, scalability, replication, sharding, ORM-tools, query caching, CAP theorem
- Protocols for remote procedure call, for example, GraphQL and Google RPC.
- Basics of the HTTP protocol and application in the form of Web APIs.
- Comprehensive introduction to the REST architecture principle: resources, URLs, CRUD, hypermedia, caching, security.
- Configuration of Web servers (Apache), load balancer, and public caches (nginx)
- Testing of backend systems, performance testing using JMeter, monitoring and logging
- Security aspects of network protocol and backend systems

In the traditional degree programme, the lecturer provides or agrees with the topics of the practical examples for the examination. In the dual study programme, the lecturer consults with the company on a task, ensuring practical relevance and feedback from the company.

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

None

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Portfolio

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Englisch

Voraussetzung für die Vergabe von Leistungspunkten

None

Lernergebnisse

- The students understand the fundamental concepts and differences of distributed systems, including their architecture and communication models.
- The students analyze various software architectures for backend systems and evaluate their suitability for different use cases.
- The students apply advanced database techniques such as replication and sharding to enhance data availability and performance.
- The students implement a backend system using a framework like Spring, following best practices for configuration, deployment, and security.
- The students compare different protocols for remote procedure calls, such as GraphQL and Google RPC, assessing their strengths and limitations.
- The students design RESTful APIs by applying the principles of the REST architecture, focusing on resources, URLs, CRUD operations, and security strategies.
- The students evaluate the security aspects of network protocols and backend systems, proposing improvements based on best practices.

Literatur

- Coulouris, J. Dollimore, and T. Kindberg, Distributed Systems: Concepts and Design (4th Edition) (International Computer Science). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.
- N. Biswas, Practical GraphQL: Learning Full-Stack GraphQL Development with Projects. Berkeley, CA: Apress, 2023.
- J. Webber, S. Parastatidis, und I. Robinson, REST in practice: hypermedia and systems architecture, 1. ed. in Theory in practice. Beijing Köln: O'Reilly, 2010.
- L. Richardson und M. Amundsen, RESTful Web APIs, First edition, Second release. Beijing Cambridge Farnham Köln Sebastopol Tokyo: O'Reilly, 2015.
- I. Dominte, Web API Development for the Absolute Beginner: A Step-by-step Approach to Learning the Fundamentals of Web API Development with .NET 7. Berkeley, CA: Apress, 2023.

Modulprofil

Prüfungsnummer

1511700

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Wintersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Englisch

Organisation

Modulverantwortung

Prof. Dr. Frank-Michael Schleif

Dozierende

Prof. Dr. Frank-Michael Schleif

Verwendbarkeit

BSED

Studiensemester

3. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

None

Empfohlene Voraussetzungen

Datenbanken, Programmieren 1, Introduction to Software Engineering

Inhalte

Most of the work in data science, and most of the risk, lies before a model is fitted. This module is about that part: how data comes to be, what is wrong with it, what may legitimately be concluded from it, and how to say so in a way that survives a question. No models are trained; the module ends where a machine learning course begins. The subject matter is worked on one real production database rather than a prepared teaching example, so the defects encountered are the defects that occur.

Core Topics:

- Data and its origin: data, information, knowledge, decision; scale levels and the operations each permits; provenance and what it implies for quality.
- Tabular data: the tidy data principle and its violations; reshaping; join types and cardinality; referential integrity on write and on read.
- From query to pipeline: operative versus dispositive data; database interfaces; parameterised queries; repeatable extraction with provenance.
- Formats, metadata and protection: CSV, XML, JSON, YAML, Parquet and HDF5; encoding and quoting defects; schema on write versus schema on read; querying hierarchical data; data dictionary, FAIR, personal data and the distance to anonymity.
- Data quality: systematic profiling; missingness as MCAR, MAR and MNAR; defect classes by the effort detection and correction require; outliers; selection, survivorship and labelling bias; cleaning decisions recorded with their risk.

- Data beyond the table: time series as a distinct data class (resampling, aggregation windows, gaps, smoothing); graph models (neighbourhood, centrality, components, querying a property graph).
- Features, similarity and distance: encoding and scaling; Euclidean, cosine, Jaccard, dynamic time warping and edit distance; metric axioms; feature engineering as the choice of an embedding.
- Reduction and structure: PCA, explained variance and loadings; what t-SNE optimises and what may never be read off its output; k-means and spectral clustering.
- Exploratory analysis and visualisation: location and spread suited to the distribution; Pearson against Spearman, correlation against causation; chart form chosen from the question; misleading axes and scales; validated colour.
- Communication and readiness: dashboard against analysis tool; what a model expects; splits and data leakage; readiness criteria and defending a decision.

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

None

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Portfolio

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Englisch

Voraussetzung für die Vergabe von Leistungspunkten

None

Lernergebnisse

Upon successful completion, students will be able to:

Knowledge and Understanding:

- Explain the consequences a scale level has for the operations it permits
- Explain the tidy data principle, and what a join cardinality does to rows
- Distinguish MCAR, MAR and MNAR, and identify selection, survivorship and labelling bias
- Explain what a model sees, and what data leakage is and how it arises

Skills and Competences:

- Turn a query into a repeatable, parameterised extraction with a provenance record
- Profile a dataset for types, completeness, cardinality, duplicates and plausibility
- Work with time series and graph data using the operations each requires
- Choose and apply an encoding, a scaling and a distance measure suited to the features
- Judge which statements a projection, a clustering or a dashboard supports and which it does not
- Produce a documented analysis dataset with a stated unit of observation and a cleaning log, and defend a go or no go verdict on its readiness

Literatur

- Skiena (2017), The Data Science Design Manual, Springer (available through the university library)
- VanderPlas (2023), Python Data Science Handbook, 2nd edition, O'Reilly
- McKinney (2022), Python for Data Analysis, 3rd edition, O'Reilly
- Wickham (2014), Tidy Data, Journal of Statistical Software 59(10)
- Wilkinson et al. (2016), The FAIR Guiding Principles for Scientific Data Management and Stewardship, Scientific Data 3
- Robinson, Webber and Eifrem (2015), Graph Databases, 2nd edition, O'Reilly
- Munzner (2014), Visualization Analysis and Design, CRC Press
- Kaufman, Rosset and Perlich (2012), Leakage in Data Mining, ACM Transactions on Knowledge Discovery from Data 6(4)
- Tukey (1977), Exploratory Data Analysis, Addison-Wesley

Modulprofil

Prüfungsnummer

1511800

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Wintersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Prof. Dr. Isabel John

Dozierende

Prof. Dr. Peter Braun,

Prof. Dr. Isabel John,

Prof. Dr.-Ing. Anne Heß

Verwendbarkeit

BSED

Studiensemester

3. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

keine

Empfohlene Voraussetzungen

keine

Inhalte

In diesem Modul werden den Studierenden theoretische und praktische Kenntnisse und Fähigkeiten vermittelt, die in einem professionellen Arbeitsumfeld in verschiedensten Bereichen Anwendung finden.

Dabei erlernen und erproben die Studierenden eine Reihe von Methoden, Techniken, und Tools, die in verschiedene Schwerpunktthemen eingeordnet sind. Dazu gehören

- Lern- und Arbeitstechniken
- Wissenschaftliches Arbeiten
- Zielgruppenorientierte fachliche Kommunikation sowie
- Arbeiten in (internationalen) Teams

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

keine

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Portfolio

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

keine

Lernergebnisse

- Die Studierenden wenden Methoden zur effektiven Planung und Strukturierung ihrer Arbeitsprozesse an
- Die Studierenden benennen die Grundprinzipien des wissenschaftlichen Arbeitens in der Informatik
- Die Studierenden führen eine Literaturrecherche durch und organisieren die Ergebnisse
- Die Studierenden beschreiben detailliert die Grundlagen zur Gestaltung effektiver wissenschaftlicher Kommunikation in Form von Texten, Präsentationen, Poster, Videos
- Die Studierenden erstellen wissenschaftlich-orientierte Texte, Präsentationen, Poster, Videos auf der Basis wissenschaftlicher Standards
- Die Studierenden können KI Tools als Hilfsmittel bei der Wissenschaftlichen Recherche und beim Wissenschaftlichen Schreiben einsetzen und kritisch hinterfragen
- Die Studierenden erlernen relevante Zielsetzungen, Fähigkeiten und Best Practices zur Planung, Durchführung und Nachbereitung verschiedener Befragungstechniken (wie Interviews, Umfragen)
- Die Studierenden wenden Methoden zur effektiven Kommunikation und zur Gesprächsführung in Teams an

Literatur

Wird in Vorlesung bekanntgegeben

Modulprofil

Prüfungsnummer

1511300

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Wintersemester

SWS

1

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 0 Std.

Selbststudienzeit: 135 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Projekt

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Prof. Dr. Peter Braun

Dozierende

Prof. Dr. Peter Braun,

Prof. Dr. Isabel John,

Michael Rott,

Prof. Dr. Rolf Schillinger,

Prof. Dr.-Ing. Tobias Fertig,

Prof. Dr. Frank-Michael Schleif,

Prof. Dr.-Ing. Sebastian

Biedermann,

Prof. Dr. Tristan Wimmer,

Prof. Dr.-Ing. Anne Heß

Verwendbarkeit

BSED

Studiensemester

3. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

keine

Empfohlene Voraussetzungen

Programmieren 1, Projekt 1, Programmieren 2, Introduction to Software Engineering, Analyse und Design, Algorithmen und Datenstrukturen

Inhalte

In Gruppen von vier bis fünf Personen entwickeln die Studierenden eine Anwendung mit dem Fokus auf das Backend und die Datenbank. Sie wenden Elemente des agilen Projektmanagement an. Die Studierenden erhalten ein Thema und ermitteln mit der betreuenden Personen die Anforderungen. Eine grafische Nutzeroberfläche ist nicht notwendig bzw. sollte sehr einfach gehalten werden. Die Software muss eine erkennbare Software Architektur (z.B. Schichten, Pipeline, Hexagonal) aufweisen. Die Qualitätssicherung erfolgt durch Unit-Tests. Die Studierenden arbeiten mit:

- Datenbank-Anbindung mit Schema-Entwurf (SQL)
- Erweiterten Software-Entwurfsmethoden (z. B. UML, Architektur-Patterns)
- Versionskontrolle mit Git (Branching, Pull Requests, Code Reviews)
- Sprint-basierten Entwicklungsprozessen (z. B. Scrum, Kanban)
- Die Dokumentation umfasst Architekturdiagramme, User Stories und eine automatische API-Dokumentation.
- Maßnahmen zur Qualitätssicherung durch automatisierte Unit-Tests

Die Studierenden werden während der Projektbearbeitung kontinuierlich von einer Dozentin oder einem Dozenten betreut. Die Betreuung erfolgt durch regelmäßige Treffen, in denen der Arbeitsfortschritt besprochen wird, sowie durch begleitende Zwischenpräsentationen, in denen die Entwicklung des Projekts reflektiert und weiterentwickelt wird.

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

keine

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Dokumentation, Präsentation

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

keine

Lernergebnisse

Nach erfolgreichem Abschluss des Moduls besitzen die Studierenden
die folgenden Kompetenzen:

- Die Studierenden verstehen grundlegende Architekturprinzipien wie das Schichtenmodell und die Modularisierung.
- Die Studierenden wenden das Schichtenmodell und Modularisierungskonzepte bei der Entwicklung einer verteilten Anwendung mit Datenbankanbindung an.
- Die Studierenden wenden agile Methoden an, um Aufgaben in Sprints zu planen und umzusetzen.
- Die Studierenden analysieren Code-Änderungen im Rahmen von Pull Requests und führen Code Reviews im Team durch.
- Die Studierenden erstellen eine strukturierte Software-Dokumentation für ihre Anwendung.
- Die Studierenden erstellen eine technische Projektpräsentation zur Vorstellung ihrer Ergebnisse.

Literatur

Wird in Abhängigkeit von den konkreten Projekten und verwendeten Technologie in der Veranstaltung bekannt gegeben.

Modulprofil

Prüfungsnummer

1511500

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Wintersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Deutsch

Organisation

Modulverantwortung

Prof. Dr.-Ing. Anne Heß

Dozierende

Prof. Dr.-Ing. Tobias Fertig,

Prof. Dr.-Ing. Anne Heß

Verwendbarkeit

BSED

Studiensemester

3. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

keine

Empfohlene Voraussetzungen

Introduction to Software Engineering

Analyse und Design

Inhalte

Dieses Modul leistet eine umfassende Einführung in den Themenkomplex Softwarequalität. Dabei werden sowohl technische Grundlagen als auch Prozess-, Management- und Usability Themen näher gebracht.

Die Inhalte umfassen die folgenden Themen:

- Softwarequalität & Softwarequalitätsmanagement
- Analytische Qualitätssicherung (Testen von Software)
- Strategien zur Testfallermittlung (Black-Box Testing, White-Box Testing)
- Testarten (Funktionales Testen, Regressionstests, Performance Tests)
- Test-Driven Development
- Testing Tools
- Bug Tracking und Reporting
- Statisches Testen (Reviews & Statische Analyse)
- Konstruktive Qualitätssicherung am Beispiel Usability / User Experience
- Human-Centered Design (Prinzipien und Kernaktivitäten (Fokus auf Interaction Design & Usability Evaluation))

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

keine

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Portfolio

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Deutsch

Voraussetzung für die Vergabe von Leistungspunkten

keine

Lernergebnisse

- Die Studierenden verstehen Begrifflichkeiten der Softwarequalität anhand eines Qualitätsmodells
- Die Studierenden können Qualitätsmodelle zur systematischen Ermittlung von Qualitätsanforderungen anwenden
- Die Studierenden verstehen relevante Aktivitäten und Artefakte im Testprozess
- Die Studierenden können im Kontext analytischer Qualitätssicherung Teststrategien entwickeln, Testfälle definieren und Testüberdeckungen ermitteln
- Die Studierenden können Testmethoden für funktionale und nicht-funktionale Anforderungen auswählen und anzuwenden.
- Die Studierenden können ein geeignetes Testmanagement etablieren und individuell auf die Anforderungen von Projekten abstimmen
- Die Studierenden verstehen die Bedeutung von Usability und User Experience
- Die Studierenden verstehen Zielsetzungen und Techniken zur konstruktiven Qualitätssicherung basierend auf dem Human-Centered Design Prozess
- Die Studierenden können die Techniken im Human-Centered Design zur Sicherstellung einer guten Usability / User Experience anwenden

Literatur

- Liggesmeyer, P. (2009). Software-Qualität. Springer-Verlag.
- Spillner, A., & Linz, T.: Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester – Foundation Level nach ISTQB®-Standard. dpunkt.verlag, 7. Auflage. 2024
- Schneider, K. (2012). Abenteuer Softwarequalität: Grundlagen und Verfahren für Qualitätssicherung und Qualitätsmanagement (2. Aufl.). dpunkt.verlag.
- Geis, T. & Tesch, G.: Basiswissen Usability und User Experience: Aus- und Weiterbildung zum UXQB® Certified Professional for Usability and User Experience (CPUX) – Foundation Level (CPUX-F), dpunkt.verlag GmbH, 2. Auflage 2023
- Holtzblatt, K., Beyer, H.: Contextual Design: Design for Life (Interactive Technologies), Morgan Kaufmann, Rheinwerk Computing, 2. Auflage, 2016
- Jacobsen, J., & Meyer, L.: Praxisbuch Usability und UX: Bewährte Usability- und UX-Methoden praxisnah erklärt, 4. Auflage, 2024 (5. Auflage erscheint im Dezember 2026)

Modul: 1511400

System-Oriented Programming

Modulprofil

Prüfungsnummer

1511400

Dauer

1 Semester

Häufigkeit des Angebots

Jedes Wintersemester

SWS

4

ECTS-Credits (CP)

5.0

Workload

Angeleitete Studienzeit:

Synchrone Teilnahme: 1 Std.

Asynchrone Teilnahme: 3 Std.

Selbststudienzeit: 90 Std.

Gesamt: 150 Std.

Lehrveranstaltungsart(en)

Seminaristischer Unterricht,
Übung

Lehrsprache

Englisch

Organisation

Modulverantwortung

Prof. Dr. Peter Braun

Dozierende

Prof. Dr. Peter Braun

Verwendbarkeit

BSED

Studiensemester

3. Semester

Art des Moduls

Pflichtmodul

Verpflichtende Voraussetzungen gemäß SPO

None

Empfohlene Voraussetzungen

Programmieren 1 and Programmieren 2

Inhalte

- Definition and meaning of system-oriented programming
- Using the command line of an operating system
- Shell programming using the example of Bash
- Working on remote computers with ssh
- Data processing on the command line with sed, awk, sort, jq
- Using AI assistance systems (e.g. GitHub Copilot, ChatGPT)
- Editing text documents with vim
- Using the version control system Git
- Introduction to the C programming language (syntax, data types, pointers, memory management)
- Build systems make, cmake, bazel
- System programming under Linux (system calls, handling files, processes)
- Debugging, profiling and performance optimisation (gdb, strace, ltrace, gprof)
- Security aspects of system-related programming and protection mechanisms
- Structure of the Linux operating system
- Processes, process management, scheduling
- Inter-process communication, race conditions, deadlocks, semaphores, Petri nets and deadlock detection, philosopher problem, producer-consumer problem
- Memory management, memory abstraction, partitioning, fragmentation, free memory management, virtual memory, page exchange algorithms
- Input and output, direct memory access, interrupts, hard disks, file systems for hard disks
- Backup methods, automation, data integrity
- Network communication and implementation of network protocols
- Hypervisor technologies, Docker containers, resource management

Prüfung

Verpflichtende Voraussetzung gemäß SPO für die Teilnahme an der Prüfung

None

Art der Prüfung

Sonstige Prüfung (soP) gemäß
§§ 26, 27 APO

Dauer/Form der Prüfung

Portfolio

Die konkrete Festlegung der
abzuleistenden Prüfung erfolgt
im Studienplan

Prüfungssprache

Englisch

Voraussetzung für die Vergabe von Leistungspunkten

None

Lernergebnisse

- The students understand the definition and principles of system-oriented programming, including its role in software development.
- The students demonstrate proficiency in using the command line of an operating system, applying advanced tools like sed, awk, sort, and jq for data processing tasks.
- The students develop shell scripts in Bash to automate system tasks and streamline operations effectively.
- The students utilize AI assistance systems, such as GitHub Copilot and ChatGPT, to improve coding efficiency and solve programming challenges.
- The students manage text documents using the vim text editor, employing advanced editing and configuration techniques.
- The students implement version control practices with Git to support collaborative software development workflows.
- The students program in C, focusing on syntax, data types, pointers, memory management, and use debugging and profiling tools like gdb, strace, ltrace, and gprof to analyze code performance.
- The students evaluate security aspects of system-related programming and apply protection mechanisms to ensure code security.

Literatur

- D. J. Barrett, Efficient Linux at the command line: boost your command-line skills, First edition. Sebastopol, CA: O'Reilly, 2022.
- A. S. Tanenbaum und H. Bos, Modern operating systems, 4th ed. Boston: Prentice Hall, 2015.
- K. Hitchcock, Linux System Administration for the 2020s: The Modern Sysadmin Leaving Behind the Culture of Build and Maintain. Berkeley, CA: Apress, 2022.
- M. Kalin, Modern C Up and Running: A Programmer's Guide to Finding Fluency and Bypassing the Quirks. Berkeley, CA: Apress, 2022.
- K. Hitchcock, The Enterprise Linux Administrator: Journey to a New Linux Career. Berkeley, CA: Apress, 2023.
- J. Varma, Pro Bash: Learn to Script and Program the GNU/Linux Shell. Berkeley, CA: Apress, 2023.
- S. M. Palakollu, Practical System Programming with C: Pragmatic Example Applications in Linux and Unix-Based Operating Systems. Berkeley, CA: Apress, 2021.