Module Handbook
Master Computer Science

Faculty for Computer Science, Electrical Engineering and Mathematics
Paderborn University

Version: October 3, 2017

Contents

Chapter 1

Description of Programme Master Computer Science

The goal of the 4-semester Master in Computer Science program is to deepen the knowledge students acquired in a Bachelor's program by providing specialized knowledge in one or several fields of computer science. The Master's program requires that one out of five areas is selected as a focus area. The students must complete at least 3 modules (of 6 ECTS each) in this focus area and must write their thesis in the focus are. Students must choose at least one module from a different area. Master's students must also participate in a project group (20 ECTS). This is a well-established 2-semester type of course in which a team cooperates together on a relevant research topic. Other than that students are free to organize their studies.

1.1 Scheme for Module Descriptions

The module descriptions are consistently structured according to the following scheme:

Module name	<Name of the module>
Courses	<List of courses in the module (incl. page references)>
Module type	<Mandatory or elective module>
Module advisor	<Person responsible for the module>
Language	<Teaching language for the module>
Implementation method	<Lectures, exercises, labs, seminars>
Contact hours (per week & semester)	<Total weekly contact hours for the module >
Credits ECTS	<Total credit points for the module>
Work load	<Total workload (ECTS), split into contact times and times for self-study. One weekly contact hour amounts to 60 minutes.>
Learning objectives	<Short summary of the main learning objectives of the module>
Assessment modalities	<Written exam, oral exam or other examination modalities>
Remarks	

1.2 Scheme for Module Descriptions

The course descriptions are consistently structured according to the following scheme:

Course	<Title of the course>
Coordination	<Lecturer>
Teaching Unit	<Institute offering the course>
Language	<Teaching language>
Type	<Contact times in weekly hours and organisational form (lecture, exerices, seminar, lab, project)>
Work load	<Total workload (ECTS), split into contact times and times for self-study.>
Course homepage	<Web page of the course, the lecturer or the institute>
Intended semester	<Winter or summer semester>
Modules using this course	<Modules containing this course (incl. page references)>

Short description
<Short summary of the contents and learning objectives>
Content
<Main contents of the course>
Learning objectives, competences
<Gained knowledge, skills and competencies>
Implementation
<Social forms and didactical-methodical practices used in the course>
Recommended knowledge
<The prerequisites are recommendations rather than requirements to be checked>
Assessment modalitites
<Assessment forms (e.g., written exam, oral exam, presentation, homework, project, lab attestation)>
Teaching material, literature
Comments

1.3 List of organization forms

The following organization forms are used in this program:

Final thesis

Lab work

Lecture with practical assignments

Lectures, seminars, projects

Seminar

1.4 List of examination forms

The following examination forms are used in this program:

Exam in general studies

Final thesis

Oral exam

Oral exam with course achievement Oral exam with course achievement, e.g. homework

Partial modul exam (100 % of module grade)

Presentation (45-60 minutes) and essay A seminar presentation should have a duration of 45 minutes and be based on a seminar paper.

Chapter 2

Focus Areas

2.1 Algorithm Design

Focus Area	Algorithm Design
Koordination	Prof. Dr. rer. nat. Johannes Blömer Codes and Cryptography Computer Science
Included Modules	• Advanced Algorithms (S. 17) • Advanced Complexity Theory (S. 21) • Advanced Distributed Algorithms and Data Structures (S. 25) • Algorithmic Game Theory (S. 29) • Algorithms for Highly Complex Virtual Scenes (S. 31) • Clustering Algorithms (S. 41) • Computational Geometry (S. 45) • Foundations of Cryptography (S. 55) • Linear and Integer Optimization (S. 74) • Public-Key Cryptography (S. 101) • Routing and Data Management in Networks (S. 105)
Description	
In this focus area students can concentrate on studying the • important techniques for the design of efficient algorithms • application areas for the design of efficient algorithms, i.e. computer graphics, networks, big data, ... • limits for the design of efficient algorithms, i.e. complexity theory • constructive use of complexity theory in cryptography and security • connection between efficient algorithms and verification and software design	

2.2 Computer Systems

Focus Area	Computer Systems
Koordination	Prof. Dr. Marco Platzner Computer Engineering Computer Science
Included Modules	• Adaptive Hardware and Systems (S. 15) • Advanced Computer Architecture (S. 23) • Algorithms for Synthesis and Optimization of Integrated Circuits (S. 33) • Architectures of Parallel Computer Systems (S. 35) • Empiric performance evaluation (S. 53) • Hardware/Software Codesign (S. 64) • High-Performance Computing (S. 66) • Intelligence in Embedded Systems (S. 68) • Reconfigurable Computing (S. 103) • VLSI Testing (S. 121)
Description	
The focus area "Computer systems" goes into technical depths of various aspects of modern computer systems. Main topics are the analysis and evaluation of computer architectures, systematic methods for design and optimisation of computer systems, in particular the interplay of hardware and software, and programming models and methods for parallel and specialised computer architectures, which are increasingly gaining importance.	

2.3 Intelligence and Data

Focus Area	Intelligence and Data
Koordination	Prof. Dr. Eyke Hüllermeier Intelligent Systems Computer Science
Included Modules	• Adaptive Hardware and Systems (S. 15) • Clustering Algorithms (S. 41) • Intelligence in Embedded Systems (S. 68) • Interactive Data Visualization (S. 70) • Logic Programming for Artificial Intelligence (S. 78) • Logic and Automated Reasoning (S. 76) • Machine Learning I (S. 80) • Machine Learning II (S. 82) • Planning and Heuristic Search (S. 97)

Description

Intelligent systems are computer systems the behavior of which is controlled by methods and algorithms from artificial intelligence (AI). Systems of that kind are becoming increasingly important, not only on a scientific level but also in a social context: Autonomous or semi-autonomous systems such as service robots, self-driving cars or medical diagnosis systems will have a deep impact on our future private and professional life. In addition to methodological advances and improved hardware, the "data explosion" can be seen as a main driving factor for the rapid development of AI-systems during the last decade: Thanks to the availability of massive amounts of data or sensory feedback from their environment, intelligent systems are able to automatically improve their behavior through adaptation and learning.
This focus area covers important aspects of intelligent systems design and conveys corresponding theoretical and methodological foundations. This includes lectures on machine learning and data analysis, data management, computer graphical and visual data analysis, as well as swarm intelligence and robotics.

2.4 Networks and Communication

Focus Area	Networks and Communication
Koordination	Prof. Dr. rer. nat. Holger Karl Computer Networks Computer Science
Included Modules	• Advanced Distributed Algorithms and Data Structures (S. 25) • Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies (S. 37) • Empiric performance evaluation (S. 53) • Future Internet (S. 60) • Mobile Communication (S. 86) • Network Simulation (S. 93) • Networked Embedded Systems (S. 95) • Routing and Data Management in Networks (S. 105) • Vehicular Networking (S. 119)
Description	
The focus area "Networks and Communication" teaches architectures, methods and systems of modern communication technology. To this end, we investigate methods of various abstractions levels, starting from the lowest level physical transmissions up to and including application design in distributed environments. Different types of systems are considered, ranging from conventional mobile communication over ad hoc networks and vehicular communication systems to networking in data centers and architectures for the future Internet at large. In doing so, we strive to build the bridge to aspects of distributed systems design. Questions on architecture design and options for protocol designs are complemented by the evaluation of such systems. To answer those questions, we introduce experimental and statistical performance evaluation techniques.	

2.5 Software Engineering

Focus Area	Software Engineering
Koordination	Prof. Dr. Gregor Engels Database and Information Systems Computer Science
Included Modules	• Advanced Compiler Construction (S. 19) • Advanced Software Engineering: Methods, Architectures, Industrial Applications (S. 27) • Build It, Break It, Fix It (S. 39) • Compiler Construction (S. 43) • Deductive Verification (S. 49) • Designing code analyses for large-scale software systems (S. 51) • Empiric performance evaluation (S. 53) • Fundamentals of Model-Driven Engineering (S. 57) • High-Performance Computing (S. 66) • Contextual Informatics (S. 47) • Language-Based Security (S. 72) • Logic Programming for Artificial Intelligence (S. 78) • Model Checking (S. 88) • Model-Based Interface Development (S. 90) • Software Analysis (S. 111) • Software Quality Assurance (S. 113) • Type Systems for Correctness and Security (S. 115) • Usability Engineering Practice (S. 117)

Description
In this focus area, students can concentrate on studying concepts, languages, methods, techniques, and tools for the systematic development of software systems. These comprise • constructive techniques for developing functional and non-functional aspects of a system, • formal and informal analytical techniques to ensure high quality of a system, • systematic techniques to enable situation-specific process models

Chapter 3

Modules

3.1 Elective Module: Adaptive Hardware and Systems

Module name	Adaptive Hardware and Systems / Adaptive Hardware and Systems
Workload	180 h
Credits	6 LP
Semester	• Adaptive Hardware and Systems : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Adaptive Hardware and Systems: Lecture (45h / 105h / EN / SS / 25) Adaptive Hardware and Systems: Tutorial (30h / 0h / EN / SS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Adaptive Hardware and Systems: Basic knowledge in computer architectures and programming languages	
Content	
Adaptive Hardware and Systems: Adaptation reflects the capability of a system to maintain or improve its performance in the context of internal or external changes, changes in the operational environment, incidental or intentional interference, or trade-offs between performance requirements and available resources. This lecture focuses on adaptive hardware systems. After a short introduction to analog and digital reconfigurable hardware the lecture focuses on algorithms from the Computational Intelligence (CI) domain for the implementation of the adaptation and optimization mechanisms. The labs include the implementation of learning strategies for run-time adaptable hardware signal classifiers, evolution of adaptable processor caches, and optimization of chip designs.	
Learning objectives	
Implementation method	
Adaptive Hardware and Systems: • Lecture with projected slides and notes on blackboard • Interactive assignments in lecture room • Self-study and discussion of scientific publications • Programming exercises • Solving of parallel optimization challenges on the PC2 compute cluster	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes)	

The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Dr. Paul Kaufmann
Learning material, literature
Adaptive Hardware and Systems: Folien; ausgewählte Fachartikel; Lehrbücher • Weicker, Karsten, "Evolutionäre Algorithmen", Springer, 2007. ISBN 978-3-8351-9203-4 • Kruse et al.: "Computational Intelligence - A Methodological Introduction", Springer, 2013. ISBN 978-1-4471-5012-1 • Kruse et al.: "Computational Intelligence [DE]", Vieweg+Teubner-Verlag, Wiesbaden, 2011. ISBN 978-3-8348-1275-9 • Wang et al.: "Electronic Design Automation", Morgan Kaufmann, 2009. ISBN: 0-1237-4364-8
Remarks
none

3.2 Elective Module: Advanced Algorithms

Module name	Advanced Algorithms / Advanced Algorithms
Workload	180 h
Credits	6 LP
Semester	• Advanced Algorithms : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Advanced Algorithms: Lecture (45h / 105h / EN / WS / 50) Advanced Algorithms: Tutorial (30h / 0h / EN / WS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Advanced Algorithms: Willingness and ability to learn the creative process of algorithm design and efficiency analysis using mathematical methods. Basic Knowledge of some basic algorithms and data structures and their analyses is assumed.	
Content	
Advanced Algorithms: This course presents advanced algorithms and algorithmic paradigms for basic problems. In particular, methods such as randomization and derandomization, as well as the concepts of approximation and online algorithms, are presented using important algorithmic problems. In all cases, proof of correctness and run-time analyzes are carried out.	
Learning objectives	
Students apply advanced algorithmic design methods such as randomization, approximation, and online algorithms to new problems and analyze them using combinatorial and probabilistic methods.	
Implementation method	
Advanced Algorithms: • Lecture with beamer and blackboard. • Exercises in small groups. • expected student activities: active participation in exercises, homework. • Exercise sheets, solutions are presented and discussed in tutorials. • In exercises and homework, design and analysis of algorithms are practiced on selected examples.	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	

Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Friedhelm Meyer auf der Heide
Learning material, literature
Advanced Algorithms: Standardlehrbücher, Foliensatz der Vorlesung, Übungsblätter
Remarks
none

3.3 Elective Module: Advanced Compiler Construction

Module name	Advanced Compiler Construction / Advanced Compiler Construction
Workload	180 h
Credits	6 LP
Semester	• Advanced Compiler Construction : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Advanced Compiler Construction: Lecture (45h / 105h / EN / WS / 25) Advanced Compiler Construction: Tutorial (30h / 0h / EN / WS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Advanced Compiler Construction:	
Content	
Advanced Compiler Construction: The goal of this course is to present advanced optimization and security techniques in compilers.	
Learning objectives	
Implementation method	
Advanced Compiler Construction:	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	
Course achievement / qualifying participation	
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.	
Prerequisites for participation in module exam	
Passing of course achievement.	

Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof.Dr. Stefan Brunthaler
Learning material, literature
Advanced Compiler Construction: none
Remarks
none

3.4 Elective Module: Advanced Complexity Theory

Module name	Advanced Complexity Theory / Advanced Complexity Theory
Workload	180 h
Credits	6 LP
Semester	• Advanced Complexity Theory : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Advanced Complexity Theory: Lecture (45h / 105h / EN / WS oder SS / 25) Advanced Complexity Theory: Tutorial (30h / 0h / EN / WS oder SS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Advanced Complexity Theory: Basic knowledge about complexity theory (e.g., Turing machines, NP-completeness)	
Content	
Advanced Complexity Theory: Complexity Theory deals with determining the amount of resources (e.g. runtime, memory consumption) necessary and sufficient for solving a given algorithmic problem (e.g. Travelling Salesperson Problem (TSP)) on a given machine model (e.g. Turing machine). One approach is to define complexity classes like P, NP, PSPACE, in order to classify problem complexity by means of completeness in such classes, like the famous class of NP-complete problems. This gives conditional results like "'If NP is not equal P, then TSP is not solvable in polynomial time."' This branch of Complexity Theory is often referred to as Structural Complexity Theory. In contrast, proving explicit lower bounds for given problems is the topic of the so-called Concrete Complexity Theory. As nobody is currently able to prove superlinear time bounds for explicitly defined problems on general computation models like Turing machines, one considers somewhat restricted models like 1-tape Turing machines, monotone Boolean circuits, Boolean circuits with bounded depth, algebraic computation models, and several kinds of parallel computation models. This lecture surveys approaches to prove such lower bound on various such models.	
Learning objectives	
The students get to know fundamental techniques in the area of complexity theory. They can decide in which complexity class the storage space and the runtime requirements of algorithmic problems can be classified. They can classify new problems into complexity classes.	
Implementation method	
Advanced Complexity Theory: • Lecture with beamer and blackboard • Practice in small groups	

• Expected activities of the students:contributions to presence exercises, homework
Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Friedhelm Meyer auf der Heide
Learning material, literature
Advanced Complexity Theory: C.H. Papadimitiriou, Computational Complexity, Addison-Wesley, S. Arora, B. Barak, Computational Complexity - A Modern Approach, Cambridge University Press; Foliensatz der Vorlesung, Übungsblätter
Remarks
none

3.5 Elective Module: Advanced Computer Architecture

Module name	Advanced Computer Architecture / Advanced Computer Architecture
Workload	180 h
Credits	6 LP
Semester	• Advanced Computer Architecture : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Advanced Computer Architecture: Lecture (45h / 105h / EN / WS / 50) Advanced Computer Architecture: Tutorial (30h / 0h / EN / WS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Advanced Computer Architecture: Basic knowledge in computer architecture.	
Content	
Advanced Computer Architecture: The course teaches concepts and methods used in modern processor architecture to exploit the available parallelism at the levels of instructions, data and threads.	
Learning objectives	
After attending the course, the students • are able to explain principles of modern memory hierarchies, • to analyze different levels of parallelism, • to assess the suitability of different architectural concepts and thus • to evaluate modern developments in computer architecture.	
Implementation method	
Advanced Computer Architecture: • Lecture with projector and board • Interactive exercises in the lecture room item Computer-based exercises with simulation tools • Analysis of case studies	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	

Course achievement / qualifying participation
none
Prerequisites for participation in module exam
none
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Marco Platzner
Learning material, literature
Advanced Computer Architecture: • Vorlesungsfolien und Übungsblaetter • Aufgabenblätter und technische Dokumentation für die Rechnerübungen • Hennessey, Patterson: Computer Architecture: A Quantitative Approach (5th edition), Morgan Kaufmann, 2012. • Aktuelle Hinweise auf alternative und ergänzende Literatur, sowie Lehrmaterialien auf der Webseite und in den Vorlesungsfolien
Remarks
none

3.6 Elective Module: Advanced Distributed Algorithms and Data Structures

Module name	Advanced Distributed Algorithms and Data Structures / Advanced Distributed Algorithms and Data Structures
Workload	180 h
Credits	6 LP
Semester	• Advanced Distributed Algorithms and Data Structures : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Advanced Distributed Algorithms and Data Structures: Lecture (45h / 105h / EN / WS / 30) Advanced Distributed Algorithms and Data Structures: Tutorial (30h / 0h / EN / WS / 30)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Advanced Distributed Algorithms and Data Structures: Algorithms and data structures, distributed algorithms and data structures	
Content	
Advanced Distributed Algorithms and Data Structures: The lecture presents advanced methods that allow the design of highly scalable distributed algorithms and data structures. It splits into separate areas that are currently very relevant in the area of distributed systems. This includes locality-preserving systems, robust information systems, and programmable matter.	
Learning objectives	
Students get to know advanced methods and algorithms for currently very relevant distributed systems. They are able to adapt algorithms to new situations and to determine their complexity. They can implement basic distributed algorithms.	
Implementation method	
Advanced Distributed Algorithms and Data Structures: Lecture with tutorials and software project	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	

Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. rer. nat. Christian Scheideler
Learning material, literature
Advanced Distributed Algorithms and Data Structures: Skript
Remarks
none

3.7 Elective Module: Advanced Software Engineering: Methods, Architectures, Industrial Applications

Module name	Advanced Software Engineering: Methods, Architectures, Industrial Applications / Advanced Software Engineering: Methods, Architectures, Industrial Applications
Workload	180 h
Credits	6 LP
Semester	• Advanced Software Engineering: Methods, Architectures, Industrial Applications : arbitrary

Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)

Advanced Software Engineering: Methods, Architectures, Industrial Applications: Lecture (45h / 105h / EN / WS / 92)
Advanced Software Engineering: Methods, Architectures, Industrial Applications: Tutorial (30h / 0h / EN / WS / 30)

Choices in Module

none

Course prerequiites

none

Recommended proficiencies

Advanced Software Engineering: Methods, Architectures, Industrial Applications: Model-based software development

Content

Advanced Software Engineering: Methods, Architectures, Industrial Applications: The course will cover advanced concepts and methods of Software Engineering. This comprises approaches for the development of architectures, modern software engineering methods as well as project management techniques. In particular, the following topics will be presented and discussed: - Architecture (Service-oriented Architecture, Microservices, Web Services, Mobile Architectures, Architectural Frameworks, Quasar Enterprise) - Methods (Agile Software Development Methods like Scrum, Situational Method Engineering, Requirements Engineering, DevOps) - Project Management (Effort Estimation, Economics of Software Projects) The course is enriched by talks of experienced industrial experts, who report and discuss the use of modern software engineering techniques in practice.

Learning objectives

The students are able to explain and deploy concepts of modern software engineering. They know and are able to deplay essential concepts to develop and maintain architectures of software systems, to customize and use software engineering methods in a specific project, and to do project management tasks like effort estimation.

Implementation method
Advanced Software Engineering: Methods, Architectures, Industrial Applications: Partially slides and partially board writing. All essential concepts and techniques will be repeatedly applied in examples during the tutorial. In a lab part, the techniques will be employed.
Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
none
Prerequisites for participation in module exam
none
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Gregor Engels
Learning material, literature
Advanced Software Engineering: Methods, Architectures, Industrial Applications: Vorlesungsfolien, Übungsaufgaben
Remarks
none

3.8 Elective Module: Algorithmic Game Theory

Module name	Algorithmic Game Theory / Algorithmic Game Theory
Workload	180 h
Credits	6 LP
Semester	• Algorithmic Game Theory : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Algorithmic Game Theory : Lecture (45h / 105h / EN / WS oder SS / 25) Algorithmic Game Theory : Tutorial (30h / 0h / EN / WS oder SS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Algorithmic Game Theory : Algorithms and data structures, linear algebra, probability theory	
Content	
Algorithmic Game Theory : This Lecture introduces the most important basic concepts and methods from the field of algorithmic game theory. It considers various topics from non-cooperative game theory as well as from social choice,	
Learning objectives	
Students get to know the important concepts of non cooperative and cooperative game theory, in particular standard game models and solution concepts. They understand a variety of algorithmic techniques and complexity results for computing game-theoretic solution concepts. They apply solution concepts, algorithms, and complexity results to unseen games that are variants of known examples and understand the state of the art in some areas of algorithmic research, including new developments and open problems.	
Implementation method	
Algorithmic Game Theory : Eine Mischung aus Folien und Tafelanschrieb. Alle wichtigen Konzepte und Techniken werden in Übungen anhand von Beispielen weiter vertieft.	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	

Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Jun.-Prof. Dr. Alexander Skopalik
Learning material, literature
Algorithmic Game Theory: Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. 2007. Algorithmic Game Theory. Cambridge University Press, New York, NY, USA. Slides Exercises
Remarks
none

3.9 Elective Module: Algorithms for Highly Complex Virtual Scenes

Module name	Algorithms for Highly Complex Virtual Scenes / Algorithms for Highly Complex Virtual Scenes
Workload	180 h
Credits	6 LP
Semester	• Algorithmen für hochkomplexe virtuelle Szenen : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Algorithms for Highly Complex Virtual Scenes: Lecture (45h / 105h / EN / WS / 40) Algorithms for Highly Complex Virtual Scenes: Tutorial (30h / 0h / EN / WS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Algorithms for Highly Complex Virtual Scenes: Willingness and ability to learn the creative process of algorithm design and efficiency analysis using mathematical methods. Basic Knowledge of some basic algorithms and data structures and their analyses is assumed.	
Content	
Algorithms for Highly Complex Virtual Scenes: Walkthrough systems allow viewing and walking through a virtual 3D scene and finds application in architecture programs, simulations or games. The efficiency of real-time rendering algorithms is crucial for a smooth and fast visualization of the virtual 3D scene in a walkthrough system. There are different algorithmic approaches to reduce highly complex 3D geometric data and to achieve a rendering of the scene in real time. The lecture introduces algorithmic approaches in the areas of visibility culling, simplification, level of detail, image-based rendering and further approaches.	
Learning objectives	
The students can apply fundamental techniques in the area of real time rendering of virtual 3D scenes. They can decide in which virtual 3D scene which algorithm is most appropriate. They can adapt algorithms to a new situation.	
Implementation method	
Algorithms for Highly Complex Virtual Scenes: • Lecture with beamer and blackboard • Practice in small groups • Expected activities of the students: Collaboration in presence exercises Homework	

- exercise sheets, sample solutions are presented in central exercises
- In exercises and homework sheets and teh analysis of algorithms of selected examples are practiced.

Assessment modalities (duration)

Oral exam (ca. 40 Minutes)
The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.

Partial module exams

none

Course achievement / qualifying participation

Studienleistung: written exercises
The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.

Prerequisites for participation in module exam

Passing of course achievement.

Prerequisites for obtaining credits

Passing of module exam.

Weight for overall grade

The module is weighted with 6 credits.

Person responsible for the module

Dr. Matthias Fischer

Learning material, literature

Algorithmen für hochkomplexe virtuelle Szenen: Standardlehrbücher, Foliensatz der Vorlesung, Übungsblätter
- Real-Time Rendering; Tomas Akenine-Möller, Eric Haines; AK Peters, 2002.
- Level of Detail for 3D Graphics; David Luebke, Martin Reddy, Jonathan D. Cohen; Morgan Kaufmann Publishers, 2002.

Remarks

none

3.10 Elective Module: Algorithms for Synthesis and Optimization of Integrated Circuits

Module name	Algorithms for Synthesis and Optimization of Integrated Circuits / Algorithms for Synthesis and Optimization of Integrated Circuits
Workload	180 h
Credits	6 LP
Semester	• Algorithms for Synthesis and Optimization of Integrated Circuits : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Algorithms for Synthesis and Optimization of Integrated Circuits: Lecture (45h / 105h / EN / SS / 30) Algorithms for Synthesis and Optimization of Integrated Circuits: Tutorial (30h / 0h / EN / SS / 30)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Algorithms for Synthesis and Optimization of Integrated Circuits: Knowledge of "Digital Design" is beneficial.	
Content	
Algorithms for Synthesis and Optimization of Integrated Circuits: The course provides the most remarkable features of digital synthesis, and explains the details of transforming hardware description languages into circuit descriptions. Besides, the major techniques for logic optimization are discussed, and then the efficient use of current design tools are exercised in practical sessions.	
Learning objectives	
After attending the course, the students are able to • select among the available optimisation methods in design of digital circuits, • identify major problems in design of integrated circuits and recognize circuit design tradeoffs • examine current digital design tools and methods	
Implementation method	
Algorithms for Synthesis and Optimization of Integrated Circuits: • Lecture with projector and board • Interactive exercises in the lecture room • Computer-based exercises with hardware synthesis tools	

Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
none
Prerequisites for participation in module exam
none
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Dr. Hassan Ghasemzadeh Mohammadi
Learning material, literature
Algorithms for Synthesis and Optimization of Integrated Circuits: • Vorlesungsfolien und Übungsblätter • Aufgabenblätter und technische Dokumentation für die Rechnerübungen • Micheli, Giovanni De. Synthesis and optimization of digital circuits. McGraw-Hill Higher Education, 1994. • Aktuelle Hinweise auf alternative und ergänzende Literatur, sowie Lehrmaterialien auf der Webseite und in den Vorlesungsfolien
Remarks
none

3.11 Elective Module: Architectures of Parallel Computer Systems

Module name	Architectures of Parallel Computer Systems / Architectures of Parallel Computer Systems
Workload	180 h
Credits	6 LP
Semester	• Architektur paralleler Rechnersysteme : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Architecture of Parallel Computer Systems: Lecture (45h / 105h / DE / SS / 20) Architecture of Parallel Computer Systems: Tutorial (30h / 0h / DE / SS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Architecture of Parallel Computer Systems: Principles of computer architectures	
Content	
Architecture of Parallel Computer Systems: The lecture considers computer architectures of actual parallel computer systems and the usage of this systems. The focus of the lecture is on high-performance computers (supercomputers).	
Learning objectives	
wird nachgetragen	
Implementation method	
Architecture of Parallel Computer Systems: Presentation of slides. Exercises on available high performance computers to practise the usage of the systems and deepen the knowledge of the lecture.	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	
Course achievement / qualifying participation	

Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Dr. Jens Simon
Learning material, literature
Architektur paralleler Rechnersysteme: Foliensatz
Remarks
none

3.12 Elective Module: Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies

Module name	Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies / Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies
Workload	180 h
Credits	6 LP
Semester	• Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies: Lecture (45h / 105h / EN / SS / 30) Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies: Tutorial (30h / 0h / EN / SS / 30)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies: Foundational knowledge of cryptography is recommended, but not mandatory.	
Content	
Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies: This course covers techniques for the construction of decentralized digital currencies like, e.g., Bitcoin, an understanding of the security properties achievable and achieved by these, and cryptographic tools for the protection of user privacy in such applications.	
Learning objectives	
Students know the basic concepts underlying modern distributed cryptographic currencies and can explain and apply them. This includes cryptographic hash functions, digital signatures, and blockchains. They know the necessary and sufficient security requirements for these building blocks and can assess the extent to which a given protocol or algorithm fulfills these. They know the terms pseudonymity and anonymity and their exact meaning and distinction, are able to investigate whether a given application possesses these features, know advanced cryptographic techniques to achieve strong anonymity, such as ring signatures, and know how these techniques are used in modern applications to protect the privacy of users.	
Implementation method	
Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies:	

- Lectures with slides and blackboard notes
- Exercises and presentation of solutions by participants

Assessment modalities (duration)

Oral exam (ca. 40 Minutes)
The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.

Partial module exams

none

Course achievement / qualifying participation

Studienleistung: written exercises
The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.

Prerequisites for participation in module exam

Passing of course achievement.

Prerequisites for obtaining credits

Passing of module exam.

Weight for overall grade

The module is weighted with 6 credits.

Person responsible for the module

Prof. Dr., Tibor Jager

Learning material, literature

Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies:
- Vorlesungsfolien
- Tafelanschrieb
- Originalliteratur

Remarks

none

3.13 Elective Module: Build It, Break It, Fix It

Module name	Build It, Break It, Fix It / Build It, Break It, Fix It
Workload	180 h
Credits	6 LP
Semester	• Build It, Break It, Fix It : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Build It, Break It, Fix It: Lecture (45h / 105h / EN / SS / 20) Build It, Break It, Fix It: Tutorial (30h / 0h / EN / SS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Build It, Break It, Fix It: A mature understanding of the Java programming language and software security requirements. Secure software development practices and experience in the field of software exploitation and vulnerability discovery will be helpful.	
Content	
Build It, Break It, Fix It: This course aims at teaching basic principles of secure software development in a very practical fashion. It is based on the "Break It, Build It, Fix It" security contest by Ruef et al. [1]. The contest is separated into three phases that test the applicant's skills in the fields of building, breaking and fixing software products. In the "Build It" phase, students will be asked to gather in teams and develop small software projects based on a formal specification, also including security requirements. In the "Break It" phase, the developed software will be exchanged between development teams to break the implementation, i.e., find and exploit security vulnerabilities in code of other teams. Afterward, in the "Fix It" phase, teams will get the chance to fix found vulnerabilities and, hence, render their software product more secure. The course will contain a theoretical part in which basic strategies of secure software development and vulnerability discovery are presented. Furthermore, specific vulnerability classes and examples of their exploitation will be presented as stimulus at the beginning of the "Break It" phase. Nevertheless, the course is generally of a very practical nature and since securing a software product, as well as breaking it, demands a wide variety of skills and creativity, a quite high amount of motivation and self-organization is required.	
Learning objectives	
After having attended this course, students will have... -gained knowledge and experience in the field of secure software development -gained knowledge and experience in the field of software exploitation as well as vulnerability discovery -learned common real	

world software vulnerabilities and ways of exploiting them
Implementation method
Build It, Break It, Fix It:
Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: Practical work Qualifizierte Teilnahme: Practical work with subsequent discussion The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Eric Bodden
Learning material, literature
Build It, Break It, Fix It: Das Kursmaterial wird auf der KoaLA Seite des Kurses angeboten werden.
Remarks
none

3.14 Elective Module: Clustering Algorithms

Module name	Clustering Algorithms / Clustering Algorithms
Workload	180 h
Credits	6 LP
Semester	• Clustering Algorithmen : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Clustering Algorithms: Lecture (45h / 105h / EN / WS oder SS / 25) Clustering Algorithms: Tutorial (30h / 0h / EN / WS oder SS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Clustering Algorithms: Algorithms and data structures, linear algebra, probability theory	
Content	
Clustering Algorithms: We study an important tool to analyze data: clustering. Clustering is the process of dividing data into useful or sensible groups. A sensible division should resemble the data's natural structure. Sometimes the goal is that each cluster should contain as many items of a similar kind as possible (for example in data compression). Clustering is a very natural way to analyze and structure data. Especially in natural sciences we are working with data whose structure is unknown to us. An example is the human DNA, that humankind is trying to decode. Clustering can be a very powerful tool in such cases. We re view standard techniques for clustering like Lloyd's algorithm and agglomerative clustering. We also look at extensions and generalizations of clusterings like estimating model parameters.	
Learning objectives	
Students know the main clustering algorithms. Depending on the problem at hand they can decide which algorithm is most appropriate. They can adapt algorithms to new situations. Students know various methods for modeling clustering problems. They can apply and adapt the methods. Depending on the modeling, they can choose appropriate algorithms and assess their quality.	
Implementation method	
Clustering Algorithms: Lectures, exercises, reading groups, short presentations	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	

Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. rer. nat. Johannes Blömer
Learning material, literature
Clustering Algorithmen: David J.C MacKay, Information Theory, Inference and Learning Algorithms Christopher M. Bishop, Pattern Recognition and Machine Learning Folien der Vorlesung
Remarks
none

3.15 Elective Module: Compiler Construction

Module name	Compiler Construction / Compiler Construction
Workload	180 h
Credits	6 LP
Semester	• Compilerbau : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Compiler Construction: Lecture (45h / 105h / EN / WS / 25) Compiler Construction: Tutorial (30h / 0h / EN / WS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Compiler Construction: An undergraduate course on compilers focusing on basic concepts of frontends (e.g., language theory, grammars, lexing, parsing). Conceptually builds on the course “Programming Languages and Compilers”.	
Content	
Compiler Construction: While the predecessor lecture ”Programming Languages and Compilers” (PLaC, Bachelor lecture) focuses on the frontend part of a compiler, this lecture focuses on the compiler backend. More precisely, we will discuss a variety of different optimizations, such as instruction scheduling, instruction selection, and register allocation. To complement the theoretical foundations of this course, we will study the concrete implementation in the LLVM compiler framework.	
Learning objectives	
Implementation method	
Compiler Construction:	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	
Course achievement / qualifying participation	
Studienleistung: written exercises	

The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof.Dr. Stefan Brunthaler
Learning material, literature
Compilerbau: • Appel: Modern Compiler Construction in ML. • Cooper, Torczon: Engineering a Compiler.
Remarks
none

3.16 Elective Module: Computational Geometry

Module name	Computational Geometry / Computational Geometry
Workload	180 h
Credits	6 LP
Semester	• Algorithmische Geometrie : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Computational Geometry: Lecture (45h / 105h / EN / SS / 40) Computational Geometry: Tutorial (30h / 0h / EN / SS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Computational Geometry: Willingness and ability to learn the creative process of algorithm design and efficiency analysis using mathematical methods. Basic Knowledge of some basic algorithms and data structures and their analyses is assumed.	
Content	
Computational Geometry: The lecture deals with algorithms and data structures in the area of computational geometry. The basic elements and input are geometric data (points, lines, circles, polygons, volumes). The problems are formulated geometrically and the task is to find an algorithmic solution using special geometric data structures. The algorithms are theoretically analyzed. For this purpose, runtime and space is analyzed and correctness of the algorithms is proved.	
Learning objectives	
The students can apply fundamental techniques in the area of computational geometry. They can decide for which geometric problem which algorithm is most appropriate. They can adapt algorithms to a new situation.	
Implementation method	
Computational Geometry: • Lecture with beamer and blackboard • Practice in small groups • Expected activities of the students: Collaboration in presence exercises Homework • exercise sheets, sample solutions are presented in central exercises • In exercises and homework sheets and teh analysis of algorithms of selected examples are practiced.	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks	

of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Dr. Matthias Fischer
Learning material, literature
Algorithmische Geometrie: Standardlehrbücher, Foliensatz der Vorlesung, Übungsblätter • Computational Geometry: Algorithms and Applications Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars, Springer-Verlag, 2008 • Algorithmische Geometrie Rolf Klein, Springer-Verlag, 2005 • Lectures on Discrete Geomtetry Jiri Matousek, Springer-Verlag, 2002.
Remarks
none

3.17 Elective Module: Contextual Informatics

Module name	Contextual Informatics / Contextual Informatics
Workload	180 h
Credits	6 LP
Semester	• Kontextuelle Informatik : arbitrary

Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)

Contextual Informatics: Lecture (30h / 105h / DE / WS / 40)
Contextual Informatics: Tutorial (45h / 0h / DE / WS / 20)

Choices in Module

none

Course prerequiites

none

Recommended proficiencies

Contextual Informatics: Proficiency of a Bachelor in Computer Science or a comparable degree

Content

Contextual Informatics: Computer scientists and programmers develop products based on symbolic descriptions (programs, specifications, documentation etc.) by modeling a specific domain. A number of different questions arise when such products are being developed: How can the developers adequately model the data that are to be processed as well as the processes that are to be implemented? Which consequences result from the possibility to develop interactive systems? When using the software, which role will the users play, which the software? Which underlying conditions of the usage context need to be considered?
The lecture "Contextual Informatics" discusses the theoretical and conceptual foundations of computer science relevant to the further considerations. Special attention will be paid to differentiate between technical concepts and the sphere of usage. Against this background, theories of interactive systems will be explored in order to examine which role digital media play with respect to processes of the mind. When developing computer systems, relevant data and processes need to be anticipated to a certain degree and modeled as formal systems. This raises issues like the question under which conditions such a formal description can be made in an adequate way and with which consequences regarding the reliability and responsible use of computer systems in a given domain.

Learning objectives

Students will learn to examine the role of interactive systems based on theories. They will learn to distinguish between technical and non-technical problems and how to relate these to each other. They will be enabled to assess current technological trends and computer systems as well as the potentials of innovation in the field of digital media.

Implementation method

Contextual Informatics: The content of the lecture will be delivered as talks with interactions from the students through questions and discussion. The tutorials will further explore the topics based on scientific literature in combination with short presentations by the students.
Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Dr. Harald Selke
Learning material, literature
Kontextuelle Informatik: Vorlesungsfolien Wardrip-Fruin, N.; Montfort, N. (eds.): The New Media Reader. Cambridge, Ma.: MIT Press, 2003. Quinn, M. J.: Ethics for the Information Age. Boston, Ma.: Pearson, 7th edition, 2016. Begleitende wissenschaftliche Literatur wird in der Vorlesung vorgestellt.
Remarks
none

3.18 Elective Module: Deductive Verification

Module name	Deductive Verification / Deductive Verification
Workload	180 h
Credits	6 LP
Semester	• Deductive Verification : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Deductive Verification: Lecture (45h / 105h / EN / WS oder SS / 40) Deductive Verification: Tutorial (30h / 0h / EN / WS oder SS / 40)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Deductive Verification: first order logic, imperative programming	
Content	
Deductive Verification: The course studies techniques for proving correctness of programs. In contrast to the model checking course, it will look at proof systems here, not automatic methods. The course starts with specifying assertions on (Java) programs which describe properties of programs at particular states. Such properties can then be checked during runtime. Assertions can however also be proven to hold using a Hoare calculus for program verification. In the course we will look at proof calculi for sequential and parallel programs, and study their soundness and completeness. At the end, proof calculi for showing temporal logic properties are introduced.	
Learning objectives	
The students know the key principles of program verification. They can show correctness of both sequential and parallel programs. They know how to use assertions to describe the functionality of programs. They know a temporal logic and are able to use it to specify program properties.	
Implementation method	
Deductive Verification: Partially slides and partially board writing. Interaction with students via questions and example exercises. All essential concepts and techniques will be repeatedly applied in examples during the tutorial.	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	

Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Heike Wehrheim
Learning material, literature
Deductive Verification: K. Apt, E.-R. Olderog: Verification of sequential and concurrent programs Vorlesungsfolien, Übungsaufgaben
Remarks
none

3.19 Elective Module: Designing code analyses for large-scale software systems

Module name	Designing code analyses for large-scale software systems / Designing code analyses for large-scale software systems
Workload	180 h
Credits	6 LP
Semester	• Designing code analyses for large-scale software systems : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Designing code analyses for large-scale software systems: Lecture (45h / 105h / EN / WS / 30) Designing code analyses for large-scale software systems: Tutorial (30h / 0h / EN / WS / 30)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Designing code analyses for large-scale software systems: The course Software Analysis is a recommended but not required prerequisite. A mature understanding of the Java programming languages and object-oriented programming will be helpful.	
Content	
Designing code analyses for large-scale software systems: Static code analysis has the goal of finding programming mistakes automatically, by searching for suspicious anti-patterns in a program's code. This course will explain how to design static code analysis that are inter-procedural, i.e., consider the whole program, across procedure boundaries. Designing such analyses is challenging, as they need to handle millions of program statements efficiently and precisely. Example applications are drawn from the area of IT security.	
Learning objectives	
After having attended this course, students will have learned... - how to make educated design decisions when designing automated code analysis for large-scale software systems, - which algorithms have which properties when using them to implement static code-analyses, - how to design real–world code analyses for practical problem cases from the area of IT security - how to interpret important terminology such as context, flow, field and object sensitivity - how to evaluate and explain the important limitations of static code analysis - which typical security code analyses exist (OWASP Top 10 etc.) and how they relate to the analysis frameworks explained in the course.	
Implementation method	
Designing code analyses for large-scale software systems: Lectures and group exercises	

Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
none
Prerequisites for participation in module exam
none
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Eric Bodden
Learning material, literature
Designing code analyses for large-scale software systems: Empfohlene Literatur: * Thomas Reps, Susan Horwitz, and Mooly Sagiv. 1995. Precise interprocedural dataflow analysis via graph reachability. POPL '95 * Shmuel Sagiv, Thomas W. Reps, and Susan Horwitz. 1995. Precise Interprocedural Dataflow Analysis with Applications to Constant Propagation. TAPSOFT '95 * Akash Lal, Thomas Reps, and Gogul Balakrishnan. 2005. Extended weighted pushdown systems. CAV 2005 * Nomair A. Naeem, Ondrej Lhoták, and Jonathan Rodriguez. 2010. Practical extensions to the IFDS algorithm. CC 2010 * Yannis Smaragdakis, Martin Bravenboer, and Ondrej Lhoták. 2011. Pick your contexts well: understanding object-sensitivity. POPL 2011 * Eric Bodden. 2012. Inter-procedural data-flow analysis with IFDS/IDE and Soot. SOAP 2012 * Rohan Padhye, Uday P. Khedker. Interprocedural Data Flow Analysis in Soot using Value Contexts. SOAP 2013
Remarks
none

3.20 Elective Module: Empiric performance evaluation

Module name	Empiric performance evaluation / Empiric performance evaluation
Workload	180 h
Credits	6 LP
Semester	• Empirische Leistungsbewertung : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Empiric performance evaluation: Lecture (45h / 105h / EN / SS / 15) Empiric performance evaluation: Tutorial (30h / 0h / EN / SS / 15)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Empiric performance evaluation: Bachelor-level stochastics.	
Content	
Empiric performance evaluation: This lecture discusses methods and procedures to conduct experimental and simulation-based performance evaluations, along with a statistically solid evaluation of results. The techniques of this class are applicable to a wide range of systems.	
Learning objectives	
Participants can determine whether a given system or modell is amenable to a particular performance evaluation method. They can design an experiment or a simulation and execute it; they can choose a suitable stochastic model and interprete the experimental results correctly. They can derive statistically justified conclusions, e.g., which system can be regarded as the best system out of a given collection.	
Implementation method	
Empiric performance evaluation: Lecture with slides and blackboard; homework assignments.	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	
Course achievement / qualifying participation	
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of	

the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. rer. nat. Holger Karl
Learning material, literature
Empirische Leistungsbewertung: Foliensatz, Übungsblätter, Lehrbuch Kelton & Law, Simulation Modelling and Analysis.
Remarks
none

3.21 Elective Module: Foundations of Cryptography

Module name	Foundations of Cryptography / Foundations of Cryptography
Workload	180 h
Credits	6 LP
Semester	• Foundations of Cryptography : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Foundations of Cryptography: Lecture (45h / 105h / EN / SS / 25) Foundations of Cryptography: Tutorial (30h / 0h / EN / SS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Foundations of Cryptography: Basic Knowledge in IT-Security and cryptography useful but not necessary, basic concepts of complexity theory and probability theory	
Content	
Foundations of Cryptography: The most important primitives of modern cryptography will be presented. These include encryption schemes, digital signatures, identification protocols, and multiparty computations. In each case we will define precise security notions. Starting from precisely stated assumptions, we develop constructions that provably satisfy these security definitions.	
Learning objectives	
Students understand fundamental concepts and methods of modern cryptography. They are able to choose appropriate cryptographic tools for various security problems. Students are able to combine and modify basic cryptographic primitives, they are able to define new security concepts, they are able to the the security of new constructions with respect to the security concepts.	
Implementation method	
Foundations of Cryptography: Lectures, exercises, reading groups	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	

Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. rer. nat. Johannes Blömer
Learning material, literature
Foundations of Cryptography: Oded Gorldreich, Foundations of Cryptography I,II, Jonathan Katz, Yehuda Lindell, Introduction to Modern Cryptography Folien der Vorlesung
Remarks
none

3.22 Elective Module: Fundamentals of Model-Driven Engineering

Module name	Fundamentals of Model-Driven Engineering / Fundamentals of Model-Driven Engineering
Workload	180 h
Credits	6 LP
Semester	• Fundamentals of Model-Driven Engineering : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Fundamentals of Model-Driven Engineering: Lecture (45h / 105h / EN / WS / 30) Fundamentals of Model-Driven Engineering: Tutorial (30h / 0h / EN / WS / 30)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Fundamentals of Model-Driven Engineering:	
Content	
Fundamentals of Model-Driven Engineering: Model-Driven Engineering focusses on the use of models, i.e., suitable abstractions, as primary artefacts in (software) engineering processes. Important tasks include specifying these models (metamodelling), deriving new models from other models (model transformation), and keeping different but related models consistent (model synchronisation). While it is important to develop an intuitive understanding for central concepts such as models, metamodels, and model transformations, it is equally important to establish solid formal foundations for at least the basic concepts. This is especially the case when providing reliable tool support (e.g., static analyses) for MDE activities. This lecture, therefore, introduces basic MDE concepts including models, metamodels, and model transformations, providing a precise and detailed formalisation using very basic category theory. The lecture is designed to be especially accessible to computer scientists, by providing a hands-on constructive mapping of all definitions and results to programs in a main stream OO language (Java). The new FMDE lecture is designed to complement the existing MDSD lecture, so students can attend both, in any order.	
Learning objectives	
Students are able to read and understand books and academic publications on graph transformation, and can learn and apply new definitions and constructions from such sources. Students are able to apply the formalisation techniques used in the lecture to formalise new structures and the rule-based manipulation of such structures. Students understand and value the advantages of a precise and constructive formalisation of pattern matching, constraints, and rules in the context of Model-Driven Engineering. Students know basic concepts of category theory and understand how and why these concepts are used	

to formalise graph transformation.

Implementation method

Fundamentals of Model-Driven Engineering: Short lectures are interspersed with practical hands-on sessions. Exercises are solved and discussed in groups as an integrated part of the course. Every exercise has a theoretical and corresponding practical part in which all new definitions and construction techniques are implemented (programmed). The practical part extends and builds upon a framework in Java, designed and implemented specially for this course.
The last 4-5 slots of the course are used to deepen the basics by exploring different topics. These advanced topics are discussed and shared among groups of 3-4 students, who work in parallel on these topics in a project-like manner. Results and experiences are presented and shared with all other groups in the last slot of the course. In this last phase of the course, students can practice how to apply the basics from the lecture to learn and implement new definitions and construction techniques from standard literature and academic sources.

Assessment modalities (duration)

Oral exam (ca. 40 Minutes)
The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.

Partial module exams

none

Course achievement / qualifying participation

Studienleistung: written exercises
The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.

Prerequisites for participation in module exam

Passing of course achievement.

Prerequisites for obtaining credits

Passing of module exam.

Weight for overall grade

The module is weighted with 6 credits.

Person responsible for the module

Jun.-Prof.Dr. Anthony Anjorin

Learning material, literature

Fundamentals of Model-Driven Engineering: Ehrig, H., Ehrig, K., Prange, U., & Taentzer, G. (2006). Fundamentals of Algebraic Graph Transformation. (W. Brauer, G. Rozenberg, & A. Salomaa, Eds.). Springer.
Awodey, S. (2006). Category Theory. Ebsco Publishing.

Remarks
none

3.23 Elective Module: Future Internet

Module name	Future Internet / Future Internet
Workload	180 h
Credits	6 LP
Semester	• Future Internet : arbitrary

Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)

Future Internet: Lecture (45h / 105h / EN / SS / 20)
Future Internet: Tutorial (30h / 0h / EN / SS / 20)

Choices in Module

none

Course prerequiites

none

Recommended proficiencies

Future Internet: Basic knowledge of computer networks is required, e.g., a Bachelor-level class "Computer networks".

Content

Future Internet: This lecture deals with up-to-date, close-to-research developments in the Future Internet context as well as data center (networking) context. The lecture is dynamically updated to reflect current research and is predominantly based on research publications.

Learning objectives

Participants of this class are introduced to the current state of the art in Internet research. They know weaknesses of today's architecture, can criticize them and contrast them with current proposals as well as discuss and assess advantages and disadvantages. For different usage scenarios, they can predict the suitability of different solution proposals. Methodically, they can design and execute networking experiments. Participants can create new Internet protocols and synthesize them into new architectures; they can compare such creations with competing approaches and assess and decide for a superior solution. Since the lecture is based on scientific publications, participants are able to make use of original work that has not been didactically prepared.

Implementation method

Future Internet: Lecture with slides and blackboard; homework assignments. The homework assignments will include architecture experiments, e.g., based on OpenFlow.

Assessment modalities (duration)

Oral exam (ca. 40 Minutes)
The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.

Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. rer. nat. Holger Karl
Learning material, literature
Future Internet: Foliensatz, insbesondere auch aktuelle Veröffentlichungen. Kein umfassendes Lehrbuch verfügbar, Teile abgedeckt durch Stallings, Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud.
Remarks
none

3.24 Mandatory Module: General Studies

Module name	General Studies / General Studies
Workload	360 h
Credits	12 LP
Semester	• Studium Generale : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
General Studies: Lecture (90h / 225h / DE / WS / 30) General Studies: Tutorial (45h / 0h / DE / WS / 30)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
General Studies:	
Content	
General Studies:	
Learning objectives	
Implementation method	
General Studies:	
Assessment modalities (duration)	
Exam in general studies The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	
Course achievement / qualifying participation	
Qualifizierte Teilnahme: Qualifying participation in general studies The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.	
Prerequisites for participation in module exam	
none	

Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 4 credits.
Person responsible for the module
Dozenten der Universität Paderborn
Learning material, literature
Studium Generale: Abhängig von den gewählten Veranstaltungen.
Remarks
none

3.25 Elective Module: Hardware/Software Codesign

Module name	Hardware/Software Codesign / Hardware/Software Codesign
Workload	180 h
Credits	6 LP
Semester	• Hardware/Software Codesign : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Hardware/Software Codesign: Lecture (45h / 105h / EN / SS / 40) Hardware/Software Codesign: Tutorial (30h / 0h / EN / SS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Hardware/Software Codesign: Digital design, computer architecture, programming skills	
Content	
Hardware/Software Codesign: Hardware/Software Codesign denotes the integrated and automated design of hardware and software in computer systems. The course Hardware/Software Codesign teaches concepts and methods that are used in computer aided design tools for design-space exploration, design optimization, and compilation for specialized computer systems.	
Learning objectives	
Participants of this course can name the goals and challenges in the design of specialized computer systems. They are able to chose the suitable modeling approaches for a given HW/SW system and its functional and non-functional requirements and to create a specification in the selected formalism. They can characterize target architectures for the implementation of HW/SW systems and evaluate the suitability of a given application. They are able to describe the design of a compiler, to understand and apply basic blocks and control-flow graphs, and to discuss optimization and code generation methods. They are able to demonstrate how programs are translated to hardware using high-level synthesis methods. The understand the integer linear programming method and can apply it to problems from the domain of synthesis, scheduling and software performance estimation.	
Implementation method	
Hardware/Software Codesign: • Lecture with projected slides and notes on blackboard • Interactive assignments in lecture room • Self-study and discussion of scientific publications • Practical programming projects for applying methods of the area Hardware/Software Codesign	

Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Christian Plessl
Learning material, literature
Hardware/Software Codesign: Folien, Standardlehrbücher, Übungsblätter, Wissenschaftliche Artikel
Remarks
none

3.26 Elective Module: High-Performance Computing

Module name	High-Performance Computing / High-Performance Computing
Workload	180 h
Credits	6 LP
Semester	• High-Performance Computing : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
High-Performance Computing: Lecture (30h / 105h / EN / WS / 40) High-Performance Computing: Tutorial (45h / 0h / EN / WS / 40)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
High-Performance Computing: • Programming skills in C/C++ • Computer architecture	
Content	
High-Performance Computing: This course teaches the foundations of high-performance computing with an emphasis on the programming of parallel computer systems and novel hardware accelerators.	
Learning objectives	
After attending this course, the students are able to • name models and programming patterns for HPC and to select patterns for a given application, • name and apply the basic constructs of frequently used HPC libraries, in particular, MPI, OpenMP and OpenCL, • analyze the performance of applications by using profiling tools and use the gathered information to create a systematic optimization strategy, • apply the taught concepts and methods for parallelizing and optimizing existing applications	
Implementation method	
High-Performance Computing: • Lecture with projected slides and blackboard notes • Interactive assignments in lecture room • Practical programming projects on parallel computer systems (teamwork in small groups)	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes)	

The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Christian Plessl
Learning material, literature
High-Performance Computing: • Vorlesungsfolien • Übungsblätter • Aufgabenblätter und technische Dokumentation für die Programmierprojekte • Lehrbuch: Pacheco: An Introduction to Parallel Programming. Morgan Kaufmann, 2011.
Remarks
none

3.27 Elective Module: Intelligence in Embedded Systems

Module name	Intelligence in Embedded Systems / Intelligence in Embedded Systems
Workload	180 h
Credits	6 LP
Semester	• Intelligenz in eingebetteten Systemen : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Intelligence in Embedded Systems: Lecture (45h / 105h / EN / SS / 25) Intelligence in Embedded Systems: Tutorial (30h / 0h / EN / SS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Intelligence in Embedded Systems:	
Content	
Intelligence in Embedded Systems: Intelligent embedded systems are technical systems that use different sensors and actors to perceive their environment and interact with it in a (partly) autonomous way. Often their behavior is controlled by methods and algorithms from artificial intelligence (AI). Such methods enable for instance that systems plan their behavior in a goal directed manner or optimize it via self-adaptation and learning. Systems of that kind are becoming increasingly important, not only on a scientific level but also in a social context: Autonomous or semi-autonomous systems such as service robots, self-driving cars or medical help and diagnosis systems will have a deep impact on our future private and professional life. This course covers important aspects for the development of intelligent embedded systems and conveys corresponding theoretical and methodological foundations. This includes lectures on architectures, intelligent sensor processing, environment modelling, intelligent behavior control and self-adaptation.	
Learning objectives	
Students know and explain methods and algorithms for intelligent sensor processing and control of actions (e.g. computer vision, sensor fusion, maps, navigation, planning and machine learning). They understand and solve problems arising when realizing them in embedded systems with restricted resources. Furthermore, they are able to understand, use and adapt new methods and algorithms especially in the context of embedded systems.	
Implementation method	
Intelligence in Embedded Systems: • Lecture with slides • Interactive exercises, where students deepen their understanding and apply their knolwledge obtained in the lectures	

Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
none
Prerequisites for participation in module exam
none
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Dr. Bernd Kleinjohann
Learning material, literature
Intelligenz in eingebetteten Systemen: Folien, Publikationen, Bücher: • St. Russel, P. Norvig: Artificial Intelligence: A Modern Approach • R. Arkin: Behavior-Based Robotics Weitere Literatur (Bücher, Publikationen) werden in der Vorlesung bekanntgegeben.
Remarks
none

3.28 Elective Module: Interactive Data Visualization

Module name	Interactive Data Visualization / Interactive Data Visualization
Workload	180 h
Credits	6 LP
Semester	• Interaktive Datenvisualisierung : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Interactive Data Visualization: Lecture (30h / 105h / EN / SS / 30) Interactive Data Visualization: Tutorial (15h / 0h / EN / SS / 30) Interactive Data Visualization: Visualization Project (30h / 0 h / EN / SS / 30)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Interactive Data Visualization: Good programming skills with at least one graphics API.	
Content	
Interactive Data Visualization: The field of visualization seeks to determine and present underlying correlated structures and relationships in data sets from a wide variety of application areas. The prime objective of the presentation is to communicate the information in a dataset so as to enhance understanding for the human viewer. In this context, a visualization (result) is a computer generated image or collection of images, using a computer representation of data as its primary source and a human as its primary target. Thus the visualization process becomes a mapping process from data to expressive and effective (visual) representations.	
Learning objectives	
Students can explain the visualization process along the Four-Component-Model (Reality, Data, Picture(s), User). Students demonstrate the ability, to explain all computational steps for selective algorithms for data preprocessing and analysis (e.g. transfer functions, filtering, statistical analysis, sampling, scaling) and to implement these algorithms with a modern API. They are able to judge the power of APIs and tools for their use in visualization. For different data types and model, multiple visualization techniques can be enumerated and explained. Students have acquired the ability to judge their own and other visualization results in terms of expressiveness, effectiveness, suitability and scalability. Evaluation methods to utilize during the visualization process can be enumerated and explained. The design process (mapping from data variables to visual variables) can be discussed in a qualified manner, including influencing properties of user, data or hardware. Students demonstrate their skills by - developing solutions for visualization problems in one or more application areas (e.g. medicine, astrophysics, computer-network analysis, text corpora analysis) in a modern computer environments - presenting solutions in a technically clear and motivating presentation.	

Implementation method
Interactive Data Visualization: Beamer and board are used in lectures. Students receive short in-class exercises to instigate a subsequent discussion on possible solutions. Students solve homework problems on their own, their solutions are being presented and discussed in lab time. During the last third of the course, students work on a group project. They present their solutions and evaluate each others' projects.
Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Gitta Domik-Kienegger
Learning material, literature
Interaktive Datenvisualisierung: Foliensatz der Vorlesung; Textbuch "Interactive Data Visualization", M. Ward, G. Grinstein, D. Keim, A K Peters.
Remarks
none

3.29 Elective Module: Language-Based Security

Module name	Language-Based Security / Language-Based Security
Workload	180 h
Credits	6 LP
Semester	• Sprachbasierte Sicherheit : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Language-Based Security: Lecture (45h / 105h / EN / WS / 25) Language-Based Security: Tutorial (30h / 0h / EN / WS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Language-Based Security: Basic knowledge on compilers, programming languages, and security. (For example, through the lectures “Programming Languages and Compilers” and “IT Security”.)	
Content	
Language-Based Security: This lecture will discuss current topics in language-based security from both, practical and theoretical perspective.	
Learning objectives	
Students will acquire fundamental knowledge about current attacks and defenses. Discussion of the techniques comprises both elements, theoretical results as well as practical implementation considerations. In addition to factual knowledge about attacks and defenses, students acquire knowledge about the underlying principles of programming languages. In consequence, students are equipped with tools and techniques to solve relevant securtiy questions in a sound fashion.	
Implementation method	
Language-Based Security:	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	
Course achievement / qualifying participation	

Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof.Dr. Stefan Brunthaler
Learning material, literature
Sprachbasierte Sicherheit: none
Remarks
none

3.30 Elective Module: Linear and Integer Optimization

Module name	Linear and Integer Optimization / Linear and Integer Optimization
Workload	180 h
Credits	6 LP
Semester	• Linear and Integer Optimization : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Linear and Integer Optimization: Lecture (45h / 105h / EN / WS / 30) Linear and Integer Optimization: Tutorial (30h / 0h / EN / WS / 30)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Linear and Integer Optimization: Algorithms and Data Structures Linear Algebra	
Content	
Linear and Integer Optimization: Nowadays, optimization problems are an omnipresent part of our daily life in industry, trade and commerce. Mostly, optimization deals with the management and efficient use of scarce or expensive resources, e.g. in transportation, scheduling, planning, or design problems. Many of these optimization problems are NP-hard and thus cannot be solved efficiently. Surprisingly, many optimization problems can naturally be modelled as linear programs (LP) or integer programs (IP) and algorithms and techniques to solve LPs or IPs perform excellent in finding solutions of these models. This lecture contains a fundamental introduction to linear and interger optimization. We will deal with techniques to model optimization problems as LP/IP. We will present algorithms for the solution of these models and succinctly analyse their behaviour. Furthermore, for IPs we will analyze polynomially solvable special cases and give examples for heuristics to increase the efficiency of the solvers in the general case. Kexwords: Simplex algorithm, Interior Point Method, Cutting Planes, Branch & Bound Algorithm, Branch & Cut	
Learning objectives	
Students get to know fundamental modelling techniques for optimization problems. They are able to identify suitable models for a given optimiazion problem from application and they can distinguish between efficiently solvable models and others. For linear and integer programs the students know fundamental algorithms as well as various techniques to speed up their computation. Furthermore, students are able to use commercial as well as uncommercial LP/IP solvers.	
Implementation method	
Linear and Integer Optimization: Lecture with tutorials	

Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Dr. Rainer Feldmann
Learning material, literature
Linear and Integer Optimization: Folien, Skript Literatur: M. Conforti, G. Cornuejols, G. Zambelli. Integer Programming. Springer 2014. A. Schrijver. Combinatorial Optimization: Polyhedra and Efficiency, Vol. A,B,C. Springer 2003. V. Chvatal. Linear Programming. Freeman 1983. L.A. Nemhauser, G.L. Wolsey. Integer and Combinatorial Optimization. Wiley 1999. A. Schrijver. Theory of Linear and Integer Optimization. Wiley, 1999.
Remarks
none

3.31 Elective Module: Logic and Automated Reasoning

Module name	Logic and Automated Reasoning / Logic and Automated Reasoning
Workload	180 h
Credits	6 LP
Semester	• Logik und automatisches Beweisen : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Logic and Automated Reasoning: Lecture (45h / 105h / EN / WS / 30) Logic and Automated Reasoning: Tutorial (30h / 0h / EN / WS / 15)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Logic and Automated Reasoning: Basics of symbolic modeling, basics of computational complexity theory	
Content	
Logic and Automated Reasoning: The course "Logic and Automated Reasoning" is dedicated to understanding different aspects of reasoning. Basis for automated reasoning is a symbolic representation of knowledge on problem domains. For this purpose, classical logics and other examples of logic-based languages are introduced and their expressiveness is explored. For these formal languages, adequate inference engines and underlying calculi are presented and analyzed. In particular, the relationship between propositional logics and first-order logics will be considered. Topics discussed include soundness, (in-) completeness, and (non-) monotonicity.	
Learning objectives	
The students are able to • name, describe, and valuate logic-based laguages for knowledge representation, • specify knowledge and tasks using a logic-based symbolic formalism, • describe seveal calculi and corresponding inference algorithms, • choose and apply approriate inference methods, • assess the complexity of tasks and the runtimes of inference algorithms based on a rough scaling.	
Implementation method	
Logic and Automated Reasoning: • Lecture • Tutorials • Discussions • Homework Assignments	

Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Dr. Theodor Lettmann
Learning material, literature
Logik und automatisches Beweisen: • Vorlesungsfolien • Lehrbücher: 1.) J. Harrison: Handbook of Practical Logic and Automated Reasoning, Cambridge University Press, 2009, 2.) M. Fitting: First-Order Logic and Automated Theorem Proving, 2nd ed., Springer 1995, 3.) j.H. Gallier: Logic for Computer Science: Foundations of Automatic Theorem Proving, 2nd ed., Dover, 2015, 4.) L. Wos, G. W. Pieper. A Fascinating Country in the World of Computing: Your Guide to Automated Reasoning, World Scientific, 1999. • Übungsaufgaben • Liste von klassischen und aktuellen Papieren, z.B. D. W. Loveland. Mechanical theorem-proving by model elimination, Journal of the ACM,15, pp. 236–251, 1968
Remarks
none

3.32 Elective Module: Logic Programming for Artificial Intelligence

Module name	Logic Programming for Artificial Intelligence / Logic Programming for Artificial Intelligence
Workload	180 h
Credits	6 LP
Semester	• Logic Programming for Artificial Intelligence : arbitrary

Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)

Logic Programming for Artificial Intelligence: Lecture (45h / 105h / EN / WS oder SS / 40)
Logic Programming for Artificial Intelligence: Tutorial (30h / 0h / EN / WS oder SS / 20)

Choices in Module

none

Course prerequiites

none

Recommended proficiencies

Logic Programming for Artificial Intelligence: Students should have previous knowledge in programming as offered in the courses "Programmierung" and "GPS" and knowledge in database query languages as offered in the course "Datenbanksysteme".

Content

Logic Programming for Artificial Intelligence: This course views various concepts and techniques from computer science, artificial intelligence, and computational linguistics from a different perspective, i.e. the perspective of programming in logic. Programming in logic in general and the programming language Prolog in particular offer the ability to describe many concepts in logic, i.e. in a declarative way, and to have them tested and executed by an interpreter at a same time. This is in particular useful for puzzles and quizes, but also for self-defined or domain specific languages.

Learning objectives

Students learn
factual knowledge about
• the transformation of knowledge given as facts and rules into an executable programs
• how to program in logic and in self-designed languages
methodological knowledge, including
• the ability to define domain specific languages
• the ability to implement interpreters for domain specific languages
• the ability to develop small question answering systems
• the ability to develop software for theorem provers or constraint solvers solving puzzles
transfer skills
• the ability to transfer the methodolies and skills gained to other data sources, knowledge representation formats, or calculi

• the ability to transfer the parsing and semantics knowledge to domain specific languages
normative evaluation skills including the ability to assess
• the suitability and limitations of different data and knowledgerepresentation formats for different tasks
• the suitability of different programming paradigms for different projects
• the effort and feasibility of projects aiming natural language understanding
• the effort and feasibility of projects aiming at automated translation

Implementation method

Logic Programming for Artificial Intelligence: The theoretical concepts are explained in the lectures and consolidated in small groups during tutorials. The tutorials are carried out as practical exercises on the computer.

Assessment modalities (duration)

Oral exam (ca. 40 Minutes)
The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.

Partial module exams

none

Course achievement / qualifying participation

Studienleistung: written exercises
The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.

Prerequisites for participation in module exam

Passing of course achievement.

Prerequisites for obtaining credits

Passing of module exam.

Weight for overall grade

The module is weighted with 6 credits.

Person responsible for the module

Prof. Dr. Stefan Böttcher

Learning material, literature

Logic Programming for Artificial Intelligence: Ivan Bratko: Prolog Programming for Artificial Intelligence. Pearson Education, Newest Edition. Hinweise auf weiteres Material werden in der Lehrveranstaltung bekannt gegeben.

Remarks

none

3.33 Elective Module: Machine Learning I

Module name	Machine Learning I / Machine Learning I
Workload	180 h
Credits	6 LP
Semester	• Maschinelles Lernen I : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Machine Learning I: Lecture (45h / 105h / EN / WS oder SS / 60) Machine Learning I: Tutorial (30h / 0h / EN / WS oder SS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Machine Learning I: Basic knowledge in mathematics (linear algebra, statistics), programming and algorithms.	
Content	
Machine Learning I: Due to the ever increasing amount of data that is routinely produced in our information society, the topic of machine learning has become increasingly important in the recent years, not only as a scientific discipline but also as a key technology of modern software and intelligent systems. This lecture provides an introduction to the topic of machine learning, with a specific focus on supervised learning for classification and regression. The lecture covers theoretical foundations of generalisation as well as practical topics and concrete learning algorithms.	
Learning objectives	
The students understand the statistical foundations of generalisation, i.e., the induction of models from data, as well as practical tools for model validation. They are able to apply basic methods of supervised learning to problems of classification and regression.	
Implementation method	
Machine Learning I: Theoretical foundations and concepts of machine learning will be taught in the form of a lecture and deepened in practical exercise courses, group work as well as individual homework.	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	

Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Eyke Hüllermeier
Learning material, literature
Maschinelles Lernen I: Skript und eine Liste mit Buchempfehlungen: • Y.S. Abu-Mostafa, M. Magdon-Ismail, H.T. Lin. Learning from Data, AMLBook, 2012. • P. Flach. Machine Learning, Cambridge Univ. Press, 2012. • E. Alpaydin. Machine Learning, Oldenbourg, 2008. • C.M. Bishop. Pattern Recognition and Machine Learning, Springer, 2006.
Remarks
none

3.34 Elective Module: Machine Learning II

Module name	Machine Learning II / Machine Learning II
Workload	180 h
Credits	6 LP
Semester	• Maschinelles Lernen II : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Machine Learning II: Lecture (45h / 105h / EN / WS oder SS / 20) Machine Learning II: Tutorial (30h / 0h / EN / WS oder SS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Machine Learning II: Basic knowledge in machine learning (as conveyed, for example, by the Machine Learning I lecture).	
Content	
Machine Learning II: This lecture, which is conceived as a continuation of the Machine Learning I, covers advanced topics in contemporary machine learning research, such as reinforcement learning, online learning and bandit algorithms, multi-task learning, multi-target and structured output prediction, preference learning, learning from weak supervision, and uncertainty in machine learning. The focus of the lecture will be on methods and algorithms, though theoretical issues and applications will be addressed, too.	
Learning objectives	
The students have an overview of methods for multi-class classification, the learning of nonlinear models, and extensions of the simple setting of supervised learning. They understand algorithmic concepts of corresponding methods and are able to apply them to real problems.	
Implementation method	
Machine Learning II: Theoretical foundations and concepts of machine learning will be taught in the form of a lecture and deepened in practical exercise courses, group work as well as individual homework.	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	

Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Eyke Hüllermeier
Learning material, literature
Maschinelles Lernen II: Skript und eine Liste mit Buchempfehlungen: • Y.S. Abu-Mostafa, M. Magdon-Ismail, H.T. Lin. Learning from Data, AMLBook, 2012. • P. Flach. Machine Learning, Cambridge Univ. Press, 2012. • E. Alpaydin. Machine Learning, Oldenbourg, 2008. • C.M. Bishop. Pattern Recognition and Machine Learning, Springer, 2006.
Remarks
none

3.35 Mandatory Module: Master Thesis

Module name	Master Thesis / Master Thesis
Workload	900 h
Credits	30 LP
Semester	• Masterarbeit : 4 • Masterararbeit Arbeitsplan : 4
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Master Thesis: (30h / 720 h / EN / WS oder SS / 1) Master Thesis Work Plan: contact hours, presentation work plan (30h / 120 h / EN / WS oder SS / 0)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Master Thesis: Depends on the thesis topic. Master Thesis Work Plan: Depends on the topic.	
Content	
Master Thesis: A Master Thesis consists of working on a demanding subject, including a written report and an oral presentation. With the thesis the student shows her/his ability to work independently and systematically on a demanding topic which also includes developing her/his own ideas. On a state-of-the-art basis the methods of computer science should be applied systematically.- Master Thesis Work Plan: After having agreed on a topic, the student draws up a work plan. The work plan includes the targeted results, the techniques and methods used and important milestones.	
Learning objectives	
Finishing their master thesis students show that they are able • to solve a problem in computer sience within an appropriate time frame using scientifically sound Methode • to apply the techniques and methods that they learned during their studies to a new and demanding problem.	
Implementation method	
Master Thesis: Independent studies supported by individual advice and supervision Master Thesis Work Plan: In agreement with supervisor.	
Assessment modalities (duration)	

Final thesis The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
none
Prerequisites for participation in module exam
none
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 50 credits.
Person responsible for the module
Prof. Dr. rer. nat. Johannes Blömer
Learning material, literature
Masterararbeit Arbeitsplan: Je nach gewähltem Thema. Masterarbeit: Je nach gewähltem Thema.
Remarks
none

3.36 Elective Module: Mobile Communication

Module name	Mobile Communication / Mobile Communication
Workload	180 h
Credits	6 LP
Semester	• Mobilkommunikation : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Mobile Communication : Lecture (45h / 105h / EN / WS / 20) Mobile Communication : Tutorial (30h / 0h / EN / WS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Mobile Communication : Basic knowledge of computer networks is required, e.g., a Bachelor-level class “Computer networks”.	
Content	
Mobile Communication : The lecture discusses foundations of mobile communication (e.g., wireless channel models) and fundamental techniques (e.g., spread spectrum communication), important protocol mechanisms (e.g., medium access in wireless systems), mobile communication systems, and MobileIP. In addition to technological and conceptual aspects, we shall also discuss approaches and methods for performance evaluation of mobile communication systems.	
Learning objectives	
Participants of this class know challenges and problems arising in design and operation of mobile communication systems. They can differentiate between challenges based in physics and those arising from a particular system design; they can choose suitable protocols or design new ones. They are able to select mechanisms from different architectural layers, integrate them into a new complete architecture and justify their selection and integration decisions. They are also able to quantitatively evaluate protocol mechanisms.	
Implementation method	
Mobile Communication : Lecture with slides and blackboard; homework assignments with (among others) some programming assignments to simulate wireless systems.	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	

Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. rer. nat. Holger Karl
Learning material, literature
Mobilkommunikation: Foliensatz; einzelne Kapitel div. Standardlehrbücher (J. Schiller, Mobile Communication, Addison Wesley, 2nd edition; D. Tse und P. Viswanath, Fundamentals of Wireless Communication, Cambridge University Press, 2005).
Remarks
none

3.37 Elective Module: Model Checking

Module name	Model Checking / Model Checking
Workload	180 h
Credits	6 LP
Semester	• Model Checking : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Model Checking: Lecture (45h / 105h / EN / WS oder SS / 40) Model Checking: Tutorial (30h / 0h / EN / WS oder SS / 40)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Model Checking: Logic, some knowledge of imperative programming, concurrency	
Content	
Model Checking: The course studies techniques for formally verifying that a system (software or hardware) is correct, i.e. adheres to requirements describing the desired functionality. For describing requirements a particular class of logics, so called temporal logics, is employed. Temporal logics can be used to describe properties of systems in time. For this class of logics there are algorithms for checking whether a property does or does not hold for a system. If the system under consideration has a finite state space, tools implementing these algorithms can fully automatically carry out the verification. The course looks at two temporal logics (LTL and CTL) and their model checking algorithms.	
Learning objectives	
The students are able to formally specify requirements on software or hardware system and know tools for automatically verifying these. They have understood the techniques underlying model checking algorithms and know the differences between linear-time and branching-time logics. They are able to understand new approaches in the area of verification and can compare them with existing methods. The students are able to develop simple formal proofs themselves and can explain the proofs which are presented in the lectures.	
Implementation method	
Model Checking: Partially slides and partially board writing. All essential concepts and techniques will be repeatedly applied in examples during the tutorial. In a lab part, the techniques will be employed using the modelchecker Spin.	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes)	

The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Heike Wehrheim
Learning material, literature
Model Checking: Christel Baier, Joost-Pieter Katoen: Principles of Model Checking Vorlesungsfolien, Übungsaufgaben
Remarks
none

3.38 Elective Module: Model-Based Interface Development

Module name	Model-Based Interface Development / Model-Based Interface Development
Workload	180 h
Credits	6 LP
Semester	• Modellbasierte Entwicklung von Benutzungsschnittstellen : arbitrary

Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)

Model-Based Interface Development: Lecture (45h / 105h / EN / WS oder SS / 40)
Model-Based Interface Development: Tutorial (30h / 0h / EN / WS oder SS / 40)

Choices in Module

none

Course prerequiites

none

Recommended proficiencies

Model-Based Interface Development: The class does not need more knowledge than what is provided through the preceding Bachelor study.

Content

Model-Based Interface Development: The class starts by dealing with the concept of modelling in a "Computer Science" way as such. Models are referred here with the following definition: "A model is an artefact, such as a drawing, a text, or a physical object supposed to reflect or represent certain relevant aspects of a real system." Based on this definition a series of different types of artefacts relevant under certain aspects are covered. This starts with task modelling, where we look into several task modelling approaches, including some tools to work with. Then we enter the area of dialogue models, which are divided into two distinct groups - the user interaction models and the control models. The two types describe "the same thing", with different goals and results. This chapter deals with several approaches and their interconnections, if applicable by looking at corresponding tools. The class finishes with describing a model-based user interface design process. These insights are then applied to the field of webmodelling. The problems are basically the same as for classical user interfaces, but in the literature alternative modelling approaches are presented and used. We look into domain model, navigation model and presentation model, and deal with approaches which are only theoretically interesting, but also use intensively the commercially used and very successfull web modelling platform WebRatio.

Learning objectives

Factual knowledge: The class teaches formal systems, which are used during a modelling process. It becomes clear, which aspect relevant for user interface or web design, can be treated with which model on which abstraction level. We cover generic basic knowledge, but also specialized approaches for the web. Methods: The students learn to apply modelling approaches in practice. We present prototypical, academic approaches, as well as commercially used ones. Their use is practiced in the exercises. Transfer knowledge: The students learn to use formal systems in realistic design scenarios. Evaluative: The

students learn about the benefits, but also the drawbacks of the concrete modelling approaches. In the end they can assess during the work on practical examples, whether it was/is worthwhile to pursue a model-based process or not, and which risks are involved.

Implementation method

Model-Based Interface Development: The class is held as a classical lecture with beamer presentation, supported by koaLA - the e-learning environment of the university. That system is used to provide the slides of the class, written homeworks, access to software used during the class and video recordings of every lecture. During the class we have interactive meetings, groupwork in several chapters, e.g. for design work with WebRatio or the creation of complex task models. During exercise hours the students present their solutions to the homework by showing a small presentation with slides.

Assessment modalities (duration)

Oral exam (ca. 40 Minutes)
The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.

Partial module exams

none

Course achievement / qualifying participation

Studienleistung: written exercises
The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.

Prerequisites for participation in module exam

Passing of course achievement.

Prerequisites for obtaining credits

Passing of module exam.

Weight for overall grade

The module is weighted with 6 credits.

Person responsible for the module

Prof. Dr. Gerd Szwillus

Learning material, literature

Modellbasierte Entwicklung von Benutzungsschnittstellen:
- Vorlesungsfolien als PDF zum Herunterladen
- Schriftliche Hausaufgaben
- Diverse, während der Vorlesung eingesetzte Modellierungswerkzeuge
- Voller Zugang zu WebRatio
- Ben Shneiderman: Designing the User Interface: Strategies for Effective Human-Computer Interaction, 2009
- David Benyon: Designing Interactive Systems: A Comprehensive Guide to HCI, UX and Interaction Design, 2009

• Stefano Ceri et al: Designing Data-Intensive Web Applications, 2003
Remarks
none

3.39 Elective Module: Network Simulation

Module name	Network Simulation / Network Simulation
Workload	180 h
Credits	6 LP
Semester	• Network Simulation : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Network Simulation: Lecture (30h / 105h / EN / SS / 20) Network Simulation: Tutorial (45h / 0h / EN / SS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Network Simulation: System software and system-level programming	
Content	
Network Simulation: This course offers the chance to get in touch with the network simulation framework OMNeT++. Simulation is one possibility (others are experiments or mathematical analysis) to evaluate the performance of systems which might even not be available in reality. After getting a basic introduction to simulation and modeling, we will study a small example project already using OMNeT++. Finally, we will work in small groups of 2-3 people on interesting research-focused projects of the areas vehicular communication and wireless sensor networks.	
Learning objectives	
The learning objective is to unserstand the fundamental concepts of network simulation. Students understand these concepts and are able to apply this knowledge.	
Implementation method	
Network Simulation: Lecture with practical exercises	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	
Course achievement / qualifying participation	
Studienleistung: written exercises	

The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Falko Dressler
Learning material, literature
Network Simulation: Folien, Lehrbücher, Papiere
Remarks
none

3.40 Elective Module: Networked Embedded Systems

Module name	Networked Embedded Systems / Networked Embedded Systems
Workload	180 h
Credits	6 LP
Semester	• Networked Embedded Systems : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Networked Embedded Systems: Lecture (45h / 105h / EN / WS / 60) Networked Embedded Systems: Tutorial (30h / 0h / EN / WS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Networked Embedded Systems: System software and system-level programming	
Content	
Networked Embedded Systems: The objective of this course is gain insights into the operation and programming of embedded systems. A strong focus is on wireless sensor networks. We study the fundamentals of such sensor networks. In the sopce of the exercises, we discuss selected topics in more detail.	
Learning objectives	
The learning objective is to unserstand the fundamental concepts of networded embedded systems. Students understand these concepts and are able to apply this knowledge.	
Implementation method	
Networked Embedded Systems: Lecture with practical exercises	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	
Course achievement / qualifying participation	
Studienleistung: written exercises	

The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Falko Dressler
Learning material, literature
Networked Embedded Systems: Folien, Lehrbücher, Papiere
Remarks
none

3.41 Elective Module: Planning and Heuristic Search

Module name	Planning and Heuristic Search / Planning and Heuristic Search
Workload	180 h
Credits	6 LP
Semester	• Planen und Heuristische Suche : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Planning and Heuristic Search: Lecture (45h / 105h / EN / WS / 40) Planning and Heuristic Search: Tutorial (30h / 0h / EN / WS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Planning and Heuristic Search: Basics of symbolic modeling: functions, relations, logical formulas; design and analysis of algorithms; basics of computational complexity theory: complexity classes, reducibility, completeness.	
Content	
Planning and Heuristic Search: The course "Planning and Heuristic Search" introduces two approaches to solve knowledge-intensive tasks. In the "Planning" part, domain tasks are represented as planning problems using e.g. state spaces or plan spaces. Several approaches to planning are discussed and analyzed. The "Heuristic Search" part generalizes the state space concept and introduces a general framework of systematic search algorithms that are guided by heuristic information on the problem domain. As an application example, planning algorithms can be implemented deterministically using heuristic search. Theoretical results will be presented and proved in both areas.	
Learning objectives	
The students are able to • name and explain relevant concepts in modeling planning tasks and search tasks, • describe the basisc steps of several planning algorithms and heuristic search algorithms, • identify and model simple tasks as planning tasks resp. search tasks, • describe approaches to compute valuable heuristcs in such tasks, • use theoretical results as guidance in selecting formalizations and algorithms.	
Implementation method	
Planning and Heuristic Search: • Lecture • Tutorials • Discussions • Homework Assignments	

• Prototype Implementations
Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Dr. Theodor Lettmann
Learning material, literature
Planen und Heuristische Suche: • Vorlesungsfolien • Lehrbücher: 1.) J. Pearl: Heuristics, Addison-Wesley (1984), 2.) S.J. Russell, P. Norvig: Artifical Intelligence: A Modern Approach, Prentice Hall, 1995, 3.) S. Edelkamp, S. Schrödl: Heuristic Search: Theory and Applications, Elsevier, 2012, 4.) M. Ghallab, D. Nau, P. Traverso: Automated Planning, Morgan Kaufmann, 2004 • Übungsaufgaben • Liste von klassischen und aktuellen Papieren, z.B. R. Ebendt, R. Drechsler: Weighted A*Search - Unifying View and Application, J. Artificial Intelligende, pp. 1310-1342, 2009 item Online Material, z.B. H. Geffner, B. Bonet: A Concise Introducation to Models and Methods for Automated Planning, doi: 10.2200/S00513ED1V01Y201306AIM022, 2013
Remarks
none

3.42 Mandatory Module: Projet Group

Module name	Projet Group / Projet Group
Workload	600 h
Credits	20 LP
Semester	• Projektgruppe : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Project Group: Lecture (30h / 540h / EN / WS oder SS / 10) Project Group: Meetings, presentation of partial results (30h / 0 h / EN / WS oder SS / 10)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Project Group: Depends on the topic.	
Content	
Project Group: In a Project Group a group of usually 8-16 students works together over a period of one year (two semesters) on a research topic determined by the group organizer. Project groups introduce students to current research topics that are usually related to the group organizer's special area of interest and the team working of the project group should be a preparation for industrial practice.	
Learning objectives	
In project groups, participating students gain first-hand practical experience in working in a team and organizing a project; in doing so, they become prepared for daily work in their later professions. The students personally experience how to carry out extensive development processes in a team. Since the tasks are divided among the individual team members, the participating students become skilled in reporting their progress and research findings to the other group members.	
Implementation method	
Project Group: • The number of participants is limited to 16 people. • Developing knowledge on the selected systematic approaches, methods and tools relevant to the research topic- usually done in an introductory seminar phase. • Logical assigning "jobs" (assigning responsibilities to the individual group members). • Discovering and promoting the participants' special individual talents, which are either already apparent or which can be developed throughout the project - such as through seminar presentations or appropriate job assignments. • Setting up a process-oriented personnel structure, similar to the structure of an industrial design team; delegating subtasks to smaller subgroups who report their findings. • Regular progress reports made by individuals and subgroups.	

• Writing a highly distributed interim report and final report.
Assessment modalities (duration)
Partial modul exam (100 % of module grade) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: Practical work The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 8 credits.
Person responsible for the module
Prof. Dr. rer. nat. Johannes Blömer
Learning material, literature
Projektgruppe: Abhängig vom Thema.
Remarks
none

3.43 Elective Module: Public-Key Cryptography

Module name	Public-Key Cryptography / Public-Key Cryptography
Workload	180 h
Credits	6 LP
Semester	• Public-Key Cryptography : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Public-Key Cryptography: Lecture (45h / 105h / EN / WS / 50) Public-Key Cryptography: Tutorial (30h / 0h / EN / WS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Public-Key Cryptography: Elementary cryptography, as usually taught in introductory courses on cryptography	
Content	
Public-Key Cryptography: This course covers foundational techniques for the construction and formal security analysis of public-key cryptosystems. This includes, in particular, digital signature schemes, public-key encryption schemes, identity-based encryption, and related concepts. The main focus lies on the introduction of well-established techniques that are commonly used to construct and prove security of modern cryptosystems.	
Learning objectives	
Students know established techniques for the construction of provably secure public-key cryptosystems. They are able to define security requirements formally, know the established standard definitions and their limitations, and are able to formally analyze the relationships between these definitions. They are also able to apply standard proof techniques to new cryptosystems.	
Implementation method	
Public-Key Cryptography: • Lectures with slides and blackboard notes • Exercises and presentation of solutions by participants	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	

Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr., Tibor Jager
Learning material, literature
Public-Key Cryptography: • Vorlesungsfolien • Tafelanschrieb • Originalliteratur
Remarks
none

3.44 Elective Module: Reconfigurable Computing

Module name	Reconfigurable Computing / Reconfigurable Computing
Workload	180 h
Credits	6 LP
Semester	• Reconfigurable Computing : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Reconfigurable Computing: Lecture (30h / 105h / EN / WS / 50) Reconfigurable Computing: Tutorial (45h / 0h / EN / WS / 25)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Reconfigurable Computing: Knowledge of "Digital Design" and "Computer Architecture" is beneficial.	
Content	
Reconfigurable Computing: This lecture provides an understanding of architectures and design methods for reconfigurable hardware systems and presents applications in the areas of high performance computing and embedded systems.	
Learning objectives	
After attending the course, the students • are able to explain the architectures of reconfigurable hardware devices, • to name and analyze the main design methods and • to judge the suitability of reconfigurable hardware for different application domains.	
Implementation method	
Reconfigurable Computing: • Lecture with projector and board • Interactive exercises in the lecture room • Computer-based exercises with reconfigurable systems	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	

Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Marco Platzner
Learning material, literature
Reconfigurable Computing: • Vorlesungsfolien und Übungsblätter • Aufgabenblätter und technische Dokumentation für die Rechnerübungen • S. Hauck and A. DeHon (editors): Reconfigurable Computing, Volume 1: The Theory and Practice of FPGA-Based Computation, Morgan Kaufmann, 2008 • Aktuelle Hinweise auf alternative und ergänzende Literatur, sowie Lehrmaterialien auf der Webseite und in den Vorlesungsfolien
Remarks
none

3.45 Elective Module: Routing and Data Management in Networks

Module name	Routing and Data Management in Networks / Routing and Data Management in Networks
Workload	180 h
Credits	6 LP
Semester	• Routing and Data Management in Networks : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Routing and Data Management in Networks: Lecture (45h / 105h / EN / SS / 40) Routing and Data Management in Networks: Tutorial (30h / 0h / EN / SS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Routing and Data Management in Networks: Algorithm design, theoretical correctness and efficiency proofs, tools from combinatorics and probability theory.	
Content	
Routing and Data Management in Networks: Routing and data management are fundamental tasks to be solved in order to ensure efficient use of large networks, e.g. the Internet, peer-to-peer systems, or wireless mobile ad-hoc networks. This lecture deals with algorithms and their analysis for routing and data management in such systems and describes, in particular, methods for dealing with their dynamics (movement of nodes, joining and exiting nodes). In particular, local, distributed algorithms, often as online algorithms, are considered.	
Learning objectives	
The students get to know fundamental techniques in the area of routing and data management of large networks. They can decide in which situation which data management, scheduling, or routing algorithm is most appropriate. They can adapt algorithms to a new situation.	
Implementation method	
Routing and Data Management in Networks: • Lecture with beamer and blackboard • Practice in small groups • Expected activities of the students: Solving homework exercises, contributing to the tutorials	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes)	

The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Friedhelm Meyer auf der Heide
Learning material, literature
Routing and Data Management in Networks: Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes, Frank Thomson Leighton, M. Kaufmann Publishers, 1992. Originalarbeiten, Skript, Foliensatz der Vorlesung, Übungsblätter
Remarks
none

3.46 Mandatory Module: Seminar I

Module name	Seminar I / Seminar I
Workload	150 h
Credits	5 LP
Semester	• Seminar : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Seminar: Seminar (30h / 120 h / EN / WS oder SS / 15)	
Choices in Module	
Seminars from the Master Program Computer Science.	
Course prerequiites	
none	
Recommended proficiencies	
Seminar: Depends on the seminar topic.	
Content	
Seminar: In seminars, students work independently on an individual research topic by using background literature from various sources. They describe their research topic in a presentation followed by discussion and a written report. The presentation material and the written report serve two different purposes: Whereas the presentation material supports the lectures (held within a specific time period), the written report provides students the opportunity to acquire detailed information on the reported topic at a later date.	
Learning objectives	
In seminars, students learn the techniques for independent research work on non-trivial topics and how to present these topics in a presentation and in written form. The seminar participants are encouraged to familiarize themselves with a research-oriented subfield of computer science. They learn how to plan a presentation and hold it within the determined time frame (usually 45 to 60 minutes), and to prioritize the contents of the presentation. The participants experience how an audience obtains knowledge from a presentation, and how to exchange opinions and information in discussions. Seminars also teach rhetorical skills for presentations and discussions. Participating students learn how to structure a presentation according to its contents and how to use various means to illustrate complex issues. They also learn how to handle the background literature appropriately.	
Implementation method	
Seminar:	
Assessment modalities (duration)	
Presentation (45-60 minutes) and essay The responsible lecturer announces type and duration of assessment modalities in the first three weeks	

of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
none
Prerequisites for participation in module exam
none
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 5 credits.
Person responsible for the module
Prof. Dr. rer. nat. Johannes Blömer
Learning material, literature
Seminar: Abhängig vom Seminarthema.
Remarks
none

3.47 Mandatory Module: Seminar II

Module name	Seminar II / Seminar II
Workload	150 h
Credits	5 LP
Semester	• Seminar : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Seminar: Seminar (30h / 120 h / EN / WS oder SS / 15)	
Choices in Module	
Seminars from the Master Program Computer Science.	
Course prerequiites	
none	
Recommended proficiencies	
Seminar: Depends on the seminar topic.	
Content	
Seminar: In seminars, students work independently on an individual research topic by using background literature from various sources. They describe their research topic in a presentation followed by discussion and a written report. The presentation material and the written report serve two different purposes: Whereas the presentation material supports the lectures (held within a specific time period), the written report provides students the opportunity to acquire detailed information on the reported topic at a later date.	
Learning objectives	
In seminars, students learn the techniques for independent research work on non-trivial topics and how to present these topics in a presentation and in written form. The seminar participants are encouraged to familiarize themselves with a research-oriented subfield of computer science. They learn how to plan a presentation and hold it within the determined time frame (usually 45 to 60 minutes), and to prioritize the contents of the presentation. The participants experience how an audience obtains knowledge from a presentation, and how to exchange opinions and information in discussions. Seminars also teach rhetorical skills for presentations and discussions. Participating students learn how to structure a presentation according to its contents and how to use various means to illustrate complex issues. They also learn how to handle the background literature appropriately.	
Implementation method	
Seminar:	
Assessment modalities (duration)	
Presentation (45-60 minutes) and essay The responsible lecturer announces type and duration of assessment modalities in the first three weeks	

of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
none
Prerequisites for participation in module exam
none
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 5 credits.
Person responsible for the module
Prof. Dr. rer. nat. Johannes Blömer
Learning material, literature
Seminar: Abhängig vom Seminarthema.
Remarks
none

3.48 Elective Module: Software Analysis

Module name	Software Analysis / Software Analysis
Workload	180 h
Credits	6 LP
Semester	• Software Analysis : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Software Analysis: Lecture (45h / 105h / EN / WS oder SS / 40) Software Analysis: Tutorial (30h / 0h / EN / WS oder SS / 40)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Software Analysis: Logic, imperative programming	
Content	
Software Analysis: The course studies techniques (mostly) statically analysing properties of programs. Such information can be valuable for optimization phases of compilers (like knowledge about live variables or constant values of variables), but might also help verifying the correctness of programs, i.e., the adherence to certain safety requirements.	
Learning objectives	
The students know the key principles of data flow analyses and their advantage and disadvantage. They can determine when to employ a data flow analysis and when a path sensitive analysis. Students can design their own analysis and know how to put them in practice. They know the principle of overapproximation.	
Implementation method	
Software Analysis: Partially slides and partially board writing. All essential concepts and techniques will be repeatedly applied in examples during the tutorial.	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	

Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Heike Wehrheim
Learning material, literature
Software Analysis: Nielson, Nielson, Hankin: Principles of Program Analysis Vorlesungsfolien, Übungsaufgaben
Remarks
none

3.49 Elective Module: Software Quality Assurance

Module name	Software Quality Assurance / Software Quality Assurance
Workload	180 h
Credits	6 LP
Semester	• Software Quality Assurance : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Software Quality Assurance: Lecture (45h / 105h / EN / SS / 90) Software Quality Assurance: Tutorial (30h / 0h / EN / SS / 30)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Software Quality Assurance: Programming, Modeling, Model-based software development	
Content	
Software Quality Assurance: The aim of the lecture is to cover approaches, technologies and strategies related to quality assurance for software systems. These include on the one hand constructive approaches such as design patterns, anti-patterns, domain-specific languages, model driven development, model quality analysis, and architectural styles, and on the other hand analytic approaches such as static reviewing techniques and dynamic testing techniques.. Furthermore, approaches for the improvement of the software development process and international standards like ISO 9001, 9126, CMM etc. are covered.	
Learning objectives	
The students are able to explain quality characteristics of software development processes, software models as well as software systems. They have understood constructive and analytical techniques used to ensure quality properties, and they are able to apply them. They can describe standards for measuring process and product quality. They are able to understand new research approaches in the area of process and product quality.	
Implementation method	
Software Quality Assurance: Partially slides and partially board writing. All essential concepts and techniques will be repeatedly applied in examples during the tutorial. In a lab part, the techniques will be employed using tools, particularly testing tools.	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes)	

The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Gregor Engels
Learning material, literature
Software Quality Assurance: Daniel Galin: Software Quality Assurance: From Theory to Implementation, Pearson / Addison Wesley, 2004 Vorlesungsfolien, Übungsaufgaben
Remarks
none

3.50 Elective Module: Type Systems for Correctness and Security

Module name	Type Systems for Correctness and Security / Type Systems for Correctness and Security
Workload	180 h
Credits	6 LP
Semester	• Typsysteme für Korrektheit und Sicherheit : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Type Systems for Correctness and Security: Lecture (45h / 105h / EN / WS / 30) Type Systems for Correctness and Security: Tutorial (30h / 0h / EN / WS / 30)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Type Systems for Correctness and Security: From the lecture "Foundations of programming languages": Language properties, Syntactical program structures, Data types and type hierarchies, Functional programming	
Content	
Type Systems for Correctness and Security: Type systems in programming languages prevent illicit behavior of programs from the very start. They provide valuable feedback for programmers to prevent bugs, crashes or even security vulnerabilities. In this lecture we will study the theory, the properties and the implementation of modern type systems. We will take a pragmatic view on type systems and will develop type checkers along our way in this course to put theory into practice immediately. We will also take a closer look at type systems of well-known programming languages like Java or Scala.	
Learning objectives	
The participant will be able to understand and develop the definition and implementation of type systems. Acquired theoretical and practical knowledge will be discussed in the course such that participants can transfer the methods, techniques, and practices of this course in different problem settings.	
Implementation method	
Type Systems for Correctness and Security: • Lecture • Discussions • Reading • Exercises with accompanying implementation	

Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Dr. Ben Hermann
Learning material, literature
Typsysteme für Korrektheit und Sicherheit: • Vorlesungsfolien • Übungsaufgaben Weiterhin wird folgende Literatur empfohlen: • Benjamin C. Pierce. 2002. Types and Programming Languages. The MIT Press. • Benjamin C. Pierce. 2004. Advanced Topics in Types and Programming Languages. The MIT Press.
Remarks
none

3.51 Elective Module: Usability Engineering Practice

Module name	Usability Engineering Practice / Usability Engineering Practice
Workload	180 h
Credits	6 LP
Semester	• Praxis des Usability Engineering : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Usability Engineering Practice: Lecture (45h / 105h / EN / WS oder SS / 40) Usability Engineering Practice: Tutorial (30h / 0h / EN / WS oder SS / 40)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Usability Engineering Practice: It is expected that the audience has attended the Bachelor class "Interaction Design", as there the basic knowledge about usability methods are treated.	
Content	
Usability Engineering Practice: The class „Usability Engineering Practice" is a continuation of the Bachelor class "Interaction Design". There the basic methods of usability engineering (user tests, inspection methods) have been introduced. In the class single selected and currently updated methods (such as cognite walkthrough, card sorting, value-centered design) are treated in detail and with a lot of practical exercises. In addition, the class covers aspects, concepts and methods grouped "around" usability, by dealing with user experience, approaches such as extreme usability, aesthetics or health- or safety-related aspects.	
Learning objectives	
Factual knowledge is presented about current usability methods and approaches. In addition, the students learn to apply these methods, as realisticly-sized examples are dealt with and treated practically in the exercises. In the end, the students are able to assess the usability of software systems, in a systematic, tool-supported way; also, they learn about the inherent limits of such an evaluation process.	
Implementation method	
Usability Engineering Practice: The class is held as a classical lecture with beamer presentation, supported by koaLA - the e-learning environment of the university. That system is used to provide the slides of the class, written homeworks, access to software used during the class and video recordings of every lecture. During the class we have interactive meetings, groupwork in several chapters, e.g. for aesthetics and value-centered design. During exercise hours the students present their solutions to the homework by showing a small presentation with slides.	

Assessment modalities (duration)
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.
Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Gerd Szwillus
Learning material, literature
Praxis des Usability Engineering: • Vorlesungsfolien als PDF zum Herunterladen • Schriftliche Hausaufgaben • Diverse, während der Vorlesung eingesetzte Modellierungswerkzeuge • Ben Shneiderman: Designing the User Interface: Strategies for Effective Human-Computer Interaction, 2009 • David Benyon: Designing Interactive Systems: A Comprehensive Guide to HCI, UX and Interaction Design, 2009 • Jakob Nielsen und Raluca Budiu: Mobile Usability, 2012 • Effie Lai-Chong Law et al. (ed): Maturing Usability: Quality in Software, Interaction and Value, 2007
Remarks
none

3.52 Elective Module: Vehicular Networking

Module name	Vehicular Networking / Vehicular Networking
Workload	180 h
Credits	6 LP
Semester	• Vehicular Networking : arbitrary
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
Vehicular Networking: Lecture (45h / 105h / EN / SS / 40) Vehicular Networking: Tutorial (30h / 0h / EN / SS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
Vehicular Networking: System software and system-level programming	
Content	
Vehicular Networking: Today's automotive industry is increasingly relying on computer science in product innovation. Young professionals are expected to have specialized knowledge in the fields of electronics, software and vehicular networks – both in-car networks and networks of moving cars. This lecture teaches important concepts from these domains, starting with in-car networks (from individual electronic control units, modern bus systems, system and network architectures, to driver assistance functions, security and safety). The lecture then moves to networks of moving cars (from communication technology and system architectures, to the design of advanced traffic information systems, security and safety). Particular emphasis is given to the relevant question of balancing users' privacy with their safety and security.	
Learning objectives	
The learning objective is to unserstand the fundamental concepts of vehicular networking. Students understand these concepts and are able to apply this knowledge.	
Implementation method	
Vehicular Networking: Lecture with practical exercises	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	

Partial module exams
none
Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Falko Dressler
Learning material, literature
Vehicular Networking: Folien, Lehrbücher, Papiere
Remarks
none

3.53 Elective Module: VLSI Testing

Module name	VLSI Testing / VLSI Testing
Workload	180 h
Credits	6 LP
Semester	• VLSI Testing : 1
Course: Teaching Form (Contact hours / Self study / Language / Date / Group size)	
VLSI Testing: Lecture (30h / 120h / EN / WS / 40) VLSI Testing: Tutorial (30h / 0h / EN / WS / 20)	
Choices in Module	
none	
Course prerequiites	
none	
Recommended proficiencies	
VLSI Testing: Digital Design	
Content	
VLSI Testing: The course "VLSI Testing" focuses on techniques for detecting hardware defects in micro-electronic circuits. Algorithms for test data generation and test response evaluation as well as hardware structures for design for test (DFT) and on-chip test implementation (BIST) are presented.	
Learning objectives	
After attending the course, the students will be able to • describe fault models, DFT techniques, and test tools, • explain and apply the underlying models and algorithms for fault simulation and test generation, • analyze systems with respect to their testability and to derive appropriate test strategies.	
Implementation method	
VLSI Testing: • Lecture based on slide presentation, extensions on blackboard • Exercises in small groups based on exercise sheets with students presenting their own solutions • Hands-on exercises using various software tools	
Assessment modalities (duration)	
Oral exam (ca. 40 Minutes) The responsible lecturer announces type and duration of assessment modalities in the first three weeks of the lecture period at latest.	
Partial module exams	
none	

Course achievement / qualifying participation
Studienleistung: written exercises The responsible lecturer announces the requirements for course achievements in the first three weeks of the lecture period at latest.
Prerequisites for participation in module exam
Passing of course achievement.
Prerequisites for obtaining credits
Passing of module exam.
Weight for overall grade
The module is weighted with 6 credits.
Person responsible for the module
Prof. Dr. Sybille Hellebrand
Learning material, literature
VLSI Testing: • Vorlesungsfolien • M. L. Bushnell, V. D. Agrawal, Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits, Boston, Dordrecht, London: Kluwer Academic Publishers, 2000 • L.-T. Wang, C.-W. Wu, X. Wen, VLSI Test Principles and Architectures: Design for Testability, Morgan Kaufmann Series in Systems on Silicon, ISBN: 0123705975 • Aktuelle Hinweise auf ergänzende Literatur und Lehrmaterialien im koala-Kurs
Remarks
none

Appendix A

Summary Tables

A.1 Focus Areas and Modules

	Algorithm Design (S. 8)	Computer Systems (S. 9)	Intelligence and Data (S. 10)	Networks and Communication (S. 11)	Software Engineering (S. 12)
Adaptive Hardware and Systems (S. 15)	-	X	X	-	-
Advanced Algorithms (S. 17)	X	-	-	-	-
Advanced Compiler Construction (S. 19)	-	-	-	-	X
Advanced Complexity Theory (S. 21)	X	-	-	-	-
Advanced Computer Architecture (S. 23)	-	X	-	-	-
Advanced Distributed Algorithms and Data Structures (S. 25)	X	-	-	X	-
Advanced Software Engineering: Methods, Architectures, Industrial Applications (S. 27)	-	-	-	-	X
Algorithmic Game Theory (S. 29)	X	-	-	-	-
Algorithms for Highly Complex Virtual Scenes (S. 31)	X	-	-	-	-
Algorithms for Synthesis and Optimization of Integrated Circuits (S. 33)	-	X	-	-	-
Architektur paralleler Rechnersysteme (S. 35)	-	X	-	-	-
Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies (S. 37)	-	-	-	X	-
Build It, Break It, Fix It (S. 39)	-	-	-	-	X
Clustering Algorithms (S. 41)	X	-	X	-	-
Compiler Construction (S. 43)	-	-	-	-	X
Computational Geometry (S. 45)	X	-	-	-	-

Designing code analyses for large-scale software systems (S. 51)	-	-	-	-	X
Empiric performance evaluation (S. 53)	-	X	-	X	X
Foundations of Cryptography (S. 55)	X	-	-	-	-
Fundamentals of Model-Driven Engineering (S. 57)	-	-	-	-	X
Future Internet (S. 60)	-	-	-	X	-
Hardware/Software Codesign (S. 64)	-	X	-	-	-
High-Performance Computing (S. 66)	-	X	-	-	X
Intelligence in Embedded Systems (S. 68)	-	X	X	-	-
Interactive Data Visualization (S. 70)	-	-	X	-	-
Kontextuelle Informatik (S. 47)	-	-	-	-	X
Language-Based Security (S. 72)	-	-	-	-	X
Linear and Integer Optimization (S. 74)	X	-	-	-	-
Logic and Automated Reasoning (S. 76)	-	-	X	-	-
Logic Programming for Artificial Intelligence (S. 78)	-	-	X	-	X
Machine Learning I (S. 80)	-	-	X	-	-
Machine Learning II (S. 82)	-	-	X	-	-
Master-Abschlussarbeit (S. 84)	-	-	-	-	-
Mobile Communication (S. 86)	-	-	-	X	-
Model Checking (S. 88)	-	-	-	-	X
Model-Based Interface Development (S. 90)	-	-	-	-	X
Network Simulation (S. 93)	-	-	-	X	-
Networked Embedded Systems (S. 95)	-	-	-	X	-
Planning and Heuristic Search (S. 97)	-	-	X	-	-
Projektgruppe (S. 99)	-	-	-	-	-
Public-Key Cryptography (S. 101)	X	-	-	-	-
Reconfigurable Computing (S. 103)	-	X	-	-	-
Routing and Data Management in Networks (S. 105)	X	-	-	X	-
Seminar I (S. 107)	-	-	-	-	-
Seminar II (S. 109)	-	-	-	-	-
Software Analysis (S. 111)	-	-	-	-	X
Software Quality Assurance (S. 113)	-	-	-	-	X
Studium Generale (S. 62)	-	-	-	-	-
Type Systems for Correctness and Security (S. 115)	-	-	-	-	X
Usability Engineering Practice (S. 117)	-	-	-	-	X
Vehicular Networking (S. 119)	-	-	-	X	-

VLSI Testing (S. 121)	-	X	-	-	-

A.2 Modules and Courses

	Adaptive Hardware and Systems (S. 15)	Advanced Algorithms (S. 17)	Advanced Compiler Construction (S. 19)	Advanced Complexity Theory (S. 21)	Advanced Computer Architecture (S. 23)	Advanced Distributed Algorithms and Data Structures (S. 25)	Advanced Software Engineering: Methods, Architectures, Industrial Applications (S. 27)	Algorithmic Game Theory (S. 29)	Algorithms for Highly Complex Virtual Scenes (S. 31)	Algorithms for Synthesis and Optimization of Integrated Circuits (S. 33)	Architektur paralleler Rechnersysteme (S. 35)	Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies (S. 37)	Build It, Break It, Fix It (S. 39)	Clustering Algorithms (S. 41)	Compiler Construction (S. 43)	Computational Geometry (S. 45)	Deductive Verification (S. 49)	Designing code analyses for large-scale software systems (S. 51)	Empirische performance evaluation (S. 53)	Foundations of Cryptography (S. 55)	Fundamentals of Model-Driven Engineering (S. 57)	Future Internet (S. 60)	Hardware/Software Codesign (S. 64)	High-Performance Computing (S. 66)	Intelligenz in Embedded Systems (S. 68)	Interactive Data Visualization (S. 70)	Kontextuelle Informatik (S. 47)	Language-Based Security (S. 72)	Linear and Integer Optimization (S. 74)	Logic and Automated Reasoning (S. 76)	Logic Programming for Artificial Intelligence (S. 78)	Machine Learning I (S. 80)	Machine Learning II (S. 82)	Master-Abschlussarbeit (S. 84)	Mobile Communication (S. 86)	Model Checking (S. 88)	Model-Based Interface Development (S. 90)	Network Simulation (S. 93)	Networked Embedded Systems (S. 95)	Planning and Heuristic Search (S. 97)	Projektgruppe (S. 99)	Public-Key Cryptography (S. 101)	Reconfigurable Computing (S. 103)	Routing and Data Management in Networks (S. 105)	Seminar I (S. 107)	Seminar II (S. 109)	Software Analysis (S. 111)	Software Quality Assurance (S. 113)	Studium Generale (S. 62)	Type Systems for Correctness and Security (S. 115)	Usability Engineering Practice (S. 117)	Vehicular Networking (S. 119)	VLSI Testing (S. 121)
Adaptive Hardware and Systems (S. ??)	–																																																
Advanced Algorithms (S. ??)		–																																															
Advanced Compiler Construction (S. ??)			–																																													
Advanced Complexity Theory (S. ??)				–																																												
Advanced Computer Architecture (S. ??)				6	–																																											
Advanced Distributed Algorithms and Data Structures (S. ??)						–																																										
Advanced Software Engineering: Methods, Architectures, Industrial Applications (S. ??)							–																																									
Algorithmic Game Theory (S. ??)								–																																									
Algorithmen für hochkomplexe virtuelle Szenen (S. ??)								6	–																																							
Algorithmic Game Theory (S. ??)										–																																						
Algorithmische Geometrie (S. ??)								6			–																																					
Algorithms for Synthesis and Optimization of Integrated Circuits (S. ??)										6		–																																				
Architektur paralleler Rechnersysteme (S. ??)											6		–																																			
Bitcoins, Cryptocurrencies, and Privacy-Enhancing Technologies (S. ??)												–																																				
Build It, Break It, Fix It (S. ??)													–																																			
Clustering Algorithms (S. ??)													6	–																																		
Compilerbau (S. ??)															–																																	
Deductive Verification (S. ??)																–																																
Designing code analyses for large-scale software systems (S. ??)																	–																															
Empirische Leistungsbewertung (S. ??)																		–																														
Foundations of Cryptography (S. ??)																		6	–																													
Fundamentals of Model-Driven Engineering (S. ??)																			6	–																											
Future Internet (S. ??)																					–																											
Hardware/Software Codesign (S. ??)																					6	–																										
High-Performance Computing (S. ??)																							–																								
Intelligenz in eingebetteten Systemen (S. ??)																							6	–																							
Interaktive Datenvisualisierung (S. ??)																									–																						
Kontextuelle Informatik (S. ??)																									6	–																					
Linear and Integer Optimization (S. ??)																											–																			
Logic Programming for Artificial Intelligence (S. ??)																												–																		
Logik und automatisches Beweisen (S. ??)																													–																	
Maschinelles Lernen I (S. ??)																														–																
Maschinelles Lernen II (S. ??)																															–															
Masterarbeit Arbeitsplan (S. ??)																																–														
Masterarbeit (S. ??)																																	–													
Mobilkommunikation (S. ??)																																		–												
Model Checking (S. ??)																																			–												
Modellbasierte Entwicklung von Benutzmgsschnittstellen (S. ??)																																				–											
Network Simulation (S. ??)																																					–										
Networked Embedded Systems (S. ??)																																						–									
Planen und Heuristische Suche (S. ??)																																							–									
Praxis des Usability Engineering (S. ??)																																								–								
Projektgruppe (S. ??)																																									–							
Public-Key Cryptography (S. ??)																																										–						
Reconfigurable Computing (S. ??)																																											–					
Routing and Data Management in Networks (S. ??)																																												–				
Seminar (S. ??)																																													–			
Software Analysis (S. ??)																																														–		
Software Quality Assurance (S. ??)																																														–		
Sprachbasierte Sicherheit (S. ??)																																															–	
Studium Generale (S. ??)																																															–	
Typsysteme für Korrektheit und Sicherheit (S. ??)																																															–	
Vehicular Networking (S. ??)																																																–	
VLSI Testing (S. ??)																																																	–