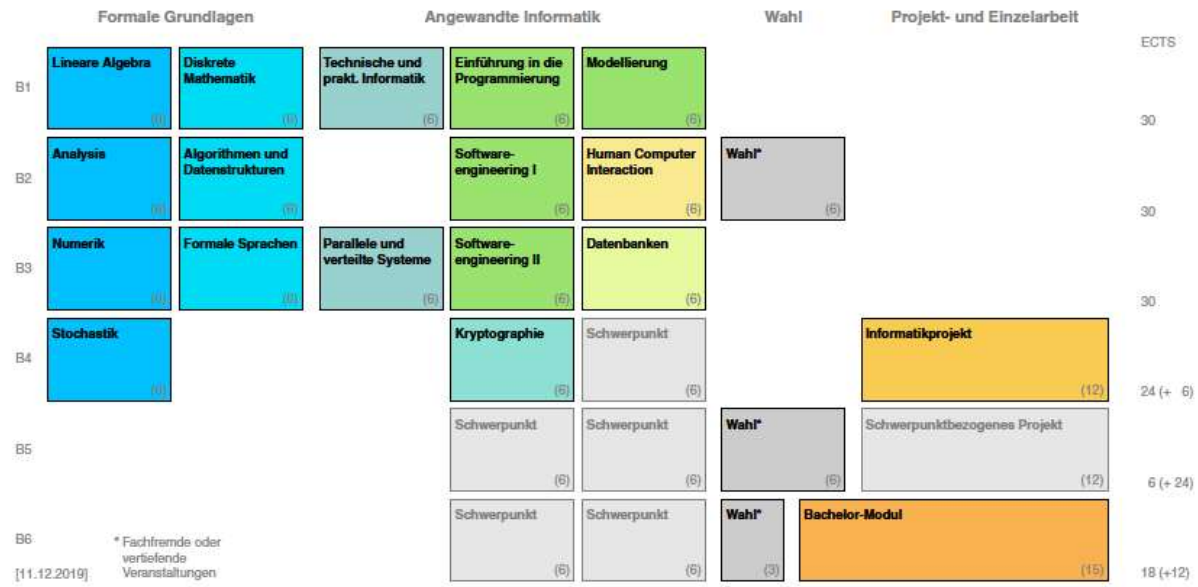
Modulkatalog Informatik

Stand: Februar 2021

Der Bachelor-Studiengang Informatik mit den Schwerpunkt-Bereichen Medieninformatik und Security and Data Science ist forschungs- bzw. grundlagenorientiert, dauert 6 Semester (180 ECTS) und bereitet die Studierenden auf den Beruf des Informatikers/der Informatikerin mit einem Anwendungsschwerpunkt im Bereich der Digitalen Medien bzw. Sicherheit und Datenwissenschaft vor. Der Modulplan orientiert sich an den Empfehlungen der Gesellschaft für Informatik für sogenannte *Typ-2 Studiengänge*. Er ist untergliedert in Pflichtmodule, Wahl- bzw. Wahlpflichtmodule und die Anleitung zum wissenschaftlichen Arbeiten. Die folgende Abbildung gibt eine Übersicht über die Struktur des Studiums:



Schwerpunktübergreifende Pflichtmodule aus Informatik und Mathematik:

Der Studiengang beinhaltet 12 Pflichtmodule, die in den ersten 4 Semestern liegen und in denen die Grundlagen der Mathematik und Informatik vermittelt werden:

- Lineare Algebra (1. Semester)
- Analysis (2. Semester)
- Numerik (3. Semester)
- Stochastik (4. Semester)
- Diskrete Mathematik (1. Semester)
- Algorithmen und Datenstrukturen (2. Semester)
- Formale Sprachen (3. Semester)
- Technische und Praktische Informatik (1. Semester)
- Einführung in die Programmierung (1. Semester)

- Softwareengineering I (2. Semester)
- Softwareengineering II (3. Semester)
- Modellierung (1. Semester)

Schwerpunktübergreifende Anwendungsspezifische Pflichtmodule:
Im Lauf des Studiums werden zunehmend auch Kompetenzen mit einem dezidierten Anwendungsbezug zur Medieninformatik sowie zur Sicherheit und Data Science vermittelt. Dies geschieht im Rahmen Schwerpunkte sowie der Anleitung zur selbstständigen wissenschaftlichen Arbeit ab dem 4. Semester (siehe unten), aber auch schon ab dem 2. Semester im Rahmen der folgenden Pflichtmodule:

- Mensch-Maschine Interaktion (2. Semester)
- Parallele und Verteilte Systeme (ab dem 3. Semester)
- Datenbanken (3. Semester)
- Kryptographie (4. Semester)

Wahlpflicht- und Wahlmodule:

Ab dem 2. Semester ermöglichen die Wahlmodule (15 ECTS), den Studierenden, ihr Wissen durch den Besuch von Veranstaltungen aus anderen Bereichen und Fakultäten zu erweitern. Auch der Besuch von Englischveranstaltungen ist im Rahmen dieses Moduls möglich.

Anleitung zur selbstständigen wissenschaftlichen Arbeit:

Bereits ab Beginn der zweiten Studienhälfte werden Studierende im Rahmen von Projekten in Kleingruppen an das selbstständige wissenschaftliche Arbeiten herangeführt. Die Projekte dienen auch dem Erwerb von berufsrelevanten „Soft Skills“ wie Kommunikationskompetenz, Grundkenntnissen des Projektmanagements, etc. Im 6. Semester ist das Verfassen einer wissenschaftlichen Arbeit vorgesehen – natürlich unter Anleitung, durch einen Betreuer –, die, im Rahmen eines Vortrages vor den Mitgliedern der Fakultät, „verteidigt“ wird. Insgesamt erfolgt die Anleitung zur selbständigen wissenschaftlichen Arbeit in drei Modulen:

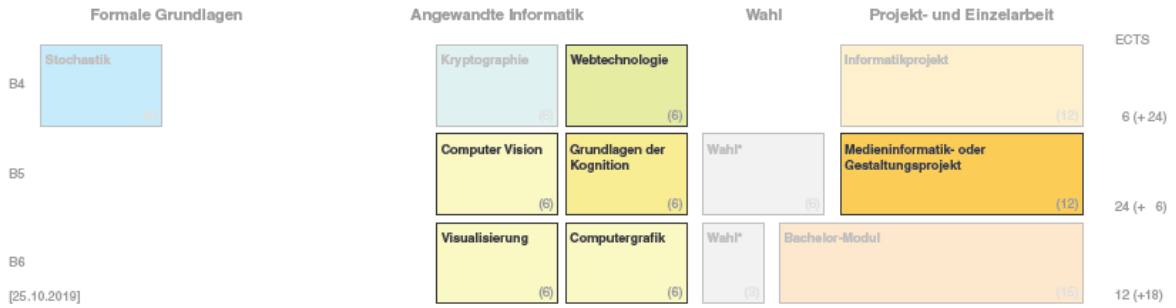
- erstes Projekt/Informatikprojekt (4. Semester)
- zweites Projekt/schwerpunktbezogenes Projekt (5. Semester)
- Bachelor-Modul (6. Semester)

Schwerpunkt-Bereiche:

Am Ende des dritten Semesters haben die Studierenden die Möglichkeit sich zwischen den beiden Schwerpunkt-Bereiche Medieninformatik und Security and Data Science zu entscheiden. Abhängig von dieser Wahl können die Studierenden unterschiedliche Anwendungsspezifische Pflichtmodule belegen. Des Weiteren soll im schwerpunktbezogenen Projekt vertieft auf das selbstständige wissenschaftliche Arbeiten zu Schwerpunkt spezifischen Fragestellungen eingegangen werden.

Schwerpunkt-Bereich Medieninformatik:

Die folgende Abbildung gibt eine Übersicht über die Struktur des Studiums ab dem 4. Semester bei der Wahl des Schwerpunkt-Bereichs Medieninformatik:



Spezifische Pflichtmodule des Schwerpunkt-Bereich Medieninformatik:

Neben den Schwerpunkt übergreifenden Pflichtmodulen wird der Schwerpunkt-Bereich Medieninformatik durch die folgenden für diesen Schwerpunkt spezifischen 5 Pflichtmodulen komplettiert:

- Webtechnologie (4. Semester)
- Grundlagen der Kognition (5. Semester)
- Computer Vision (5. Semester)
- Visualisierung (6. Semester)
- Computergraphik (6. Semester)

Schwerpunkt-Bereich Security and Data Science:

Die folgende Abbildung gibt eine Übersicht über die Struktur des Studiums ab dem 4. Semester bei der Wahl des Schwerpunkt-Bereichs Security and Data Science:

Spezifisches Pflichtmodul des Schwerpunkt-Bereich Security and Data Science:

Neben den Schwerpunkt übergreifenden Pflichtmodulen wird der Schwerpunkt-Bereich Security and Data Science durch ein Pflichtmodule ergänzt:

- Information und Codierung (4. Semester)

Spezifisches Wahlpflichtmodule des Schwerpunkt-Bereich Security and Data Science:

Neben den Schwerpunkt übergreifenden Pflichtmodulen und dem spezifischen Pflichtmodul wird der Schwerpunkt-Bereich Security and Data Science durch 4 Wahlpflichtmodule komplettiert:

- Ein Wahlpflichtmodul aus dem Bereich Theoretische Informatik.
- Ein Wahlpflichtmodul aus dem Bereich Advanced Security.
- Ein Wahlpflichtmodul aus dem Bereich Advanced Data Science.
- Ein Wahlpflichtmodul aus dem Bereich Grafische Informationssysteme.

Für das Wahlpflichtmodul aus dem Bereich *Theoretische Informatik* kann eine der folgenden Veranstaltungen aus dem Masterprogramm Computer Science for Digital Media gewählt werden:

- Advanced Numerical Mathematics
- Complexity Theory
- Discrete Optimisation
- Introduction to Functional Programming with Haskell
- Randomised Algorithms

Für das Wahlpflichtmodul aus dem Bereich *Advanced Security* kann eine der folgenden Veranstaltungen aus dem Masterprogramm Computer Science for Digital Media gewählt werden:

- Advanced Cryptography: Cryptographic Hash Functions
- Advanced Cryptography: Secure Channels
- Digital Watermarking & Steganography
- Quantum Algorithms & Cryptanalysis
- Security Engineering

Für das Wahlpflichtmodul aus dem Bereich *Advanced Data Science* kann entweder das Modul

- Webtechnologie (4./6. Semester)

oder eine der folgenden Veranstaltungen aus dem Masterprogramm Computer Science for Digital Media gewählt werden:

- Introduction to Machine Learning
- Introduction to Natural Language Processing
- Search Algorithms
- Web Search and Information Retrieval

Für das Wahlpflichtmodul aus dem Bereich *Grafische Informationssysteme* kann eines der folgenden Pflichtmodule des Schwerpunt-Bereichs Medieninformatik gewählt werden:

- Computer Vision (5. Semester)
- Visualisierung (6. Semester)
- Computergraphik (6. Semester)

Da insbesondere die Veranstaltungen des Masterprogramms Computer Science for Digital Media nicht in jedem Semester angeboten werden, ist es Ratsam, wenn interessierte Studierende den Besuch dieser Vorlesungen langfristig planen.

Leistungspunkte/Credits:

Den einzelnen Modulen sind jeweils ECTS-Leistungspunkte zugeordnet, entsprechend dem European Credit Transfer System. Sie sind quantitatives Maß für die zeitliche Belastung der Studierenden. In der Regel werden pro Studienjahr etwa 60 ECTS-Punkte vergeben. Nach nationalen und internationalen Standards (für Deutschland: Beschluss der Kultusministerkonferenz vom 24.10.1997) wird pro Leistungspunkt eine Arbeitsbelastung (workload) für Studierende im Präsenz- und Selbststudium von 30 Stunden angenommen. Etwa 60 ECTS-Punkte im Jahr entsprechend damit einem Arbeitsaufwand von etwa 45 Wochen zu jeweils 40 Stunden.

Bewertung:

Jedes Modul wird mit einer Note abgeschlossen. Diese Note kann sich ggf. auf mehrere Teilleistungen stützen. Die Details dazu sind in den jeweiligen Modulbeschreibungen festgehalten.

Sieht ein Modul eine oder mehrere Teilleistungen vor, die als Klausur zu erbringen sind, kann die Prüfungsleistung abweichend im Rahmen einer mündlichen Prüfung erbracht werden, wenn einschlägige pädagogische Gründe vorliegen. Solche Gründe sind insbesondere:

- Eine sehr kleine Teilnehmerzahl (z.B. weniger als 10). Klausuren mit sehr wenigen Teilnehmern sind schwierig skalierbar. In solchen Fällen kann eine mündliche Prüfung das fairere Bewerten der Studierenden durch die Lehrenden erleichtern.
- Es handelt sich um die letzte Wiederholungsprüfung des Prüflings, und bei ihrem Nichtbestehen ist der Prüfling endgültig durchgefallen. Insbesondere Studierende, die besondere Schwierigkeiten mit der Prüfungsform der Klausur haben, können im Rahmen einer mündlichen Prüfung ggf. zeigen, dass sie die erwarteten Kompetenzen sehr wohl erworben haben.

Titel **Lineare Algebra** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
1	jährlich	Wöchentlich im Wintersemester	6	Präsenz (Vorlesung + Übung): 67,5 Belegbearbeitung: 20 Selbststudium: 62,5 Prüfungsvorbereitung (inklusive Klausur): **30** **Summe 180**	Deutsch	Dr. Bock

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Bachelor of Science	Grundwissen Abitur	Klausur (Dauer 120 Minuten). In Sonderfällen kann aus didaktischen Gründen stattdessen auch eine mündliche Prüfung (Dauer: 30 Minuten – 45 Minuten) angeboten werden. Die erfolgreiche Bearbeitung der Belegaufgaben ist eine Voraussetzung für die Zulassung zur Klausur.

Qualifikationsziele

Das Modul zielt auf das Verständnis von Sachverhalten, Grundlagen und Methoden, die für ein Studium der Informatik bzw. Medieninformatik unverzichtbar sind. Den Studierenden wird ein Einstieg in die Grundlagen der Teilgebiete der Mathematik ermöglicht, die für Informatiker besonders wichtig sind. Zu diesem Zweck wird ein Teil Elementarmathematik vorgeschaltet, der die Studierenden auf ein einheitliches, die Studierfähigkeit garantierendes Niveau bringt. Ihnen werden Begriffe und Verfahren der Linearen Algebra vermittelt. Nach einem erfolgreichen Besuch der Lehrveranstaltung können die Studierenden konkrete Probleme mit Begriffen der Mathematik formulieren und analysieren, die wesentlichen Merkmale erfassen (Abstraktion) und mit Standardmethoden der Linearen Algebra geeignete Lösungsansätze entwickeln. Dabei werden sowohl die geistigen Grundtechniken Konkretisieren bzw. Spezialisieren als auch Verallgemeinern vermittelt. Spezielle Aufmerksamkeit wird dem Training des logischen Denkens und des korrekten Schließens gewidmet. Sie sind in der Lage, unter unterschiedlichen Problemlösungsansätzen bzw. unterschiedlichen Algorithmen einen geeigneten auszuwählen und diese Wahl nachvollziehbar zu begründen. Nicht zuletzt soll das Modul zur Förderung des objektiven und sicheren Denkens beitragen sowie zur Urteilsfähigkeit und Selbstkontrolle erziehen.

Spezielle Qualifikationsziele aus der Linearen Algebra:

- Die Studierenden sind sicher im Umgang mit den folgenden Begriffen und können sie aktiv verwenden:
  - Mengen und ihre Operationen
  - Grundbegriffe der Logik
  - Natürliche Zahlen, ganze Zahlen, rationale Zahlen, komplexe Zahlen
  - Funktionen und Abbildungen
  - Elementare Funktionen und ihre Umkehrfunktionen
  - Vektoren in 2 und 3D, Vektoralgebra
  - Vektoren im R^n
  - Geometrie im R^n, Cauchy-Schwarzsche Ungleichung, allgemeine Dreiecksungleichung, Orthogonalität
  - Basis, Dimension, lineare Unabhängigkeit
  - Teilraum, lineare Mannigfaltigkeit
  - Orthogonalisierung von Vektoren, Projektionen
  - Euklidische Vektorräume
  - Matrizen und Matrixoperationen
  - Matrizen als lineare Abbildungen, Darstellungsmatrix
  - Rang, Determinante, Regularität, inverse Matrix
  - Lösung linearer Gleichungssysteme
  - Matrizeneigenwertprobleme

- Die Studierenden sind in der Lage, ihr Wissen unter anderem in den folgenden Bereichen bzw. zur Beantwortung der folgenden Fragestellungen anzuwenden:
    - Qualitative und quantitative Charakterisierung elementarer Funktionen
    - Analytische Geometrie der Ebene und des Raumes
    - Anwendung der Theorie linearer Vektorräume zur Beschreibung geometrischer Objekte in höheren Dimensionen
    - Beschreibung und Konstruktion spezieller linearer Abbildungen
    - Lösung linearer Gleichungssysteme
    - Bestimmung der Eigenwerte einer Matrix und Charakterisierung ihrer Invarianten

- Die Studierenden können ihr Wissen auf anspruchsvolle Probleme bzw. im Zusammenhang mit komplexen Beispielen anwenden:
    - Lösung komplexer geometrischer Probleme in der Ebene und im Raum
    - Koordinatentransformationen
    - Auswahl und Anpassung entsprechender Algorithmen
    - Konstruktionen spezieller Abbildungsmatrizen
    - Modellierung komplexer linearer Systeme
    - Lösung entsprechender linearer Gleichungssysteme
    - Bestimmung der Invarianten linearer Systeme

- Die Studierenden können Probleme der Informatik mit Hilfe von Konzepten
    - der linearen Algebra vereinfachend beschreiben, algorithmisch bearbeiten und adäquate Datenstrukturen auswählen und zur Lösung auf dem Computer vorbereiten.
    - Sie können formalisieren und abstrahieren, konkrete Fragestellungen analysieren, systematisch nach möglichen mathematischen Lösungen suchen und selbst entsprechende Algorithmen entwickeln und zur Implementation auf dem Computer vorbereiten. Sie sind in der Lage, die Auswahl der Verfahren zu beurteilen und die erhaltenen Ergebnisse qualitativ und quantitativ zu bewerten.

Lehrinhalte

- Zahlenbereiche, Axiomatik, Rechengesetze.
- Einführung in die Mengenlehre
- Grundlagen der Logik
- Funktionen und Abbildungen
- Elementare Funktionen und ihre Umkehrfunktionen
- Vektorrechnung im R^2 und R^3, analytische Geometrie der Ebene und des Raumes
- Vektorraum R^n und allgemeine Euklidische Vektorräume
- Skalarprodukt, Cauchy-Schwarz Ungleichung und ihre Anwendung auf die Lösung geometrischer Probleme
- Orthogonalisierungsverfahren
- Matrizen und Matrixoperationen
- Matrizen und lineare Abbildungen
- Determinante, Rang
- Lineare Gleichungssysteme
- Eliminationsverfahren, Cramersche Regel
- Matrixeigenwertprobleme
- Invarianten einer Matrix
- Koordinatentransformationen

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert (Präsenzstudium). Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (nun wieder Präsenzstudium).). Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen. Zur Unterstützung des Selbststudiums kann ein Tutorium angeboten werden. Zur Selbstkontrolle sind studienbegleitende Belegaufgaben zu bearbeiten.

Bestimmte Lehrinhalte überlappen sich mit Lehrinhalten aus anderen Modulen. Dieses Vorgehen soll eine Orientierung in der Stofffülle vermitteln und Zusammenhänge erkennbar machen. Es basiert auf einem didaktischen Konzept, das sich an das Spiralprinzip aus der Schuldidaktik anlehnt.

Da sich gezeigt hat, dass viele StudienanfängerInnen Schwierigkeiten beim Einstieg in die Mathematik haben, wurde ein Mathematik-"Liftkurs" in die Veranstaltung Lineare Algebra integriert. Der Liftkurs ermöglicht es den Studierenden, beim Studienanfang ihr Mathematik-Schulwissen wieder aufzufrischen.

Hinweise

Literatur:

- Lothar Papula: Mathematik für Ingenieure und Naturwissenschaftler, Band 1,2,3. Vieweg+Teubner Verlag
- Burg, Haf, Wille, Höhere Math. für Ingenieure, Bde 1-2, Vieweg+Teubner Verlag
- Jänich, Lineare Algebra,Springer

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Lineare Algebra (wöchentlich im Wintersemester)	6 ECTS-Punkte (SWS: V3+Ü3)

Titel **Diskrete Mathematik** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
1	jährlich	Wöchentlich im mit jeweils einer Lehrveranstaltung im Wintersemester	12	Präsenz (Vorlesung + Übung): **45** Selbststudium (einschließlich Tutorium, falls angeboten): **90** Prüfungsvorbereitung einschließlich der Prüfung: **45** **Summe 180**	Deutsch	Stefan Lucks

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Bachelor of Science	Grundwissen Abitur	Klausur (2 Stunden). In Sonderfällen kann aus didaktischen Gründen stattdessen auch eine mündliche Prüfung (30 - 45 Minuten) angeboten werden. Die regelmäßige Bearbeitung der Übungsaufgaben ist eine Voraussetzung für die Zulassung zu den Klausuren.

Qualifikationsziele

Das Modul zielt auf den Erwerb der zentraler Grundlagen und Methoden, die für ein Studium der Informatik bzw. Medieninformatik unverzichtbar sind. Den Studierenden wir ein Einstieg in Teilgebiete der Mathematik vermittelt, die für Informatiker besonders wichtig sind, ihnen werden Begriffe und Methoden der Algorithmik vermittelt. Nach einem erfolgreichen Besuch der Lehrveranstaltung können die Studierenden konkrete Probleme mit Begrifflichkeiten der Mathematik abstrakt darstellen und mit Methoden der Mathematik algorithmische Lösungsansätze entwickeln.

Spezielle Qualifikationsziele:

- Die Studierenden kennen insbesondere die folgenden Begriffe und Zusammenhänge und können sie anderen erklären:
  - grundlegende Eigenschaften der natürlichen Zahlen
  - grundlegende Beweismethoden der Mathematik bzw. der Theoretischen Informatik: direkte und indirekte Beweise, vollständige Induktion
  - Endlichkeit, Abzählbarkeit, Über-Abzählbarkeit
  - elementare algebraische Algorithmen (Addition, Multiplikation, Potenzbildung)
  - die Laufzeiten dieser Algorithmen
  - den Zusammenhang von vollständiger Induktion und rekursiven Programmen
  - Syntax und Semantik einer einfachen Programmiersprache
  - Rechenregeln in der Menge der Restklassen mod n, Besonderheiten, wenn n prim ist, Inverse mod n
  - Grundlegende algebraische Strukturen (Gruppe, Ring, Körper)
  - Beispiele für die Anwendung der Arithmetik mod n in der Informatik (Prüfusummen, Public-Key Kryptographie)
  - Probabilistische Algorithmen, insbesondere probabilistische Primzahltests
  - Polynomringe und ihre Eigenschaften
  - Endliche Körper und ihre Eigenschaften
  - das Geburtstagsparadoxon
  - Wesentliche Begriffe der Diskreten Wahrscheinlichkeit (Wahrscheinlichkeitsraum, Zufallsvariable, bedingte Wahrscheinlichkeit, …)
  - Grundlegende Begriffe der Graphentheorie (gerichtete und ungerichtete Graphen, Zusammenhangskomponenten, Wege, Kreise, …)
  - Beispiele für effizient lösbare Probleme der Graphentheorie (Eulerkreis, TSP-Approximation) und für NP-harte Probleme (Hamiltonkreis, TSP-Suchproblem)
- Die Studierenden sind in der Lage, ihr Wissen unter anderem in den folgenden Bereichen bzw. zur Beantwortung der

folgenden Fragestellungen anzuwenden:

- die grundlegenden Beweistechniken einsetzen, um einfache Beweise zu führen
- entscheiden, ob eine gegebene Menge endlich, abzählbar oder über-abzählbar ist
- einfache Verfahren zum Berechnen von Prüfsummen anwenden
- einfache Algorithmen in einer einfachen Programmiersprache implementieren
- mit Hilfe des Erweiterten Euklidischen Algorithmus feststellen, ob eine Zahl ein Inverses mod n hat, und wenn ja, dieses Inverse mod n berechnen
- fehlererkennende Codes als Anwendung der Arithmetik in Polynomringen berechnen
- Secret-Sharing-Verfahren als Anwendung der Arithmetik in endlichen Körpern einsetzen
- einfache Aufgaben der Wahrscheinlichkeitsrechnung und der bedingten Wahrscheinlichkeit lösen

- Die Studierenden können ihr Wissen auf anspruchsvolle Probleme bzw. im Zusammenhang mit komplexen Beispielen anwenden:

  - mittelschwere Algorithmen in einer einfachen Programmiersprache implementieren (z.B. Simulation von Zufallsereignissen, probabilistische Primzahltests, Zeichnen von Fraktalen, …)
  - offen gestellte mathematische Fragestellungen mit Hilfe der oben genannten Beweistechniken beweisen oder, durch Angabe von Gegenbeispielen, widerlegen
  - selbstständig abstrakte Begriffe in mathematisch-formaler Darstellung verstehen sich selbständig in die Anwendung grundlegender Methoden einarbeiten
  - das WWW als gerichteten Graphen modellieren und die Funktion einer Suchmaschine auf Basis des Page-Rank Algorithmus nachvollziehen

- Die Studierenden können Probleme der Informatik mit Hilfe von Konzepten

  - der Zahlentheorie,
  - der diskreten algebraischen Strukturen,
  - der diskreten Wahrscheinlichkeit und
  - der Graphentheorie

  formalisieren, formale Fragestellungen analysieren, systematisch nach möglichen Lösungen suchen und selbst entsprechende Algorithmen entwickeln. In einfachen Fällen können sie die Korrektheit von Algorithmen begründen und ihre Laufzeit grob abschätzen (linear, quadratisch, kubisch, …).

- Die Studierenden können in Anwendungsfällen entscheiden und bewerten, welche Konzepte und Methoden

  - der Zahlentheorie,
  - der diskreten algebraischen Strukturen,
  - der diskreten Wahrscheinlichkeit und
  - der Graphentherie

  zur Modellbildung geeignet sind. Insbesondere können sie diese Entscheidung auch nachvollziehbar begründen.

Lehrinhalte

- Beweistechniken
- Abzählbarkeit und Über-Abzählbarkeit
- Einführung in eine einfache Programmiersprache (Teilmenge von Python)
- Restklassen mod n
- Primzahlen
- Gruppen
- Ringe und Körper
- Diskrete Wahrscheinlichkeit
- Graphentheorie

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert (Präsenzstudium). Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (nun wieder Präsenzstudium). Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen. Zur Unterstützung des Selbststudiums kann ein Tutorium angeboten werden.

Bestimmte Lehrinhalte überlappen sich mit Lehrinhalten aus anderen Modulen. Zum Beispiel werden Beweismethoden auch in

Algebra und Analysis behandelt, und im Modul Einführung in die Programierung werden Programmierkenntnisse vermittelt. Dieses Vorgehen soll eine Orientierung in der Stofffülle vermitteln und Zusammenhänge erkennbar machen. Es basiert auf einem didaktischen Konzept, das sich an das Spiralprinzip aus der Schuldidaktik anlehnt.
Hinweise
Handouts und Literatur zu den Diskreten Mathematik: • Lucks: Handout für den Start • Lucks: Einfache Algorithmen mit Python • Jukna: Crashkurs Mathematik für Informatiker • Albertson, Hutchinson: Discrete Mathematics with Algorithms

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Diskrete Mathematik (wöchentlich im Wintersemester)	6 ECTS-Punkte (SWS: V3+Ü1)

Titel **Technische und Praktische Informatik** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
1	jährlich	Wöchentlich im Wintersemester	6	Präsenz (Vorlesung + Übung): **45** Selbststudium (einschließlich Tutorien, falls angeboten): **110** Prüfungsvorbereitung einschließlich der Prüfung: **25** **Summe 180**	Deutsch	Dr. A. Jakoby

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Bachelor of Science	Grundwissen Abitur	Die Prüfungsleistung der Lehrveranstaltung ‚Technische und Praktische Informatik' ist eine Klausur. Dauer: 120 Minuten. Die regelmäßige Bearbeitung der Übungsaufgaben ist eine Voraussetzung für die Zulassung zur Prüfung. In Sonderfällen kann aus didaktischen Gründen stattdessen auch eine mündliche Prüfung (30 - 45 Minuten) angeboten werden.

Qualifikationsziele

Im Rahmen des Moduls werden die zentralen Grundlagen der Informatik vermittelt, welche für ein Studium der Informatik bzw. Medieninformatik besonders wichtig sind. Den Studierenden wird ein Einstieg in die Begriffe und Methoden der Analyse von Problemen und deren Modellierung und algorithmischen Lösung, sowie deren Umsetzung in ein Programm vermittelt. Nach einem erfolgreichen Besuch der Lehrveranstaltungen dieses Moduls können die Studierenden konkrete einfache Aufgaben mit Begrifflichkeiten und Methoden der Informatik darstellen und mit Hilfe eines Programms lösen.

Spezielle Qualifikationsziele aus der Technische und Praktische Informatik

Die Studierenden kennen die grundlegenden Prinzipien einer problemorientierten Vorgehensweise eines Informatikers: algorithmisches Denken ausgehend von klaren Spezifikationen. Darüber hinaus kennen die Studierenden typische Programmierparadigmen und grundlegende Begrifflichkeiten wie Felder, lineare Suche, binäre Suche, Sortierverfahren sowie Rekursion und O-Notation und deren typische Anwendungsfälle. Außerdem sind grundlegende Konzepte der Technischen Informatik (wie die Boolesche Algebra, Funktionsweise von Transistoren, Schaltungsaufbau und –analyse), Rechnerarchitektur (von-Neumann-Modell) und von Betriebssystemen (Ressourcenverwaltung, Shells) bekannt und in ihren Aufgaben erfasst.

Die Studierenden verstehen, wozu algorithmisches Denken eingesetzt wird und warum es als zentrales Handwerkszeug eines Informatikers gilt. Sie verstehen grundlegende Fragestellungen wie Suchen, Sortieren und rekursive Problemlöseansätze und die Wichtigkeit strukturierten Vorgehens bei der Problemanalyse und genauer Analysen zur Effizienzbewertung verschiedener Lösungsansätze. Zusätzlich verstehen die Studierenden den Zusammenhang zwischen Rechnermodell, praktischer Umsetzung in Hardware, und den Verwaltungsaufgaben eines Betriebssystems. Aus dem Umfeld der Computerarchitektur kennen die Studierenden insbesondere die folgenden Begriffe und Zusammenhänge und können diese anderen erklären: Funktionsprinzipien von informationsverarbeitenden Geräten, Begriff Gerät, Architektur, Ubicomp, mobile Systeme, Abstraktionseben von Software und Hardware, Hardware-Schichtenmodell, Moorsches Gesetz, Perspektiven, Siliziumtechnologie und künftige Alternativen, Speichersystem, Anforderungen und Grundstrukturen, Arbeitsprinzipien und Technologie, Register, Cache, Hauptspeicher, Virtueller Speicher, Externer Speicher, Kommunikation, Systematik, interne und externe Bussysteme, Interfaces. Die Studierenden haben kennen die grundlegenden Aspekte und Umsetzungsformen von Booleschen Funktunen und deren Realisierung mit Hilfe von Schaltungen von Logikgattern.

Die Studierenden können einfache Problemstellungen spezifizieren, algorithmisch erfassen und in Pseudocode umsetzen. Vorgegebene Lösungsansätze können unter Effizienzgesichtspunkten analysiert und ggf. leicht verbessert werden. Einfache Abschätzungen zur Performanceverbesserung durch Parallelisierbarkeit können durchgeführt und berechnet werden. Die Studierenden sind in der Lage die Effizienz der Speicherhierarchien und Parallelisierungen nach Amdahl zu analysieren. Sie kennen externe Schnittstellen der Medieninformatik, und sind in der Lage diese problemspezifische auszuwählen.

Lehrinhalte

Die Veranstaltung gibt eine Einführung in die zentralen Konzepte der Informatik und die algorithmische Denkweise als Problemlösestrategie.

- Informatikbegriff und Teilgebiete der Informatik
- Algorithmusbegriff (Algorithmen und Spezifikationen, Übergang zu Programmiersprachen)
- Informationen und Daten (Bits und Bytes, Kodierung)
- Boolesche Funktionen und deren Realisierungen
- Funktionsweise von Transistoren und Relais, Gatter und deren Realisierung
- Optimierung der disjunktive Normalform mit dem Quine-McCluskey-Verfahren
- Nicht-Funktionale Aspekte von Schaltung (Zeitverhalten von Logikzellen, Hazards)
- Betriebssysteme (Ressourcen, Aufbau von Betriebssystemen, Treiber)
- Shell-Programmierung (Shell-Begriff, Shell-Befehle, Shell-Skripte)
- Imperative Programmierung (Arten von Programmiersprachen, Berechnungen, Steuerungsanweisungen)
- Suchen und Sortieren (Lineare Suche, Binäre Suche, Bubble-Sort, Insertion-Sort, Selection-Sort)
- Laufzeitverhalten (Laufzeit von Programmen, O-Notation,Einfache vs. schwierige Probleme)
- Rekursion (Begriff der Rekursion, Rekursives Sortieren, Merge-Sort, Quick-Sort)
- Objektorientierte Programmierung (Modularisierung von Software, Attribute, Methoden, Objektorientierte Modellierung)
- Rechnerarchitektur (von-Neumann-Modell)
- Funktionsprinzipien von informationsverarbeitenden Geräten, Begriff Gerät, Architektur, Ubicomp, mobile Systeme,
- Abstraktionseben Software und Hardware, Hardware-Schichtenmodell,
- Moorsches Gesetz, Perspektiven, Siliziumtechnologie und künftige Alternativen,
- Rechnerinterne Informationsdarstellung, Logikpegel, n-Tupel-Verarbeitung, Datentypen, Bustypen,
- Organisationsprinzip, Rechnergenerationen und -klassen, von-Neumann-Maschine und ihre Evolution,
- Taxonomie insbes. der Parallelverarbeitung, Flynn, Betriebsarten Bit-, Instruction-, Task-, Processor-Level,
- CPU- und Busstrukturen, CPU-Philosophien, Ebenen der Parallelverarbeitung, Registerstrukturen, Multimedia-Erweiterungen, Innovation, Technologie, Realisierung,
- Leistungsbewertung, Amdahl-Prinzip der Effizienz von Parallelstrukturen,
- Speichersystem, Anforderungen und Grundstrukturen, Arbeitsprinzipien und Technologie, Register, Cache, Hauptspeicher, Virtueller Speicher, Externer Speicher,
- Kommunikation, Systematik, interne und externe Bussysteme, Interfaces.

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert. Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (in Vorlesung oder Übung). Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen. Zur Unterstützung des Selbststudiums kann ein Tutorium angeboten werden.

Über regelmäßig zu bearbeitende Übungsaufgaben wird eine kontinuierliche Aufarbeitung der Inhalte bewirkt.

Im Sinne des Spiralprinzips werden Inhalte, die in höheren Semestern im Detail behandelt werden, hier einführend behandelt.

Wissenschaftliche Mitarbeiter und studentische Tutoren aus höheren Semestern betreuen die Studierenden in den Übungen und stehen für Rückfragen und Diskussion zur Verfügung

Die ‚Grundlagen der Informatik' besteht aus einer wöchentlichen 135-minütigen Vorlesung (mit einer 15-minütigen Pause) sowie aus einer vierzehntägigen 90-minütigen Übung.

Hinweise

Literatur

- Jakoby: Folien zur Vorlesung Technische und Praktische Informatik
- Gumm, Sommer: Einführung in die Informatik. Oldenbourg Verlag.
- Hennessy, J.L.; Patterson, D.A.: Computer Architecture. A Quantiative Approach.
- Hoffmann, D. W.: Grundlagen der Technischen Informatik, Hanser Verlag.
- Tanenbaum, A.: Structured Computer Organization.

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)

Technische und Praktische Informatik	6.0 ECTS-Punkte (SWS: V3+Ü1)

Titel **Einführung in die Programmierung** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
1	jährlich	Wöchentlich im Wintersemester	6	Präsenz (Vorlesung + Übung): **45** Projektarbeit: **60** Selbststudium: **75** **Summe 180**	Deutsch	Professur Software Engineering, voraussichtliche Neubesetzung Sommersemester 2021

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Bachelor of Science	Grundwissen Abitur	Die Veranstaltung ‚Einführung in die Programmierung' wird durch bewertete Programmieraufgaben und die bewertete Vorstellung der Programmierprojekte geprüft.

Qualifikationsziele

Im Rahmen des Moduls werden die zentralen Grundlagen der Informatik und der Programmierung vermittelt, welche für ein Studium der Informatik bzw. Medieninformatik besonders wichtig sind. Den Studierenden wird ein Einstieg in die Begriffe und Methoden der Analyse von Problemen und deren Modellierung und algorithmischen Lösung, sowie deren Umsetzung in ein Programm vermittelt. Nach einem erfolgreichen Besuch der Lehrveranstaltung können die Studierenden konkrete einfache Aufgaben mit Begrifflichkeiten und Methoden der Informatik darstellen und mit Hilfe eines Programms lösen.

Spezielle Qualifikationsziele der Einführung in die Programmierung:

Die Studierenden kennen die Grundlagen, Konzepte und Eigenschaften der Programmiersprache Python, dieses umfasst insbesondere Datentypen, Variablen, Ausdrücke, Operatoren, Kontrollstrukturen, Blöcke, Methoden, Klassen, Objekte, Vererbung, Pakete, Schnittstellen und die Oberflächenprogrammierung.

Die Studierenden können Classen für einfache Datentypen eigenständig implementieren und grundlegende Algorithmen für diese Datentypen in Methoden dieser Klassen umsetzen (z.B. für Komplexe Zahlen). Basierend auf dem Konzept der Vererbung können bestehende Datentyp-Klassen erweitert werden (z.B. Stacks). Die Studierenden können eine einfache Oberfläche zur Eingabe von Zeichenketten implementiert.

Die Studierenden wissen wie in Kleingruppen einfache Schnittstellen entworfen werden und wie basierend auf diesen Schnittstellen Programme in einem Team entwickelt werden können.

Lehrinhalte

Einführung in die Programmierung

Die Veranstaltung gibt eine Einführung in die Implementierung von Programmen basierend auf der Programmiersprache Python.

- Datentypen, Variablen
- Arithmetische und Boolesche Ausdrücke
- Operatoren, Kontrollstrukturen, Blöcke, Methoden
- Klassen, Objekte, Vererbung
- Pakete, Schnittstellen
- Oberflächenprogrammierung
- Umgang mit Programmierwerkzeug zur Entwicklung von Software

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert. Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (in Vorlesung oder Übung). Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen. Zur Unterstützung des Selbststudiums kann ein Tutorium angeboten werden.

Über regelmäßig zu bearbeitende Übungsaufgaben wird eine kontinuierliche Aufarbeitung der Inhalte bewirkt.

Im Sinne des Spiralprinzips werden Inhalte, die in höheren Semestern im Detail behandelt werden, hier einführend behandelt.

Zur Reduzierung der Prüfungsbelastung und im Sinne einer Prüfung praktischer Fähigkeiten erfolgt für zwei der Fächer eine studienbegleitende Prüfungsform durch Haus- und Programmieraufgaben. Dies gilt insbesondere für die ‚Einführung in die Modellierung'. Diese Veranstaltung wird durch mehrere zu bearbeitete Hausaufgaben geprüft (maximal 6 verteilt über das Semester).

Die Vorlesung ‚Einführung in die Programmierung' zielt auf das Erlernen einer konkreten Programmiersprache und die Programmentwicklung in dieser Sprache. Daher finden die Übungen als betreute Lab-Stunden in einem Rechnerpool statt. Die Veranstaltung wird durch drei kleinere und eine größere (auch außerhalb der eingeplanten Lab-Stunden zu bearbeitende) über das Semester verteilte Programmieraufgaben geprüft.

Wissenschaftliche Mitarbeiter und studentische Tutoren aus höheren Semestern betreuen die Studierenden in den Übungen und stehen für Rückfragen und Diskussion zur Verfügung

Hinweise

Literatur zu Einführung in die Programmierung:

- Johannes Ernesti, Peter Kaiser: Python 3: Das umfassende Handbuch: Über 1.000 Seiten Sprachgrundlagen, Objektorientierte Programmierung und Beispielprogramme, Rheinwerk Computing, 2020

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Einführung in die Programmierung	6.0 ECTS-Punkte (SWS: V2+Ü2)

Titel **Modellierung** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
1	jährlich	Wöchentlich im Wintersemester	6	Präsenz (Vorlesung + Übung):45 Selbststudium und Hausarbeiten 135 Summe 180	Deutsch	Hornecker

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Bachelor of Science	Grundwissen Abitur	Die Veranstaltung wird durch mehrere bewertete Hausaufgaben geprüft. Eine finale Aufgabe die den Stoff des Semesters umfasst muss alleine bearbeitet und abgegeben werden.

Qualifikationsziele

Im Rahmen des Moduls werden zentrale Grundlagen der Informatik vermittelt. Den Studierenden wird ein Einstieg in die Begriffe und Methoden der Analyse von Problemen und deren Modellierung vermittelt. Nach einem erfolgreichen Besuch der Lehrveranstaltungen dieses Moduls können die Studierenden konkrete einfache Aufgaben mit Begrifflichkeiten und Methoden der Informatik darstellen und analysieren.

Die Studierenden kennen Grundbegriffe der Modelltheorie (Verkürzung, Pragmatik, Abbild) sowie zentrale Modellierungsmethoden der Informatik (Daten-, Steuer-, Verhaltens-, sowie Interaktionsmodelle, Klassen und Objekte, Vererbung, Logik). Sie kennen die Grundlagen von UML.

Die Studierenden verstehen, warum Modellbildung ein zentraler Bestandteil der Informatik ist. Sie verstehen, dass jeder Modellierungsprozess einem bestimmten Modellierungszweck dient und dass Modellbildung keine wertfreie Tätigkeit ist. Sie verstehen die Wichtigkeit einer definierten Notation für die Modellierung.

Sie können vorgegebene einfache Diagramme (Entitity Relationship, Flowchart, Zustandsautomat, Klassenmodell, etc.) lesen bzw. interpretieren und können solche Diagramme für einfache Beispiele selber erstellen.

Lehrinhalte

Die Veranstaltung gibt eine Einführung in zentrale Modellierungsmethoden der Informatik sowie Grundprinzipien der Modellierung.

- Der Modellbegriff, Grundprinzipien der Modellierung (Abstraktion, Klassifikation, Generalisierung, Komposition, Benutzung), Original-Abbild-Modell Relation, Verkürzung und Pragmatik der Modellierung, Modellbildung als Realitätskonstruktion, deskriptive und präskriptive Modellbildung
- Rolle von Notationen, diagrammatisches Denken
- Logik als Modellierungssprache (einfache Aussagen- und Prädikatenlogik)
- Modellierungsmethoden: statisch: Datenmodelle (Entity Relationship), Klassen- und Objektmodelle (mit UML Notation),
- Dynamische Verhaltensmodellierung (Flußdiagramme, Aktivitätsmodelle, Zustandsautomaten und –diagramme, Statecharts, Petrinetze), Datenflussdiagramme,
- Interaktionsmodelle (Kontextdiagramme, Anwendungsfälle / Use Cases, Szenarien, Aktivitätsdiagramme)

Lehr- und Lernmethoden / Didaktisches Konzept

Die Veranstaltung besteht aus jeweils einer wöchentlichen 90-minütigen Vorlesungen sowie einer wöchentlichen 90-minütigen Übung.

Die Lehrinhalte werden in einer Vorlesung präsentiert. Sie werden im Rahmen der Übung vertieft. Dabei sind zunächst im Selbststudium in Gruppenvorgegebene kleinere Aufgaben zu bearbeiten, die anschließend diskutiert werden. Die Hausaufgaben vertiefen sodann die Thematik und bauen auf den in Präsenz diskutierten Aufgaben auf. Letztere werden abgegeben, korrigiert und bewertet. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in der Übung besprochen.

Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an Übungen dient unter anderem der Selbstkontrolle der Studierenden, bewirkt eine kontinuierliche Aufarbeitung der Inhalte und ist für die kontinuierliche Vertiefung der Lehrinhalte zu empfehlen.

Im Sinne des Spiralprinzips werden in dieser Vorlesung Inhalte, die in höheren Semestern im Detail behandelt werden, in der ‚Modellierung' einführend behandelt. Zudem wird das Grundprinzip der Modellierung und dessen Rolle in der Informatik vermittelt.

Zur Reduzierung der Prüfungsbelastung und im Sinne einer Prüfung praktischer Fähigkeiten erfolgt eine studienbegleitende Prüfung durch Hausaufgaben. Es gibt mehrere zu bearbeitete Hausaufgaben (maximal 6 verteilt über das Semester) zum jeweils aktuellen Stoff. Das Gesamtverständnis wird durch eine umfangreichere, individuell zu bearbeitende Hausaufgabe nach Abschluss der Vorlesungszeit geprüft. Diese ist zum Bestehen der Veranstaltung erforderlich. Die Gesamtnote ergibt sich gewichtet aus allen Abgaben, wobei die übergreifende finale Aufgabe mindestens 35% der Endnote ausmacht.

Wissenschaftliche Mitarbeiter und studentische Tutoren aus höheren Semestern betreuen die Studierenden in den Übungen und stehen für Rückfragen und Diskussion zur Verfügung

Hinweise

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Modellierung	6.0 ECTS-Punkte (SWS: V2+Ü2)

Titel **Analysis** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
2	jährlich	Wöchentlich im Sommersemester	6	Präsenz (Vorlesung + Übung): 45 Belegbearbeitung: 30 Selbststudium: 80 Prüfungsvorbereitung (inklusive Klausur): 25 **Summe 180**	Deutsch	Dr. Bock

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Bachelor of Science	Grundwissen Abitur	Klausur (120 Minuten). In Sonderfällen kann aus didaktischen Gründen stattdessen auch eine mündliche Prüfung (30 - 45 Minuten) angeboten werden. Die erfolgreiche Bearbeitung der Belegaufgaben ist eine Voraussetzung für die Zulassung zur Klausur.

Qualifikationsziele

Das Modul zielt auf das Verständnis von Sachverhalten, Grundlagen und Methoden, die für ein Studium der Informatik bzw. Medieninformatik unverzichtbar sind. Den Studierenden wird ein Einstieg in die Grundlagen der Teilgebiete der Mathematik ermöglicht, die für Informatiker besonders wichtig sind. Zu diesem Zweck wird ein Teil Elementarmathematik vorgeschaltet, der die Studierenden auf ein einheitliches, die Studierfähigkeit garantierendes Niveau bringt. Ihnen werden Begriffe und Verfahren der Analysis vermittelt. Nach einem erfolgreichen Besuch der Lehrveranstaltung können die Studierenden konkrete Probleme mit Begriffen der Mathematik formulieren und analysieren, die wesentlichen Merkmale erfassen (Abstraktion) und mit Standardmethoden der Analysis geeignete Lösungsansätze entwickeln. Dabei werden sowohl die geistigen Grundtechniken Konkretisieren bzw. Spezialisieren als auch Verallgemeinern vermittelt. Spezielle Aufmerksamkeit wird dem Training des logischen Denkens und des korrekten Schließens gewidmet. Sie sind in der Lage, unter unterschiedlichen Problemlösungsansätzen bzw. unterschiedlichen Algorithmen einen geeigneten auszuwählen und diese Wahl nachvollziehbar zu begründen. Nicht zuletzt soll das Modul zur Förderung des objektiven und sicheren Denkens beitragen sowie zur Urteilsfähigkeit und Selbstkontrolle erziehen.

Spezielle Qualifikationsziele aus der Analysis:

- Die Studierenden kennen insbesondere die folgenden Begriffe und Zusammenhänge und können sie anderen erklären:
  - Zahlenfolgen und Zahlenreihen
  - Grenzwert und Grenzwertsätze
  - Konvergenz von Zahlenfolgen und Zahlenreihen
  - Konvergenzkriterien
  - Funktionsbegriff; Stetigkeit und Differenzierbarkeit
  - Bestimmung von Nullstellen
  - Iterative Lösung nichtlinearer Gleichungen, Intervallhalbierung, Newtonverfahren, Banachscher Fixpunktsatz
  - Approximation von Funktionen
  - Taylorsche Formel
  - Funktionenreihen – Taylorreihe
  - Bestimmtes und unbestimmtes Integral
  - Zusammenhang von Integration, Flächeninhalt und Stammfunktion
  - Integrationsregeln
  - Approximation von Integralen
  - Funktionenreihen – Fourierreihen
- Die Studierenden sind in der Lage, ihr Wissen unter anderem in den folgenden Bereichen bzw. zur Beantwortung der folgenden Fragestellungen anzuwenden:
  - Berechnung des Grenzwertes von Zahlenfolgen und Zahlenreihen
  - Analyse des Konvergenzverhaltens
  - Klassifikation von Funktionen

- Berechnung von Extremwerten, Nullstellen, Fixpunkten nichtlinearer Funktionen
- Beschreibung von Kurven und Flächen und ihre geometrische Analyse
- Entwicklung von Funktionen in Potenzreihen
- Entwicklung von Funktionen in trigonometrische Reihen
- Fehleranalyse für die Partialsummen dieser Reihen
- Berechnung von Kurvenlängen, Flächeninhalten und Volumina
- Bestimmung von Stammfunktionen
- Elementare Aufgaben der Differentialgeometrie nach entsprechender Aufbereitung
- Konstruktion und Charakterisierung allgemeiner krummliniger Koordinatensysteme

- Die Studierenden können ihr Wissen auf anspruchsvolle Probleme bzw. im Zusammenhang mit komplexen Beispielen anwenden:

  - Berechnung von Flächeninhalten und Volumina kompliziert geformter Körper
  - Analyse von Signalen durch Fourierreihen
  - Approximation von Funktionen durch Potenzreihen
  - Iterative Lösung nichtlinearer Gleichungen

- Die Studierenden können Probleme der Informatik mit Hilfe von Konzepten

  - der Analysis der Funktionen einer oder mehrerer Veränderlicher,
  - der Differential- und Integralrechnung,
  - durch Reihenentwicklungen

  formalisieren, konkrete Fragestellungen analysieren, systematisch nach möglichen Lösungen suchen und selbst entsprechende Algorithmen entwickeln. In einfachen Fällen können sie die Angemessenheit der angewandten Verfahren und die Aussagekraft der erzielten Ergebnisse bewerten und Fehlerabschätzungen erarbeiten.

- Die Studierenden können in Anwendungsfällen entscheiden und bewerten, welche Konzepte und Methoden der Differential- und Integralrechnung der Funktionen einer oder mehrerer Veränderlicher zur Modellbildung geeignet sind. Insbesondere können sie diese Entscheidung auch nachvollziehbar begründen.

Lehrinhalte

- Zahlenfolgen
- Grenzwert
- Konvergenz
- Zahlenreihen
- Stetige Funktionen
- Differenzierbare Funktionen
- Nullstellen, Fixpunkte, Extremwerte
- Bestimmtes und unbestimmtes Integral
- Taylorpolynom und Taylorreihe, Potenzreihen
- Fourierreihen
- Beschreibung von Kurven und Flächen, geometrische Charakterisierung

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert (Präsenzstudium). Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (nun wieder Präsenzstudium).). Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen. Zur Unterstützung des Selbststudiums kann ein Tutorium angeboten werden. Zur Selbstkontrolle sind studienbegleitende Belegaufgaben zu bearbeiten.

Bestimmte Lehrinhalte überlappen sich mit Lehrinhalten aus anderen Modulen. Dieses Vorgehen soll eine Orientierung in der Stofffülle vermitteln und Zusammenhänge erkennbar machen. Es basiert auf einem didaktischen Konzept, das sich an das Spiralprinzip aus der Schuldidaktik anlehnt.

Hinweise

Literatur:

- Lothar Papula: Mathematik für Ingenieure und Naturwissenschaftler, Band 1,2,3. Vieweg+Teubner Verlag
- Burg, Haf, Wille, Höhere Math. für Ingenieure, Bde 1-2, Vieweg+Teubner Verlag

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Analysis (wöchentlich im Sommersemester)	6 ECTS-Punkte (SWS: V2+Ü2)

Titel **Algorithmen und Datenstrukturen** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
2	jährlich	Wöchentlich im mit jeweils einer Lehrveranstaltung im Sommersemester	6	Präsenz (Vorlesung + Übung): **45** Selbststudium (einschließlich Tutorium, falls angeboten): **120** Prüfungsvorbereitung einschließlich der Prüfungen: **15** **Summe 180**	Englisch	Charles Wuethrich

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für nformatik, Bachelor of Science	Grundwissen Abitur	Klausur (120 Minuten)

Qualifikationsziele

Das Modul zielt auf den Erwerb der zentraler Grundlagen und Methoden, die für ein Studium der Informatik unverzichtbar sind. Den Studierenden wir ein Einstieg in Teilgebiete der Mathematik vermittelt, die für Informatiker besonders wichtig sind, ihnen werden Begriffe und Methoden der Algorithmik vermittelt. Nach einem erfolgreichen Besuch der beiden Lehrveranstaltungen können die Studierenden konkrete Probleme mit Begrifflichkeiten der Mathematik abstrakt darstellen, mit Methoden der Mathematik algorithmische Lösungsansätze entwickeln, diese Algorithmen analysieren (vor allem in Bezug auf Laufzeit und Korrektheit) und die Algorithmen implementieren. Sie sind in der Lage unter unterschiedlichen Problemlösungsansätzen bzw. unterschiedlichen Algorithmen einen geeigneten auszuwählen und diese Wahl nachvollziehbar zu begründen.

Spezielle Qualifikationsziele aus dem Bereich Algorithmen und Datenstukturen:

- Die Studierenden kennen insbesondere die folgenden Begriffe und Zusammenhänge und können sie anderen erklären:
  - grundlegende Methoden der Organisation von Daten
  - Analyse und Klassifikation von Algorithmen (Untersuchung der Leistungsfähigkeit von Algorithmen und Berechnungskomplexität im ungünstigsten Fall)
  - Suchalgorithmen, Sortierverfahren und Algorithmen für Graphen
  - Laufzeiten dieser Algorithmen und ihre Eigenschaften
  - „Teile und Herrsche“ Prinzip für die Entwicklung von Algorithmen
  - Geometrische Algorithmen wie Bestimmung der konvexen Hülle und das Problem des nächsten Punktes
  - Fluss in einem Netzwerk
  - Mathematische Algorithmen (Zufallszahlen, Arithmetik, …)
  - Verfahren zur Lösung des Problems der Datenanpassung (Interpolation mit Hilfe von Polynomen, Spline-Interpolation und Methode der kleinsten Quadrate)
  - Einige NP-vollständige Probleme wie Erfassung von Knoten, Hamilton-Zyklus und das Problem des Handlungsreisenden
- Die Studierenden sind in der Lage, ihr Wissen unter anderem in den folgenden Bereichen bzw. zur Beantwortung der folgenden Fragestellungen anzuwenden und zu programmieren:
  - die Wahl der richtigen Datenstruktur bei der Implementation der Algorithmen
  - die Leistungsfähigkeit von Algorithmen beurteilen und ihre Komplexität berechnen
  - die Wahl der geeigneten Algorithmen zur Lösung von Problemen und der Implementierung auf einem Computer
  - die Besonderheiten des Problems analysieren und eine gut angepasste Lösung finden
  - Algorithmen entwickeln und implementieren

- Die Studierenden können wesentliche Merkmale von

    - Suchalgorithmen
    - Algorithmen für Graphen
    - Geometrische Algorithmen
    - Sortieralgorithmen
    - Mathematische Algorithmen

  verstehen und Ihre Besonderheiten beachten, um sie anwenden zu können. Sie können entscheiden, welche Algorithmen für eine bestimmte Problemstellung geeignet sind, welche nicht, und sie können diese Entscheidung auch begründen.

- Die Studierenden sind in der Lage, ihr Wissen auch auf algorithmische Fragestellungen, jenseits der Suchalgorithmen, Algorithmen für Graphen, Geometrischen Algorithmen, Sortieralgorithmen und Mathematische Algorithmen anzuwenden.

Lehrinhalte

- Datenstrukturen
- Analyse von Algorithmen
- Hashing
- Suchalgorithmen
- Algorithmen für Graphen
- Geometrische Algorithmen
- Sortieralgorithmen
- Teile und Herrsche
- Mathematische Algorithmen
- NP-vollständige Probleme

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert (Präsenzstudium). Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (nun wieder Präsenzstudium). Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen. Zur Unterstützung des Selbststudiums kann ein Tutorium angeboten werden.

Bestimmte Lehrinhalte überlappen sich mit Lehrinhalten aus anderen Modulen. Zum Beispiel werden Beweismethoden auch in Mathematik I behandelt, und im Modul Praktische Informatik werden Programmierkenntnisse vermittelt. Dieses Vorgehen soll eine Orientierung in der Stofffülle vermitteln und Zusammenhänge erkennbar machen. Es basiert auf einem didaktischen Konzept, das sich an das Spiralprinzip aus der Schuldidaktik anlehnt.

Hinweise

Literatur zu Algorithmen und Datenstukturen:

- R. Sedgewick, „Algorithmen“
- M. Goodrich and R. Tamassia ‚Algorithm Design”

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Algorithmen und Datenstrukturen (wöchentlich im Sommersemester)	6 ECTS-Punkte (SWS: V2+Ü2)

Titel **Software Engineering I** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
2	jährlich	Wöchentlich im Sommersemester	6	Präsenz (Vorlesung + Übung): **56.25** Selbststudium: **93.75** Prüfungsvorbereitung und Prüfung: **30** **Summe 180**	Deutsch	Bernd Fröhlich

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Bachelor of Science	Modul „Einführung in die Programmierung“	Eine Klausur mit Dauer 90-120 Minuten Aus didaktischen Gründen kann stattdessen auch eine mündliche Prüfung (30 Minuten – 45 Minuten) angeboten werden. Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen sind eine Voraussetzung für die Zulassung zur Prüfung.

Qualifikationsziele

Das Modul zielt auf den Erwerb zentraler Grundlagen und Methoden für moderne imperative, objektorientierte, funktionale und generische Programmierung. Nach einem erfolgreichen Besuch der Lehrveranstaltung können die Studierenden konkrete Probleme mit Begrifflichkeiten der Programmiermodelle abstrakt darstellen, Lösungsansätze für einfach Programmieraufgaben entwickeln und diese implementieren. Sie sind in der Lage ihre Vorgehensweise nachvollziehbar zu begründen.

Qualifikationsziele aus Software Engineering I:

- Die Studierenden kennen und beherrschen wesentliche Konzepte imperativer, objektorientierter, funktionaler und generischer Programmierung und können diese erklären, anwenden und gegeneinander abgrenzen.
  - Objekte, Zustand, Verhalten, Botschaften, Methoden, Klassen, Instanzen, Klassenhierarchien
  - Klassendefinition, Konstruktor, Überladung, Initialisierung vs. Zuweisung, Deklaration vs. Definition, grundlegende Regeln für den Aufbau einer Klasse (class mechanics), Gültigkeitsbereich und Lebensdauer von Objekten
  - const correctness als wichtiges Element zur sicheren Programmierung und als Interface-Versprechen: konstante Objekte, const Parameter und Methoden
  - Übergabe- und Rückgabemechanismen für Methoden und Funktionen: Korrekte Verwendung von Übergabe und Rückgabe per value und per reference in Kombination mit const
  - Templates, Standard Template Library in C++, Container, Iteratoren, Algorithmen
  - Funktoren, anonyme Funktionen (Lambdas), Closures, Funktionen höherer Ordnung
  - Speicherverwaltung, Stack, Freestore, Größe zusammengesetzter Objekte, globale und automatische Variablen, Zeiger, Smart Pointer, Zeigersemantik vs. Wertsemantik, Einsatz von Freestore-Objekten vs. automatische Variablen, Klassendesign für die Verwaltung dynamischer Ressourcen
  - Klassenhierarchien und Vererbung, virtuelle Funktionen, Überschreiben, Interfaces und abstrakte Klassen, polymorphe Variablen, statische und dynamische Typisierung, Einsatz von Vererbung vs. Templates vs. Containment, Liskov-Prinzip
  - Überladen vs. Überschreiben vs. Überdecken

- Für einfache Aufgabenstellungen können die Studierenden die passenden Konzepte imperativer, objektorientierter und generischer Programmierung auswählen und kombinieren sowie eine Lösung in modernem C++ umsetzen. Sie können die Auswahl begründen und die Funktionsweise der Implementierung im Detail erklären.
- Die Studierenden kennen grundlegende Regeln für das robuste Design einzelner Klassen und Klassenhierarchien und können diese anwenden
- Die Studierenden haben grundlegende Programmiererfahrung durch die Übungen und ein abschließendes Miniprojekt zum Thema Ray Tracing erworben. Sie kennen sich mit einer Entwicklungsumgebung, Versionsverwaltung und Software Tests aus und können diese gekonnt einsetzen.
- Zusätzlich werden soziale Fähigkeiten und Schlüsselqualifikationen durch Gruppenarbeit basierend auf konkreten Problemen und Aufgaben geübt.

Lehrinhalte

- Klassen und Klassenhierarchien
- Übergabe- und Rückgabemechanismen für Funktionen und Methoden
- const correctness
- Speicherverwaltung und Zeiger
- generische und funktionale Programmierung
- Robustes Design einzelner Klassen und Klassenhierarchien

Dieses Modul wird durch die Vorlesung *Grundlagen Programmiersprachen* abgedeckt.

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert (Präsenzstudium). Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (nun wieder Präsenzstudium). Zur Unterstützung des Selbststudiums gibt es das Angebot einer freiwilligen Übung pro Woche mit 45 Minuten Dauer.

Bestimmte Lehrinhalte überlappen sich mit Lehrinhalten aus anderen Modulen. Zum Beispiel werden Modellierungskonzepte auch im Modul Praktische Informatik behandelt. Dieses Vorgehen soll eine Orientierung in der Stofffülle vermitteln und Zusammenhänge erkennbar machen. Es basiert auf einem didaktischen Konzept, das sich an das Spiralprinzip aus der Schuldidaktik anlehnt.

Hausaufgaben umfassen ein Maximum an 8 Aufgabenzetteln verteilt über das gesamte Modul. Wissenschaftliche Mitarbeiter und studentische Tutoren aus höheren Semestern betreuen die Studierenden in den Übungen und stehen für Rückfragen und Diskussion zur Verfügung.

Nach der Korrektur der eingereichten Übungsaufgaben durch die Betreuenden werden diese mit Anmerkungen an die Studierenden zurückgegeben und beispielhafte Lösungen sowie typische Fehler werden in der Präsenzphase besprochen.

Die einzelnen Veranstaltungen bestehen aus wöchentlichen 90-minütigen Vorlesungen sowie zwei betreuten Übungsterminen mit 90 und 45 Minuten Dauer.

Hinweise

Literatur:

- B. Stroustrup: Programming: Principles and Practice Using C++

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Grundlagen Programmiersprachen	6.0 ECTS-Punkte (SWS: V2+Ü3)

Titel **Human-Computer Interaction** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
2	jährlich	Wöchentlich im Wintersemester	6	Präsenz (Vorlesung + Übung): 34 Selbststudium und Übungsaufgaben: 120 Prüfungsvorberei-tung einschließ-lich der Prüfungen: 26 Summe 180	Englisch	Hornecker

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informa-tik, Bachelor of Science	Grundwissen Abitur	Klausur. Dauer: 120-150 Minuten. In Sonderfällen kann aus didaktischen Gründen stattdessen auch eine mündliche Prüfung (30 - 45 Minuten) angeboten werden. Die regelmäßige Bearbeitung der Übungsaufgaben ist eine Voraussetzung für die Zulassung zur Klausur.

Qualifikationsziele

Im Rahmen des Moduls

- Die Studierenden kennen die unter ‚Lehrinhalte' genannten Begriffe und Methoden und können sie erklären (z.B. in Bezug auf relevante Theorien und Methoden).
- Sie verstehen und können begründen:
  - was Usability ausmacht und warum gute Usability wichtig für den erfolgreichen Einsatz von Softwaresystemen ist,
  - welche Eigenschaften menschlicher Wahrnehmung und Kognition Einfluss auf gute Usability besitzen,
  - warum ein Verstehen des Nutzungskontextes sowie der Benutzerbedürfnisse und –eigenschaften zentral für die Entwicklung aufgabenangemessener Systeme ist,
  - was die Grundprinzipien eines nutzerzentrierten Designprozesses sind
  - welche Methoden zur Erfassung der Anforderungen (Requirements) eingesetzt werden können und was dabei zu beachten ist
  - den Stellenwert von Prototyping, Nutzertests und sowie von Theorien und Modellen über Nutzer und deren Verhalten,
  - warum neue Technologien zur Entwicklung neuer Interaktionsmethoden führen und neue Usabilityprobleme erzeugen können,
  -
- Die Studierenden sind in der Lage, dieses Wissen auf einfache Beispielszenarien anzuwenden.
  - Sie können Interfaces bezüglich wichtiger Konsequenzen für Prozesse menschlicher Wahrnehmung und Kognition analysieren und bewerten sowie gängige Probleme erkennen.
  - Sie können Beispielszenarien und Systeme bezüglich ihrer Usability analysieren und bewerten.
  - Sie können unter Berücksichtigung des Wissens über Usabilityprinzipien sowie Grundlagen menschlicher Wahrnehmung und Kognition einfache Benutzungsschnittstellen entwerfen.
- Sie können ihr Wissen auch auf anspruchsvollere Probleme bzw. komplexere Beispielszenarien übertragen und anwenden
  - Sie können in Anwendungsfällen (Beispielszenarien und Systeme) entscheiden, ob diese Usabilityprinzipien bzw. Prinzipien des User-Centered Designs einhalten oder verletzen, und ihre Bewertung nachvollziehbar begründen
  - Sie können für ausgewählte Klassen von Usabilityproblemen Lösungsansätze generieren. Dies umfasst auch die wichtigsten Klassen von Usabilityproblemen, die in Eigenschaften menschlicher Wahrnehmung und Kognition fußen.
  - Sie können für vorgegebene Szenarien Vorgehensvorschläge für einen benutzerzentrierten Designprozess entwickeln und ihre Methodenwahl begründen.

- Die Studierenden können die gelehrten Methoden und ihr Wissen im Kontext einfacher Beispielszenarien praktisch anwenden
  - z.B. Papierprototypen entwickeln, einen einfachen Usability Test durchführen, eine KLM-Analyse durchführen, Systemanforderungen bestimmen, ...
  - z.B. ein Interface bezüglich auftretender (Farb-)Kontraste, Gruppierung von Bedienelementen/Anzeige oder Anforderungen an Aufmerksamkeit und Gedächtnissysteme analysieren und in Hinblick auf eine Eignung für generelle wie spezielle Nutzergruppen bewerten.
  -

Zusätzlich werden soziale Fähigkeiten und Schlüsselqualifikationen durch Gruppenarbeit basierend auf konkreten Problemen und Aufgaben geübt

Lehrinhalte

- Prozess des User-Centered Designs (Requirements Analyse, Prototyping, Evaluation)
- Grundprinzipien der Software-Ergonomie und Usability, Designregeln und -prinzipien, wie z.B. Affordances, Constraints, Mapping, etc.
- Mentale Modelle, Interface Metaphern, Interaction Styles,
- Grundlegende Methoden der Benutzerforschung (Interviews, Fokusgruppen, Beobachtung, Logfile-Analyse, Usability-Tests, Feldstudien, Experiment)
- Die Rechte von Teilnehmern an Benutzerstudien (Informed Consent etc)
- Dokumentationsmethoden der Anforderungsanalyse (Personas, Storyboards, User Profile),
- Designmethoden und -werkzeuge (Paper Prototyping, Wizard of Oz, Videoprototypen, horizontale und vertikale Prototypen)
- Grundlegende Interaktionstypen und –stile sowie Eingabe- und Ausgabetechnologien (historische sowie moderne, u.a. Robotik, VR, AR, Tangible Interaction...)
- die Rolle von Interaktionsmetaphern sowie deren Vor- und Nachteile
- Deskriptive und vorhersagende Modellierungsmethoden der HCI (Hierarchical Task Analysis, Fitts Law, Keystroke-Level Model etc.)

Lehr- und Lernmethoden / Didaktisches Konzept

Vorlesungen und praktische Übungen kombiniert mit individueller und Gruppenarbeit zu theoretischen und praktischen Aspekten der Inhalte.

In praktischen Übungen werden die Inhalte vertieft. Über regelmäßig zu bearbeitende Übungsaufgaben wird eine kontinuierliche Aufarbeitung der Inhalte eingefordert. Die Übungen enthalten Elemente projektorientierter Gruppenarbeit zu konkreten Beispielproblemen und –szenarien (problembasiertes Lernen). Die vermittelten Methoden werden exemplarisch eingesetzt und angewendet sowie das vermittelte Wissen praktisch umgesetzt.

Nach der Korrektur der eingereichten Übungsaufgaben durch die Betreuenden werden diese mit Anmerkungen an die Studierenden zurückgegeben und beispielhafte Lösungen sowie typische Fehler werden in der Präsenzphase besprochen.

Hausaufgaben umfassen ein Maximum an 6 Aufgabenzetteln verteilt über das Semester. Wissenschaftliche Mitarbeiter und studentische Tutoren aus höheren Semestern betreuen die Studierenden in den Übungen und stehen zusammen mit den Lehrenden für Rückfragen und Diskussion zur Verfügung

Die Veranstaltung besteht aus wöchentlichen 90-minütigen Vorlesungen sowie einer Einzelstunde (45 min) Übung, welche typischerweise zweiwöchentlich mit 90 Minuten Dauer stattfinden. Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient unter anderem der Selbstkontrolle der Studierenden und ist für die kontinuierliche Vertiefung der Lehrinhalte zu empfehlen.

Das Modul vermittelt praktisch-methodisches und theoretisches Wissen auf Bachelorniveau, das in einer Klausur geprüft wird.

Hinweise

Literatur
J. Preece, H. Sharp, Y. Rogers. Interaction Design: Beyond Human-Computer Interaction, 4th Edition. Wiley 2015
D. Norman. The Design of Everyday Things. Basic Books
A. Dix, J. Finlay, G. Abowd, R. Beale. Human-Computer Interaction. Prentice Hall 2014

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Human-Computer Interaction	6.0 ECTS-Punkte (SWS: V2+Ü1)

Titel **Numerik** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
3 und 4	jährlich	Wöchentlich mit jeweils einer Lehrveranstaltung im Wintersemester	6	Präsenz (Vorlesung + Übung): **45** Belegbearbeitung: **30** Selbststudium: **80** Prüfungsvorbereitung einschließlich der Prüfungen: **25** **Summe 180**	Deutsch	Prof. Gürlebeck

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Bachelor of Science	Inhalte von Algebra und Analysis	Mündliche Prüfung (30 - 45 Minuten). Die erfolgreiche Bearbeitung der Belegaufgaben ist eine Voraussetzung für die Zulassung zu den Klausuren.

Qualifikationsziele

Das Modul zielt auf das Verständnis von Sachverhalten, Grundlagen und Methoden, die für ein Studium der Informatik bzw. Medieninformatik unverzichtbar sind. Den Studierenden wird ein Einstieg in Teilgebiete der Mathematik ermöglicht, die für Informatiker besonders wichtig sind. Ihnen werden Begriffe und Verfahren der Numerischen Mathematik vermittelt. Nach einem erfolgreichen Besuch der Lehrveranstaltungen können die Studierenden konkrete Probleme mit Begriffen der Mathematik formulieren und analysieren, die wesentlichen Merkmale erfassen (Abstraktion) und mit Standardmethoden der Numerik geeignete Lösungsansätze entwickeln. Dabei werden sowohl die geistigen Grundtechniken Konkretisieren bzw. Spezialisieren als auch Verallgemeinern vermittelt. Sie sind in der Lage, unter unterschiedlichen Problemlösungsansätzen bzw. unterschiedlichen Algorithmen einen geeigneten auszuwählen und diese Wahl nachvollziehbar zu begründen. Nicht zuletzt soll das Modul zur Förderung des objektiven und sicheren Denkens beitragen sowie zur Urteilsfähigkeit und Selbstkontrolle erziehen.

Spezielle Qualifikationsziele aus der Numerik:

- Die Studierenden sind sicher im Umgang mit den folgenden Begriffen und können sie aktiv verwenden:
  - Computerzahlen, Rechnen mit Computerzahlen
  - Rundungsfehler für Grundrechenarten, Funktionen einer und mehrerer Veränderlicher und für Matrixoperationen
  - Interpolation
  - Steigungen
  - Interpolationsfehler
  - Splinefunktionen, Basissplines
  - Approximation
  - Finite Differenzen, numerische Differentiation
  - Taylorabgleich
  - Numerische Integration, Quadraturformeln, Exaktheitsgrad, Stabilität
  - Stabilität

- Die Studierenden sind in der Lage, ihr Wissen unter anderem in den folgenden Bereichen bzw. zur Beantwortung der folgenden Fragestellungen anzuwenden:
  - Rundungsfehler und ihre Fortpflanzung in elementaren Rechnungen zu beschreiben und abzuschätzen
  - Die Fehler beim numerischen Lösen von Gleichungssystemen zu beurteilen
  - Messdaten zu interpolieren bzw. zu approximieren sowie Auswahl einer geeigneten Methode
  - Aufwandsanalyse für Interpolation und Approximation

- Ableitungen erster und höherer Ordnung zu approximieren
- Funktionen numerisch zu integrieren

- Die Studierenden können ihr Wissen auf anspruchsvolle Probleme bzw. im Zusammenhang mit komplexen Beispielen anwenden:
  - Rundungsfehleranalyse komplexer Berechnungen und Anpassung entsprechender Algorithmen basierend auf der Abschätzung der Konditionszahlen.
  - Darstellung von Signalen und Messdaten mit Hilfe geeignet ausgewählter Interpolationsverfahren bzw. auf der Basis von Approximationsverfahren.
  - Erkennen von Querbezügen zur Stochastik
  - Approximation von Ableitungen direkt oder im Zusammenhang mit der Lösung von Differentialgleichungen, näherungsweise Lösung von Problemen aus der Analysis der Funktionen einer oder mehrerer Veränderlicher
  - Numerische Integration im Zusammenhang mit Längenberechnungen, Flächenberechnungen, Fourierkoeffizienten, Stammfunktionen

- Die Studierenden können Probleme der Informatik mit Hilfe von Konzepten
  - der Interpolations- und Approximationstheorie vereinfachend beschreiben, diskrete Daten glätten und die Fehlereinflüsse durch die Implementierung auf dem Computer erfassen und zum Teil steuern
  - Probleme aus der Differential- und Integralrechnung diskretisieren und formalisieren, konkrete Fragestellungen analysieren, systematisch nach möglichen mathematischen Lösungen suchen und selbst entsprechende Algorithmen entwickeln und zur Implementation auf dem Computer vorbereiten. Sie sind in der Lage, die Auswahl der Verfahren zu beurteilen und die erhaltenen Ergebnisse qualitativ und quantitativ zu bewerten.

Lehrinhalte

- Rechnen mit Computerzahlen
- Rundungsfehler, Fortpflanzung und Abschätzung, relative Konditionszahlen
- Konditionszahl einer Matrix, Einfluss auf die numerische Lösung linearer Gleichungssysteme
- Interpolation; Lagrange-, Newton-, Hermite-, Spline- und trigonometrische Interpolation
- Abschätzung des Interpolationsfehlers
- Approximation; beste Approximation, kleinste Fehlerquadratmethode
- Numerische Differentiation; Konstruktionsprinzipien, Taylorabgleich, Fehlerabschätzung
- Numerische Integration; elementare Formeln vom Newton und Newton-Cotes Typ, Gauß-Quadraturformeln, Integration periodischer Funktionen

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert (Präsenzstudium). Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (nun wieder Präsenzstudium). Die aktive Teilnahme an den Übungen und die regelmäßige Bearbeitung der Übungsaufgaben wie auch der studienbe-gleitenden Belegaufgaben (Prüfungszulassung) dienen hierbei unteranderem der Selbstkontrolle der Studierenden und ist für die kontinuierliche Vertiefung der Lehrinhalte zu empfehlen. Zur Unterstützung des Selbststudiums kann ein Tutorium angeboten werden.

Hinweise

Literatur:

- Lothar Papula: Mathematik für Ingenieure und Naturwissenschaftler, Band 1,2,3. Vieweg+Teubner Verlag
- Gerald Teschl: Mathematik für Informatiker, Band 1,2. Springer-Verlag
- H. Schwetlick, H. Kretzschmar: Numerische Verfahren für Naturwissenschaftler und Ingenieure, Fachbuchverlag Leipzig
- W. Dahmen, A. Reusken: Numerik für Ingenieure und Naturwissenschaftler, Springer

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Numerik (wöchentlich im Wintersemester)	6 ECTS-Punkte (SWS: V2+Ü2)

Titel **Formale Sprachen** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
3	jährlich	Wöchentlich im mit jeweils einer Lehrveranstaltung im Winter-semester	6	Präsenz (Vorlesung + Übung): **45** Selbststudium: **110** Prüfungsvorbereitung einschließlich der Prüfungen: **25** **Summe 180**	Deutsch	Dr. PD A. Jakoby

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Bachelor of Science	Grundwissen Abitur	Eine Klausur. Dauer: 120 Minuten In Sonderfällen kann aus didaktischen Gründen stattdessen auch eine mündliche Prüfung (30 - 45 Minuten) angeboten werden. Die regelmäßige Bearbeitung der Übungsaufgaben ist eine Voraussetzung für die Zulassung zu den Klausuren.

Qualifikationsziele

Im Rahmen des Moduls werden die theoretischen Grundlagen der Informatik vermittelt, welche für ein Studium der Informatik besonders wichtig sind. Den Studierenden wird ein Einstieg in die Begriffe und Methoden der Analyse von Problemen vermittelt. Nach einem erfolgreichen Besuch der beiden Lehrveranstaltungen können die Studierenden konkrete Probleme mit Begrifflichkeiten der Mathematik abstrakt darstellen, mit Methoden der Theoretischen Informatik analysieren und bezüglich ihrer Lösbarkeit und Zugehörigkeit innerhalb der Chomsky charakterisieren. Sie sind somit in der Lage verschiedene grundlegende Fragen für unterschiedliche Probleme mit Hilfe von Standardverfahren algorithmisch zu lösen bzw. deren Unlösbarkeit nachvollziehbar nachzuweisen.

Spezielle Qualifikationsziele aus den Formalen Sprachen:

- Die Studierenden kennen insbesondere die folgenden Begriffe und Zusammenhänge und können sie anderen erklären:
  - Formale Sprachen und deren Grammatiken
  - grundlegende Formen von Grammatiken und deren Charakterisierung bezüglich der Chomsky-Hierarchie
  - reguläre Sprachen, kontextfreie Sprachen, kontextsensitive Sprachen
  - grundlegende Automaten-Modelle und deren Zuordnung zu den unterschiedlichen Stufen der Chomsky-Hierarchie: Endliche Automaten, deterministischer Kellerautomat, Kellerautomaten, Turing Maschine (mit und ohne Platzbeschränkung)
  - reguläre Ausdrücke und deren Anwendungen
  - Verfahren für Umwandlungen zwischen Grammatiken und den jeweiligen Automaten
  - Verfahren für Umwandlungen zwischen den verschiedenen Beschreibungen für reguläre Sprachen: deterministischer endlicher Automat, nicht-deterministischer endlicher Automat, Epsilon nicht-deterministischer endlicher Automat, reguläre Grammatiken, reguläre Ausdrücke, minimaler endlicher Automat
  - Normalformen von Grammatiken
  - Pumping-Lemmata und deren Beweis: für reguläre Sprachen und für kontextfreie Sprachen
  - Anwendbarkeit der Pumping-Lemmata und deren Schranken

- Satz von Nerode und dessen Beziehung zu minimalen endlichen Automaten
- LL und LR Grammatiken
- Algorithmen zur Umwandlung einer Grammatik in eine Normalform
- elementare Eigenschaften der Stufen der Chomsky-Hierarchie
- elementare Problemstellungen für Formale Sprachen
- elementare Algorithmen für Formale Sprachen: Membership, Gleichheit, Teilmenge
- die Laufzeit und Korrektheit der entsprechenden Algorithmen
- Kenntnis der Abschlusseigenschaften bezüglich Mengenoperationen der jeweiligen Stufen der Chomsky-Hierarchie
- Endlichkeit, Entscheidbarkeit, Abzählbarkeit, Über-Abzählbarkeit
- grundlegende Beweismethoden der Mathematik bzw. der Theoretischen Informatik: direkte und indirekte Beweise, vollständige Induktion, Diagonalisierung
- das Halteproblem
- rekursiv, rekursiv aufzählbar oder nicht rekursiv aufzählbar Mengen

- Die Studierenden sind in der Lage, ihr Wissen unter anderem in den folgenden Bereichen bzw. zur Beantwortung der folgenden Fragestellungen anzuwenden:
  - die grundlegenden Beweistechniken einsetzen, um einfache Beweise zu führen
  - entscheiden, ob eine Menge (bzw. Sprache) endlich, regulär, kontextfrei, kontextsensitiv, abzählbar oder über-abzählbar ist
  - Grammatiken für grundlegende Formale Sprachen angeben und den Beweis der Korrektheit führen
  - die Nicht-Zugehörigkeit einer Sprache zu den regulären bzw. zu den kontextfreien Sprachen mit Hilfe des jeweiligen Pumping-Lemmas zeigen
  - die Zugehörigkeit bzw. die Nicht-Zugehörigkeit einer Sprache zu den regulären Sprachen mit Hilfe der Neroden Relation nachweisen
  - einen Automaten für eine Sprache aus einer gegebenen Grammatik herzuleiten (und umgekehrt)
  - Abschlusseigenschaften bezüglich Mengenoperationen der jeweiligen Stufen der Chomsky-Hierarchie herzuleiten
  - grundlegende Algorithmen (z.B. der CYK-Algorithmus) aus dem Bereich der Formalen Sprachen anzuwenden und zu simulieren
  - die Chomsky- und die Greibach-Normalform für kontextfreie Grammatiken herzuleiten
  - eine einfache Diagonalisierung durchzuführen
  - entscheiden ob eine Menge rekursiv, rekursiv aufzählbar oder nicht rekursiv aufzählbar ist
  - den Satz von Rice zur Entscheidung der Nicht-Rekursivität einer Menge anwenden

- Die Studierenden können ihr Wissen auf anspruchsvolle Probleme bzw. im Zusammenhang mit komplexen Beispielen anwenden:
  - Mittelschwere Sprachen untersuchen (z.B. Herleiten einer entsprechenden Grammatik und Beweis der Korrektheit, Herleiten eines Automaten für diese Sprache, Zeigen der Nicht-Zugehörigkeit zu einer Hierarchiestufe durch Widerspruchsannahme, Pumping-Lemma, Satz von Nerode, Anwendung von Sätzen aus der Vorlesung, …)
  - offen gestellte mathematische Fragestellungen mit Hilfe der oben genannten Beweistechniken beweisen oder, durch Angabe von Gegenbeispielen widerlegen

- selbstständig abstrakte Begriffe in formaler Darstellung verstehen, sich selbständig in die Anwendung grundlegender Methoden einarbeiten
- Anwendung von regulären Ausdrücken zur Suche in großen Datenmengen

- Die Studierenden können Probleme der Informatik mit Hilfe von Konzepten
  - der Automatentheorie,
  - der Grammatiken von Formalen Sprachen,
  - der Algorithmik und
  - der Berechenbarkeitstheorie

  formalisieren, formale Fragestellungen analysieren, systematisch nach möglichen Lösungen suchen und selbst entsprechende Algorithmen entwickeln. In einfachen Fällen können sie die Korrektheit von Algorithmen begründen und ihr asymptotisches Laufzeitverhalten abschätzen.
- Die Studierenden können in Anwendungsfällen entscheiden und bewerten, welche Konzepte und Methoden
  - der Automatentheorie,
  - der Grammatiken von Formalen Sprachen,
  - der Algorithmik und
  - der Berechenbarkeitstheorie

  zur Modellbildung geeignet sind. Insbesondere können sie diese Entscheidung auch nachvollziehbar begründen.

.

Lehrinhalte

- Formale Sprachen
  - Beweistechniken
  - Grammatiken
  - Chomsky-Hierarchie
  - Endliche Automaten
  - Kellerautomaten
  - Turing-Maschine
  - Reguläre Sprachen
  - Kontextfreie Sprachen
  - LL und LR Grammatiken
  - Kontextsensitive Sprachen
  - Chomsky-0 Sprachen
  - Abschlusseigenschaften von Sprachklassen
  - Algorithmen zur Lösung des Membership-Problems (CYK-Algorithmus)

- Algorithmen zur Überführung zwischen Automaten und Grammatiken (bzw. regulären Ausdrücken)
- Nerode Klassen
- Pumping-Lemma für reguläre Sprachen
- Pumping-Lemma für kontextfreie Sprachen
- Normalformen für kontextfreie Sprachen
- Halteproblem
- Diagonalisierung
- Entscheidbarkeit, Abzählbarkeit und Über-Abzählbarkeit
- Loop-Berechenbar
- While-Berechenbar
- Turing-Maschine
- Primitive Rekursion
- Partielle Rekursion
- elementare Techniken zum Umgang mit Turing-Maschinen
- Universelle Turing-Maschine
- Entscheidbarkeit
- Nicht-Entscheidbarkeit
- Church-Turing-These
- Many-One-Reduktion
- Turing Reduktion
- Rekursionstheorie
- S-m-n Theorem
- Rekursionstheorem
- Fixpunktsatz
- Satz von Rice

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert (Präsenzstudium). Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (nun wieder Präsenzstudium). Zur Unterstützung des Selbststudiums kann ein Tutorium angeboten werden. Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen.

Bestimmte Lehrinhalte überlappen sich mit Lehrinhalten aus anderen Modulen. Zum Beispiel werden Beweismethoden auch in Mathematik Vorlesungen und Informatik Strukturen behandelt, und im Modul Technische und Praktische Informatik werden Grundkenntnisse der Analyse des asymptotischen Laufzeitverhaltens und der Effizienz vermittelt. Dieses Vorgehen soll eine Orientierung in der Stofffülle vermitteln und Zusammenhänge erkennbar machen. Es basiert auf einem didaktischen Konzept, das sich an das Spiralprinzip aus der Schuldidaktik anlehnt.

Die jeweiligen Veranstaltungen bestehen aus wöchentliche 135 Minuten Vorlesungen sowie einer Einzelstunde (45 min) Übung.

Hinweise

Handouts und Literatur zu Formalen Sprachen:

- Jakoby: Folien zur Vorlesung Formalen Sprachen
- Engeler, Peter Läuchli: Berechnungstheorie für Informatiker (Leitfäden und Monographien der Informatik); Vieweg+Teubner Verlag; 1992
- Hopcroft, Motwani, Ullman: Einführung in Automatentheorie, Formale Sprachen und Berechenbarkeit, Pearson Studium, 2011
- Tourlakis: Theory of Computation; Wiley, 2014

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Formale Sprachen (wöchentlich im Wintersemester)	6 ECTS-Punkte (SWS: V3+Ü1)

Titel **Parallele und verteilte Systeme** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
1	jährlich	Wöchentlich im Wintersemester	6	Präsenz (Vorlesung + Übung): **45** Projektarbeit: **30** Selbststudium: **75** Prüfungsvorbereitung: **30** **Summe 180**	Deutsch	Prof. Dr.-Ing. Volker Rodehorst

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Bachelor of Science	• Einführung in die Programmierung, • Software-Engineering I, • Algorithmen und Datenstrukturen	Klausur (120 Minuten). Die erfolgreiche Bearbeitung der Übungsaufgaben ist eine Voraussetzung für die Zulassung zur Klausur.

Qualifikationsziele

Die Veranstaltung gibt eine Einführung in die grundlegenden Konzepte paralleler und verteilter Programmierung. Behandelt werden aber auch praktische Aspekte zur Programmierung von Mehrkern-Systemen, die verteilte Berechnung auf Rechnercluster und die massive Parallelität mittels Grafikprozessoren.

Spezielle Qualifikationsziele:

- Die Studierenden sind in der Lage, ihr theoretisches und praktisches Wissen zur Analyse und effektiven Beschleunigung von aufwändigen Algorithmen einzusetzen.
- Sie kennen folgende Grundbegriffe und können die wesentlichen Zusammenhänge erläutern:
  - Möglichkeiten und Grenzen der Parallelprogrammierung (kontrollparallel, datenparallel)
  - Wettrennen um Ressourcen (Synchronisation, Barriere, Mutex, Semaphor)
  - Analyse und Beseitigen von Datenabhängigkeiten
  - Effiziente Parallelisierung (Granularität, Lastenausgleich)
  - Mehrkern- bzw. Multiprozessor-Berechnung von Threads mit gemeinsamem Speicher
  - Verteiltes Rechnen durch expliziten Nachrichtenaustausch von Prozessen (Cluster, Grid, Cloud)
  - Interprozesskommunikation ((nicht-)blockierend, (a)synchron, gepuffert, kollektiv)
  - Speichermodelle (Cache, global, lokal, privat)
  - Ausführungsmodelle (Online-Kompilierung, Topologie, Befehlswarteschlange)
  - Mathematische Darstellung zur Modellierung und Verifikation
- Durch die begleitenden Übungen mit den Entwicklungsumgebungen OpenMP, MPI und OpenCL sind die Studierenden in der Lage ihr theoretisches Wissen auch in der Praxis auf anspruchsvolle Probleme und komplexe Beispiele zu übertragen.
- Sie können für eigene Anwendungen abwägen, welche Verfahren zur Beschleunigung am besten geeignet sind und die Entscheidung nachvollziehbar begründen.
- Die Studierenden können Fragestellungen von verteilten Systemen mit Hilfe
  - von Petrinetzen
  - der (linearen) temporalen Logik

  mathematisch formalisieren, Spezialfälle untersuchen und geeignete Lösungen finden.

Lehrinhalte

- Parallele und verteilte Systeme
  - Grundlagen und Architekturen
  - Mehrkern-Programmierung mit OpenMP
  - Programmierung verteilter Systeme mit MPI
  - Massive Parallelisierung auf Grafikkarten mit OpenCL
  - Modellierung und Verifikation mit Petrinetzen und temporaler Logik

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in Vorlesungen präsentiert und im Rahmen von Übungen vertieft (Präsenzstudium). Übungsaufgaben sind im Selbststudium zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden ausgewählte Lösungen in der folgenden Übung besprochen. Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen.

Über regelmäßig zu bearbeitende Übungsaufgaben wird eine kontinuierliche Aufarbeitung der Inhalte bewirkt.

Hinweise
Literatur: • V. Rodehorst: Vorlesungsfolien, online. • G. Bengel, C. Baun, M. Kunze und K.U. Stucky: Masterkurs Parallele und Verteilte Systeme: Grundlagen und Programmierung von Multicore-Prozessoren, Multiprozessoren, Cluster, Grid und Coud, 2. Aufl., Springer, 503 S., 2015. • S. Hoffmann und R. Lienhart: OpenMP - Eine Einführung in die parallele Programmierung mit C/C++, Informatik im Fokus, Springer, 172 S., 2009. • A.S. Tanenbaum und M. van Steen: Verteilte Systeme: Prinzipien und Paradigmen, 2. Aufl., Pearson Studium - IT, 768 S., 2007. • T. Rauber und G. Rünger: Parallele Programmierung, 2. Aufl., Springer, 485 S., 2007.

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
• Parallele und verteilte Systeme, Vorlesung • Parallele und verteilte Systeme, Übung • Parallele und verteilte Systeme, Projekt	• 2 SWS / 3 ECTS-Punkte • 1 SWS / 1,5 ECTS-Punkte • 1 SWS / 1,5 ECTS-Punkte

Titel	**Software Engineering II**	Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
3	jährlich	Wöchentlich im mit jeweils einer Lehrveranstaltung im Wintersemester	6	Präsenz (Vorlesung + Übung): **60** Belegbearbeitung: **30** Selbststudium (einschließlich Tutorien, falls angeboten): **70** Prüfungsvorbereitung einschließlich der Prüfung: **20** Summe **180**	Deutsch	Professur Software Engineering, voraussichtliche Neubesetzung Sommersemester 2021

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Medieninformatik, Bachelor of Science	Modul Software Engineering I	Klausur. Dauer: 90-120 Minuten In Sonderfällen kann aus didaktischen Gründen stattdessen auch eine mündliche Prüfung (30 - 45 Minuten) angeboten werden. Die regelmäßige Bearbeitung der Übungsaufgaben ist eine Voraussetzung für die Zulassung zu den Klausuren.

Qualifikationsziele

Das Modul zielt auf den Erwerb zentraler Grundlagen und Methoden ab, die für die Entwicklung von Software unverzichtbar sind. Den Studierenden wird ein Einstieg in die moderne imperative, objektorientierte und generische Programmierung ermöglicht, und ihnen werden Begriffe und Methoden des Softwaredesigns vermittelt. Nach einem erfolgreichen Besuch der Lehrveranstaltung können die Studierenden konkrete Probleme mit Begrifflichkeiten der objektorientierten Programmierung abstrakt darstellen, mit Methoden des Softwareentwurfs Lösungsansätze entwickeln, diese Ansätze analysieren (vor allem in Bezug auf Vor- und Nachteile bezüglich konkreter Einsatzszenarien) und die Lösungsansätze implementieren. Sie sind in der Lage unter unterschiedlichen Problemlösungsansätzen bzw. unterschiedlichen Prozessmodellen einen geeigneten auszuwählen und diese Wahl nachvollziehbar zu begründen.

Spezielle Qualifikationsziele aus dem Software Engineering II:

- Die Studierenden kennen insbesondere die folgenden Begriffe und Zusammenhänge und können sie anderen erklären:
  1. Softwareprozessmodelle und -methoden
     1. traditionell: Wasserfallmodell, Spiralmodell
     2. agil: Scrum, Extreme Programming
     3. open-source: „Bazaar“-Modell
  2. Revision Control System & Build-System
  3. Unit Testing & Continuous Integration
  4. Design Patterns
     1. Creation: Abstract Factory, Factory Method, Singleton, Prototype
     2. Behaviour: Command, Iterator, Visitor, Observer
     3. Structure: Adapter, Facade, Proxy, Composite, Decorator
     4. UI Patterns
  5. Compiler, Linker, Library, Object, Debugger
  6. Qualitätsmetriken für Software
  7. Requirements Engineering

- Die Studierenden sind in der Lage, ihr Wissen unter anderem in den folgenden Bereichen bzw. zur Beantwortung der folgenden Fragestellungen anzuwenden:
  1. die behandelten Design Patterns in einer objektorientierten Programmiersprache implementieren (z.B. Composite zur Darstellung von Baumstrukturen)
  2. ein Lasten- und Pflichtenheft für ein Softwareentwicklungsprojekt erstellen
  3. User Stories und Tasks für einen agilen Softwareentwicklungsprozess erstellen
  4. ein objektorientiertes Softwaredesign in UML modellieren
  5. traditionelle (SVN) und verteilte (Git) Revisionskontrollsysteme sinnvoll einzusetzen

- Die Studierenden können ihr Wissen auf anspruchsvolle Probleme bzw. im Zusammenhang mit komplexen Beispielen anwenden:
  1. die Zusammenhänge zwischen bestimmten Design Patterns nachvollziehen (z.B. AbstractFactory, FactoryMethod & Prototype)
  2. ein gegebenes UML-Diagramm oder „Box&Line"-Diagramm analysieren und in der dargestellten Softwarearchitektur verwendete Konzepte identifizieren
  3. eine Anforderungsanalyse für ein gegebenes Software-Entwicklungsproblem durchführen und ein geeignetes Software-Prozessmodell auswählen
  4. gegebenen Source-Code analysieren, Qualitätsmetriken dafür bestimmen sowie darin verwendete Design Patterns (auch UI-Patterns) erkennen und beschreiben
  5. Fehler in einem bestehenden Softwareprojekt mit Hilfe eines Debuggers finden und analysieren

- Die Studierenden können Probleme der Informatik mit Hilfe von Konzepten aus
  1. Unified Modelling Language (UML)
  2. Design Patterns
  3. Software-Prozessmodellen

  formalisieren, formale Fragestellungen analysieren, systematisch nach möglichen Lösungen suchen und selbst entsprechende Lösungsansätze entwickeln. Die Studierenden können in Anwendungsfällen zudem entscheiden und bewerten, welche dieser Konzepte und Methoden zur Modellbildung geeignet sind. Insbesondere können sie diese Entscheidung auch nachvollziehbar begründen.

- Zusätzlich werden soziale Fähigkeiten und Schlüsselqualifikationen durch Gruppenarbeit basierend auf konkreten Problemen und Aufgaben geübt.

Lehrinhalte

- Best Practices der Softwareentwicklung
- Software-Prozessmodelle (traditionell & agil)
- Design Patterns für Architektur & Code
- Codequalität, Testing & Continuous Integration
- Build-Systeme, Compiler/Linker & Debugger
- Open-Source-Software
- Requirements Engineering

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert (Präsenzstudium). Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (nun wieder Präsenzstudium). Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen. Zur Unterstützung des Selbststudiums kann ein Tutorium angeboten werden.

Bestimmte Lehrinhalte überlappen sich mit Lehrinhalten aus anderen Modulen. Zum Beispiel werden Modellierungskonzepte auch im Modul Praktische Informatik behandelt. Dieses Vorgehen soll eine Orientierung in der Stofffülle vermitteln und Zusammenhänge erkennbar machen. Es basiert auf einem didaktischen Konzept, das sich an das Spiralprinzip aus der Schuldidaktik anlehnt.

Hausaufgaben umfassen ein Maximum an 7 Aufgabenzetteln verteilt über das gesamte Modul. Wissenschaftliche Mitarbeiter und studentische Tutoren aus höheren Semestern betreuen die Studierenden in den Übungen und stehen für Rückfragen und Diskussion zur Verfügung.

Nach der Korrektur der eingereichten Übungsaufgaben durch die Betreuenden werden diese mit Anmerkungen an die Studierenden zurückgegeben und beispielhafte Lösungen sowie typische Fehler werden in der Präsenzphase besprochen.

Die einzelnen Veranstaltungen bestehen aus wöchentlichen 90-minütigen Vorlesungen sowie einer Doppelstunde Übung.

Hinweise

Literatur zu Software-Entwurf:	
Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Software Engineering II	6.0 ECTS-Punkte (SWS: V2+Ü2)

Titel **Datenbanken** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
3	jährlich	Wöchentlich mit jeweils einer Lehrveranstaltung im Winter-semester	6	Präsenz (Vorlesung + Übung): **60** Selbststudium: 9**0** Prüfungsvorbereitung einschließlich der Prüfungen: 3**0** **Summe 180**	Deutsch	Benno Stein

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Medieninformatik, Bachelor of Science	Erfolgreiche Absolvierung der Vorlesungen • Technische und Praktische Informatik • Diskrete Mathematik	Klausur. Dauer: 1.5h In Sonderfällen kann aus didaktischen Gründen stattdessen auch eine mündliche Prüfung (30 - 45 Minuten) angeboten werden. Die regelmäßige Bearbeitung der Übungsaufgaben ist eine Voraussetzung für die Zulassung zu den Klausuren.

Qualifikationsziele

Im Rahmen des Moduls werden Grundlagen moderner Informationssysteme mit dem Schwerpunkt Datenbanken behandelt. Die Vorlesung ist grundlagen- und methodenorientiert ausgerichtet. Die Qualifikationsziele im Einzelnen sind:

- Grundbegriffe von Datenbanken kennen und einordnen können
- Kenntnis von und sicherer Umgang mit Techniken zur Modellierung von Datenbankanwendungen
- Beherrschung der Umsetzung externer Schemata in relationale Schemata
- Beherrschung der Logik-basierten Grundlagen von Anfragesprachen
- Erfahrung und verbesserter Umgang mit formalen Methoden
- Praktische Erfahrung bei der Datenanfrage und Datenmanipulation auf der Basis von SQL
- Beherrschung der theoretischen Grundlagen von Datenbanksystemen
- Verständnis für die Grenzen von Datenbanksystemen
- Beherrschung von Vorgehensweisen, um sich selbst weiterbilden können

Lehrinhalte

Die Vorlesung gibt eine Einführung in die Konzepte moderner Datenbanksysteme und stellt den Datenbankentwurf für klassische Datenmodelle, insbesondere für das Relationenmodell vor. Hierzu gehören folgende Vorlesungseinheiten:

- Grundlegendes zum Datenbankentwurf und Datenbankmodelle
- Konzeptueller Datenbankentwurf mit einem syntaktisch reichem Modell
- Logischer Datenbankentwurf mit dem relationalen Modell
- Grundlagen relationaler Anfragesprachen: Relationale Algebra
- Grundlagen relationaler Anfragesprachen: Tupel- und Domänenkalkül
- SQL: Datenanfrage, Datendefinition, Datenmanipulation
- SQL: Anwendungsprogrammierung

- Relationale Entwurfstheorie: Funktionale Abhängigkeiten und Normalformen
- Relationale Entwurfstheorie: Entwurfsalgorithmen

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert (Präsenzstudium). Es werden statische und dynamische Beamer-Präsentationen mit dem Einsatz der Tafel kombiniert. Der Beamer-Einsatz ermöglicht u.a. die schnelle und übersichtliche Präsentation und Kombination von Inhalten in verschiedenen Kontexten. Beispiele: gleichzeitige Darstellung von ER-Diagrammen und ihrer relationalen Entsprechung, Animation von Relationenzerlegungen, Demonstration vom SQL-Aufrufen an Datenbanken.

Die Tafel wird eingesetzt, um Beweise und formale Definitionen schrittweise zu entwickeln und die Herausforderungen und Schwierigkeiten bei der Entwicklung zu diskutieren. Beispiel: Schrittweise Herleitung eines Tupel-Kalkülausdrucks auf Basis einer relational spezifizierten Anfrage.

Weiterhin wird die Tafel dafür eingesetzt, um auf Rückfragen von Studierenden spontan zu reagieren und entsprechende Grundlagen oder Beispiele ad-hoc herzuleiten.

Alle Lehrinhalte werden im Rahmen von Übungen vertieft. Hierzu zählt sowohl die Anwendung der Theorie im Rahmen praktischer Modellierungs- und Programmieraufgaben als auch das Wiederholen und Nachvollziehen der theoretischen Grundlagen anhand von kleineren Beweisen.

Pro Übungsserie gibt es Pflichtaufgaben und freiwillige Aufgaben. Die Bearbeitung der Pflichtaufgaben geschieht in Gruppen von 2-3 Studierenden. Diese Gruppen bleiben typischerweise in ihrer Zusammensetzung im Verlauf einer Vorlesung unverändert. Die Pflichtaufgaben werden durch die Betreuenden korrigiert und die Lösungen in einer gemeinsamen Sitzung besprochen. Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen.

Zur Unterstützung des Selbststudiums kann ein Tutorium z bestimmten Inhalten angeboten werden.

Hinweise

Empfohlene Literatur :

- Ramez Elmasri, Shamkant B. Navathe. Fundamentals of Database Systems. 6th edition, Addison Wesley, 2010.
- Alfons Kemper, Andre Eickler. Datenbanksysteme - Eine Einführung. 8. Auflage, Oldenbourg, 2011.
- Andreas Heuer, Gunter Saake. Datenbanken: Konzepte und Sprachen. 5. Auflage, mitp, 2013.
- Gottfried Vossen. Datenmodelle, Datenbanksprachen und Datenbankmanagement-Systeme.5. Auflage, Oldenbourg, 2008.

Umfangreiche und aktuelle Literaturangaben sind im Script verlinkt.

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Datenbanken (wöchentlich im Wintersemester)	6 ECTS-Punkte (SWS: V2+Ü1)

Titel **Stochastik** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
4	jährlich	Wöchentlich im mit jeweils einer Lehrveranstaltung im Sommersemester	6	Präsenz (Vorlesung + Übung): **45** Belegbearbeitung: **30** Selbststudium: **80** Prüfungsvorbereitung einschließlich der Prüfungen: **25** **Summe 180**	Deutsch	Prof. Gürlebeck

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Bachelor of Science	Inhalte von Algebra und Analysis	Klausur (120 Minuten). In Sonderfällen kann aus didaktischen Gründen stattdessen auch eine mündliche Prüfung (30 - 45 Minuten) angeboten werden. Die erfolgreiche Bearbeitung der Belegaufgaben ist eine Voraussetzung für die Zulassung zu den Klausuren.

Qualifikationsziele

Das Modul zielt auf das Verständnis von Sachverhalten, Grundlagen und Methoden, die für ein Studium der Informatik bzw. Medieninformatik unverzichtbar sind. Den Studierenden wird ein Einstieg in Teilgebiete der Mathematik ermöglicht, die für Informatiker besonders wichtig sind. Ihnen werden Begriffe und Verfahren der Stochastik vermittelt. Nach einem erfolgreichen Besuch der Lehrveranstaltung können die Studierenden konkrete Probleme mit Begriffen der Mathematik formulieren und analysieren, die wesentlichen Merkmale erfassen (Abstraktion) und mit Standardmethoden der Stochastik geeignete Lösungsansätze entwickeln. Dabei werden sowohl die geistigen Grundtechniken Konkretisieren bzw. Spezialisieren als auch Verallgemeinern vermittelt. Sie sind in der Lage, unter unterschiedlichen Problemlösungsansätzen bzw. unterschiedlichen Algorithmen einen geeigneten auszuwählen und diese Wahl nachvollziehbar zu begründen. Nicht zuletzt soll das Modul zur Förderung des objektiven und sicheren Denkens beitragen sowie zur Urteilsfähigkeit und Selbstkontrolle erziehen.

Spezielle Qualifikationsziele aus der Stochastik:

- Die Studierenden kennen insbesondere die folgenden Begriffe und Zusammenhänge und können sie anderen erklären:
  - Zufällige Ereignisse und relative Häufigkeiten
  - Grundbegriffe der Kombinatorik
  - Axiomatische Definition der Wahrscheinlichkeit, verschiedene Wahrscheinlichkeitsmodelle
  - Bedingte Wahrscheinlichkeit und stochastische Unabhängigkeit
  - Zufallsgrößen und Verteilungsfunktionen
  - Wichtige Beispiele diskreter Zufallsvariabler (Binomialverteilung, Poissonverteilung,…)
  - Wichtige Beispiele stetiger Zufallsvariabler, insbesondere Gaußsche Normalverteilung
  - Erwartungswerte und Varianzen
  - Stochastische Konvergenz und Grenzwertsätze
  - Beschreibende Statistik: Verschiedene Skalenniveaus der Merkmale, Grafische Darstellung, Boxplots
  - Verschiedene Lage- und Streuungsparameter, deren Berechnung und Eigenschaften
  - Zweidimensionale Darstellungen, Kovarianz und Korrelation
  - Lineare Regression
  - Schließende Statistik: Stichprobenfunktionen
  - Punktschätzungen
  - Intervallschätzungen, Zusammenhang mit Parametertests
- Die Studierenden sind in der Lage, ihr Wissen unter anderem in den folgenden Bereichen bzw. zur Beantwortung der folgenden Fragestellungen anzuwenden:
  - Die Struktur zufälliger Ereignisse mittels Ereignisoperationen analysieren
  - Die Sätze über bedingte Wahrscheinlichkeiten in konkreten Situationen einsetzen
  - Die Abhängigkeit bzw. Unabhängigkeit von Ereignissen zu erkennen (insbesondere Bernoulli-Ketten)
  - Den Typ der Verteilung einer konkreten Zufallsgröße erkennen

- Die zur Beschreibung einer Verteilung notwendigen Parameter identifizieren und bei Bedarf zu schätzen
- Vorliegende Daten anschaulich, übersichtlich und kompakt darzustellen
- Das Skalenniveau eines konkreten Merkmals zu erkennen
- Zweidimensionale Merkmale übersichtlich darzustellen und mittels Korrelations- und Regressionsanalyse zu untersuchen
- In einfachen Fällen von Eigenschaften eines Merkmals in der Stichprobe auf Gesetzmäßigkeiten schließen, die dem Gesamtvorgang zu Grunde liegen

- Die Studierenden können ihr Wissen auf anspruchsvolle Probleme bzw. im Zusammenhang mit komplexen Beispielen anwenden:

  - Die verschiedenen Möglichkeiten der Approximation von Verteilungen effizient einsetzen
  - Den notwendigen Stichprobenumfang für eine statistische Analyse abzuschätzen
  - Bei nur wenig Information über einen zufälligen Vorgang die Grenzwertsätze anwenden
  - Verhältnis von Sicherheit und Genauigkeit bei statistischen Schätzverfahren abwägen

- Die Studierenden können Probleme der Informatik mit Hilfe von Konzepten

  - der Wahrscheinlichkeitsrechnung,
  - der Theorie der Zufallsgrößen,
  - der beschreibenden Statistik und
  - der schließenden Statistik

  formalisieren, konkrete Fragestellungen analysieren, systematisch nach möglichen Lösungen suchen und selbst entsprechende Algorithmen entwickeln. In einfachen Fällen können sie die Angemessenheit der angewandten Verfahren und die Aussagekraft der erzielten Ergebnisse bewerten.

- Die Studierenden können in Anwendungsfällen entscheiden und bewerten, welche Konzepte und Methoden

  - der Wahrscheinlichkeitsrechnung,
  - der Theorie der Zufallsgrößen,
  - der beschreibenden Statistik und
  - der schließenden Statistik

  zur Modellbildung geeignet sind. Insbesondere können sie diese Entscheidung auch nachvollziehbar begründen.

Lehrinhalte

- Zufallsereignisse und deren Wahrscheinlichkeit
- Bedingte Wahrscheinlichkeit und Unabhängigkeit von Zufallsereignissen
- Verteilungen diskreter und stetiger Zufallsgrößen
- Summen unabhängiger Zufallsgrößen und zentraler Grenzwertsatz
- Beschreibende Statistik
- Schließende Statistik, Parameter- und Intervallschätzungen, statistische Tests
- Korrelation und Regression

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert (Präsenzstudium). Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (nun wieder Präsenzstudium). Die aktive Teilnahme an den Übungen und die regelmäßige Bearbeitung der Übungsaufgaben wie auch der studienbe-gleitenden Belegaufgaben (Prüfungszulassung) dienen hierbei unteranderem der Selbstkontrolle der Studierenden und ist für die kontinuierliche Vertiefung der Lehrinhalte zu empfehlen. Zur Unterstützung des Selbststudiums kann ein Tutorium angeboten werden.

Bestimmte Lehrinhalte überlappen sich mit Lehrinhalten aus anderen Modulen. Zum Beispiel werden Mengenoperationen (hier: Ereignisoperationen) und diskrete Wahrscheinlichkeiten auch im Modul Informatik-Strukturen behandelt. Dieses Vorgehen soll eine Orientierung in der Stofffülle vermitteln und Zusammenhänge erkennbar machen. Es basiert auf einem didaktischen Kon-zept, das sich an das Spiralprinzip aus der Schuldidaktik anlehnt.

Hinweise

Literatur:

- Lothar Papula: Mathematik für Ingenieure und Naturwissenschaftler, Band 1,2,3. Vieweg+Teubner Verlag
- Gerald Teschl: Mathematik für Informatiker, Band 1,2. Springer-Verlag

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Stochastik (wöchentlich im Sommersemester)	6 ECTS-Punkte (SWS: V2+Ü2)

Titel **Kryptographie** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
4	jährlich	Wöchentlich im Sommersemester eine Vorlesung, alle zwei Wochen eine Übung. Ein Mini-Projekt gegen Ende der Vorlesungszeit.	6	*Präsenz (Vorlesung + Übung): 34* *Selbststudium: 79* *Mini-Projekt/Hausarbeit/Vortrag: 22* *Prüfungsvorbereitung einschließlich der Prüfungen: 45* *Summe 180*	Englisch	Stefan Lucks

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Bachelor of Science	• Diskrete Mathematik • Technische und Praktische Informatik • Formale Sprachen	Eine Klausur (2 Stunden). In Sonderfällen kann aus didaktischen Gründen stattdessen auch eine mündliche Prüfung (30 - 45 Minuten) angeboten werden. Die regelmäßige Bearbeitung der Übungsaufgaben sowie die Bearbeitung des Mini-Projektes sind Voraussetzungen für die Zulassung zur Klausur.

Qualifikationsziele

Im Rahmen des Moduls werden verteilte Systeme behandelt, die versuchen, im Beisein von Gegenspielern bzw. Angreifern miteinander zu kommunizieren. Es geht um grundlegende Sicherheitsmerkmale wie Vertraulichkeit, Authentizität und Nicht-Abstreitbarkeit, aber auch um theoretische und praktische Angriffe auf kommunizierende Systeme.

- Die Studierenden kennen die folgenden Begriffe und Zusammenhänge und können sie anderen erklären:
  - klassische Kryptosysteme (Caesar, Substitution) und perfekte Kryptosysteme (Vernam)
  - Entropie und Passwort-Sicherheit
  - Theorie der Abstrakten Strom- und Blockchiffren
  - Praktische Blockchiffren (DES, AES)
  - Theorie der Public-Key Kryptographie
  - Praktische Public-Key Kryptosysteme (RSA, Rabin, Diffie-Hellman, ElGamal)
  - Digitale Unterschriften
  - Angriffe auf fehlerhaft implementierte Public-Key Kryptosystem, Digitale Unterschriften und auf Textbook-RSA, sowie Gegenmaßnahmen
  - Definition der Vertraulichkeit (Real-or-Random) und der Authentizität
  - Privacy Modes of Operation für Blockchiffren
  - Blockchiffren-basierte Message Authentication Codes und Authenticated Encryption Modes
  - Beweisbar sichere Public-Key Kryptographie (RSA-OAEP und Full-Domain-Hash Unterschriften)
- Die Studierenden sind in der Lage, ihr Wissen zur Beantwortung von Sicherheitsfragen anzuwenden.
- Sie können ihr Wissen auch auf anspruchsvolle Probleme bzw. im Zusammenhang mit komplexen Beispielen übertragen. Sie können vorgegebene unsichere Systeme angreifen. Ebenso können Sie zu gegebenen sicheren Systemen begründen, warum diese, nach dem Stand der Technik, als sicher anzusehen sind.
- Die Studierenden können Probleme der Kryptographie bzw. IT-Sicherheit mit Hilfe von Konzepten
  - der Kryptanalyse,
  - der symmetrischen Kryptographie und
  - der asymmetrischen Kryptographie

  formalisieren, formale Fragestellungen analysieren und systematisch nach möglichen Lösungen suchen.
- Die Studierenden können in Anwendungsfällen entscheiden und bewerten, welche Konzepte und Methoden der Kryptographie geeignet sind, ein Sicherheitsproblem zu lösen, und welche nicht. Insbesondere können sie eine

derartige Entscheidung auch nachvollziehbar begründen.

Lehrinhalte

- Grundlagen
- Passwörter und Entropie
- Stromchiffren
- Blockchiffren
- Public-Key Kryptographie
- Unsichere Systeme
- Vertraulichkeit
- Authentizität
- Authentisierte Verschlüsselung
- Public-Key Sicherheit

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert (Präsenzstudium). Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (nun wieder Präsenzstudium). Zur Unterstützung des Selbststudiums kann ein Tutorium angeboten werden. Zur Vertiefung des Praxisbezugs sollen die Studierenden im Rahmen eines Mini-Projektes eine praxisbezogene Aufgabe bearbeiten und ihre Ergebnisse in Form einer Hausarbeit schriftlich darstellen oder im Rahmen eines Kurzvortrages präsentieren. Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unter anderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen.

Hinweise

- J. Buchmann: Einführung in die Kryptographie, Springer Verlag.
- A. Beutelspacher: Kryptologie, Vieweg Verlag.
- Beutelspacher, Schwenk, Wolfenstetter: Moderne Verfahren der Kryptographie, Vieweg Verlag.
- D. R. Stinson: Cryptography Theory and Practice, CRC Press.
- C. Eckert: IT-Sicherheit. Konzepte - Verfahren - Protokolle, Oldenbourg.
- A. J. Menezes, P. C. van Oorschot, S. A. Vanstone: Handbook of Applied Cryptography, CRC Press.

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Kryptographie	6.0 ECTS-Punkte (SWS: V2+Ü1)

Titel **Web-Technologie** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
4	jährlich	Wöchentlich mit jeweils einer Lehrveranstaltung im Sommer-semester	6	Präsenz (Vorlesung + Übung): **60** Selbststudium: **90** Prüfungsvorbereitung einschließlich der Prüfungen: **30** **Summe 180**	Deutsch	Benno Stein

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Schwerpunkt Medieninformatik, Bachelor of Science	Erfolgreiche Absolvierung der Vorlesungen • Technische und Praktische Informatik • Diskrete Mathematik	Klausur. Dauer: 1.5h In Sonderfällen kann aus didaktischen Gründen stattdessen auch eine mündliche Prüfung (30 - 45 Minuten) angeboten werden. Die regelmäßige Bearbeitung der Übungsaufgaben ist eine Voraussetzung für die Zulassung zu den Klausuren.

Qualifikationsziele

Im Rahmen des Moduls werden Grundlagen moderner Informationssysteme mit den Schwerpunkt Web-Technologie behandelt. Die Vorlesungen dieses Moduls sind grundlagen- und methodenorientiert ausgerichtet. Die Qualifikationsziele im Einzelnen sind:

- Grundbegriffe des Webs und der Internettechnologie einordnen können
- Verständnis für den technischen Aufbau des Internets
- Erfahrung in der Zuordnung von Server- und Client-Technologien
- Beherrschung wichtiger Auszeichnungssprachen, um Web-Inhalte zu repräsentieren
- Praktische Erfahrung im Umgang ausgewählter Scriptsprachen zur Verarbeitung von Web-Inhalten
- Beherrschung der grundlegenden Verarbeitungsabläufe zur Verwaltung und Präsentation von Web-Inhalten
- Förderung der Fähigkeit, um zwischen Web-Grundlagen und Web-Werkzeugen zu unterscheiden
- Förderung der Fähigkeit, die Tragfähigkeit neuer Entwicklungen einzuschätzen
- Entwicklung von Prinzipien, um die Informationsmenge die durch die Web-Technologievielfalt entsteht, effektiv zu filtern
- Beherrschung von Vorgehensweisen, um sich selbst weiterbilden können

Lehrinhalte

Web-basierte Systeme werden hinsichtlich ihrer Architektur und Funktionsweise eingeführt, wobei ein Schwerpunkt auf der Vorstellung Web-basierter Sprachen liegt. Grundlagen aus benachbarten Gebieten wie der Rechnerkommunikation werden behandelt, soweit dies für das Verständnis erforderlich ist. Hierzu gehören folgende Vorlesungseinheiten:

- Überblick über die historische Entwicklung des Internets
- Begriffe, Problemstellungen und Herausforderungen im Web-Kontext
- Internetprotokollaufbau und das HTTP-Protokoll
- Dokumentsprachen: HTML, CSS
- Dokumentsprachen: XML-Grundlagen

- Dokumentsprachen: XML-Schema
- Dokumentsprachen: XSL
- XML-Programmierung und APIs
- Ausgewählte Server-Technologien
- Ausgewählte Client-Technologien
- Einführung in Web-Middleware

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert (Präsenzstudium). Es werden statische und dynamische Beamer-Präsentationen mit dem Einsatz der Tafel kombiniert. Der Beamer-Einsatz ermöglicht u.a. die schnelle und übersichtliche Präsentation und Kombination von Inhalten in verschiedenen Kontexten. Beispiele: Darstellung von Protokollabläufen, Interaktive Manipulation von Programmen und Web-Sprachen im Web-Client.

Die Tafel wird eingesetzt, um Beweise und formale Definitionen schrittweise zu entwickeln und die Herausforderungen und Schwierigkeiten bei der Entwicklung zu diskutieren. Beispiel: Schrittweise Herleitung der Struktur eines SAX-Parsers auf Basis eines endlichen Automaten.

Weiterhin wird die Tafel dafür eingesetzt, um auf Rückfragen von Studierenden spontan zu reagieren und entsprechende Grundlagen oder Beispiele ad-hoc herzuleiten.

Alle Lehrinhalte werden im Rahmen von Übungen vertieft. Hierzu zählt sowohl die Anwendung der Theorie im Rahmen praktischer Modellierungs- und Programmieraufgaben als auch das Wiederholen und Nachvollziehen der theoretischen Grundlagen anhand von kleineren Beweisen.

Insbesondere werden in den Übungen zur Web-Technologie die verschiedenen Techniken und Konzepte zur Modellierung und Programmierung von Web-Inhalten sowohl isoliert (pro Übung) als auch im Zusammenspiel (über mehrere Übungen hinweg) behandelt.

Pro Übungsserie gibt es Pflichtaufgaben und freiwillige Aufgaben. Die Bearbeitung der Pflichtaufgaben geschieht in Gruppen von 2-3 Studierenden. Diese Gruppen bleiben typischerweise in ihrer Zusammensetzung im Verlauf einer Vorlesung unverändert. Die Pflichtaufgaben werden durch die Betreuenden korrigiert und die Lösungen in einer gemeinsamen Sitzung besprochen. Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen.

Zur Unterstützung des Selbststudiums kann ein Tutorium z bestimmten Inhalten angeboten werden.

Hinweise

Empfohlene Literatur aus dem Bereich Web-Technologie:

- Comer. Computer Networks and Internets with Internet Applications. 5. Auflage, Pearson Prentice Hall, 2008.
- Meinel/Sack. WWW - Kommunikation, Internetworking, Web-Technologien. Springer, 2013.
- Meinel/Sack. Internetworking: Technische Grundlagen und Anwendungen.. Springer, 2012.
- Harold/Means. XML in a Nutshell. 3. Auflage, OReilly, 2004.

Umfangreiche und aktuelle Literaturangaben sind im Script verlinkt.

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Web-Technologie (wöchentlich im Sommersemester)	6 ECTS-Punkte ($\mathrm{SWS}: \mathrm{V2}+\ddot{\mathrm{U}}1$)

Titel **Information und Codierung** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
4	jährlich	Wöchentlich im Sommersemester	6	Präsenz (Vorlesung + Übung): **45** Selbststudium: **110** Prüfungsvorbereitung einschließlich der Prüfungen: **25** **Summe 180**	Deutsch	Dr, A. Jakoby

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Schwerpunkt Security und Data-Science Bachelor of Science	Erfolgreiche Absolvierung der Vorlesungen • Algebra • Diskrete Mathematik	Die Prüfungsleistung der Lehrveranstaltung , Information und Codierung ' ist eine Klausur. Dauer: 100 Minuten. Die regelmäßige Bearbeitung der Übungsaufgaben ist eine Voraussetzung für die Zulassung zur Prüfung. In Sonderfällen kann aus didaktischen Gründen stattdessen auch eine mündliche Prüfung (30 - 45 Minuten) angeboten werden.

Qualifikationsziele

Das Modul vermittelt erforderliche Kenntnisse an der Schnittstelle von Informationstechnik und Informatik. Nach einem erfolgreichen Besuch der Lehrveranstaltungen sind die Studierenden in der Lage, informationstechnische Denkvorstellungen auf konkrete Probleme der Informatik zu projizieren. Dies umfasst Informations- und codierungstechnische Fragestellungen zur Beschreibung medialer. Damit können die Studierenden konkrete Probleme mit den Begrifflichkeiten der Informationstheorie abstrakt darstellen und Lösungsansätze entwickeln, als auch Anforderungen an die Kommunikationstechnik schlüssig formulieren.

Spezielle Qualifikationsziele aus Information und Codierung:

- Die Studierenden kennen insbesondere die folgenden Begriffe und Zusammenhänge und können sie anderen erklären:
  - Zeichentheorie als Grundlage von Verständigungsvorgängen,
  - Semiotisches Dreieck, Zeichentypen, Zeichen in Kalkülen, Alphabet,
  - Signale als Zeichen, Signalkennwerte, Pegel,
  - Beschreibung und Eigenschaften kontinuierlicher und diskreter Signale,
  - Quantisierungs- und Samplingtheorem, Abweichungen und Artefakte,
  - Verfahren zu Komprimierung (LZW, JPEG)
  - Mathematischer Informationsbegriff,
  - Zusammenhang von Entropie und Redundanz, Kompression
  - Codierungstheorem, Decodierungsbedingung, Entropiecodierung, Bitverarbeitung
  - Kanalcodierung, lineare Blockcodes, Hammingdistanz, Fehlererkennung und -korrektur,
  - Hammingcode, Hammingschranke,
  - Irrelevanzcodierung, Typologie medialer Dateiformate.
  - Spezielle Codes wie
    - Huffman-Codes und dynamischer Huffman-Code
    - Einfache Lineare Codes (insbesondere den Reed-Muller-Code)

        - Zyklische Codes (insbesondere den Reed-Solomon-Code)
    - Algebraische Beschreibungen von linearen und zyklischen Codes
    - Verfahren zur Beschreibung von Varianten bekannter Codes
    - Fehlerfreie, fehlerhafte und Erasure-Kanäle

- Die Studierenden sind in der Lage, ihr Wissen unter anderem in den folgenden Bereichen bzw. zur Beantwortung der folgenden Fragestellungen anzuwenden:
    - entscheiden, welche Quantisierungs- und Abtastbedingungen zur Informationsgewinnung erforderlich sind,
    - Kennwerte für Analog-Digital- bzw. Digital-Analog-Umsetzer selbstständig auswählen,
    - Zusammenhang zwischen Datenvolumen, Übertragungszeit und Datenrate sicher beherrschen,
    - einfache Algorithmen zur Entropiecodierung anwenden,
    - Codiereffizienz und Decodierfähigkeit bewerten,
    - einfache Sicherungs- und Korrekturalgorithmen anwenden,
    - Headerstrukturen von Mediendateien analysieren und manipulieren,
    - entscheiden welche Codes für welche Form von Kanälen eingesetzt werden können
    - Analyse der Effizienz von Codes

- Die Studierenden können ihr Wissen auf anspruchsvolle Probleme bzw. im Zusammenhang mit komplexen Beispielen anwenden:
    - Datenkompressionsverfahren mathematisch als Entropieproblem bzw. sinnesphysiologisch als Irrelevanzproblem bearbeiten,
    - Fehlererkennungs- und -korrekturprobleme analysieren, bearbeiten und mit Hilfe linearer Codes lösen
    - selbstständig abstrakte Begriffe in mathematisch-formaler Darstellung verstehen sich selbständig in die Anwendung grundlegender Methoden der Informations- und Codierungstechnik einarbeiten.

- Die Studierenden können Probleme der Medieninformatik formalisieren und systematisch nach Lösungen suchen mit Hilfe von Konzepten
    - der Signaltheorie,
    - der Informationstheorie,
    - der Codierungstheorie.

- Die Studierenden können in Anwendungsfällen entscheiden und bewerten, welche Konzepte und Methoden
    - der Signaltheorie,
    - der Informationstheorie,
    - der Codierungstheorie.

  zur Modellbildung geeignet sind. Insbesondere können sie diese Entscheidung auch nachvollziehbar begründen.

Lehrinhalte

- Information und Codierung
    - Zeichen, Signale, Information, Codierung
    - Komprimierung mit und ohne Verluste (LZW, JPEG)
    - Codes für Bilder: GIF, JPEG
    - Quantisierungs- und Abtasttheorem,
    - Mathematischer Informationsbegriff,
    - Entropie und Redundanz,
    - Diskrete Quellen und Kanäle,
    - Entropie- und Irrelevanzcodierung,
    - Kanalcodierung, Modulation.
    - Huffman Codes, Algebraische Codes (Reed-Muller und Reed Solomon Codes)
    - Fehlerhafte Signale und Rrasure-Kanäle
    - RAID-Systeme

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in einer Vorlesung präsentiert. Sie werden im Rahmen einer Übung vertieft. Dabei sind zunächst im Selbststudium vorgegebene Übungsaufgaben zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden die Lösungen in einer gemeinsamen Sitzung besprochen (in Vorlesung oder Übung). Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen. Zur Unterstützung des Selbststudiums kann ein Tutorium angeboten werden.

Über regelmäßig zu bearbeitende Übungsaufgaben wird eine kontinuierliche Aufarbeitung der Inhalte bewirkt.

Im Sinne des Spiralprinzips werden Inhalte, die in höheren Semestern im Detail behandelt werden, hier einführend behandelt.

Wissenschaftliche Mitarbeiter und studentische Tutoren aus höheren Semestern betreuen die Studierenden in den Übungen und stehen für Rückfragen und Diskussion zur Verfügung

Hinweise

Jakoby: Folien zur Vorlesung Information und Codierung

Klimant, Herbert u.a.: Informations- und Kodierungstheorie.
Shannon, Claude; Weaver, Warren: The Mathematical Theory of Communication.
Werner, Martin: Nachrichtentechnik.
Christoph Meinel, Harald Sack; Digitale Kommunikation: Vernetzen, Multimedia, Sicherheit (X.media.press); Springer
Dave K. Kythe, Prem K. Kythe; Algebraic and Stochastic Coding Theory; CRC Press

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Information und Codierung (wöchentlich im Sommersemester)	6 ECTS-Punkte (SWS: V3+Ü1)

Titel **Computer Vision** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
5	jährlich	Wöchentlich im Wintersemester	6	Präsenz (Vorlesung + Übung + Projekt): 4**5** Projektarbeit: **30** Selbststudium: 7**5** Prüfungsvorbereitung: **30** **Summe 180**	Englisch	Prof. Dr.-Ing. Volker Rodehorst

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Wahlpflichtmodul für Informatik mit Schwerpunkt Medieninformatik, Bachelor of Science	• Einführung in die Programmierung, • Software-Engineering I, • Algorithmen und Datenstrukturen	Klausur (120 Minuten). Die erfolgreiche Bearbeitung der Übungsaufgaben ist eine Voraussetzung für die Zulassung zur Klausur.

Qualifikationsziele

Die Veranstaltung gibt eine Einführung in die Grundlagen der Sensor-Orientierung und 3D-Rekonstruktion. Das Ziel ist ein Verständnis der Prinzipien, Methoden und Anwendungen der bildbasierten Vermessung. Behandelt werden unter anderem die algebraische projektive Geometrie, Abbildungsgeometrie, Kalibrierung, Orientierungsverfahren, Stereo-Bildzuordnung und weitere Verfahren zur Oberflächenrekonstruktion.

Spezielle Qualifikationsziele:

Die Studierenden kennen die wesentlichen Grundlagen der algebraischen projektiven Geometrie und können dieses Wissen für die räumliche 3D-Rekonstruktion von Objekten aus Bildern einsetzen. Sie sind in der Lage die bei der Bildaufnahme herrschende Abbildungsgeometrie zu modellieren und für eine Oberflächenrekonstruktion zu invertierten. Die Studierenden können perspektivische Abbildungen mit Hilfe der direkten linearen Transformation bestimmen und beherrschen auch die optischen Abbildungseigenschaften von Linsen. Nach der Veranstaltung sind sie in der Lage grundlegende Rekonstruktionsverfahren zu beurteilen und auszuwählen. Die begleitenden Übungen versetzen sie in die Lage eigene Verfahren zu implementieren, um wichtige Kameraeigenschaften zu kalibrieren, die einzelnen Sensorpositionen und –ausrichtungen aus den Bildern zu bestimmen und räumliche Objektpunkte zu triangulieren.

Lehrinhalte

Computer Vision
- Projektive Geometrie
  - Perspektivische Abbildungen mit homogenen Koordinaten
  - Bestimmung linearer Transformationen
  - Robuste Parameterschätzung
- Kameramodellierung
  - Kalibrierung, Sensororientierung und Triangulation
  - Optische Abbildung mit Linsen
  - Bündelausgleich für Mehrbild-Geometrien
- Stereobildverarbeitung
  - Erstellung von Stereo-Normalbildern
  - Globale Strategien für die dichte Bildzuordnung

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in Vorlesungen präsentiert und im Rahmen von Übungen vertieft (Präsenzstudium). Übungsaufgaben sind im Selbststudium zu bearbeiten. Nach der Korrektur der Übungsaufgaben durch die Betreuenden werden ausgewählte Lösungen in der folgenden Übung besprochen. Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen dient hierbei unteranderem der Selbstkontrolle der Studierenden und ist zum für die kontinuierliche Vertiefung der Lehrinhalte sehr zu empfehlen.

Hinweise

Literatur:

- V. Rodehorst: Vorlesungsfolien, online.
- W. Förstner and B.P. Wrobel: Photogrammetric Computer Vision – Statistics, Geometry, Orientation and Reconstruction, Springer, 2016.
- R. Hartley und A. Zisserman: Multiple View Geometry in Computer Vision, 2. Aufl., Cambridge University Press, 655 S., 2003.
- O. Faugeras und Q.-T. Luong: The Geometry from Multiple Images, MIT Press, 646 S., 2004.

• R. Szeliski: Computer Vision - Algorithms and Applications, Springer, 812 S., 2010.	
Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
• Photogrammetric Computer Vision, Vorlesung • Photogrammetric Computer Vision, Übung • Photogrammetric Computer Vision, Projekt	• 2 SWS / 3 ECTS-Punkte • 1 SWS / 1,5 ECTS-Punkte • 1 SWS / 1,5 ECTS-Punkte

Titel **Grundlagen der Kognition** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
5	jährlich	Wöchentlich im Wintersemester	6	Präsenz (Vorlesung + Übung): **34** Selbststudium (einschließlich Tutorien, falls angeboten): **34** Übungsarbeiten: **112** Summe **180**	Deutsch	Ehlers

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Schwerpunkt Medieninformatik, Bachelor of Science	Grundwissen Abitur	Die Veranstaltung erfordert die Anfertigung von (empirischen) Übungsberichten im Rahmen praktischer Kleingruppenarbeit. Zum Ende des Semesters wird das theoretische sowie das praktisch-methodische Wissen in einer Modulklausur geprüft (120 Minuten).

Qualifikationsziele

Im Rahmen des Moduls werden zentrale Grundlagen der Kognitionspsychologie gelehrt. Den Studierenden wird ein Einstieg in Begrifflichkeiten, Theorien, Modelle sowie in messmethodische Ansätze vermittelt. Nach Abschluss der Veranstaltung sind sie in der Lage, die denkpsychologische Dynamik der System1 – System 2 Interaktion fachwissenschaftlich und metatheoretisch zu analysieren und auf lebensweltliche Probleme anzuwenden. Sie sind vertraut mit den ordnungsbildenden Prozessen der Wahrnehmungsorganisation und dem Prinzip der perzeptuellen Ambiguität. Sie verstehen, welche Eigenschaften menschlicher Kognition Einfluss auf Aspekte der Gebrauchstauglichkeit und des Nutzererlebens haben, insbesondere im Hinblick auf Aufmerksamkeitsleistungen und die Grenzen mentaler Belastung. Sie sind vertraut mit den psychologischen Motiven nach Selbstwirksamkeit und Kontrollausübung und kennen die gedächtnispsychologischen Gesetzmäßigkeiten der Wissensaneignung und des Vergessens. Sie sind sensibilisiert für die Rolle des Nutzungskontextes und in der Lage, ihre Kenntnisse auf die Konzeption und Evaluation multimodaler Interaktionssystemen anzuwenden.

Zudem haben sich die Studierenden ein umfassendes Verständnis intra- und interindividueller Abweichungen innerhalb der kognitiven und psychomotorischen Leistungen erarbeitet. Sie sind in der Lage, einfache experimentelle Designs zu konzipieren und ihre Hypothesen empirisch zu testen.

Lehrinhalte

- Grundlagen der Denkpsychologie: Zwei Systeme des Denkens (Eindrücke vs Überzeugungen, Selbstkontrolle und Erschöpfung), Rationalität und Intuition, (Vor)Urteile und Schlussfolgern, Heuristiken, kognitive Verzerrungen („Debiasing“ und Dissonanzreduktion)
    - Konsequenzen für Interfacedesign
- Wahrnehmungspsychologie: Grundlagen der Sinnesphysiologie (visuelles System), Elektromagnetisches Spektrum und sichtbares Licht, Absorption und Reflektanz, Wahrnehmen und Erkennen (top-down und bottom-up Verarbeitung), Multistabilität (Prägnanztendenz, Gestaltgesetze und Ambiguitätsreduktion), Farb- und Objektwahrnehmung (Helligkeit, Kontraste, Figur-Grund Trennung), Bewegungssehen, Fechnersches Gesetz und Schwellenmessung, Subliminale Wahrnehmung und Priming
- Grundlagen der (visuellen) Aufmerksamkeit: Gesichtsfeld, offene und verdeckte Aufmerksamkeit (Posner-Paradigma), Blickbewegungen und visuelle Suche, selektive Aufmerksamkeit (Filter- und Dämpfungstheorie, späte vs frühe Selektion, serieller Flaschenhals, dichotisches Hören), zentrale und geteilte Aufmerksamkeit, Vigilanz und Daueraufmerksamkeit (Go/noGo-Tests, Schlafdeprivation, Aufmerksamkeitstrainings), Übung und Automatisierung (Stroop-Effekt)
    - Konsequenzen für das Design multimodaler Interaktionssysteme
- Mentale Belastung: Cognitive load theory, Arbeitsgedächtnis nach Baddeley, Messung der kognitiven Belastung (kognitive Pupillometrie, rating scales, secondary task techniques (Yerkes-Dodson-law)

- Lernen und Gedächtnis: Modelle, Lern- und Vergessensfunktionen, prozedurales und deklaratives Wissen, Theorie der dualen Kodierung, Prototypen, Schemata
- Psychomotorik: Sensomotorik, Handlungssteuerung, motorisches Lernen, zielgerichtete Bewegungen und Fitt’s law
- Selbstwirksamkeit und Kontrollüberzeugungen in interaktiven Systemen: Placebo Funktionen, „locus of control“ Theorie, Kontrollverlust und erlernte Hilflosigkeit, Entscheidungsverhalten („choice paralysis“)
- Grundlagen des Experimentaldesigns (Vorbereitung auf Usability Engineering & Testing)

Lehr- und Lernmethoden / Didaktisches Konzept

Die Veranstaltung besteht aus einer wöchentlichen Vorlesung (90 min) sowie einer Übung (45 min). Die Übungen finden typischerweise in einem zweiwöchentlichen Turnus zu jeweils 90 Minuten statt.

In sechs Übungen werden die Vorlesungsinhalte anhand von Beispielszenarien veranschaulicht und in praktischer Arbeit vertieft. In Kleingruppenarbeit ist dabei eine lösungsorientierte Aufarbeitung ausgewählter Problemstellungen gefordert. Die Studierenden sollen das vermittelte Methodenwissen exemplarisch anwenden und Vorlesungsinhalte vor dem Hintergrund der eigenen Ergebnisse reflektieren. Ausgewählte Resultate werden in der Vorlesung aufgegriffen und unter Berücksichtigung der jeweiligen Settings sowie interindividueller Unterschiede kontrastiert und diskutiert.

Jede Übung wird mit einem (empirischen) Arbeitsbericht abgeschlossen. Die Bearbeitung der Übungsaufgaben sowie das aktive Engagement in der Kleingruppenarbeit sollen ein durchdringungstiefes Verständnis der Vorlesungsinhalte befördern und die Selbstkontrolle der Studierenden im Hinblick auf das Erreichen der Lernziele des Moduls gestatten. Mitarbeiter- und studentische TutorInnen aus höheren Semestern sichern eine engmaschige Betreuung sämtlicher Übungsarbeiten, indem sie für Rückfragen zur Verfügung stehen und Feedback beim Erreichen von Teilzielen anbieten. Nach der Korrektur der eingereichten Berichte werden den Studierenden die Arbeitsleistungen zeitnah zurückgemeldet und innerhalb der Kleingruppe anhand beispielhafter Lösungen sowie typische Fehler gemeinsam rückbesprochen.

Das theoretische sowie das praktisch-methodische Wissen wird auf Bachelorniveau vermittelt und in einer Modulklausur zum Ende des Semesters geprüft.

Hinweise

Literatur

Hagendorf, H., Krummenacher, J., Müller, H. J., & Schubert, T. (2011). Wahrnehmung und Aufmerksamkeit: Allgemeine Psychologie für Bachelor. Berlin: Springer.

Funke, J., & Frensch, P. A. (Eds.). (2006). Handbuch der Allgemeinen Psychologie-Kognition. Hogrefe Verlag.

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Grundlagen der Kognition (wöchentlich im Wintersemester)	6 ECTS-Punkte (SWS: V2+Ü1)

Titel **Visualisierung** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
6	jährlich	Wöchentlich im Sommersemester	6	Präsenz (Vorlesung + Übung): **45** Selbststudium: **90** Prüfungsvorbereitung: **45** **Summe 180**	Deutsch und Englisch	Bernd Fröhlich

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Schwerpunkt Medieninformatik Bachelor of Science	• HCI	Klausur 90-120 Minuten. Aus didaktischen Gründen kann auch eine mündliche Prüfung (30 - 45 Minuten) angeboten werden. Die regelmäßige Bearbeitung der Übungsaufgaben und die aktive Teilnahme an den Übungen sind eine Voraussetzung für die Zulassung zur Prüfung.

Qualifikationsziele

Im Rahmen des Moduls lernen Studierende grundlegende mathematische, algorithmische und technische Verfahren und Vorgehensweisen zur Darstellung von und Interaktion mit gemessenen, simulierten oder gesammelten Daten kennen. Sie können diese für neue Problemstellungen auswählen, anpassen, implementieren und bewerten.

Qualifikationsziele der Visualisierung

- Die Studierenden sind in der Lage auf verschiedenen Ebenen zu abstrahieren:
  - Problemabstraktion: Übersetzung des Problems aus einer spezifischen Anwendungsdomäne in das Visualisierungsvokabular
  - Datenabstraktion: Was soll visualisiert werden?
  - Aufgabenabstraktion: Warum wollen Nutzer die Daten visualisiert haben?
- Sie kennen eine große Auswahl an grundlegenden Visualisierungstechniken und deren Aufbau
  - Visuelle Kodierung: Wie werden die Daten dargestellt
  - Interaktionstechniken: Wie kann die Sicht auf die Daten angepasst werden und wie können verschiedene Sichten kombiniert werden
- Die Studierenden sind in der Lage, Daten aus eine Anwendungsdomäne zu analysieren und abstrahieren sowie systematisch geeignete Visualisierungstechniken auszuwählen und mit passenden Interaktionstechniken zu kombinieren.
- Sie können ihr Wissen auch auf anspruchsvolle und komplexere Probleme übertragen und dabei die Skalierbarkeit der ausgewählten Techniken beurteilen.
- Die Übungen versetzen Studierende in die Lage, grundlegende Visualisierungstechniken selbst zu entwickeln, implementieren und testen.

Lehrinhalte

- Vorlesung Visualisierung
  - Informationsvisualisierung
    - Munzners what-why-how Analyse-Framework
    - Grundlegende Interaktionstechniken
    - Visualisierungstechniken für tabellarische Daten, zeitabhängige Daten, Bäume, Graphen, kartographische Daten und Text
  - Wissenschaftliche Visualisierung
    - Datentypen und mathematische Grundlagen
    - Isolinen und Isoflächen
    - Direkte Volumendarstellung mittels Ray Casting
    - Multi-Resolution Verfahren für Volumendaten

- Strömungsvisualisierung

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in Vorlesungen präsentiert und im Rahmen von Übungen vertieft (Präsenzstudium). Übungsaufgaben sind im Selbststudium zu bearbeiten. Nach der Korrektur und Bewertung der Übungsaufgaben durch die Betreuenden werden ausgewählte Lösungen in der folgenden Übung besprochen.

Hinweise

Literatur zu Visualisierung:

- R. Spence: Information Visualization: An Introduction (3rd Edition)
- T. Munzner: Visualization Analysis and Design

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Visualisierung (wöchentlich im Sommersemester)	6.0 ECTS-Punkte (SWS: V2+Ü2)

Titel **Computergrafik** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
6	jährlich	Wöchentlich im Sommersemester	6	Präsenz (Vorlesung + Übung): **45** Selbststudium: **90** Prüfungsvorbereitung: **45** **Summe 180**	Deutsch und Englisch	Charles Wuethrich

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul für Informatik, Schwerpunkt Medieninformatik Bachelor of Science	• Algebra • Diskrete Mathematik • Algorithmen und Datenstrukturen	Klausur (120 Minuten)

Qualifikationsziele

Das Ziel der Computergrafik besteht darin, mit Hilfe von Computern visuelle Darstellungen zu erzeugen. Die Studierenden kennen nach dieser Vorlesung die grundlegenden Verfahren und Komponenten, die den Stand der Technik darstellen. Diese beinhalten Hardwarekomponenten, Farbräume sowie grundlegende Rasterungsverfahren bis hin zu Verfahren zur Elimination verdeckter Flächen. Weiterhin verfügen sie über einen Überblick zu Modellierungsverfahren, Ansichtstransformationen, lokalen und globalen Beleuchtungsverfahren sowie grundlegende Ansätzen der computergestützten Animation.

Durch die Durchführung eines studienbegleitenden Belegs sind die Studierenden in der Lage, den Stoff aus der Vorlesung in konkrete Programme umzusetzen.

Sie kennen und beherrschen die wesentlichen Konstrukte aus OpenGL oder Vulkan und Shading Languages. Studierende sind am Ende der Veranstaltung in der Lage, geeignete grafische Algorithmen für ein Problem auszuwählen, diese zu programmieren und komplexe grafische Anwendungen zu entwickeln.

Lehrinhalte

- Grundlagen: Licht, menschliches visuelles Sysem, Farbe, Farbräume, Ausgabegeräte
- 3D Modellierung: 3D Koordnaen, Polygone, Splines, Patches, 3D Rotationen und Translationen, Hoomogene Koordinaten, hierarchische Modelle, Szenengraphen.
- Lokale Beleuchtungsmodelle: ambiente Beleuchtung, diffuse Beleuchtung, spekulare Beleuchtung.
- Texturen, Bump und Displacement Maps, Environment Maps, Projection Maps, Shadow Maps.
- Grafikpipeline, Vertexshader, Fragmentshader
- Rasterisierung: Geraden, Kreise, Polygone.
- Clippingsalgorithmen: Cohen-Sutherland, Sutherland-Hodgeman
- Hidden Surface Removal: Painter-Algorithmus, Z-buffer, Warnock-Algorithmus, Binäres Space-Tree
- Kajiyas globale Beleuchtungsglechung.
- Raycasting, Rekursives Raytracing, Beschleunigungsmethoden.
- Radiosity, Progressive Radiosity, Zonalmethoden.
- Aliasing und Antialiasing
- Computeranimation: Pfade in 3D, direkte und inverse Kinematik.

Lehr- und Lernmethoden / Didaktisches Konzept

Die Lehrinhalte werden in Vorlesungen präsentiert und im Rahmen von Übungen vertieft (Präsenzstudium). Übungsaufgaben sind im Selbststudium zu bearbeiten. Nach der Korrektur und Bewertung der Übungsaufgaben durch die Betreuenden werden ausgewählte Lösungen in der folgenden Übung besprochen.

Hinweise

Literatur:

- F. Cohen, J. Wallace: Radiosity and Realistic Image Synthesis, Academic Press, 1993, ISBN 978-0121782702
- A. Glassner: Principles of Digital Image Synthesis, Morgan Kaufman, 1995, ISBN 978-1558602762
- A. Watt: 3D Computer Graphics, Addison-Wesley, 1999
- Akenine-Möller, Haines, Hoffman: Real-Time Rendering, A K Peters, 2008
- Foley, Van Dam, Feiner, Hughes: Computer Graphics. Principles and practice, Addison Wesley, 2009, ISBN 978-0201357172
- Shirley, Marschner: Fundamentals of Computer Graphics, A K Peters, 2009

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)
Computergrafik (wöchentlich im Sommersemester)	6.0 ECTS-Punkte (SWS: V2+Ü2)

Titel **Wahlmodul** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
5 und 6	Jedes Semester	Wöchentlich jedes Semester	15	**450**	Deutsch, ggf. andere Sprache	Prüfungsausschuss

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Wahlmodul für Medieninformatik, Bachelor of Science	Grundwissen Abitur Abhängig von der gewählten Veranstaltung können weitere Zulassungsvoraussetzungen gelten.	Abhängig von der gewählten Veranstaltung. Die Endnote des Moduls wird berechnet anhand des gewichteten Mittelwerts der Noten in den gewählten Kursen, gewichtet anhand ECTS.

Qualifikationsziele

Das Wahlmodul ermöglicht den Studierenden wahlweise

1. den Besuch von Lehrveranstaltungen der Medieninformatik, um die Kompetenzen als Medieninformatiker zu vertiefen,
2. den Besuch von Lehrveranstaltungen, die von Professoren anderer Fakultäten bzw. von Professoren anderer Bereiche der Fakultät Medien angeboten werden, um die Kompetenzen zu verbreitern
3. und den Besuch von Englisch-Veranstaltungen, um die Berufsqualifikation zu verbessern bzw. um sich auf den Besuch eines englischsprachigen Master-Studiums vorzubereiten.

Im Zusammenhang mit einem Studienaufenthalt im Ausland können auch Veranstaltungen zum Erwerb der jeweiligen Landessprache im Rahmen des Wahlmoduls belegt werden.

Lehrinhalte

(abhängig von den gewählten Veranstaltungen)

Lehr- und Lernmethoden / Didaktisches Konzept

(abhängig von den gewählten Veranstaltungen)

Hinweise

Grundsätzlich können Studierende frei aus den folgenden Veranstaltungen wählen:

1. Bachelorveranstaltungen des Bereichs Medieninformatik, soweit sie nicht an anderer Stelle angerechnet werden.

   Handelt es sich bei der Veranstaltung um ein Projekt, haben Studierende, die dieses Projekt im Rahmen ihres Erst- bzw. Zweitprojekts belegen wollen, Vorrang vor Studierenden, die sich das Projekt im Wahlmodul anrechnen lassen wollen. Insbesondere gibt es keinen Anspruch darauf, das im Rahmen des Wahlmoduls ein Projekt belegen zu können.

   Der Prüfungsausschuss kann auf Antrag auch die Anrechnung einzelner Master-Veranstaltungen aus dem Bereich der Medieninformatik im Wahlmodul erlauben.
2. Fachfremde Veranstaltungen, die von Professoren oder Dozenten anderer Fakultäten bzw. anderer Bereiche der Fakultät Medien angeboten werden.

Als fachfremd gilt eine Veranstaltung, deren Inhalte bzw. Lernziele nicht Informatik-typisch sind (im Gegensatz, z.B., zum Programmieren, zum Schreiben von Skripten in einer ausführbaren Sprache oder zu der Beschäftigung mit den Details binärer Kommunikationsprotokolle). Es obliegt dem Prüfungsausschuss, zu entscheiden, ob eine Veranstaltung fachfremd ist, oder nicht. Bei Veranstaltungen, die nicht fachfremd sind, kann der Prüfungsausschuss die Anrechnung der Leistungspunkte begrenzen oder ganz versagen.

3. Sprachkurse in Englisch, die an der Bauhaus-Universität angeboten werden.

Über die Anrechnung anderer Veranstaltungen im Wahlmodul entscheidet der Prüfungsausschuss. Unter anderem sollen fachfremde Veranstaltungen, die formal nicht von Professoren oder Dozenten angeboten werden, sondern von Universitätsmitarbeitern, trotzdem angerechnet werden können, wenn sich ein Professor oder Dozent als zuständig für die fachliche Qualität der Veranstaltung und eine angemessene Bewertung der Studierenden erklärt.

Im Zusammenhang mit einem Studienaufenthalt an einer ausländischen Hochschule können im Wahlmodul auch

- Veranstaltungen (auch fachfremde) der gastgebenden Universität und
- Veranstaltungen zum Erwerb der jeweiligen Landessprache

angerechnet werden.

Die Anrechnung von Sprachkursen im Rahmen des Wahlmoduls (Englisch und, ggf., die Landessprache für einen Studienaufenthalt) ist auf insgesamt maximal 6 Leistungspunkte beschränkt.

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)

Titel	**Informatikprojekt / Schwerpunktbezogenes Projekt**	Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
4 bzw. 5	jedes Semester	regelmäßige Termine über das gesamte Semester hinweg	12	Präsenz: **50** (davon Treffen mit Betreuenden **45**, öffentliche Präsentation des Projektes: **5)** Selbststudium: **260** **Summe 360**	Deutsch, ggf. Englisch	Jeweilige Professur

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Zwei Pflichtmodule für Medieninformatik, Bachelor of Science	Abhängig vom Projektthema Kenntnisse aus einzelnen Pflichtmodulen des 1.-3. Semesters (Informatikprojekt) bzw. des 1.-4. Semesters sowie dem gewählten Schwerpunkt (Schwerpunktbezogenes Projekt)	• Regelmäßige Zwischenpräsentationen (mündlich) • ggf. Zwischenberichte (schriftlich) • hochschul- oder allgemein öffentliche Präsentation des Projektes (mündlich) • Dokumentation der Ergebnisse, wissenschaftlicher Report oder Software-Quelltext (schriftlich)

Qualifikationsziele

Abhängig vom Projektthema lernen die Studierenden, wie man bei anspruchsvollen Fragestellungen nach geeigneten Lösungsansätzen recherchiert, deren Vor- und Nachteile versteht, und die Ergebnisse derartiger Recherchen dokumentiert, oder die Studierenden machen praktische Erfahrungen mit dem Entwurf, der Implementation und der Evaluation von Software- und ggf. Hardwaresystemen und Benutzerinterfaces, oder sie lernen, wie man wissenschaftliche Experimente und insbesondere Benutzerstudien plant, ausführt und auswertet.

Aufbauend auf den bereits im Rahmen der für das Projektthema relevanten Pflichtmodule erworbenen fachlichen Kompetenzen können die Studierenden zeigen, dass sie in der Lage sind, Sachverhalte zu verstehen, Methoden der Informatik bzw. des gewählten Schwerpunkt-Bereichs anzuwenden und auf verwandte Probleme zu übertragen. In der Projektgruppe sollen sie Probleme analysieren und Lösungen bewerten. Außerdem wird der Projektgruppe die Gelegenheit geboten, gemeinsam mit den Betreuern/innen an der Schöpfung neuer wissenschaftlicher Erkenntnisse bzw. an der Erstellung neuer praktischer Lösungen mitzuwirken.

Neben der Vermittlung fachlicher Kompetenzen, entsprechend dem jeweiligen Projektthema, dienen die Projekte vor allem der Vermittlung von

- Sozialen Kompetenzen und
- Selbstkompetenz

Unabhängig vom Projektthema lernen die Studierenden, wie man als Team zusammenarbeitet, auch autonom, ohne Beisein von Betreuern, wie man Gruppensitzungen protokolliert, wie man Ergebnisse mündlich vorträgt und diese später schriftlich fixiert. Die Studierenden verstehen die Bedeutung von Projektmanagement und Organisation für komplexe Projekte.

Eine Konsequenz des "learning by doing" Ansatzes des projektbezogenen Studiums ist, dass Studierende ihre Arbeits- und Kommunikationsweise, ihre Selbstdisziplin und ihre Erwartungen an andere Projektteilnehmer im Laufe des Projektes hinterfragen und ihre eigenen Fehler nachträglich erkennen können. Das schwerpunktbezogene Projekt soll dazu dienen, Lehren aus dem Informatikprojekt zu ziehen und die dort bereits erworbenen Kompetenzen sowie erste Erfahrungen aus dem gewählten Schwerpunkt entsprechend zu vertiefen.

Lehrinhalte

Die konkreten Lehrinhalte sind abhängig vom Projektthema, beinhalten jedoch stets mehrere der folgenden Punkte:

- Literaturrecherche zu einem Thema aus der Wissenschaft oder der Praxis der Medieninformatik
- Spezifikation eines Systems der Medieninformatik
- Implementation eines Systems in Soft- oder Hardware.
- Evaluation eines Systems
- Planen Durchführen und Auswerten von Benutzerstudien

- Schreiben eines wissenschaftlichen Berichts

Lehr- und Lernmethoden / Didaktisches Konzept

Typischerweise treffen sich die Projektgruppen einmal die Woche mit den Betreuenden, um Zwischenergebnisse zu präsentieren und Feedback für die weitere Arbeit zu erhalten. In manchen Fällen finden zu Projektbeginn, in Rahmen einer Einarbeitungsphase, mehrere Treffen pro Woche statt, dafür können die Treffen im weiteren Verlauf des

Semesters entsprechend seltener sein. Immer liegt der Schwerpunkt der Arbeit auf dem Selbststudium.

Das projektbezogene Studium ist ein zentraler didaktischer Bestandteil der Lehre an der Bauhaus-Universität.

Unter Anleitung der zuständigen Professuren werden die Studierenden mit komplexen wissenschaftlichen oder praktischen Problemen der Medieninformatik bzw. aus den Bereichen Sicherheit und Data Science konfrontiert, die sie so selbstständig wie möglich lösen sollen. Projektteilnehmer tauschen sich sehr regelmäßig untereinander aus, treffen sich regelmäßig mit den Betreuenden, diskutieren ihre Zwischenergebnisse innerhalb der Projektgruppe und mit den Betreuern, präsentieren ihr Projekt wenigstens einmal hochschulöffentlich oder allgemein öffentlich, und verfassen einen Abschlussbericht.

Hinweise

Um im 4. Semester ein Mobilitätsfenster zu schaffen, kann das Erstprojekt durch beliebige Module/Veranstaltungen aus der Informatik oder Medieninformatik ersetzt werden, wenn diese im Rahmen eines Auslandsstudiums an einer ausländischen Hochschule erworben wurden. Es wird davon ausgegangen, dass die Aufnahme eines Auslandsstudiums dazu beiträgt, unter anderem

- Sozialen Kompetenzen und
- Selbstkompetenz

zu erwerben, die dann, im Schwerpunktbezogenen Projekt, an der Bauhaus-Universität weiter vertieft werden sollen.

Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)

Titel **Bachelor-Modul** Modulnummer

Semester (optional)	Häufigkeit des Angebots	Dauer + Turnus	ECTS-Punkte	Studentischer Arbeitsaufwand (in Stunden)	Sprache	Verantwortlich
6	jedes Semester	4 Monate, jederzeit	15	Verfassen der Arbeit: **360**, davon • Präsenz (Treffen mit Betreuer(in)): **45** • Selbststudium: **315** Verteidigung der Arbeit: **90**, davon • Präsenz (Treffen mit Betreuer(in) und Halten des eigentlichen Vortrages): **5** • Selbststudium: **85** Summe: **450**	Deutsch, auf Antrag Englisch	Betreuerin / Betreuer

Modultyp / Verwendbarkeit	Voraussetzungen für die Teilnahme	Prüfungsleistung(en)
Pflichtmodul Medieninformatik Bachelor of Science	Zum Zeitpunkt der Anmeldung der Arbeit müssen die Studierenden in den anderen Modulen mindestens 120 Leistungspunkte nachgewiesen haben. Die Verteidigung ist öffentlich und die letzte Prüfungsleistung. Deshalb müssen die Studierenden zum Zeitpunkt der Verteidigung alle anderen Module erfolgreich abgeschlossen haben.	• Wissenschaftlicher Report (schriftlich, 80%) • Verteidigung der Arbeit (mündlich, 20%)

Qualifikationsziele

Eine Bachelor-Arbeit ist die Bearbeitung eines wissenschaftlichen Themas mit

- schriftlicher Ausarbeitung
- und anschließender mündlicher Präsentation.

Die Aufgabe einer Bachelorarbeit kann die Entwicklung von Software oder Hardware, die Durchführung einer empirischen Studie, eine formale Beweisführung oder eine Literaturrecherche umfassen.

Die schriftliche Ausarbeitung soll die Struktur einer wissenschaftlichen Veröffentlichung haben und muss den Grundsätzen guter wissenschaftlicher Praxis genügen, insbesondere bei der Verwendung von Referenzen und Zitaten sowie bei der Nutzung möglicherweise erhobener oder genutzter empirischer Daten. Es muss klar erkennbar sein, welche eigenen Beiträge der Student geleistet hat und welche er von anderen Quellen übernommen hat.

Im Rahmen der Bachelor-Arbeit soll der Studierende zeigen, dass er

- dass er grundlegende wissenschaftliche Methoden beherrscht,
- dass er die Grundsätze guter wissenschaftlicher Praxis kennt und ihnen entsprechend handelt
- und dass er innerhalb einer vorgegebenen Frist ein Thema der Informatik bzw. Medieninformatik mit wissenschaftlichen Methoden erarbeiten und für Fachkollegen verständlich darstellen und präsentieren kann.

Lehrinhalte

- Literaturrecherche zu einem Thema aus der Wissenschaft oder der Praxis der Medieninformatik
- Experimente, Tests, Implementationsarbeit oder Benutzerstudien
- Schriftliches Fixieren der Ergebnisse in Form einer wissenschaftlichen Arbeit

Lehr- und Lernmethoden / Didaktisches Konzept

Weitgehend Selbststudium, Feedback durch regelmäßige Betreuung.

Hinweise

Das Bachelor-Modul ist ein zentraler Bestandteil des Studiums, und das Verfassen der Arbeit ist der wichtigste Teil dieses Moduls. Die Gewichtung der Noten reflektiert diese Bedeutung.	
Zum Modul gehörende Lehrveranstaltungen / Kurse (optional)	SWS / ECTS (optional)

Beschreibungen der Veranstaltungen aus dem Masterprogramm *Computer Science for Digital Media* für das Wahlpflichtmodul aus dem Bereich

Theoretische Informatik

Course/Module Title	Advanced Numerical Mathematics
Coordinator	Dr. rer. nat. Dmitrii Legatiuk
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	Courses: Analysis, Linear Algebra, and Numerical Mathematics
Exam requirements	Project report and presentation
Specific target qualifications	The lecture course introduces concepts, algorithms, and theoretical background for the numerical solution of partial differential equations. The accompanying practical classes are concerned with theoretical, as well as applied tasks, for expanding students understanding of the field. The theoretical lectures will be completed by exercises and classes in a computer lab, where the MATLAB will be used as an implementation and simulation environment. The course will deal with the following topics: • numerical linear algebra • the iterative solution of linear and non-linear systems of algebraic equations • discretization and numerical solution of ordinary and partial differential equations • finite difference method • approximation, stability and convergence • error estimates • implementation of concrete problems in MATLAB and their analysis Students should be able to apply the above tools and the theory to solve practical problems. Furthermore, they should be able to implement the algorithms discussed in the course in MATLAB and perform computer simulations. Students should be able to understand • the role of discretization methods in mathematical modelling of engineering problems • how to create a system of linear algebraic equation by using finite difference method • how to apply numerical algorithms for solving systems of linear equations • how to improve the efficiency of a numerical method in order to solve practical problems from mathematical physics and engineering. The students should be able to adapt standard models to the given situation if necessary. Students should be able to understand special problems at research level and be able to work with them in the form of supervised projects.
Contents	• Numerical linear algebra • Discretization of ordinary and partial differential equations • Finite difference method, approximation, stability, and convergence • Numerical simulations
Special information	Literatur: Varga, Matrix iterative analysis. Hermann, Numerische Mathematik Kress, Numerical Analysis Matlab

Course/Module Title	Complexity Theory
Coordinator	Andreas Jakoby
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	Formal Languages
Exam requirements	Final oral exam (max. 45 min.).
Specific target qualifications	Complexity theory deals with methods to analyse the solvability of problems and the classification of the problems according to the required complexity resources. Based on this goal diffened complexity classes and the relations between this classes are analysed. The aim of this module is to understand the basic concepts of complexity classes and the major techniques to analyse problems with the aim to get a classification of intractability of solvability of the problem. Students should understand the major techniques to analyse concrete problems. They should be able to distinguish efficiently solvable and intractability. Students shall master the following concepts: • complexity measures and classes • NP-hardness reductions • analysing the efficiency of algorithms and circuits • finding gap-versions of a problem to show non-approximability • analysing the circuit complexity of problems in NC_1 and AC_0 Students should understand the current state of research in complexity theory, specifically of the design, analysis and application of efficient algorithms and the complexity analysis of concrete problems. With appropriate supervision, students should be able to tackle research problems in complexity theory.

Contents	The course deals with the following topics: • general complexity measures • time and space • circuit depth and size • Turing-machine model • universal Turing-machine • main techniques to work with Turing-machines • Bachmann-Landau notation • complexity classes L, NL, P, NP, PSPACE, NPO, APX, PTAS, FPTAS, NC, AC, P/ploy • tape reduction • Sacitch's Theorem, Immerman–Szelepcsenyi Theorem • hierarchies and universal languages • many-one reduction (polynomial time and logarithmic space) • linear reducibility • properties of reductions • completeness and hardness • clique, independent set, vertex cover, versions of satisfiability, Hamiltonian cycle, CVP, QBF, graph isomorphism, reachability, knapsack, TSP, maximization/minimization versions of several problems • gap versions of several problems • probabilistically checkable proofs, PCP-theorem • non-approximability • (non)-uniform circuit families • sensitivity • general bounds for circuits • relativizing proofs and oracle Turing-machines • relativizing proof for P=NP and for P=/=NP • a satisfiability version between P and NPC • natural proofs
Special information	Jakoby: Lecture slides Papadimitriou: Computational Complexity (Addison Wesley, 1993) Garey, Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness (W.H. Freeman and Company, 1979) Reischuk: Komplexitätstheorie Band I: Grundlagen: Maschinenmodelle, Zeit- Und Platzkomplexität, Nichtdeterminismus (Teubner Verlag, 1999) Sipser: Introduction to the Theory of Computation (Wadsworth Publishing Co Inc, 2012) Wegener: Theoretische Informatik - eine algorithmenorientierte Einführung (B.G. Teubner Verlag, 1999) Goldreich: Computational Complexity: A Conceptual Perspective (Cambridge University Press, 2008) Baier, Asteroth: Theoretische Informatik (Pearson Studium, 2002) Hopcroft, u.a: Introduction to Automata Theory, Languages, and Computation (Addison-Wesley Longman, 2006)

Course/Module Title	Discrete Optimisation
Coordinator	Andreas Jakoby
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	(no specific requirements for this course)
Exam requirements	Final oral exam (max. 45 min.).
Specific target qualifications	Discrete optimisation is about finding optimal solutions for discrete problems. Finding efficient algorithms for discrete optimisation problems is one of the main topics in algorithm design. The goal of this course is to understand the principles of analysing discrete optimisation problems and designing efficient algorithms for such problems. Students should understand the following topics and methods: • discrete optimisation problems and complexity theory • heuristic and local search strategies for optimisation problems • backtracking for discrete optimisation problems • branch-and-bound schema • convex optimisation problems and linear programming • Simplex-Algorithm and Ellipsoid-Algorithm • greedy algorithms • approximability of concrete problems • tree decomposition and dynamic programming Students should be able to apply the above concepts to solve concrete problems. Furthermore, they should appreciate the limits and constraints of the above schemes. Students should be able formalise and generalise their own solutions using the above tools. Students should master concepts and approaches such as • analyzing the intractability of discrete optimisation problems • using different algorithmic concepts to design efficient algorithms for discrete optimisation problems • using transformations to solve optimisation problems • knowing the limits of efficient optimisation algorithms • using approximations to find adequate solutions • knowing that input of a specific form can be solved efficiently by dynamic programming in order to tackle optimisation problems. They should understand the current state of research in discrete optimisation, specifically of the design, analysis and application of schemes for optimisation problems. With appropriate supervision, students should be able to tackle research problems in discrete optimisation.

Contents	• Introduction • Heuristic Search - a general algorithm for search problems • Heuristic Search - best first search • Best First Search with Duplicate Elimination • The A*-algorithm • Backtracking for Discrete Optimisation Problems • Knapsack Problem, Traveling Salesperson Problem, MAXCLIQUE Problem • General Backtracking Strategy • Backtracking with Bounding Function • Branch-and-Bound Schema • Alpha-Beta-Search for 2-Party-Games • Local Search • Hopfield neural networks • Maximum-Cut Approximation via Local Search • Application to Vertex Cover • Local Search for Discrete Optimisation Problems • The Metropolis Algorithm and Simulated Annealing • Linear Programming • Complexity of Linear Programs • Geometry of Linear Programs • Basic Feasible Solution • The Simplex-Algorithm • The Ellipsoid-Algorithm • Affine Transformations and Ellipsoids • Precision of Computation • Greedy Algorithms and Bounds on the Optimum: a load balancing problem • Greedy Algorithms and Knowing the Optimum: the center selection problem • A First Step to the Pricing Method: the Set Cover Problem • Approximations via Reductions: the Vertex Cover Problem • The Pricing Method for the Vertex Cover Problem • Arbitrary Good Approximations: the Knapsack Problem • An Introduction to Linear Programming and Rounding: Vertex Cover revisited • Linear Programming and Rounding: generalized load balancing • Dynamic Programming and Tree-decomposition • Treewidth
Special information	Jakoby: Lecture slides T. Cormen, C. Leiserson, R. Rivest, Introduction to Algorithms, MIT Press, 1990 J. Kleinberg, E. Tardos, Algorithm Design, Addison Wesley, 2005 W. Kocay, D. Kreher, Graphs, Algorithms and Optimisation, CRC 2005 D. Kreher, D. Stinson, Combinatorial Algorithms, CRC Press, 1999 C. H. Papadimitriou, K. Steiglitz, Combinatorial Optimisation: Algorithms and Complexity, Dover Books on Computer Science, 2000

Course/Module Title	Introduction to Functional Programming with Haskell
Coordinator	Dr. rer. nat. Dmitrii Legatiuk
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	No specific requirements for this course
Exam requirements	Submission of a project given during semester with weight of 50% of total grade. The project should be presented at the final oral examination (max. 30 min) with time for preparation (max. 30 min).
Specific target qualifications	Functional programming is a modern programming paradigm based on lambda calculus and recursive functions as model of computation. A program in functional programming is a function in a strong mathematical sense, and the output of a program is application of the function to its arguments. Haskell is a brilliant example of a well-designed programming language illustrating all the advantages of functional programming. The goal of this course is to present basic concepts of the functional paradigm and their realization in Haskell. Students should understand the following topics: • syntax of pure lambda calculus • reduction order and normal forms • evaluation strategies for lambda terms • typing rules and typing relations • syntax of simply-typed lambda calculus • relation between simply-typed lambda calculus and propositional logic • syntax of Haskell • use of lists in Haskell • types and type classes • higher order functions • creating own modules in Haskell Students should be able to apply the above tools and theories to solve concrete problems. Furthermore, they should appreciate the limits and constraints of the above theories, e.g. limitations of pure lambda calculus and the need for types. Students should master concepts and approaches such as • thinking functionally • understanding the mathematical background of functional programming • reasoning about their programs in order to tackle problems from functional programming and their application to digital media. They should be able to understand proposed programming problems, to make well-informed decisions about the preferred proposal and, if necessary, to find their own solutions to given programming problems. Students should develop an understanding of the current state of research in functional programming. With appropriate supervision, students should be able to tackle research problems.
Contents	• Pure lambda calculus • Simply-typed lambda calculus • Curry-Howard isomorphism • Evaluation strategies
Special information	Literature: R. Bird, Thinking Functionally with Haskell Haskell Platform

Course/Module Title	Randomised Algorithms
Coordinator	Andreas Jakoby
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	(no specific requirements for this course)
Exam requirements	Final oral exam (max. 45 min.).
Specific target qualifications	For many problems, randomised algorithms are the only known efficient solution method. For some other problems we can find randomised algorithms that are much simpler and more understandable than any known deterministic method. The goal of this course is to understand the principles of designing and analysing randomised algorithms. The course deals with the following topics: • basic concepts and inequalities from probability (including Chernoff bounds, Markov's, Chebyshev's, and Jensen's inequality) • various randomised algorithms (e.g. for verifying matrix multiplication, sorting, computing the median, cut problems, packet routing, hashing, satisfiability, Hamiltonian cycles, independent sets, K4- subgrapaph problem, etc.) • average-case analysis of algorithms • balls and bins • random graphs • basic probabilistic methods • Lovasz Local Lemma • derandomisation • Markov chains Students should be able to apply the above tools, algorithm, and concepts to solve concrete problems. Furthermore, they should appreciate the limits and constraints of the above topics. Students should be able formalise and generalise their own solutions using randomization. Students should master concepts and approaches such as • using the principle of deferred decisions to analyse randomised algorithms • decreasing error probability by running randomised algorithms multiple times • distinguishing between Las Vegas and Monte Carlo algorithms • using the moment-generating function and Chertoff bounds for analysing the tail probabilities • using the balls and bins model to solve concrete problems and analyse randomised algorithms • using randomised graphs to analyse the average complexity of hard problems • using basic probabilistic methods to solve problems • knowing how to apply Lovasz Local Lemma to analyse hard problems • knowing techniques for derandomisation based on probabilistic methods Students should understand the current state of research in randomised algorithms, specifically of the design, analysis and application of randomised algorithms. With appropriate supervision, students should be able to tackle research problems in randomised algorithms.

Contents	• Some Notes on Probability • Verifying Matrix Multiplication • A Randomised Min-cut Algorithm: Karger's min-cut algorithm • A Randomised Version of Quicksort • Coupon Collector's Problems • A Randomised Algorithm for Computing the Median • Types of Randomised Aalgorithms: Las Vegas versus Monte Carlo • Randomised Computational Complexity Ttheory • Moment-Generating Functions and Cherno ffBounds (versions for independent Poisson trials) • Estimating Probability from Samples • Packet Routing in Sparse Networks (including bit-fixing routing mechanism hypercubes) • Balls and Bins • Applications for Balls and Bins: bucket sort, hashing with bit strings, Bloom filters, breaking symmetry • Models for Random Graphs • Hamiltonian Cycles in Random Graphs • Basic Probabilistic Methods • The Counting Argument and Edge Coloring • The Expectation Argument • Applications for Probabilistic Methods: maximum satisfiability, finding a large cut • Derandomisation Using Conditional Expectations and Finding a Large Cut • Applications for Sample and Modify: independent sets, graphs with large girth • The Second Moment Method and Applications for Threshold Behaviour in Random Graphs • Conditional Expectation Inequality and Applications for K4-Subgraph Problem • Lovasz Local Lemma • Lovasz Local Lemma and Application to Edge-Disjoint Paths • Lovasz Local Lemma and Application to Satisfiability • Explicit Constructions Using the Local Lemma • Explicit Constructions for Satisfiability • Lovasz Local Lemma: the General Case • Markov Chains • A Randomised Algorithm for 2-Satisfiability
Special information	Jakoby: Lecture slides Michael Mitzenmacher, Eli Upfal, Probability and Computing - Randomised Algorithms and Probabilistic Analysis, CAMBRIDGE UNIVERSITY PRESS, 2005

Beschreibungen der Veranstaltungen aus dem Masterprogramm *Computer Science for Digital Media* für das Wahlpflichtmodul aus dem Bereich

Advanced Security

Course/Module Title	Advanced Cryptography: Cryptographic Hash Functions
Coordinator	Stefan Lucks
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	Basic knowledge of cryptography, either from a “Introduction to Modern Cryptography” or from another introduction course
Exam requirements	Active participation in problem session (minimum 25% of achievable points per problem set). Final oral exam (max. 45 min.).
Specific target qualifications	A cryptographic hash function serves as a “fingerprint” to uniquely identify data. The goal of this course is to understand the principles of designing and analysing cryptographic hash functions, and to apply them to the context of digital media. The course deals with the following topics: • the distinction between cryptographic and combinatorial hash functions • security requirements for cryptographic hash functions such as collision resistance, preimage resistance and second preimage resistance • design principles for iterated hash functions • MD4, its internals and attacks on MD4 • the wider MD4 family of hash functions (including SHA-0, SHA-1, SHA-256, and SHA-512) • generic attack techniques, such as cycle finding, time-memory tradeoffs, and distinguished points • block-cipher-based hash functions (single- and double-block) • number-theory-based hash functions • tree hashing • SHA-3 and SHA-3 candidates The course also considers hash function applications, such as • password hashing, • key stretching, • proofs of work, • proofs of space, and • blockchaining Students should understand the application of hash functions for solving concrete problems and be able to distinguish secure from insecure designs. Students should be able to master the following concepts: • formalising security properties using ideal block ciphers and random oracles • falsifying security claims by specific attacks • analysing the security of hash functions • using hash functions for key-stretching and memory-intense password scrambling Students should understand the current state of research in cryptography, specifically of the design, analysis and application of cryptographic hash functions. With appropriate supervision, students should be able to tackle research problems in cryptography.
Contents	• Introduction • Iterated Hash Functions • Generic Attacks • Block-Cipher-Based Hashing • Dedicated Compression Functions • Tree Hashing • The SHA-3 Competition • Proofs of Work, Proofs of Space • Password Hashing and Blockchaining
Special information	The course is based on recent publications; which will be provided during the semester.

Course/Module Title	Advanced Cryptography: Secure Channels
Coordinator	Stefan Lucks
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	Basic knowledge of cryptography, either from a “Introduction to Modern Cryptography” or from another introduction course
Exam requirements	Active participation in problem session (minimum 25% of achievable points per problem set). Final oral exam (max. 45 min.).
Specific target qualifications	A secure channel between two or more participants provides the privacy and integrity of the transmitted data. The goal of this course is to understand the principles of designing and analysing secure channels. Students should understand the following topics: • encryption • semantic security, find-then-guess security, left-or-right security, real-or-random security • nonce-based encryption • authentication • specific message authentication codes (variants of the CBC-MAC, the PMAC) • MACs based on universal hash functions and polynomial hashing • authenticated encryption • the generic composition of secure encryption and secure authentication • the handling of associated data for authenticated encryption • dedicated block-cipher modes for authenticated encryption (OCB, EAX, GCM) • the failure of insecure modes • resistance to nonce-reuse • resistance to the release of unverified plaintexts • on-line authenticated encryption • side-channel attacks and leakage resilience Students should master the design of secure channels from secure components, such as block ciphers, stream ciphers, MACs or universal hash functions. Students should understand the limits and constraints of the approaches and formalisms presented in the course. They should know how to distinguish secure from insecure designs for secure channels. Students should recognise the following concepts: • formalising security requirements for secure channels • analysing existing protocol and channel designs • the provable security approach in symmetric cryptography • the implementation of secure channels Students should develop an understanding of the current state of research in cryptography, specifically in cryptography as applied to enhance confidentiality and authenticity. With appropriate supervision, students should be able to tackle research problems in the area.
Contents	• Encryption • Authentication • Authenticated Encryption • Dedicated Schemes • Robustness • Side-Channels
Special information	Introduction to Modern Cryptography by Mihir Bellare and Phillip Rogaway and recent publications

Course/Module Title	Digital Watermarking & Steganography
Coordinator	Andreas Jakoby
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	(no specific requirements for this course)
Exam requirements	Final oral exam (max. 45 min.).
Specific target qualifications	A digital watermarking and steganography deals with hiding additional information in digital data such as audio data or pictures. The main goal of digital watermarking is to embed information about the content of data within the content, for instance copyrights. Steganography, on the other hand, deals with the aspect of hiding the existence an embedded message. The goal of this course is to understand the principles of designing and analysing schemes for digital watermarking and steganography, and to apply them to the context of digital media. The course deals with the following topics: • the distinction between cryptography, digital watermarking, and steganography • the distinction between application areas for cryptography, digital watermarking, and steganography • design principles for information hiding within multimedia data • properties of areas and digital values within multimedia data for information hiding • tools for measuring the quality of information hiding systems (including Watson's DCT-based visual model) • basic transformations from image processing (including DCT, FFT, wavelet transformation) • JPEG-compression • using the LSB for information hiding • statistical test to detect embedded information within the LSBs (including ?2-test) • information-theoretically secure embedding scheme • practical embedding schemes (including OutGuess and F5) • PRNG based on linear recurrences, linear feedback shift registers, the security of a crypto system, and on the computational difficulty of mathematical problems (including the generator of Blum, Blum, Shup) • linear codes (including Reed-Muler code) • the distinction between informed and blind systems • attacks (including calibration and Markov process based features) • robustness test for digital watermarks (including JPEG compression with different compression rates, stirmark attack, averaging filter, noise, row and column exchange) • rightful ownership problems and schemes to solve them • copy attack and schemes to solve the corresponding problem • audio watermarking Students should understand the application of digital watermarking and steganography for solving concrete problems. They should be able to distinguish secure from insecure designs. Students shall maser the following concepts: • falsifying security claims by specific statistical tests • analyzing the security of stego systems • analysing the robustness and security of digital watermarks • using digital watermarks to solve copy right problems • using PRNG and linear coding theory to reduce embedding distortion Students should understand the current state of research in digital watermarking and steganography, specifically of the design, analysis and application of schemes for digital watermarking and steganography. With appropriate supervision, students should be able to tackle research problems in digital watermarking and steganography.

Contents	• Introduction • Applications and Properties of digital Watermarking • Applications and Properties of Steganography • Basic Notations • Theoretical Observations on Steganography • A Model for Steganography • Information-Theoretically Secure Steganography • Computationally Secure Steganography • Cachin's Definition of Steganographic Security • Basic Transformations from Image Processing • The Embedding Distortion • The Perceptual Model • Watson's DCT-based visual model • Building Blocks of a Steganographic Algorithm • Information-Theoretical Foundations of Steganography • Practical Steganographic Methods • Statistics Preserving Steganography • Statistical Tests • The OutGuess Algorithm – Preserving DCT Statistics • Pseudorandom Number Generators • Model-Based Steganography • Masking Embedding as Natural Processing • The F5 Embedding Algorithm • Minimizing the Embedding Impact • Some Notes on Linear Codes • Feature-Based Steganalysis • Communication-Based Models of Watermarking • Simple Informed Watermark System for 1-Bit Payload • Spread-Spectrum Approach • Strength of the Similarity Measure • Extension to Blind Detection • Experimental Analysis of Robustness • Security of Watermarking • Feature-Based Approach • Audio-Watermarking and Echo Hiding • Rightful Ownership Problem: Single Public Watermarked Image • Invertible and Noninvertible Schemes • Rightful Ownership Problem: Multiple Public Watermarked Image • Copy Attack
Special information	Jakoby: Lecture slides I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, T. Kalker, Digital Watermarking and Steganography (Second Edition), Korgan Kaufmann, 2008. Octave or Matlab will be used in the Lab

Course/Module Title	Quantum Algorithms & Cryptanalysis
Coordinator	Stefan Lucks
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	(none)
Exam requirements	Active participation at the problem session (minimum 25% of achievable points per problem set). A final oral exam (at most 45 min.).

Specific target qualifications	The computational model of a quantum computer is fundamentally different from the classical model of computation. Quantum computers can solve certain problems efficiently, which, to the best of our knowledge, are infeasible on a classical computer. E.g., Shor's celebrated period-finding algorithm, can be used to factorise huge numbers and compute discrete logarithms in huge groups, thus breaking almost all currently used asymmetric cryptosystems. Such exploits assume ECLSQ (Error-Correcting Large-Scale Quantum) computers, which will not be available for many years (if ever). Nevertheless, with the current advent of the first NISQ ("Noisy Intermediate- Scale Quantum") computers, it becomes increasingly important for computer scientists – and especially for cryptographers – to understand how quantum computers work, what quantum computers can do, and what quantum computers can't do.
	
	The students will master the following topics:
	• The fundamental difference between a classical state and a quantum state.
	• Operations over quantum states, modeled as linear operations over the complex numbers.
	• Basic quantum algorithms, such as
	  • amplitude Amplification and Grover's algorithm to "find a needle in a haystack",
	  • the quantum Fourier transform and Shor's factorization method.
	• The application of basic quantum algorithms to quantum cryptanalysis:
	  • symmetric key-recovery in time 2**(n/2) using Grover's algorithm,
	  • hash collisions using Grover's algorithm in time 2**(n/3),
	  • factorization and discrete logarithm using Shor's method.
	• The polynomial method in quantum complexity theory.
	• Post-quantum asymmetric cryptography.
	• Quantum error correction.
	
	The students will understand
	• The quantum model of computation.
	• Its application to design and analyse quantum algorithms.
	• The application of such algorithms to cryptanalysis.
	• Some of the limits of quantum computers.
	
	The students will conceive knowledge about the state of research in quantum algorithms, with a focus on the application to quantum cryptanalysis. Given some guidance, they will be able to tackle current research problems in quantum cryptanalysis.

Contents	• classical bits, quantum bits, classical states, and quantum states • quantum gates and quantum circuits • quantum key exchange • Deutsch's problem and Simon's problem • Grover's amplitude amplification: how to find a needle in a haystack • the application of Grover's algorithm to symmetric cryptanalysis • quantum Fourier analysis and Shor's algorithm for period finding • the application of period finding to asymmetric cryptanalysis • lower bounds: the limits of quantum computing • symmetric cryptanalysis: Grover's algorithm and beyond • post-quantum asymmetric cryptography • quantum error correction

Special information	Students are required to understand Mathematics (namely Linear Algebra, Complex Numbers, and Probability Theory) and Theoretical Computer Science (Complexity Theory). N. David Mermin: Quantum Computer Science: An Introduction John Preskill: Quantum Computing in the NISQ era and beyond <https://arxiv.org/abs/1801.00862> Introduction to Modern Cryptography by Mihir Bellare and Phillip Rogaway and recent publications

Course/Module Title	Security Engineering
Coordinator	Stefan Lucks
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	(no specific requirements for this course)
Exam requirements	Active participation in problem session: solving at least two problems identified in the session and presenting at least one solution. Final oral exam (max. 45 min.).
Specific target qualifications	Safety is about systems running reliably under normal and exceptional circumstances. Security is about systems defending themselves against malicious manipulation and attacks. The goal of this course is to provide an introduction to the specific skills and the mindset which the designers of such systems need. Students should understand the following tools and theories: • the programming languages Ada and SPARK • various strategies for white-box and black-box testing • preconditions, postconditions and invariants • the Hoare logic • data-flow analysis, information-flow analysis and the static verification of pre- and postconditions • the theory of distributed and failure-tolerant systems • algorithms for failure-tolerant distributed systems Students should be able to apply the above theories and tools to solve concrete problems. Furthermore, they should appreciate the limits and constraints of the above theories and tools. Students should be able formalise and generalise their own solutions using the above tools and theories. Students should master concepts and approaches such as • systematic testing • design by contract • static verification • formal models for failure modes (fail-stop, Byzantine) • algorithms for failure-tolerant distributed system in order to tackle problems from safe and secure system development and its application to digital media. They should be able to understand proposed solutions to safety and security problems, to compare different proposals for safe and secure systems, to make well-informed decisions about the preferred proposal and, if necessary, to find their own solutions to given safety and security problems. Students should be able to apply formal methods in practical settings. Specifically, students should be demonstrate their ability to formaly specify software properties, to implement such software, and to verify their implementation. In the lab, the SPARK toolset will be used.
Contents	• An Introduction to the Ada Programming Language • Software Testing • Design by Contract • The Hoare Logic • The SPARK Specification and Programming Language • Distributed Systems • The Concept of Tasks in Ada • The Development of Failure-Tolerant and Reliable Systems • Formal Language Theory for Security
Special information	Participants will need compilers for the Ada and SPARK programming languages (gnat, gnatprove), for the generation of automatic tests (testgen or AUnit) and for test covereage evaluation (gcov, lcov). All these tools are available at no cost under GPL.

Beschreibungen der Veranstaltungen aus dem Masterprogramm *Computer Science for Digital Media* für das Wahlpflichtmodul aus dem Bereich

Advanced Data Science

Course/Module Title	Introduction to Machine Learning
Coordinator	Benno Stein
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	(no specific requirements for this course)
Exam requirements	Active participation in lab classes. Final written exam (120 minutes).
Specific target qualifications	Given a task and a performance measure, a computer program (and hence a machine) is said to learn from experience if its performance at the task improves with experience. In this course, students will learn to understand machine learning as a guided search in a space of possible hypotheses. The mathematical means of formulating a particular hypothesis class determines the learning paradigm, the discriminative power of a hypothesis, and the complexity of the learning process. Students should understand the following concepts and theories: • model functions and hypothesis spaces • model formulation and model bias • unrestricted versus linear decision boundaries • input spaces versus feature spaces • how domain space structures determine learning approaches • means to specify loss and regularization objectives • discriminative versus generative learning approaches • statistical learning approaches Students should be able to formalize real-world classification tasks as machine learning problems. They should be able to apply the above mentioned concepts and theories to solve concrete learning problems. In particular, they should be able to choose the appropriate learning paradigm within a concrete setting. Students should master concepts and approaches such as • classifier programming • classifier application • classifier evaluation in order to tackle machine learning problems and their application to digital media. They should be able to analyze machine learning problems, to compare different learning algorithms, and to make well-informed decisions about the preferred learning paradigm. Students should develop an understanding of current developments in machine learning. With appropriate supervision, they should be able to tackle research problems.
Contents	• Model Formation • Concept Learning • Performance Evaluation • Linear Models • Neural Networks • Support Vector Machines • Decision Trees • Statistical Learning • Learning Theory • Principles of Deep Learning

Special information	Course material: https://lecturenotes.webis.de/#machine-learning Tools: Weka, scikit-earn, R, SciPy, GNU Octave Literature: • C.M. Bishop. *Pattern Recognition and Machine Learning* • T. Hastie, R. Tibshirani, J. Friedman. *The Elements of Statistical Learning* • T. Mitchell. *Machine Learning* • P.N. Tan, M. Steinbach, V. Kumar. *Introduction to Data Mining*

Course/Module Title	Introduction to Natural Language Processing
Coordinator	Benno Stein
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	(no specific requirements for this course)
Exam requirements	Active participation in lab classes. Final written exam (120 minutes).
Specific target qualifications	The course gives an overview of basic techniques of working with language data. It will introduce basic linguistic notions, issues involved in building and working with language corpora, current standard techniques for preparing text for analysis, and methods of computational processing of a subset of language phenomena. As part of practical lab classes, the students will get hands-on experience with language processing technology. Students should understand the following concepts and theories: • the basics of language processing • background on selected elements of language processing theory • practical considerations Students should get an understanding of key word-level, syntactic, semantic, and discourse phenomena, and be aware of issues involved in building text corpora. Students should master concepts and approaches such as • implement and apply NLP algorithms in practice • perform tasks that are part of a standard NLP pipeline • enable comparative evaluations • being able to educate oneself in order to assess the challenges and difficulties when solving natural language processing. They should be able to analyze language technologies, to compare different NLP algorithms, and to make well-informed decisions about the possibilities and limits of important NLP algorithms. Students should develop an understanding of current developments in natural language processing. With appropriate supervision, they should be able to tackle research problems.
Contents	• NLP Problems • Corpus Linguistics • Words • Syntax • Semantics • Pragmatics • NLP Architectures
Special information	Course material: https://lecturenotes.webis.de/#natural-language-processing Literature: • D. Jurafsky, J. H. Martin. *Speech and Language Processing* • C. D. Manning, H. Schütze. *Foundations of Statistical Natural Language Processing*

Course/Module Title	Search Algorithms
Coordinator	Benno Stein
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	(no specific requirements for this course)
Exam requirements	Active participation in lab classes. Final written exam (120 minutes).
Specific target qualifications	The course will introduce search algorithms as a means of solving combinatorial problems such as constraint satisfaction and optimization problems. Tackling such problems by machine often follows a two-step approach: (1) definition of a space of solution candidates followed by (2) intelligent exploration of this space. We will cover the modeling of search problems, basic (uninformed) search algorithms, informed search algorithms, as well as hybrid combinations. Special focus will be placed on heuristic search approaches. Students should understand the following concepts and theories: • state space versus problem reduction space • uninformed search • weight functions • cost measures • informed search • admissibility of search algorithms • search monotonicity and consistency Students should be able to model a search space by selecting the appropriate representation principle and by devising encoding for partial solution bases. They should understand and describe how different search algorithms will explore the search space differently. With regard to informed search algorithms, they should understand the principle of admissible search and be able to prove basic properties of the search algorithms (completeness, soundness, admissibility). The students will learn to analyze the nature of search problems, this way being able to • devise adequate search space representations • (heuristically) inform an uninformed strategy • develop admissible search strategies • combine informed with uninformed strategies • prove important properties such as admissibility or monotonicity. Students should eventually be able to tackle non-trivial search and constraint satisfaction problems and their application to digital media. In this regard, they should be able to make well-informed decisions and explain their approach to finding solutions, considering the theoretical background. With appropriate supervision, students should be able to tackle research problems. Students should develop an understanding of the current developments of the semantic web. With appropriate supervision, they should be able to tackle research problems.
Contents	• Search Examples • Search Space Representations • Algorithms for Uninformed Search • Hybrid Search Algorithms • Algorithms for Informed Search • Theoretical Properties of Search Algorithms

Special information	Course material: https://lecturenotes.webis.de/#search Literature: • Edmund K. Burke, Graham Kendall. *Search Methodologies* • Nils J. Nilsson. *Artificial Intelligence: A New Synthesis* • Judea Pearl. *Heuristics* • Stuart Russel, Peter Norvig. *Artificial Intelligence: A Modern Approach*

Course/Module Title	Web Search and Information Retrieval
Coordinator	Benno Stein
Assigned Module Groups(s)	Security and Data Science, Specialization, Electives
Formal requirements for participation	(no specific requirements for this course)
Exam requirements	Active participation in lab classes. Final written exam (120 minutes).
Specific target qualifications	Web search engines and information retrieval today have the world's information at the users' fingertips. Research from the last few decades now helps users to find what they want for a variety of information needs in a split second. The goal of this course is to understand how search engines and IR systems work, to acquire the necessary theoretical background that enables comprehension of practical considerations, and to develop an understanding of comparison and evaluations issues. Limits and constraints and latest trends are part of the curriculum. The students should understand the following topics: • the relationship of information retrieval and web search • indexing process (offline) and query process (online) • crawling large collections with storage issues and up-to-date checks • text-processing techniques, including subtleties of parsing, stopping, stemming and anchor texts • PageRank concept and algorithm • NLP techniques for information extraction • MapReduce technique for index creation • various types of inverted indexes • index-compression concepts for efficiency • information need vs. query formulation • techniques for transforming and improving human queries • comprehending components of result presentation • mathematical models of relevance • distinguishing probabilistic retrieval models from language modeling approaches • machine learning techniques for improving ranking • Cranfield paradigm for evaluating retrieval performance • effectiveness and efficiency metrics for retrieval systems Students should be able to apply the above theories and topics to solve concrete problems in the field of retrieval systems. Furthermore, they should appreciate the limits and constraints of the respective tools and methods that make them suitable approaches in specific scenarios. Students should be able to formalize and generalize their own solutions for retrieval problems using the above theories and methods. Students should master concepts and approaches such as • freshness vs. age as crawling update policies • stopping and stemming to reduce index sizes vs. store everything approaches • authority ranking approaches such as PageRank • delta encoding and skip pointes for efficient small indexes • the similarity and importance of tf-components in BM25 and query likelihood retrieval models • pooling strategies in search result evaluation • precision vs. recall in effectiveness evaluations to tackle search and retrieval problems and their application to digital media. They should be able to understand typical problems faced when developing retrieval systems, to compare different approaches suited to the different components of such systems, to make well-informed decisions about the preferred approach and, if necessary, to find their own solutions to given retrieval and search problems. Students should develop an understanding of the current state of research in web search and information retrieval. With appropriate supervision, students should also be able to tackle research problems.

Contents	• Introduction • Architecture of a Search Engine • Crawling, Parsing, Information Extraction • Inverted Indexes and Index Compression • Query Processing • Retrieval Models
Special information	Course material: https://lecturenotes.webis.de/#information-retrieval Literature: • R. Baeza-Yates, B. Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology behind Search* • C.D. Manning, P. Raghavan, H. Schütze. *Introduction to Information Retrieval* • S. Büttcher, C.L.A. Clarke, G.V. Cormack. *Information Retrieval: Implementing and Evaluating Search Engines*