Modul 10: Vertiefendes Wahlpflichtmodul, hier:

Modul 10(a): Vertiefende Aspekte der Algorithmik

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
	450 h	15	1-2	jährlich	2 Semester

1	**Lehrveranstaltungen** a) Vorlesung und Übung Ausgewählte Kapitel aus Algorithmen und Datenstrukturen *oder* Algorithmische Geometrie b) Vorlesung und Übung Netzwerkalgorithmen *oder* Algorithm Engineering	**Kontaktzeit** 9 SWS / 135h	**Selbst-studium** 315 h	**geplante Gruppengröße** 30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden gewinnen einen vertieften Einblick in einen selbst gewählten Bereich der Informatik. Die zu erwerbenden Kenntnisse in diesem Bereich können bis an den Stand der Forschung heranreichen.

Ausgewählte Kapitel aus Algorithmen und Datenstrukturen:

Die Vorlesung vermittelt den Studierenden Fähigkeiten

- zum Lesen und Verstehen aktueller Forschungsartikel aus dem Gebiet Datenstrukturen und Algorithmen
- zum Verständnis fortgeschrittener Datenstrukturen und typischer algorithmischer Paradigmen aus verschiedenen Anwendungsgebieten
- zur Analyse, zum Entwurf und zur Implementierung von effizienten Algorithmen
- zur Einschätzung der praktischen Einsatzmöglichkeiten von Algorithmen und Datenstrukturen

Algorithmische Geometrie:

- Erlernen verschiedener Algorithmen und Datenstrukturen für geometrische Probleme, sowie deren Entwurf, Analyse und Anwendung.
- Einschätzen der Besonderheiten diskreter geometrischer Probleme und Lösungen, etwa im Vergleich zu numerischen Verfahren,
- Entwurf und Implementierung neuer Verfahren für bestimmte Anwendungen
- Einsatz des Repertoires der entwickelten Datenstrukturen und Methoden für neue Probleme

Algorithm Engineering:

- Das Gebiet der Algorithmen und Datenstrukturen wird traditionell der theoretischen Informatik zugeordnet. Die hier entworfenen und untersuchten Algorithmen verwenden Häufig komplizierte Datenstrukturen und haben selten einen direkten Bezug zur Praxis, wo die asymptotische Komplexität der Probleme und Algorithmen und das worst-case Verhalten selten eine Rolle spielt. Hier sind häufig andere Aspekte, wie z.B. die Effiziente Nutzung des Caches oder die leichte Implementier- und Wartbarkeit viel wichtiger.
- Algorithm Engineering versucht die Kluft zwischen Theorie und Praxis zu schließen. Entsprechend werden wir in dieser Vorlesung Algorithmen und Datenstrukturen betrachten, die in der Praxis tatsächlich Verwendung finden.E

Netzwerkalgorithmen:

- Erlernen der wichtigsten algorithmischen Techniken zur Lösung von Netzwerkproblemen (kürzeste Wege, Flussprobleme, Matchings)
- Untersuchung der Komplexität dieser Probleme

	• Fähigkeiten zum Entwurf, der Analyse und Implementierung von entsprechenden Algorithmen • Untersuchung der praktischen Effizienz, insbesondere die Auswirkung verschiedener Datenstrukturen zur Darstellung von Graphen **Softskills bei allen genannten Veranstaltungen:** • Erwerb von Fertigkeiten und Methoden beim Durcharbeiten der Vorlesungsinhalte, • Selbstständiges Arbeiten beim Lösen von Übungsaufgaben, • Argumentation und Präsentation eigener Ergebnisse in den Übungsgruppen.
3	**Inhalte** **Ausgewählte Kapitel aus Algorithmen und Datenstrukturen:** Die Vorlesung befasst sich mit ausgewählten Themen aus dem Gebiet der Algorithmen und Datenstrukturen. Aufbauend auf den Grundkenntnissen aus Informatik II werden wir Probleme aus folgenden Bereichen behandeln: • Datenstrukturen für Mengen • Graph- und Netzwerkalgorithmen • Algorithmische Geometrie • Parallele Algorithmen Außer dem Entwurf und der Analyse entsprechender Algorithmen werden auch Fragen der Implementierung bzw. Programmierung behandelt. **Algorithmische Geometrie:** Die Vorlesung behandelt den Entwurf, die Analyse und die Implementierung von Algorithmen und Datenstrukturen für geometrische Probleme. Dabei werden grundlegende Vorgehensweisen und Paradigmen, wie „Teile und Beherrsche", Plane-Sweep, Dualität, und Randomisierung vorgestellt und auf Problemstellungen aus verschiedenen Bereichen der graphischen Datenverarbeitung angewandt, wie z.B. die Berechnung konvexer Hüllen, Bewegungsplanung für Roboter, Eliminierung von verborgenen Linien und Flächen, Boolesche Operationen auf Polygonen oder die Berechnung der nächsten Nachbarn. Ein zentrales Problem bei der Implementierung von geometrischen Algorithmen ist die Tatsache, dass Computer keine beliebig genauen reellen Zahlen, sondern nur Fließkommazahlen zur Verfügung stellen. Die dadurch entstehenden Rundungsfehler können nicht nur zu ungenauen Ergebnissen sondern zum völligen Versagen der Programme führen. Dieses Robustheitsproblem wird in der Vorlesung genauer untersucht und es werden Methoden zu seiner Lösung entwickelt. **Algorithm Engineering:** • Algorithm Engineering, was ist das ? • Implementierung grundlegender Datenstrukturen • Cache-Effizienz • Effizientes Sortieren • Mengen, Dictionaries und Priority Queues • Graphen und Netzwerke • Repräsentation von Graphen und Netzwerken • grundlegende Graphalgorithmen • Kürzeste Wege • Flussprobleme • Algorithmische Geometrie • Exaktes geometrisches Rechnen • Programmiertechniken zur effizienten Implementierung von Algorithmen **Netzwerkalgorithmen:**

	Die Vorlesung befasst sich mit der Entwicklung, Analyse und Implementierung von Algorithmen für Graphen und Netzwerkprobleme. Dabei sollen die wichtigsten Resultate dieses Gebiets behandelt werden, wie Algorithmen zur Berechung maximaler und billigster Flüsse. Insbesondere sollen einige der behandelten Algorithmen unter Verwendung der LEDA-Bibliothek implementiert und experimentell untersucht werden.
4	**Lehrformen** Vorlesungen und Übungen
5	**Teilnahmevoraussetzungen** **Formal:** keine **Inhaltlich:** keine
6	**Prüfungsformen** Abschlussklausur oder mündliche Prüfung
7	**Voraussetzungen für die Vergabe von Kreditpunkten** Erfolgreiche Übungsteilnahme, bestandene Modulprüfung
8	**Verwendung des Moduls** (in anderen Studiengängen)
9	**Stellenwert der Note für die Endnote** 15/120
10	**Modulbeauftragte/r und hauptamtlich Lehrende** Näher
11	**Sonstige Informationen** Veranstaltung i.d.R. in deutscher Sprache, bei Bedarf jedoch auf Englisch Letztes Bearbeitungsdatum: 22.04.2014

Modul 10: Vertiefendes Wahlpflichtmodul, hier:

Modul 10(b): Vertiefende Aspekte Theoretischer Informatik

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
	450 h	15	1-2	jährlich	2 Semester

1	**Lehrveranstaltungen** *3 Gebiete (jeweils V+Ü) aus dem folgenden Angebot:* Komplexitätstheorie A, Komplexitätstheorie B, Datenkompression, Berechenbarkeit und Logik, Lernalgorithmen, Formale Sprachen A, Formale Sprachen B, Rechnerarithmetik, Berechenbare Analysis, parameterisierte Algorithmen, Approximationsalgorithmen	**Kontakt-zeit** 9 SWS / 135h	**Selbst-studium** 315 h	**geplante Gruppengröße** 30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden gewinnen einen vertieften Einblick in einen selbst gewählten Bereich der Informatik. Die zu erwerbenden Kenntnisse in diesem Bereich können bis an den Stand der Forschung heranreichen.

Komplexitätstheorie A/B:

- Faktenwissen: Struktur der wichtigsten Komplexitätsklasssen
- Methodisches Wissen: Reduktionstechniken zum Vergleich der Komplexität
- Transferkompetenz:Übertragung theoretischer Komplexitätsbetrachtungen auf reale Probleme
- Normativ-bewertende Kompetenzen: Einschätzung realer Probleme bzgl. ihrer Komplexität

Datenkompression:

- Umgang mit dem mathematischen Hintergrund moderner Datenkompressionsverfahren üben und deren Notwendigkeit einsehen.

Berechenbarkeit und Logik:

- Aufbauend auf die Einführungsveranstaltung im Bachelor-Studium werden Ergänzungen aus dem klassischen Stoff der Berechenbarkeit und der Logik angeboten und entsprechendes Faktenwissen vermittelt.
- liefert Grundlagen für Masterarbeiten

Lernalgorithmen:

- Einführung in das Maschinelle Lernen, einem Kerngebiet der Künstlichen Intelligenz, vom Standpunkt der Theoretischen Informatik
- liefert Grundlagen für Masterarbeiten

Formale Sprachen A/B:

- Aufbauend auf die Einführungsveranstaltung im Bachelor-Studium werden Ergänzungen aus dem klassischen Stoff der Formalen Sprachen angeboten und entsprechendes Faktenwissen vermittelt.
- liefert Grundlagen für Masterarbeiten

Rechnerarithmetik:

- Faktenwissen: Grundkonzepte bei Darstellungen von Zahlen im Computer, Methoden zum Entwurf mehrfach-genauer Algorithmen
- Methodisches Wissen: Einblick in die Grundlagen numerischer Bibliotheken, Exakte

	Analysen arithmetischer Algorithmen • Normativ-bewertende Kompetenzen: Erkennen der Problematik arithmetisch ungenauer Algorithmen **Berechenbare Analysis:** • Methodisches Wissen: Berechenbarkeit auf nicht-abzählbaren Mengen • Transferkompetenz: Kombination der Gebiete Berechenbarkeit (Informatik) und Analysis (Mathematik) • liefert Grundlagen für Masterarbeiten **Parameterisierte Algorithmen/ Approximative Algorithmen** • praktischer Umgang mit dem grundlegenden P/NP-Problem der (Theoretischen) Informatik • liefert Grundlagen für Masterarbeiten **Für alle genannten Veranstaltungen gilt:** • Neben der Vermittlung von Faktenwissen wird auch stets Wert gelegt auf die Vermittlung und Einübung der Methoden innerhalb insbesondere der Theoretischen Informatik mit den entsprechenden Spezialisierungen (Methodenwissen). • Heranführung an den aktuellen Forschungsstand im Rahmen eines forschungsorientierten Studiums • Der Selbststudiumsanteil umfasst jeweils auch eigenständigen Umgang mit entsprechender Fachliteratur und dient so zur Einübung / Vorbereitung auf lebenslanges Lernen. **Softskills bei allen genannten Veranstaltungen:** • Erwerb von Fertigkeiten und Methoden beim Durcharbeiten der Vorlesungsinhalte, • Selbstständiges Arbeiten beim Lösen von Übungsaufgaben, • Argumentation und Präsentation eigener Ergebnisse in den Übungsgruppen.
3	**Inhalte** **Komplexitätstheorie A:** Die im Bachelorstudium erworbenen Kenntnisse zum Thema Komplexität werden in vieler Hinsicht vertieft. Konkret werden folgende Bereiche behandelt werden: • Berechnungsressourcen Zeit und Speicherplatz, • Komplexitätsklassen L, NL, P, NP, PSPACE, • Klassifikation konkreter Probleme, • Reduzierbarkeit und Vollständigkeit, • Harte Probleme der obigen Komplexitätsklassen, • Hierarchiesätze, • Orakel-Maschinen, • relativierte Berechenbarkeit • abschließend wird ein aktuelles Thema aus dem Bereich der Komplexitätstheorie behandelt. **Komplexitätstheorie B:** Die im Bachelorstudium erworbenen Kenntnisse zum Thema Komplexität werden in vieler Hinsicht vertieft. Konkret können folgende Bereiche behandelt werden: • Zum Umgang mit dem P/NP-Problem: parameterisierte Komplexitätsklassen wie W[t], W[P] usf., Approximations-Komplexitätsklassen wie APX etc., das PCP Theorem

- Schaltkreis-Komplexitätsklassen wie NC[1]
- Klassifikation konkreter Probleme,
- Reduzierbarkeit und Vollständigkeit,
- Harte Probleme der obigen Komplexitätsklassen,
- abschließend wird ein aktuelles Thema aus dem Bereich der Komplexitätstheorie behandelt.

Datenkompression:

Es werden verschiedene Techniken zum Entwurf von Datenkompressionsalgorithmen vorgestellt. Zum näheren Ablauf:
Hinweis: Aufgrund der Dynamik des Gebietes werden wir uns erlauben, von Zeit zu Zeit einige der unten bezeichneten konkreten Kompressionsverfahren und -methoden durch modernere zu ersetzen.

- Motivation / Grundlagen:informationstheoretische Grundlagen (Wahrscheinlichkeitsmodell; Entropie), Verzerrungsmaße für verlustbehaftete Kompression
- Präfixcodes: Der Satz von McMillan/Kraft, Shannon / Shannon-Fano / Huffmancodierungen, erweiterte und adaptive Codierungen, arithmetische Codierung / PPM (prediction by partial match)
- Wörterbuchverfahren: LZ77 / LZSS (g)zip, LZ78 / LZW gif, compress
- Weitere verlustfreie Techniken: Burrows-Wheeler-Transformation (BWT) / Blocksortierung, bedingte Entropie: eine Anwendung bei der Fax-Codierung, spezielle Anwendungen (fortlaufende Bildübertragung; Gensequenzen; etc.)
- Allgemeine Überlegungen zur verlustbehafteten Codierung
- Skalarquantisierung: Gleichquantisierer, adaptive Quantisierer, Lloyd-Max Verfahren
- Vektorquantisierung: Lindo-Buzo-Gray-Verfahren, Codebuchentwurf, Codebuchübertragung: lohnt sich der Aufwand?
- Differentialcodierung und Teilbandcodierung: Prädikative Differentialcodierung (Auto-)Korrelation, DPCM (auch mit Adaption), Delta-Modulierung, Frequenzfilter, Shapiros Nullbaumübertragung, Einführung in Wavelets
- Transformationscodierung: Passfilter, Diskrete Fourier-Transformation DFT, Diskrete Cosinus-Transformation DCT, Fraktale Codierung
- Anwendungen / Standards: JPEG, MPEG, Audiodaten

Berechenbarkeit und Logik:

- Es werden wechselnde Vertiefungen im Bereich Berechenbarkeit und Logik vorgestellt.
- Mögliche Vertiefungen können sein, wobei auch Kombinationen denkbar sind:
  - Klassische Rekursionstheorie
  - Lerntheorie
  - Mathematische Logik
  - Logische Kennzeichnungen Formaler Sprachen
  - Kolmogorov-Komplexität
  - Deduktionssysteme (Automatisches Beweisen)
  - Reduktionssysteme
  - Informationstheorie
  - Codierungstheorie

Lernalgorithmen:

Es werden verschiedene Techniken zum Entwurf und zur Analyse von Lernalgorithmen vorgestellt. Zum näheren Ablauf:

- Motivation / Grundlagen: Was ist „Lernen“ (Induktion) ? Ein einfaches mathematisches Modell (nach Gold)
- Lerntheorie: Schlussfolgen; der Satz von Blum und Blum, charakteristische Mengen; der Satz von Angluin, Modellvarianten, Lernen von Funktionen
- Zur Identifikation regulärer Sprachen: Minimal- und Quotientenautomaten, Verbandsstrukturen bei Zustandsverschmelzungsalgorithmen; Automatenlernen als Suchaufgabe,

testbare Sprachen, reversible Sprachen, eine Anwendung im Bereich XML / SGML
- Anfragelernen: Vorstellung des Lernmodells, Lernen von regulären Sprachen: der L* Algorithmus von Angluin, eine Anwendung zur Roboterorientierung, Erweiterungen für nicht-reguläre Sprachen
- Hidden Markov Models HMM: Parameterschätzung als Lernprozess, Viterbi-Algorithmus,, Anwendungen, speziell in der Bioinformatik und bei der Spracherkennung
- Wahrscheinlich annähernd korrektes Lernen, PAC: das Lernmodell, Lernen von Formeln, Stichprobenkomplexität; Vapnik/Chervonenkis Dimension, eine PAC-Variante des Lernalgorithmus L*
- Wir werden die Vorlesung mit einem „freien" Kapitel abschließen, das knapp in einen aktuellen Forschungsgegenstand einführt.

Formale Sprachen A:

Es werden wechselnde Vertiefungen im Bereich Formale Sprachen vorgestellt, aufbauend auf entsprechenden Veranstaltungen aus dem Bachelorprogramm.
Mögliche Vertiefungen können sein:
- Parallele Grammatiken
- weitere Automatenmodelle
- Regulierte Ersetzung
- Theoretische Modelle des Biocomputing
- Nicht-Standardmodelle für Berechenbarkeit
- Abstrakte Familien von Sprachen
- Algorithmische Fragestellungen bei Formalen Sprachen
- Beschreibungskomplexität von Formalen Sprachen

Formale Sprachen B:

Es werden wechselnde Vertiefungen im Bereich Formale Sprachen vorgestellt, aufbauend auf entsprechenden Veranstaltungen aus dem Bachelorprogramm.
Mögliche Vertiefungen können sein:
- Automaten auf Bäumen und Baumsprachen
- Automaten auf unendlichen Strukturen, z.B. omega-Wörtern, und entsprechende Wortmengen
- Schwach kontextsensitive Grammatikmodelle (wichtig für Linguistische Datenverarbeitung)
- Syntaktische Verfahren bei der Mustererkennung
- Formale Potenzreihen
- Syntaktische Methoden der Bildbeschreibung

Rechnerarithmetik:

Grundlagen der Computerarithmetik und des wissenschaftlichen Rechnens:
elementare Integer-Arithmetik – schnelle Multiplikation – modulare und redundante Zahl-Darstellungen – rationale Arithmetik – Fließkomma-Zahlen – IEEE 754/854 Fließkomma-Standards – Multiple-Precision-Arithmetik – Intervall-Arithmetik – Reduktionsmethoden in der Arithmetik – Komplexitätsbetrachtungen in der Arithmetik – exaktes Rechnen mit reellen und arithmetischen Zahlen

Berechenbare Analysis:

Viele mathematische Modelle in Wissenschaft und Technik verwenden die reellen Zahlen und darauf aufbauende Strukturen. Die berechenbare Analysis beschreibt und untersucht, wie man auf solchen Strukturen mit digitalen Computern rechnen kann. Es sollen u.a. folgende Themen behandelt werden:
- Typ-2-Berechenbarkeit,
- berechenbare reelle Zahlen und Funktionen,
- Darstellungen überabzählbarer Mengen,
- Zusammenhänge zwischen Berechenbarkeit und Stetigkeit,
- Komplexität reeller Zahlen und Funktionen, Differentiation und Integration,
- andere Berechenbarkeitsmodelle für die Analysis

	Parameterisierte Algorithmen: Es werden verschiedene Techniken zum Entwurf und zur Analyse parameterisierter Algorithmen vorgestellt. Zum näheren Ablauf: • Motivation / Grundlagen: P/NP Problematik, Parameterisierte Probleme, die Komplexitätsklasse FPT (fixed parameter tractable), graphentheoretische Grundbegriffe • Problemkerne: Problemkerneigenschaft = FPT, Kernreduktion aus Datenreduktionsregeln, lineare Kerne, Greedy-Algorithmen, Parameterisierte Dualität • Suchbäume und ihre Analyse: das Knotenüberdeckungsproblem, systematische Verbesserung von Suchbaumalgorithmen, Laufzeitberechnung von Suchbaumalgorithmen • Graphparameter: Baumzerlegungen, exakte Algorithmen, parameterisiert nach der Baumweite, Algorithmen auf planaren Graphen • Weitere Verfahren: parameterisiertes Zählen und Aufzählen, dynamisches Programmieren auf Teilmengen, iterative Kompression,Farbkodierung • Parameterisierte Komplexitätstheorie: parameterisierte Reduktionen, die Klassen W[1] und W[2] • In einem abschließenden Kapitel werden wir speziell auf einen aktuellen Forschungsgegenstand aus dem Bereich der Parameterisierten Algorithmen eingehen. **Approximative Algorithmen:** Es werden verschiedene Techniken zum Entwurf und zur Analyse approximativer Algorithmen vorgestellt. Zum näheren Ablauf: • Motivation / Einführung: P/NP Problematik, Optimierungsprobleme, das Knotenüberdeckungsproblem. lokale Verhältnisse (local ratio) • Grundlagen:Fragestellungen / Aufgaben bei Optimierungsproblemen, Reduktionen • Grundtechniken / Algorithmenklassen: Greedy-Algorithmen, Aufteilen und Anordnen (Partitionieren / Scheduling), Lokale Suche, Lineares Programmieren, Dynamisches Programmieren, Approximationstheorie / Approximationsklassen, absolute / relative Approximation (APX), Gap-Technik (Grenzen der Approximierbarkeit),Approximationsschemata: PTAS und FPTAS, verschiedene Reduktionsbegriffe • abschließend wird in einem Kapitel mit wechselnden Inhalten knapp in einen Bereich der aktuellen Forschung auf dem Gebiet approximativer Algorithmen eingeführt. **Softskills bei allen genannten Veranstaltungen:** • Erwerb von Fertigkeiten und Methoden beim Durcharbeiten der Vorlesungsinhalte, • Selbstständiges Arbeiten beim Lösen von Übungsaufgaben, • Argumentation und Präsentation eigener Ergebnisse in den Übungsgruppen.
4	**Lehrformen** Vorlesungen und Übungen
5	**Teilnahmevoraussetzungen** **Formal:** keine **Inhaltlich:** keine
6	**Prüfungsformen** Abschlussklausur oder mündliche Prüfung
7	**Voraussetzungen für die Vergabe von Kreditpunkten** Erfolgreiche Übungsteilnahme, bestandende Modulprüfung
8	**Verwendung des Moduls** (in anderen Studiengängen)
9	**Stellenwert der Note für die Endnote** 15/120
10	**Modulbeauftragte/r und hauptamtlich Lehrende**

	Fernau, Müller
11	**Sonstige Informationen** Veranstaltung i.d.R. in deutscher Sprache, bei Bedarf jedoch auf Englisch **Letztes Bearbeitungsdatum: 22.04.2014**

Modul 10: Vertiefendes Wahlpflichtmodul, hier:

Modul 10(c): Systemsoftware und Anwendungsarchitekturen (vertiefend)

Kennnummer	Workload	Credits	Studiensemester	Häufigkeit des Angebots	Dauer
	450 h	15	1-2	jährlich	2 Semester

1	Lehrveranstaltungen	Kontaktzeit	Selbststudium	geplante Gruppengröße
	Systemsoftware (V+Ü) *und 2 Gebiete (jeweils V+Ü) aus dem folgenden Angebot:* Verteilte Systeme, Spieleprogrammierung *oder* Grundlagen soziotechnischer Systeme	9 SWS / 135h	315 h	30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden gewinnen einen vertieften Einblick in einen selbst gewählten Bereich der Informatik. Die zu erwerbenden Kenntnisse in diesem Bereich können bis an den Stand der Forschung heranreichen.

Systemsoftware:

- Grundlegende Konzepte und Techniken moderner Betriebssysteme
- Überblick über aktuelle Betriebssysteme
- Ausgewählte Implementierungsaspekte
- Basiswissen systemnahe Programmierung

Verteilte Systeme:

- Vermittlung detaillierter Kenntnisse über die Möglichkeiten und Grenzen verteilter Systeme
- Einblick in gängige Middleware-Architekturen
- Fähigkeit zur Lösung anspruchsvoller Aufgabenstellungen aus dem Bereich verteilter Systeme in Kleingruppen

Spieleprogrammierung:

- Überblick über die Anforderungen an moderne multimediale Spiele
- Detaillierte Kenntnisse über Techniken der 3D-Graphikprogrammierung in Echtzeit
- Grundlegende Programmierkenntnisse in 3D-Echtzeitgraphik (DirectX und OpenGL)
- Lösung von Programmierprojekten in Kleingruppen

Grundlagen soziotechnischer Systeme:

- Vermittlung grundlegender Kenntnisse über die Möglichkeiten und Grenzen soziotechnischer Informationssysteme
- Beherrschen der Softwaretechniken, die für die Implementierung dieser Systeme notwendig sind
- Einblicke in die relevanten Systemarchitekturen
- Fähigkeit zur Realisierung anspruchsvoller Aufgabenstellungen im Bereich soziotechnischer Informationssysteme
- Reflektion gesellschaftlicher Chancen und Risiken

Softskills bei allen genannten Veranstaltungen:

- Erwerb von Fertigkeiten und Methoden beim Durcharbeiten der Vorlesungsinhalte,
- Selbstständiges Arbeiten beim Lösen von Übungsaufgaben,

	Argumentation und Präsentation eigener Ergebnisse in den Übungsgruppen.
3	**Inhalte** **Systemsoftware:** • Einführung: Zentrale Funktionen und Abstraktionen der Systemsoftware • Speicher: Reale und virtuelle Adressierungstechniken auf Betriebssystemebene (insbesondere Paging), Speicherverwaltung (u.a. Heap) • Threads: Abstraktion virtueller Prozessor, Scheduling, Schedulingverfahren, Thread- und Scheduling-spezifische APIs • Synchronisation: Synchronisationproblematik für nebenläufiger Threads und Prozesse, Unterschiedliche Lösungsansätze (u.a. Hardware-gestützt, Spinlocks, Semaphore, Monitore, bedingte kritische Abschnitte), Klassische Synchronisationsprobleme (u.a. Wechselseitiger Ausschluß, Producer/ Consumer, Reader/Writer), Deadlocks (Dining Philosophers Problem), Synchronisationsspezifische APIs • Kommunikation: Grundlagen der speicher- und nachrichtenbasierten Kommunikation (u.a. Shared Memory, Pipes, Message Queues, UDP und TCP, RPC, Remote Method Invocation bzw. Remoting), Kommunikationsspezifische APIs (u.a. POSIX 1003.4 und Win32 bzw. .NET 2.0 Framework) • Dateisysteme: Architektur von Dateisystemen (u.a. Datenträgerorganisation, Blockorientiertes Dateisystem, Virtuelles Dateisystem, RAID, LVM), Struktur aktueller Dateisysteme (u.a. NTFS, ext3, reiserfs), Memory-mapped Files, Datei- und Directory-APIs von Linux/Unix und Windows **Verteilte Systeme:** • Motivation: Grenzen verteilter Systeme (fehlender globaler Zustand, fehlende gemeinsame Zeit), Vorteile verteilter Systeme (Leistungssteigerung, Fehlertoleranz, Skalierbarkeit),Grundlagen Rechnernetze (Netzarchitekturen, Protokolle, Protokollfamilien) • Verteilte Algorithmen: Logische Uhren, Schnappschüsse, verteilter wechselseitiger Ausschluß, … • Load Balancing (inkl. Initial Placement und Migration) • Fehlertolerante Systeme (Primary-Backup, Hochverfügbarkeit, State Machines) • Verteilte Dateisysteme • GRIDs • Systemstrukturen (u.a. zentralisierte Ansätze, Client/Server, P2P, Ad-Hoc-Netze, SOA) • Middleware-Plattformen für verteilte Anwendungen • Heranführung an aktuell bearbeite Forschungsthemen in diesem Bereich **Spieleprogrammierung:** • Einführung: Anforderungen moderne Computerspiele insbesondere im Bereich 3D-Echtzeitgraphik • 3D-Graphik: Mathematische Grundlagen (Vektorräume, Homogene Koordinatensysteme, Affine Transformationen), Raytracing- und Radiosity-Verfahren, Fixed-Function und Programmable Rendering Pipeline moderner Graphikkarten, Einführung in DirectX- und OpenGL-Programmierung, Texturen und Licht, Pixel- und Vertex-Shader, Aufbau und Leistungsmerkmale moderner 3D-Hardware, Organisation komplexer Szenen (Oct-Trees, BSPs), 3D-Engines • Spielphysik: Grundlegende physikalische Gesetze (Kraft, Geschwindigkeit, Beschleunigung, Feder, Dämpfung, Reibung, Kollisionen…), Numerische Integration und Simulation, Physikprozessoren • Character Animation: Skeletthierarchie, Kinematik, Inverse Kinematik, Transformationen (Eulerwinkel, Quaternions, Lerps), Motion Extraction, Motion Capturing, Wechselwirkungen Animation und Rendering • Multiplayer: Netzwerke, Client/Server- und P2P-Ansätze, Verteilte Simulation, Dead Reckoning, Technische, gesellschaftliche und soziale Auswirkungen (insbesondere von Online-Spielen) • Spiele-KI: Non-Player Characters, Machine Learning, Expertensysteme, CBR, Bayesian Networks, Neuronale Netze, Sensing, Thinking, Acting

	• Spiele-Entwicklung: Übersicht und Diskussion über die einzelnen Phasen der Spielentwicklung, Wechselwirkungen Software-Engineering und Computerspiele, Wirtschaftsfaktor Computerspiele **Grundlagen soziotechnischer Systeme:** • Formale Grundlagen (Skalenfreie Netze, Power Law, ...) • Technische Grundlagen (Web-APIs, OAuth, P2P-Overlays, Reputationssysteme, Clouds) • Systemstrukturen (Soziale Utilities, P2P-Netzwerke, Crowd Sourcing) • Forschungstrends **Softskills bei allen genannten Veranstaltungen:** • Erwerb von Fertigkeiten und Methoden beim Durcharbeiten der Vorlesungsinhalte, • Selbstständiges Arbeiten beim Lösen von Übungsaufgaben, • Argumentation und Präsentation eigener Ergebnisse in den Übungsgruppen.
4	**Lehrformen** Vorlesungen und Übungen
5	**Teilnahmevoraussetzungen** **Formal:** keine **Inhaltlich:** keine
6	**Prüfungsformen** Abschlussklausur oder mündliche Prüfung
7	**Voraussetzungen für die Vergabe von Kreditpunkten** Erfolgreiche Übungsteilnahme, bestandende Modulprüfung
8	**Verwendung des Moduls** (in anderen Studiengängen)
9	**Stellenwert der Note für die Endnote** 15/120
10	**Modulbeauftragte/r und hauptamtlich Lehrende** Sturm
11	**Sonstige Informationen** Veranstaltung i.d.R. in deutscher Sprache, bei Bedarf jedoch auf Englisch **Letztes Bearbeitungsdatum: 22.04.2014**

Modul 10: Vertiefendes Wahlpflichtmodul, hier:

Modul 10(d): Datenbanksysteme und Information Retrieval

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
	450 h	15	1-2	jährlich	2 Semester

1	Lehrveranstaltungen	Kontakt-zeit	Selbst-studium	geplante Gruppengröße
	Datenbanksysteme 1 (V+Ü) *Datenbanksysteme 2 (V+Ü)* *Information Retrieval (V+Ü)*	9 SWS / 135h	315 h	30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden gewinnen einen vertieften Einblick in einen selbst gewählten Bereich der Informatik. Die zu erwerbenden Kenntnisse in diesem Bereich können bis an den Stand der Forschung heranreichen.

Datenbanksysteme 1:

- Grundlegende Kenntnisse über Datenbanksysteme
- Fakten- und Methodenwissen über konzeptuelle Modellierung, Design Relationaler Datenbanksysteme sowie Abfrage und Manipulation Relationaler Daten.
- Praktischer Umgang mit Entwurfsmethoden und mit einem aktuellen Datenbanksystem

Datenbanksysteme 2:

- Fakten- und Methodenwissen über Modellierung, Abfrage und Manipulation komplexer Objekte in Datenbanken
- Praktischer Umgang mit einem entsprechenden Datenbanksystem

Information Retrieval:

- Grundkenntnisse über Information Retrieval (IR) und Web Mining
- Faktenwissen über
  - typische Datenstrukturen und Algorithmen für IR-Systeme
  - theoretische Grundlagen der IR-Modelle
- Methodisches Wissen über
  - die Konstruktion von IR-Systemen
  - die experimentelle Evaluation von Retrieval-Mechanismen

Softskills bei allen genannten Veranstaltungen:

- Erwerb von Fertigkeiten und Methoden beim Durcharbeiten der Vorlesungsinhalte,
- Selbstständiges Arbeiten beim Lösen von Übungsaufgaben,
- Argumentation und Präsentation eigener Ergebnisse in den Übungsgruppen.

3 Inhalte

Datenbanksysteme 1:

- Motivation
- Datenmodellierung
- Konzeptuelle Datenmodellierung
  - Entity-Relationship-Modell
  - UML-Klassendiagramme
- Datenbank-Design
- Relationales Modell
- Abbildung Entity Relationship auf Relationales Modell
- Normalisierung
- Datenbanksprachen
- Relationenalgebra, Relationenkalkül
- SQL
- Query By Example

	• Erweiterungen des relationalen Datenmodells **Datenbanksysteme 2:** • Modellierung komplexer Daten ◦ Evolution von Datenmodellen • Objektorientierte Datenbanksysteme ◦ Objekt-Daten-Modell ◦ Anfragesprache OQL • Objektrelationale Datenbanksysteme ◦ Objekt-relationales Modell ◦ Anfrage- und Manipulationssprache SQL:2003/2010 • XML-Datenbanken ◦ XML als Datenmodell ◦ Abfrage und Manipulation von XML-Daten, SQL/XML, XQuery **Information Retrieval:** • Motivation ◦ Daten, Wissen, Information ◦ Pragmatische Aspekte des IR (Precision & Recall, Zipf'sche Verteilungen) • Grundtechniken ◦ Zugriffspfade für Volltexte ◦ Vorverarbeitung von Dokumenten (Zoning, Stemming etc.) ◦ Suche in Volltexten ◦ Boole'sches Retrieval ◦ Evaluierung mittels Standard-Textsammlungen • Vector-Space-Modell ◦ Darstellung von Dokumenten und Anfragen ◦ Gewichtung ◦ Ähnlichkeitsmaße • Alternative IR-Modelle • Anfragenverarbeitung ◦ Relevance-Feedback ◦ Anfrageerweiterungen • IR und Web-Mining ◦ Architektur von Suchmaschinen für das Internet ◦ Internet-spezifische Problemstellungen
4	**Lehrformen** Vorlesungen und Übungen
5	**Teilnahmevoraussetzungen** **Formal:** keine **Inhaltlich:** keine
6	**Prüfungsformen** Abschlussklausur oder mündliche Prüfung
7	**Voraussetzungen für die Vergabe von Kreditpunkten** Erfolgreiche Übungsteilnahme, bestandende Modulprüfung
8	**Verwendung des Moduls** (in anderen Studiengängen)
9	**Stellenwert der Note für die Endnote** 15/120
10	**Modulbeauftragte/r und hauptamtlich Lehrende** Walter
11	**Sonstige Informationen** Veranstaltung i.d.R. in deutscher Sprache, bei Bedarf jedoch auf Englisch **Letztes Bearbeitungsdatum: 22.04.2014**

Modul 10: Vertiefendes Wahlpflichtmodul, hier:

Modul 10(e): Vertiefende Aspekte der Informationssicherheit

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
	450 h	15	1-2	jährlich	2 Semester

1	**Lehrveranstaltungen** System- und Netzwerksicherheit (V+Ü) *und 1 Gebiet (jeweils V+Ü) aus dem folgenden Angebot:* Moderne Kryptographie *oder* Ausgewählte Kapitel der Informationssicherheit und Kryptographie	**Kontakt-zeit** 9 SWS / 135h	**Selbst-studium** 315 h	**geplante Gruppengröße** 30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden gewinnen einen vertieften Einblick in einen selbst gewählten Bereich der Informatik. Die zu erwerbenden Kenntnisse in diesem Bereich können bis an den Stand der Forschung heranreichen.

System- und Netzwerksicherheit:

- Studierende sind für Schwachstellen und Angriffe auf Ebene der System- und Netzwerksicherheit sensibilisiert;
- Studierende sind mit konkreten System- und Netzwerkangriffen sowie den zugrundeliegenden Prinzipien vertraut;
- Studieren kennen Verfahren und Mechanismen zur Abwehr dieser Angriffe.

Moderne Kryptographie:

- rigoroser, mathematisch präziser Zugang zur Kryptographie,
- Kenntnis von Sicherheitsdefinitionen für ausgewählte kryptographische Primitive,
- Fähigkeit einfache Sicherheitsbeweise selbst durchzuführen

Ausgewählte Kapitel der Informationssicherheit und Kryptographie:

- Fachkenntnis und Methoden zu aktuellen forschungsnahen Themen der Informationssicherheit und Kryptographie

Softskills bei allen genannten Veranstaltungen:

- Erwerb von Fertigkeiten und Methoden beim Durcharbeiten der Vorlesungsinhalte,
- Selbstständiges Arbeiten beim Lösen von Übungsaufgaben,
- Argumentation und Präsentation eigener Ergebnisse in den Übungsgruppen.

3 Inhalte

System- und Netzwerksicherheit:

- Angriffe und Abwehrmaßnahmen im Kontext der Software- und Betriebsystemsicherheit, einschließlich:
  - Stack und Heap Overflows, Format String Schwachstellen, Integer Overflows, etc.
  - Abwehrmaßnahmen auf Basis des Betriebsystems sowie statische und dynamische Abwehrmaßnahmen
- Web Sicherheit, einschließlich
  - Cross-Site-Scripting (CSS/XSS), SQL Injection, Cross-Site-Request-Forgery
  - client- und server-seitige Abwehrmaßnahmen
  - Sitzungsmanagement
- nicht-kryptographische Aspekte der Netzwerksicherheit, einschließlich
  - Angriffe auf den Domain Name Service (DNS)

	○ Denial of Service Angriffe ○ Firewalls **Moderne Kryptographie:** • Grundlegende kryptographische Primitive (z.B. Verschlüsselung, MACs, digitale Signaturen) und Protokolle (z.B.-Authentifizierung und Schlüsselaustausch) werden aufgegriffen und deren Sicherheit wird definiert. Außerdem werden mögliche Realisierungen dieser Primitive vorgestellt und deren Sicherheit wird auf Basis der formulierten Sicherheitsdefinitionen bewiesen (beweisbare Sicherheit).Zudem werden Public-Key-Infrastrukturen diskutiert. **Ausgewählte Kapitel der Informationssicherheit und Kryptographie:** • Die Inhalte dieser Veranstaltung richten sich nach aktuellen Forschungsthemen der Arbeitsgruppe. Mögliche Themen sind die folgenden: • Verifikation und Analyse kryptographischer Protokolle • simulationsbasierte Sicherheit • elektronische Wahlen • Sicherheit in Sensor- und Ad-Hoc-Netzwerken • Anonymität und Privatsphäre (Privacy) • Seitenkanalangriffe • Web Security
4	**Lehrformen** Vorlesungen und Übungen
5	**Teilnahmevoraussetzungen** **Formal:** keine **Inhaltlich:** keine
6	**Prüfungsformen** Abschlussklausur oder mündliche Prüfung
7	**Voraussetzungen für die Vergabe von Kreditpunkten** Erfolgreiche Übungsteilnahme, bestandende Modulprüfung
8	**Verwendung des Moduls** (in anderen Studiengängen)
9	**Stellenwert der Note für die Endnote** 15/120
10	**Modulbeauftragte/r und hauptamtlich Lehrende** Küsters
11	**Sonstige Informationen** Veranstaltung i.d.R. in deutscher Sprache, bei Bedarf jedoch auf Englisch **Letztes Bearbeitungsdatum: 22.04.2014**

Modul 10: Vertiefendes Wahlpflichtmodul, hier:

Modul 10(f): Softwaretechnik (vertiefend)

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
	450 h	15	1-2	jährlich	2 Semester

1	**Lehrveranstaltungen** Softwaretechnik (V+Ü) *und 2 Gebiete (jeweils V+Ü) aus dem folgenden Angebot:* Grundlagen der Computergraphik, Fortgeschrittene Softwaretechnik, Übersetzung und Analyse von Programmen *oder* Informationsvisualisierung	**Kontakt-zeit** 9 SWS / 135h	**Selbst-studium** 315 h	**geplante Gruppengröße** 30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden gewinnen einen vertieften Einblick in einen selbst gewählten Bereich der Informatik. Die zu erwerbenden Kenntnisse in diesem Bereich können bis an den Stand der Forschung heranreichen.

Softwaretechnik:

- Kenntnis der Phasen des Softwareentwicklungsprozesses
- Kenntnis gängiger Modellierungsmethoden und -notationen (UML)
- Kenntnis grundlegender Methoden zur Qualitätssicherung
- Verständnis der grundlegenden Probleme bei der Erstellung großer Softwaresysteme
- Fähigkeit zur kritischen Auseinandersetzung mit Originalliteratur im Bereich der Softwaretechnik

Grundlagen der Computergraphik:

- Kenntnis der Grundlagen der 2D und 3D Computergrafik
- Fähigkeit mit Hilfe von Werkzeugen, 3D-Modelle und Computeranimationen zu erstellen

Fortgeschrittene Softwaretechnik:

- Kenntnis aktueller Entwicklungen in der Softwaretechnik und Fähigkeit, diese zu beurteilen
- Fähigkeit fortgeschrittene Techniken einzusetzen
- Fähigkeit empirische Studien zu entwerfen und einzuschätzen

Übersetzung und Analyse von Programmen:

- Kenntnis der Struktur von Compilern
- Kenntnis der gängigen Erzeugungsverfahren für die verschiedenen Phasen eines Compilers
- Verständnis der Übersetzung einer realen Programmiersprache (z.B. Java)
- Kenntnis verschiedener Analyseverfahren

Informationsvisualisierung:

- Kenntnis grundlegender Visualisierungstechniken
- Kenntnis physiologischer und psychologischer Faktoren
- Kenntnis der wichtigsten Anwendungsgebiete von Visualisierungen (Schwerpunkt: Softwarevisualisierung)

Softskills bei allen genannten Veranstaltungen:

- Erwerb von Fertigkeiten und Methoden beim Durcharbeiten der Vorlesungsinhalte,
- Selbstständiges Arbeiten beim Lösen von Übungsaufgaben,

Argumentation und Präsentation eigener Ergebnisse in den Übungsgruppen.

3 Inhalte

Softwaretechnik:

- Einführung
- Die Phasen des Softwareentwicklungsprozesses Prozessmodelle
- Objekt-orientierte Analyse und Entwurf (Grundlagen der Objektorientierung, Anwendungsfallanalyse, Struktur- und Verhaltensdiagramme, Systementwurf)
- Entwurfsmuster
- Teamarbeit, Konfigurationsmanagement
- Qualitätssicherung, Testmethoden (Softwaremetriken, Testwerkzeuge)

	• Benutzerschnittstellen **Grundlagen der Computergraphik:** • Grundlagen: 2D Rastergrafik, Vektorgrafik • Bild und Videoformate • Grundlagen: 3D Computergrafik • Renderingpipeline • Beleuchtung • Transformationen • 3D Modellierungssprachen (z.B. X3D) • Grundlagen der Computeranimation **Fortgeschrittene Softwaretechnik:** • Bad smells, Anit-Patterns, Refactorings • Aspektorientierte Softwareentwicklung • Agile Softwareentwicklung, Extreme Programming • Methoden und Ergebnisse der empirischen Softwaretechnik • Weitere aktuelle Themen aus der Softwaretechnik **Übersetzung und Analyse von Programmen:** • Struktur eines Compilers • Lexikalische und syntaktische Analyse • Datenflussanalyse • Übersetzung von Java • Abstrakte Maschinen, insbesondere die JVM • Erkennen von Code-Klonen • Erkennen von Entwurfsmustern • Kenntnis der Struktur von Compilern **Informationsvisualisierung:** • Visualisierung von textuellen und numerischen Daten • Visualisierung von hierarchischen Informationen • Visualisierung von Graphen • Visuelle Wahrnehmung • Softwarevisualisierung • Evaluation von Visualisierungstechniken
4	**Lehrformen** Vorlesungen und Übungen
5	**Teilnahmevoraussetzungen** **Formal:** keine **Inhaltlich:** keine
6	**Prüfungsformen** Abschlussklausur oder mündliche Prüfung
7	**Voraussetzungen für die Vergabe von Kreditpunkten** Erfolgreiche Übungsteilnahme, bestandende Modulprüfung
8	**Verwendung des Moduls** (in anderen Studiengängen)
9	**Stellenwert der Note für die Endnote** 15/120
10	**Modulbeauftragte/r und hauptamtlich Lehrende** Diehl
11	**Sonstige Informationen** Veranstaltung i.d.R. in deutscher Sprache, bei Bedarf jedoch auf Englisch **Letztes Bearbeitungsdatum: 22.04.2014**

Modul 11: Wahlpflichtmodul, hier:

Modul 11(a): Algorithmik

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
	300 h	10	1-2	jährlich	1-2 Semester

1	**Lehrveranstaltungen** a) 1 Gebiet (Vorlesung) aus folgendem Angebot: *Netzwerkalgorithmen* oder *Algorithm Engineering* b) Übung zur gewählten Veranstaltung c) Seminar zur gewählten Veranstaltung	**Kontaktzeit** 5 SWS / 75h	**Selbst-studium** 225 h	**geplante Gruppengröße** 30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden gewinnen einen Einblick in einen selbst gewählten Bereich der Informatik. Die zu erwerbenden Kenntnisse in diesem Bereich können bis an den Stand der Forschung heranreichen. Durch ein Seminar zum gewählten Bereich erlernen die Studierenden, wie man sich neue Themen selbständig erarbeitet.

Algorithm Engineering:

- Das Gebiet der Algorithmen und Datenstrukturen wird traditionell der theoretischen Informatik zugeordnet. Die hier entworfenen und untersuchten Algorithmen verwenden Häufig komplizierte Datenstrukturen und haben selten einen direkten Bezug zur Praxis, wo die asymptotische Komplexität der Probleme und Algorithmen und das worst-case Verhalten selten eine Rolle spielt. Hier sind häufig andere Aspekte, wie z.B. die Effiziente Nutzung des Caches oder die leichte Implementier- und Wartbarkeit viel wichtiger.
- Algorithm Engineering versucht die Kluft zwischen Theorie und Praxis zu schließen. Entsprechend werden wir in dieser Vorlesung Algorithmen und Datenstrukturen betrachten, die in der Praxis tatsächlich Verwendung finden.E

Netzwerkalgorithmen:

- Erlernen der wichtigsten algorithmischen Techniken zur Lösung von Netzwerk-problemen (kürzeste Wege, Flussprobleme, Matchings)
- Untersuchung der Komplexität dieser Probleme
- Fähigkeiten zum Entwurf, der Analyse und Implementierung von entsprechenden Algo-rithmen
- Untersuchung der praktischen Effizienz, insbesondere die Auswirkung verschiedener Datenstrukturen zur Darstellung von Graphen

Seminar zur gewählten Veranstaltung:

- Selbststudium von ergänzender Literatur
- Befähigung, sich neue Themen selbständig zu erarbeiten
- Übung von Präsentationstechniken
- Übung von Kommunikationsfähigkeiten in der Diskussion
- Übung im Abfassen wissenschaftlicher Aufsätze durch Ausarbeitung
- i.d.R Übung des Leseverständnisses englischsprachiger Literatur

3 Inhalte

Algorithm Engineering:

- Algorithm Engineering, was ist das ?

	• Implementierung grundlegender Datenstrukturen • Cache-Effizienz • Effizientes Sortieren • Mengen, Dictionaries und Priority Queues • Graphen und Netzwerke • Repräsentation von Graphen und Netzwerken • grundlegende Graphalgorithmen • Kürzeste Wege • Flussprobleme • Algorithmische Geometrie • Exaktes geometrisches Rechnen • Programmiertechniken zur effizienten Implementierung von Algorithmen **Netzwerkalgorithmen:** Die Vorlesung befasst sich mit der Entwicklung, Analyse und Implementierung von Algorithmen für Graphen und Netzwerkprobleme. Dabei sollen die wichtigsten Resultate dieses Gebiets behandelt werden, wie Algorithmen zur Berechung maximaler und billigster Flüsse. Insbesondere sollen einige der behandelten Algorithmen unter Verwendung der LEDA-Bibliothek implementiert und experimentell untersucht werden.
4	**Lehrformen** Vorlesungen, Übungen, Seminar
5	**Teilnahmevoraussetzungen** **Formal:** keine **Inhaltlich:** keine
6	**Prüfungsformen** mündliche Prüfung
7	**Voraussetzungen für die Vergabe von Kreditpunkten** ErfolgreicheTeilnahme an den Übungen und am Seminar, bestandende Modulprüfung
8	**Verwendung des Moduls** (in anderen Studiengängen)
9	**Stellenwert der Note für die Endnote** 10/120
10	**Modulbeauftragte/r und hauptamtlich Lehrende** Näher
11	**Sonstige Informationen** Veranstaltung i.d.R. in deutscher Sprache, bei Bedarf jedoch auf Englisch Letztes Bearbeitungsdatum: 22.04.2014

Modul 11: Wahlpflichtmodul, hier:

Modul 11(b): Theoretische Informatik

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
	300 h	10	1-2	jährlich	1-2 Semester

1	**Lehrveranstaltungen** *a) 1 Gebiet (Vorlesung) aus dem folgenden Angebot:* Komplexitätstheorie A, Komplexitätstheorie B, Datenkompression, Berechenbarkeit und Logik, Lernalgorithmen, Formale Sprachen A, Formale Sprachen B, Rechnerarithmetik, Berechenbare Analysis, parameterisierte Algorithmen, Approximationsalgorithmen b) Übung zur gewählten Veranstaltung c) Seminar zur gewählten Veranstaltung	**Kontakt-zeit** 5 SWS / 75h	**Selbst-studium** 225 h	**geplante Gruppengröße** 30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden gewinnen einen Einblick in einen selbst gewählten Bereich der Informatik. Die zu erwerbenden Kenntnisse in diesem Bereich können bis an den Stand der Forschung heranreichen. Durch ein Seminar zum gewählten Bereich erlernen die Studierenden, wie man sich neue Themen selbständig erarbeitet.

Komplexitätstheorie A/B:

- Faktenwissen: Struktur der wichtigsten Komplexitätsklasssen
- Methodisches Wissen: Reduktionstechniken zum Vergleich der Komplexität
- Transferkompetenz:Übertragung theoretischer Komplexitätsbetrachtungen auf reale Probleme
- Normativ-bewertende Kompetenzen: Einschätzung realer Probleme bzgl. ihrer Komplexität

Datenkompression:

- Umgang mit dem mathematischen Hintergrund moderner Datenkompressionsverfahren üben und deren Notwendigkeit einsehen.

Berechenbarkeit und Logik:

- Aufbauend auf die Einführungsveranstaltung im Bachelor-Studium werden Ergänzungen aus dem klassischen Stoff der Berechenbarkeit und der Logik angeboten und entsprechendes Faktenwissen vermittelt.
- liefert Grundlagen für Masterarbeiten

Lernalgorithmen:

- Einführung in das Maschinelle Lernen, einem Kerngebiet der Künstlichen Intelligenz, vom Standpunkt der Theoretischen Informatik
- liefert Grundlagen für Masterarbeiten

Formale Sprachen A/B:

- Aufbauend auf die Einführungsveranstaltung im Bachelor-Studium werden Ergänzungen aus dem klassischen Stoff der Formalen Sprachen angeboten und entsprechendes Faktenwissen vermittelt.
- liefert Grundlagen für Masterarbeiten

Rechnerarithmetik:

	• Faktenwissen: Grundkonzepte bei Darstellungen von Zahlen im Computer, Methoden zum Entwurf mehrfach-genauer Algorithmen • Methodisches Wissen: Einblick in die Grundlagen numerischer Bibliotheken, Exakte Analysen arithmetischer Algorithmen • Normativ-bewertende Kompetenzen: Erkennen der Problematik arithmetisch ungenauer Algorithmen **Berechenbare Analysis:** • Methodisches Wissen: Berechenbarkeit auf nicht-abzählbaren Mengen • Transferkompetenz: Kombination der Gebiete Berechenbarkeit (Informatik) und Analysis (Mathematik) • liefert Grundlagen für Masterarbeiten **Parameterisierte Algorithmen/ Approximative Algorithmen** • praktischer Umgang mit dem grundlegenden P/NP-Problem der (Theoretischen) Informatik • liefert Grundlagen für Masterarbeiten **Für alle genannten Veranstaltungen gilt:** • Neben der Vermittlung von Faktenwissen wird auch stets Wert gelegt auf die Vermittlung und Einübung der Methoden innerhalb insbesondere der Theoretischen Informatik mit den entsprechenden Spezialisierungen (Methodenwissen). • Heranführung an den aktuellen Forschungsstand im Rahmen eines forschungsorientierten Studiums • Der Selbststudiumsanteil umfasst jeweils auch eigenständigen Umgang mit entsprechender Fachliteratur und dient so zur Einübung / Vorbereitung auf lebenslanges Lernen. **Seminar zur gewählten Veranstaltung:** • Selbststudium von ergänzender Literatur • Befähigung, sich neue Themen selbständig zu erarbeiten • Übung von Präsentationstechniken • Übung von Kommunikationsfähigkeiten in der Diskussion • Übung im Abfassen wissenschaftlicher Aufsätze durch Ausarbeitung • i.d.R Übung des Leseverständnisses englischsprachiger Literatur
3	**Inhalte** **Komplexitätstheorie A:** Die im Bachelorstudium erworbenen Kenntnisse zum Thema Komplexität werden in vieler Hinsicht vertieft. Konkret werden folgende Bereiche behandelt werden: • Berechnungsressourcen Zeit und Speicherplatz, • Komplexitätsklassen L, NL, P, NP, PSPACE, • Klassifikation konkreter Probleme, • Reduzierbarkeit und Vollständigkeit, • Harte Probleme der obigen Komplexitätsklassen, • Hierarchiesätze, • Orakel-Maschinen, • relativierte Berechenbarkeit • abschließend wird ein aktuelles Thema aus dem Bereich der Komplexitätstheorie behandelt. **Komplexitätstheorie B:**

Die im Bachelorstudium erworbenen Kenntnisse zum Thema Komplexität werden in vieler Hinsicht vertieft. Konkret können folgende Bereiche behandelt werden:

- Zum Umgang mit dem P/NP-Problem: parameterisierte Komplexitätsklassen wie W[t], W[P] usf., Approximations-Komplexitätsklassen wie APX etc., das PCP Theorem
- Schaltkreis-Komplexitätsklassen wie NC^1
- Klassifikation konkreter Probleme,
- Reduzierbarkeit und Vollständigkeit,
- Harte Probleme der obigen Komplexitätsklassen,
- abschließend wird ein aktuelles Thema aus dem Bereich der Komplexitätstheorie behandelt.

Datenkompression:

Es werden verschiedene Techniken zum Entwurf von Datenkompressionsalgorithmen vorgestellt. Zum näheren Ablauf:
Hinweis: Aufgrund der Dynamik des Gebietes werden wir uns erlauben, von Zeit zu Zeit einige der unten bezeichneten konkreten Kompressionsverfahren und -methoden durch modernere zu ersetzen.

- Motivation / Grundlagen:informationstheoretische Grundlagen (Wahrscheinlichkeitsmodell; Entropie), Verzerrungsmaße für verlustbehaftete Kompression
- Präfixcodes: Der Satz von McMillan/Kraft, Shannon / Shannon-Fano / Huffmancodierungen, erweiterte und adaptive Codierungen, arithmetische Codierung / PPM (prediction by partial match)
- Wörterbuchverfahren: LZ77 / LZSS (g)zip, LZ78 / LZW gif, compress
- Weitere verlustfreie Techniken: Burrows-Wheeler-Transformation (BWT) / Blocksortierung, bedingte Entropie: eine Anwendung bei der Fax-Codierung, spezielle Anwendungen (fortlaufende Bildübertragung; Gensequenzen; etc.)
- Allgemeine Überlegungen zur verlustbehafteten Codierung
- Skalarquantisierung: Gleichquantisierer, adaptive Quantisierer, Lloyd-Max Verfahren
- Vektorquantisierung: Lindo-Buzo-Gray-Verfahren, Codebuchentwurf, Codebuchübertragung: lohnt sich der Aufwand?
- Differentialcodierung und Teilbandcodierung: Prädikative Differentialcodierung (Auto-)Korrelation, DPCM (auch mit Adaption), Delta-Modulierung, Frequenzfilter, Shapiros Nullbaumübertragung, Einführung in Wavelets
- Transformationscodierung: Passfilter, Diskrete Fourier-Transformation DFT, Diskrete Cosinus-Transformation DCT, Fraktale Codierung
- Anwendungen / Standards: JPEG, MPEG, Audiodaten

Berechenbarkeit und Logik:

- Es werden wechselnde Vertiefungen im Bereich Berechenbarkeit und Logik vorgestellt.
- Mögliche Vertiefungen können sein, wobei auch Kombinationen denkbar sind:
  - Klassische Rekursionstheorie
  - Lerntheorie
  - Mathematische Logik
  - Logische Kennzeichnungen Formaler Sprachen
  - Kolmogorov-Komplexität
  - Deduktionssysteme (Automatisches Beweisen)
  - Reduktionssysteme
  - Informationstheorie
  - Codierungstheorie

Lernalgorithmen:

Es werden verschiedene Techniken zum Entwurf und zur Analyse von Lernalgorithmen vorgestellt. Zum näheren Ablauf:

- Motivation / Grundlagen: Was ist „Lernen“ (Induktion) ? Ein einfaches mathematisches

Modell (nach Gold)
- Lerntheorie: Schlussfolgen; der Satz von Blum und Blum, charakteristische Mengen; der Satz von Angluin, Modellvarianten, Lernen von Funktionen
- Zur Identifikation regulärer Sprachen: Minimal- und Quotientenautomaten, Verbandsstrukturen bei Zustandsverschmelzungsalgorithmen; Automatenlernen als Suchaufgabe, testbare Sprachen, reversible Sprachen, eine Anwendung im Bereich XML / SGML
- Anfragelernen: Vorstellung des Lernmodells, Lernen von regulären Sprachen: der L* Algorithmus von Angluin, eine Anwendung zur Roboterorientierung, Erweiterungen für nicht-reguläre Sprachen
- Hidden Markov Models HMM: Parameterschätzung als Lernprozess, Viterbi-Algorithmus,, Anwendungen, speziell in der Bioinformatik und bei der Spracherkennung
- Wahrscheinlich annähernd korrektes Lernen, PAC: das Lernmodell, Lernen von Formeln, Stichprobenkomplexität; Vapnik/Chervonenkis Dimension, eine PAC-Variante des Lernalgorithmus L*
- Wir werden die Vorlesung mit einem „freien" Kapitel abschließen, das knapp in einen aktuellen Forschungsgegenstand einführt.

Formale Sprachen A:

Es werden wechselnde Vertiefungen im Bereich Formale Sprachen vorgestellt, aufbauend auf entsprechenden Veranstaltungen aus dem Bachelorprogramm.
Mögliche Vertiefungen können sein:
- Parallele Grammatiken
- weitere Automatenmodelle
- Regulierte Ersetzung
- Theoretische Modelle des Biocomputing
- Nicht-Standardmodelle für Berechenbarkeit
- Abstrakte Familien von Sprachen
- Algorithmische Fragestellungen bei Formalen Sprachen
- Beschreibungskomplexität von Formalen Sprachen

Formale Sprachen B:

Es werden wechselnde Vertiefungen im Bereich Formale Sprachen vorgestellt, aufbauend auf entsprechenden Veranstaltungen aus dem Bachelorprogramm.
Mögliche Vertiefungen können sein:
- Automaten auf Bäumen und Baumsprachen
- Automaten auf unendlichen Strukturen, z.B. omega-Wörtern, und entsprechende Wortmengen
- Schwach kontextsensitive Grammatikmodelle (wichtig für Linguistische Datenverarbeitung)
- Syntaktische Verfahren bei der Mustererkennung
- Formale Potenzreihen
- Syntaktische Methoden der Bildbeschreibung

Rechnerarithmetik:

Grundlagen der Computerarithmetik und des wissenschaftlichen Rechnens:
elementare Integer-Arithmetik – schnelle Multiplikation – modulare und redundante Zahl-Darstellungen – rationale Arithmetik – Fließkomma-Zahlen – IEEE 754/854 Fließkomma-Standards – Multiple-Precision-Arithmetik – Intervall-Arithmetik – Reduktionsmethoden in der Arithmetik – Komplexitätsbetrachtungen in der Arithmetik – exaktes Rechnen mit reellen und arithmetischen Zahlen

Berechenbare Analysis:

Viele mathematische Modelle in Wissenschaft und Technik verwenden die reellen Zahlen und darauf aufbauende Strukturen. Die berechenbare Analysis beschreibt und untersucht, wie man auf solchen Strukturen mit digitalen Computern rechnen kann. Es sollen u.a. folgende Themen behandelt werden:
- Typ-2-Berechenbarkeit,
- berechenbare reelle Zahlen und Funktionen,

	• Darstellungen überabzählbarer Mengen, • Zusammenhänge zwischen Berechenbarkeit und Stetigkeit, • Komplexität reeller Zahlen und Funktionen, Differentiation und Integration, • andere Berechenbarkeitsmodelle für die Analysis **Parameterisierte Algorithmen:** Es werden verschiedene Techniken zum Entwurf und zur Analyse parameterisierter Algorithmen vorgestellt. Zum näheren Ablauf: • Motivation / Grundlagen: P/NP Problematik, Parameterisierte Probleme, die Komplexitätsklasse FPT (fixed parameter tractable), graphentheoretische Grundbegriffe • Problemkerne: Problemkerneigenschaft = FPT, Kernreduktion aus Datenreduktionsregeln, lineare Kerne, Greedy-Algorithmen, Parameterisierte Dualität • Suchbäume und ihre Analyse: das Knotenüberdeckungsproblem, systematische Verbesserung von Suchbaumalgorithmen, Laufzeitberechnung von Suchbaumalgorithmen • Graphparameter: Baumzerlegungen, exakte Algorithmen, parameterisiert nach der Baumweite, Algorithmen auf planaren Graphen • Weitere Verfahren: parameterisiertes Zählen und Aufzählen, dynamisches Programmieren auf Teilmengen, iterative Kompression,Farbkodierung • Parameterisierte Komplexitätstheorie: parameterisierte Reduktionen, die Klassen W[1] und W[2] • In einem abschließenden Kapitel werden wir speziell auf einen aktuellen Forschungsgegenstand aus dem Bereich der Parameterisierten Algorithmen eingehen. **Approximative Algorithmen:** Es werden verschiedene Techniken zum Entwurf und zur Analyse approximativer Algorithmen vorgestellt. Zum näheren Ablauf: • Motivation / Einführung: P/NP Problematik, Optimierungsprobleme, das Knotenüberdeckungsproblem. lokale Verhältnisse (local ratio) • Grundlagen:Fragestellungen / Aufgaben bei Optimierungsproblemen, Reduktionen • Grundtechniken / Algorithmenklassen: Greedy-Algorithmen, Aufteilen und Anordnen (Partitionieren / Scheduling), Lokale Suche, Lineares Programmieren, Dynamisches Programmieren, Approximationstheorie / Approximationsklassen, absolute / relative Approximation (APX), Gap-Technik (Grenzen der Approximierbarkeit),Approximationsschemata: PTAS und FPTAS, verschiedene Reduktionsbegriffe • abschließend wird in einem Kapitel mit wechselnden Inhalten knapp in einen Bereich der aktuellen Forschung auf dem Gebiet approximativer Algorithmen eingeführt.
4	**Lehrformen** Vorlesungen, Übungen, Seminar
5	**Teilnahmevoraussetzungen** **Formal:** keine **Inhaltlich:** keine
6	**Prüfungsformen** mündliche Prüfung
7	**Voraussetzungen für die Vergabe von Kreditpunkten** Erfolgreiche Teilnahme an den Übungen und am Seminar, bestandende Modulprüfung
8	**Verwendung des Moduls** (in anderen Studiengängen)
9	**Stellenwert der Note für die Endnote** 10/120
10	**Modulbeauftragte/r und hauptamtlich Lehrende** Fernau, Müller
11	**Sonstige Informationen** Veranstaltung i.d.R. in deutscher Sprache, bei Bedarf jedoch auf Englisch Letztes Bearbeitungsdatum: 22.04.2014

Modul 11: Wahlpflichtmodul, hier:

Modul 11(c): Systemsoftware und Anwendungsarchitekturen

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
	300 h	10	1-2	jährlich	1-2 Semester

1	Lehrveranstaltungen	Kontakt-zeit	Selbst-studium	geplante Gruppengröße
	a) 1 Gebiet (Vorlesung) aus dem folgenden Angebot: Systemsoftware, Verteilte Systeme, Spieleprogrammierung *oder* Grundlagen soziotechnischer Systeme b) Übung zur gewählten Veranstaltung c) Seminar zur gewählten Veranstaltung	5 SWS / 75h	225 h	30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden gewinnen einen Einblick in einen selbst gewählten Bereich der Informatik. Die zu erwerbenden Kenntnisse in diesem Bereich können bis an den Stand der Forschung heranreichen. Durch ein Seminar zum gewählten Bereich erlernen die Studierenden, wie man sich neue Themen selbständig erarbeitet.

Systemsoftware:

- Grundlegende Konzepte und Techniken moderner Betriebssysteme
- Überblick über aktuelle Betriebssysteme
- Ausgewählte Implementierungsaspekte
- Basiswissen systemnahe Programmierung

Verteilte Systeme:

- Vermittlung detaillierter Kenntnisse über die Möglichkeiten und Grenzen verteilter Systeme
- Einblick in gängige Middleware-Architekturen
- Fähigkeit zur Lösung anspruchsvoller Aufgabenstellungen aus dem Bereich verteilter Systeme in Kleingruppen

Spieleprogrammierung:

- Überblick über die Anforderungen an moderne multimediale Spiele
- Detaillierte Kenntnisse über Techniken der 3D-Graphikprogrammierung in Echtzeit
- Grundlegende Programmierkenntnisse in 3D-Echtzeitgraphik (DirectX und OpenGL)
- Lösung von Programmierprojekten in Kleingruppen

Grundlagen soziotechnischer Systeme:

- Vermittlung grundlegender Kenntnisse über die Möglichkeiten und Grenzen soziotechnischer Informationssysteme
- Beherrschen der Softwaretechniken, die für die Implementierung dieser Systeme notwendig sind
- Einblicke in die relevanten Systemarchitekturen
- Fähigkeit zur Realisierung anspruchsvoller Aufgabenstellungen im Bereich soziotechnischer Informationssysteme
- Reflektion gesellschaftlicher Chancen und Risiken

Softskills bei allen genannten Veranstaltungen:

	• Erwerb von Fertigkeiten und Methoden beim Durcharbeiten der Vorlesungsinhalte • Selbstständiges Arbeiten beim Lösen von Übungsaufgaben • Argumentation und Präsentation eigener Ergebnisse in den Übungsgruppen. **Seminar zur gewählten Veranstaltung:** • Selbststudium von ergänzender Literatur • Befähigung, sich neue Themen selbständig zu erarbeiten • Übung von Präsentationstechniken • Übung von Kommunikationsfähigkeiten in der Diskussion • Übung im Abfassen wissenschaftlicher Aufsätze durch Ausarbeitung • i.d.R Übung des Leseverständnisses englischsprachiger Literatur
3	**Inhalte** **Systemsoftware:** • Einführung: Zentrale Funktionen und Abstraktionen der Systemsoftware • Speicher: Reale und virtuelle Adressierungstechniken auf Betriebssystemebene (insbesondere Paging), Speicherverwaltung (u.a. Heap) • Threads: Abstraktion virtueller Prozessor, Scheduling, Schedulingverfahren, Thread- und Scheduling-spezifische APIs • Synchronisation: Synchronisationproblematik für nebenläufiger Threads und Prozesse, Unterschiedliche Lösungsansätze (u.a. Hardware-gestützt, Spinlocks, Semaphore, Monitore, bedingte kritische Abschnitte), Klassische Synchronisationsprobleme (u.a. Wechselseitiger Ausschluß, Producer/ Consumer, Reader/Writer), Deadlocks (Dining Philosophers Problem), Synchronisationsspezifische APIs • Kommunikation: Grundlagen der speicher- und nachrichtenbasierten Kommunikation (u.a. Shared Memory, Pipes, Message Queues, UDP und TCP, RPC, Remote Method Invocation bzw. Remoting), Kommunikationsspezifische APIs (u.a. POSIX 1003.4 und Win32 bzw. .NET 2.0 Framework) • Dateisysteme: Architektur von Dateisystemen (u.a. Datenträgerorganisation, Blockorientiertes Dateisystem, Virtuelles Dateisystem, RAID, LVM), Struktur aktueller Dateisysteme (u.a. NTFS, ext3, reiserfs), Memory-mapped Files, Datei- und Directory-APIs von Linux/Unix und Windows **Verteilte Systeme:** • Motivation: Grenzen verteilter Systeme (fehlender globaler Zustand, fehlende gemeinsame Zeit), Vorteile verteilter Systeme (Leistungssteigerung, Fehlertoleranz, Skalierbarkeit),Grundlagen Rechnernetze (Netzarchitekturen, Protokolle, Protokollfamilien) • Verteilte Algorithmen: Logische Uhren, Schnappschüsse, verteilter wechselseitiger Ausschluß, … • Load Balancing (inkl. Initial Placement und Migration) • Fehlertolerante Systeme (Primary-Backup, Hochverfügbarkeit, State Machines) • Verteilte Dateisysteme • GRIDs • Systemstrukturen (u.a. zentralisierte Ansätze, Client/Server, P2P, Ad-Hoc-Netze, SOA) • Middleware-Plattformen für verteilte Anwendungen • Heranführung an aktuell bearbeite Forschungsthemen in diesem Bereich **Spieleprogrammierung:** • Einführung: Anforderungen moderne Computerspiele insbesondere im Bereich 3D-Echtzeitgraphik • 3D-Graphik: Mathematische Grundlagen (Vektorräume, Homogene Koordin-

	atensysteme, Affine Transformationen), Raytracing- und Radiosity-Verfahren, Fixed-Function und Programmable Rendering Pipeline moderner Graphikkarten, Einführung in DirectX- und OpenGL-Programmierung, Texturen und Licht, Pixel- und Vertex-Shader, Aufbau und Leistungsmerkmale moderner 3D-Hardware, Organisation komplexer Szenen (Oct-Trees, BSPs), 3D-Engines • Spielphysik: Grundlegende physikalische Gesetze (Kraft, Geschwindigkeit, Beschleunigung, Feder, Dämpfung, Reibung, Kollisionen…), Numerische Integration und Simulation, Physikprozessoren • Character Animation: Skeletthierarchie, Kinematik, Inverse Kinematik, Transformationen (Eulerwinkel, Quaternions, Lerps), Motion Extraction, Motion Capturing, Wechselwirkungen Animation und Rendering • Multiplayer: Netzwerke, Client/Server- und P2P-Ansätze, Verteilte Simulation, Dead Reckoning, Technische, gesellschaftliche und soziale Auswirkungen (insbesondere von Online-Spielen) • Spiele-KI: Non-Player Characters, Machine Learning, Expertensysteme, CBR, Bayesian Networks, Neuronale Netze, Sensing, Thinking, Acting • Spiele-Entwicklung: Übersicht und Diskussion über die einzelnen Phasen der Spielentwicklung, Wechselwirkungen Software-Engineering und Computerspiele, Wirtschaftsfaktor Computerspiele **Grundlagen soziotechnischer Systeme:** • Formale Grundlagen (Skalenfreie Netze, Power Law, ...) • Technische Grundlagen (Web-APIs, OAuth, P2P-Overlays, Reputationssysteme, Clouds) • Systemstrukturen (Soziale Utilities, P2P-Netzwerke, Crowd Sourcing) • Forschungstrends
4	**Lehrformen** Vorlesungen, Übungen, Seminar
5	**Teilnahmevoraussetzungen** **Formal:** keine **Inhaltlich:** keine
6	**Prüfungsformen** mündliche Prüfung
7	**Voraussetzungen für die Vergabe von Kreditpunkten** ErfolgreicheTeilnahme an den Übungen und am Seminar, bestandende Modulprüfung
8	**Verwendung des Moduls** (in anderen Studiengängen)
9	**Stellenwert der Note für die Endnote** 10/120
10	**Modulbeauftragte/r und hauptamtlich Lehrende** Sturm
11	**Sonstige Informationen** Veranstaltung i.d.R. in deutscher Sprache, bei Bedarf jedoch auf Englisch Letztes Bearbeitungsdatum: 22.04.2014

Modul 11: Wahlpflichtmodul, hier:

Modul 11(d): Datenbanksysteme

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
	300 h	10	1-2	jährlich	1-2 Semester

1	**Lehrveranstaltungen** *a)* Vorlesung *Datenbanksysteme 1* b) Übung zur Vorlesung c) Seminar zur Vorlesung	**Kontakt-zeit** 5 SWS / 75h	**Selbst-studium** 225 h	**geplante Gruppengröße** 30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden gewinnen einen Einblick in einen selbst gewählten Bereich der Informatik. Die zu erwerbenden Kenntnisse in diesem Bereich können bis an den Stand der Forschung heranreichen. Durch ein Seminar zum gewählten Bereich erlernen die Studierenden, wie man sich neue Themen selbständig erarbeitet.

Datenbanksysteme 1:

- Grundlegende Kenntnisse über Datenbanksysteme
- Fakten- und Methodenwissen über konzeptuelle Modellierung, Design Relationaler Datenbanksysteme sowie Abfrage und Manipulation Relationaler Daten.
- Praktischer Umgang mit Entwurfsmethoden und mit einem aktuellen Datenbanksystem

Softskills:

- Erwerb von Fertigkeiten und Methoden beim Durcharbeiten der Vorlesungsinhalte,
- Selbstständiges Arbeiten beim Lösen von Übungsaufgaben,
- Argumentation und Präsentation eigener Ergebnisse in den Übungsgruppen.

Seminar zur gewählten Veranstaltung:

- Selbststudium von ergänzender Literatur
- Befähigung, sich neue Themen selbständig zu erarbeiten
- Übung von Präsentationstechniken
- Übung von Kommunikationsfähigkeiten in der Diskussion
- Übung im Abfassen wissenschaftlicher Aufsätze durch Ausarbeitung
- i.d.R Übung des Leseverständnisses englischsprachiger Literatur

3 Inhalte

- Datenmodellierung
- Konzeptuelle Datenmodellierung
  - Entity-Relationship-Modell
  - UML-Klassendiagramme
- Datenbank-Design
- Relationales Modell
- Abbildung Entity Relationship auf Relationales Modell
- Normalisierung
- Datenbanksprachen
- Relationenalgebra, Relationenkalkül
- SQL
- Query By Example

	• Erweiterungen des relationalen Datenmodells
4	**Lehrformen** Vorlesungen, Übungen, Seminar
5	**Teilnahmevoraussetzungen** **Formal:** keine **Inhaltlich:** keine
6	**Prüfungsformen** mündliche Prüfung
7	**Voraussetzungen für die Vergabe von Kreditpunkten** ErfolgreicheTeilnahme an den Übungen und am Seminar, bestandende Modulprüfung
8	**Verwendung des Moduls** (in anderen Studiengängen)
9	**Stellenwert der Note für die Endnote** 10/120
10	**Modulbeauftragte/r und hauptamtlich Lehrende** Walter
11	**Sonstige Informationen** Veranstaltung i.d.R. in deutscher Sprache, bei Bedarf jedoch auf Englisch Letztes Bearbeitungsdatum: 22.04.2014

Modul 11: Wahlpflichtmodul, hier:

Modul 11(e): Informationssicherheit

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
	300 h	10	1-2	jährlich	1-2 Semester

1	Lehrveranstaltungen	Kontakt-zeit	Selbst-studium	geplante Gruppengröße
	a) Vorlesung *System- und Netzwerksicherheit* b) Übung zur Vorlesung *c) Seminar zur* Vorlesung	5 SWS / 75h	225 h	30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden gewinnen einen Einblick in einen selbst gewählten Bereich der Informatik. Die zu erwerbenden Kenntnisse in diesem Bereich können bis an den Stand der Forschung heranreichen. Durch ein Seminar zum gewählten Bereich erlernen die Studierenden, wie man sich neue Themen selbständig erarbeitet.

Vorlesung und Übung zur System- und Netzwerksicherheit

- Studierende sind für Schwachstellen und Angriffe auf Ebene der System- und Netzwerksicherheit sensibilisiert;
- Studierende sind mit konkreten System- und Netzwerkangriffen sowie den zugrundeliegenden Prinzipien vertraut;
- Studieren kennen Verfahren und Mechanismen zur Abwehr dieser Angriffe.

Softskills:

- Erwerb von Fertigkeiten und Methoden beim Durcharbeiten der Vorlesungsinhalte,
- Selbstständiges Arbeiten beim Lösen von Übungsaufgaben,
- Argumentation und Präsentation eigener Ergebnisse in den Übungsgruppen.

Seminar zur Veranstaltung:

- Selbststudium von ergänzender Literatur
- Befähigung, sich neue Themen selbständig zu erarbeiten
- Übung von Präsentationstechniken
- Übung von Kommunikationsfähigkeiten in der Diskussion
- Übung im Abfassen wissenschaftlicher Aufsätze durch Ausarbeitung
- i.d.R Übung des Leseverständnisses englischsprachiger Literatur

3 Inhalte

- Angriffe und Abwehrmaßnahmen im Kontext der Software- und Betriebsystemsicherheit, einschließlich:
  - Stack und Heap Overflows, Format String Schwachstellen, Integer Overflows, etc.
  - Abwehrmaßnahmen auf Basis des Betriebsystems sowie statische und dynamische Abwehrmaßnahmen
- Web Sicherheit, einschließlich
  - Cross-Site-Scripting (CSS/XSS), SQL Injection, Cross-Site-Request-Forgery
  - client- und server-seitige Abwehrmaßnahmen
  - Sitzungsmanagement
- nicht-kryptographische Aspekte der Netzwerksicherheit, einschließlich
  - Angriffe auf den Domain Name Service (DNS)
  - Denial of Service Angriffe
  - Firewalls

4	**Lehrformen** Vorlesungen, Übungen, Seminar
5	**Teilnahmevoraussetzungen** **Formal:** keine **Inhaltlich:** keine
6	**Prüfungsformen** mündliche Prüfung
7	**Voraussetzungen für die Vergabe von Kreditpunkten** ErfolgreicheTeilnahme an den Übungen und am Seminar, bestandende Modulprüfung
8	**Verwendung des Moduls** (in anderen Studiengängen)
9	**Stellenwert der Note für die Endnote** 10/120
10	**Modulbeauftragte/r und hauptamtlich Lehrende** Küsters
11	**Sonstige Informationen** Veranstaltung i.d.R. in deutscher Sprache, bei Bedarf jedoch auf Englisch Letztes Bearbeitungsdatum: 22.04.2014

Modul 11: Wahlpflichtmodul, hier:

Modul 11(f): Softwaretechnik

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
	300 h	10	1-2	jährlich	1-2 Semester

1	**Lehrveranstaltungen** *a) 1 Gebiet (Vorlesung) aus dem folgenden Angebot:* Softwaretechnik (V+Ü), Grundlagen der Computergraphik oder Übersetzung und Analyse von Programmen b) Übung zur gewählten Vorlesung c) Seminar zur gewählten Vorlesung	**Kontakt-zeit** 5 SWS / 75h	**Selbst-studium** 225 h	**geplante Gruppengröße** 30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden gewinnen einen Einblick in einen selbst gewählten Bereich der Informatik. Die zu erwerbenden Kenntnisse in diesem Bereich können bis an den Stand der Forschung heranreichen. Durch ein Seminar zum gewählten Bereich erlernen die Studierenden, wie man sich neue Themen selbständig erarbeitet.

Softwaretechnik:

- Kenntnis der Phasen des Softwareentwicklungsprozesses
- Kenntnis gängiger Modellierungsmethoden und -notationen (UML)
- Kenntnis grundlegender Methoden zur Qualitätssicherung
- Verständnis der grundlegenden Probleme bei der Erstellung großer Softwaresysteme
- Fähigkeit zur kritischen Auseinandersetzung mit Originalliteratur im Bereich der Softwaretechnik

Grundlagen der Computergraphik:

- Kenntnis der Grundlagen der 2D und 3D Computergrafik
- Fähigkeit mit Hilfe von Werkzeugen, 3D-Modelle und Computeranimationen zu erstellen

Übersetzung und Analyse von Programmen:

- Kenntnis der Struktur von Compilern
- Kenntnis der gängigen Erzeugungsverfahren für die verschiedenen Phasen eines Compilers
- Verständnis der Übersetzung einer realen Programmiersprache (z.B. Java)
- Kenntnis verschiedener Analyseverfahren

Softskills:

- Erwerb von Fertigkeiten und Methoden beim Durcharbeiten der Vorlesungsinhalte
- Selbstständiges Arbeiten beim Lösen von Übungsaufgaben
- Argumentation und Präsentation eigener Ergebnisse in den Übungsgruppen.

Seminar zur gewählten Veranstaltung:

- Selbststudium von ergänzender Literatur
- Befähigung, sich neue Themen selbständig zu erarbeiten
- Übung von Präsentationstechniken
- Übung von Kommunikationsfähigkeiten in der Diskussion
- Übung im Abfassen wissenschaftlicher Aufsätze durch Ausarbeitung
- i.d.R Übung des Leseverständnisses englischsprachiger Literatur

3 Inhalte

Softwaretechnik:

- Einführung
- Die Phasen des Softwareentwicklungsprozesses Prozessmodelle

	• Objekt-orientierte Analyse und Entwurf (Grundlagen der Objektorientierung, Anwendungsfallanalyse, Struktur- und Verhaltensdiagramme, Systementwurf) • Entwurfsmuster • Teamarbeit, Konfigurationsmanagement • Qualitätssicherung, Testmethoden (Softwaremetriken, Testwerkzeuge) • Benutzerschnittstellen **Grundlagen der Computergraphik:** • Grundlagen: 2D Rastergrafik, Vektorgrafik • Bild und Videoformate • Grundlagen: 3D Computergrafik • Renderingpipeline • Beleuchtung • Transformationen • 3D Modellierungssprachen (z.B. X3D) • Grundlagen der Computeranimation **Übersetzung und Analyse von Programmen:** • Struktur eines Compilers • Lexikalische und syntaktische Analyse • Datenflussanalyse • Übersetzung von Java • Abstrakte Maschinen, insbesondere die JVM • Erkennen von Code-Klonen • Erkennen von Entwurfsmustern • Kenntnis der Struktur von Compilern
4	**Lehrformen** Vorlesungen, Übungen, Seminar
5	**Teilnahmevoraussetzungen** **Formal:** keine **Inhaltlich:** keine
6	**Prüfungsformen** mündliche Prüfung
7	**Voraussetzungen für die Vergabe von Kreditpunkten** ErfolgreicheTeilnahme an den Übungen und am Seminar, bestandende Modulprüfung
8	**Verwendung des Moduls** (in anderen Studiengängen)
9	**Stellenwert der Note für die Endnote** 10/120
10	**Modulbeauftragte/r und hauptamtlich Lehrende** Diehl
11	**Sonstige Informationen** Veranstaltung i.d.R. in deutscher Sprache, bei Bedarf jedoch auf Englisch Letztes Bearbeitungsdatum: 22.04.2014

Zu jedem vertiefenden Wahlpflichtmodul wird ein Projektpraktikum angeboten. Diese Praktika schließen sich an das jeweils für „Modul 10“ gewählte vertiefende Wahlpflichtmodul an:

Modul 12: Projektpraktikum

Kennnummer	Workload	Credits	Studiensemester	Häufigkeit des Angebots	Dauer
	300 h	10	3	jährlich	1 Semester

1	**Lehrveranstaltungen** Praktikum	**Kontaktzeit** 6 SWS / 90 h	**Selbststudium** 210 h	**geplante Gruppengröße** 30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden

- sind in der Lage, ingenieurmäßig Methoden und Techniken zur systematischen Entwicklung von Software-Systemen in der Praxis einzusetzen;
- können eine Anwendung analysieren, entwerfen und implementieren;
- können Lösungen des Moduls 10 einsetzen;
- können Methoden des Moduls 10 umsetzen;
- können Software-Entwicklung im Team organisieren (insbesondere bezüglich der Entwicklung einer arbeitsteiligen Vorgehensweise und der Implementierung von partiellen Erkenntnissen in den Gesamtprozess).

3 Inhalte

Selbstorganisierte Entwicklung eines Softwaresystems im Team, insbesondere aufbauend auf dem gewählten vertiefenden Wahlpflichtmodul

4 Lehrformen

Praktikum

5 Teilnahmevoraussetzungen

Formal: keine

Inhaltlich: Kenntnisse aus dem gewählten vertiefenden Wahlpflichtmodul

6 Prüfungsformen

Portfolio

7 Voraussetzungen für die Vergabe von Kreditpunkten

Erfolgreiche Übungsteilnahme, bestandene Modulprüfung

8 Verwendung des Moduls (in anderen Studiengängen)

9 Stellenwert der Note für die Endnote

10/120

10 Modulbeauftragte/r und hauptamtlich Lehrende

Modulbeauftragter: Müller,
Hauptamtlich Lehrende: alle Dozenten der Informatik

11 Sonstige Informationen

Letztes Bearbeitungsdatum: 22.04.2014

Modul 13: Didaktik des Informatikunterrichts

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
	210 h	7	4	jährlich	1Semester

		Kontaktzeit	Selbst-studium	geplante Gruppengröße
1	**Lehrveranstaltungen** a) Vorlesung Informatik-Didaktik b) Übung Informatik-Didaktik c) Seminar Informatik-Didaktik	5 SWS / 75 h	135 h	30 Studierende

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden

- kennen Möglichkeiten zur didaktischen Aufbereitung schulform-spezifischer Themenbereiche, sie können diese fundiert bewerten sowie eigene Unterrichtskonzepte entwickeln;
- können ihre bisher erworbenen allgemeinen Kenntnisse der Fachdidaktik der Informatik den besonderen Bedingungen der jeweiligen Schulart, insbesondere unter Beachtung altersspezifischer lernpsychologischer Voraussetzungen, zur Planung komplexerer Unterrichtsprojekte nutzen;
- sind zu einer anwendungsbezogenen Planung von Unterrichtseinheiten in der Lage;
- können Formen projektbezogener Leistungsbewertung und Evaluation geeignet einbeziehen.

Neben den inhaltlichen Themen werden Kompetenzen im Bereich Verstehen und Wiedergeben von Sachverhalten (komplexer Inhalte) erlangt. Auch eine Reflexion des eigenen Kommunikations- und Informationsvermögens ist Ziel dieser Veranstaltung. Eine Reihe von rhetorischen und kommunikativen Kompetenzen wie z.B. die Verwendung von Frage- und Argumentationstechniken, das verständliche formulieren von Inhalten und ein überzeugendes Auftreten werden erlernt.

3 Inhalte

- Vertiefende fachdidaktische und fachmethodische Themenbereiche der jeweiligen Schulart
- objektorientierte Programmierung im Unterricht, deklarative Programmierung im Unterricht
- Kommunikation in Rechnernetzen im Unterricht, Rechnerarchitektur im Unterricht
- formale Sprachen und Automaten im Unterricht
- Grenzen algorithmisch arbeitender Systeme im Unterricht
- Datenbanken
- Auswahl, Planung, Gestaltung, Wartung und Bewertung einfacher technischer Systeme der Informatik
- Informatische Aspekte des Projektunterrichts
- Lernpsychologische Grundlagen zur Gestaltung informatischen Anfangsunterrichts
- Planung komplexer Unterrichtseinheiten unter handlungsorientierten Kriterien zu informatischen Themenbereichen

4 Lehrformen

Vorlesungen, Übungen, Seminar

5 Teilnahmevoraussetzungen

Formal: keine
Inhaltlich: keine

6 Prüfungsformen

mündliche Prüfung

7 Voraussetzungen für die Vergabe von Kreditpunkten

- Bestehen der Modulprüfung sowie
- erfolgreicheTeilnahme an Übung und Seminar

8 Verwendung des Moduls (in anderen Studiengängen)

9 Stellenwert der Note für die Endnote

7/120

10 Modulbeauftragte/r und hauptamtlich Lehrende

Löhnertz

11 Sonstige Informationen

Letztes Bearbeitungsdatum: 22.04.2014

Obwohl die Masterarbeit entsprechend der Allgemeinen Prüfungsordnungen der Universität Trier nicht als Modul angesehen wird, ist der Vollständigkeit halber im Folgenden eine analoge Formulierung beigefügt:

Masterarbeit im M.Ed. Informatik (Gymnasium)

Kennnummer	Workload	Credits	Studiensemester	Häufigkeit des Angebots	Dauer
	600 h	20	4	jedes Semester	3 Monate

1	**Lehrveranstaltungen** keine	**Kontaktzeit**	**Selbststudium**	**geplante Gruppengröße** i.d.R. Einzelarbeit

2 Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden

- besitzen die Fähigkeit, wissenschaftliche Methoden und Kenntnisse des Faches eigenständig anzuwenden und zu erweitern,
- sind in der Lage, die Zusammenhänge des Faches zu überblicken,
- können eine umfangreiche schriftliche Arbeit unter Einhaltung einer Zeitvorgabe zielorientiert planen
- und sind in der Lage, diese Arbeit, mit Interpretation und Bewertung, in einem vorgegebenen Zeitraum zu erstellen.

Insbesondere wird auch die Schlüsselqualifikation der Organisationsfähigkeit gefördert.

3 Inhalte

In der Masterarbeit soll eine komplexe Problemstellung aus dem Gebiet der Informatik selbstständig unter Anwendung des Theorie- und Methodenwissens der Informatik bearbeitet und gemäß wissenschaftlicher Standards dokumentiert werden.

4 Lehrformen

Selbststudium

5 Teilnahmevoraussetzungen

keine

6 Prüfungsformen

schriftliche Arbeit

7 Voraussetzungen für die Vergabe von Kreditpunkten

8 Verwendung des Moduls (in anderen Studiengängen)

9 Stellenwert der Note für die Endnote

20/120

10 Modulbeauftragte/r und hauptamtlich Lehrende

Müller

11 Sonstige Informationen

Laut §15(2) der entsprechenden Prüfungsordnung gilt:

- Im Studium für das Lehramt an Realschulen und Gymnasien kann die Bachelorarbeit in einem der gewählten Fächer oder den Bildungswissenschaften angefertigt werden.
- Bei der Themenvergabe können fachdidaktische Aspekte und Bezüge zu anderen Fächern berücksichtigt werden.

Die Masterarbeit muss in einem anderen Fach als die Bachelorarbeit angefertigt werden.